IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 June 2023, accepted 26 July 2023, date of publication 31 July 2023, date of current version 7 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3300034

RESEARCH ARTICLE

Advanced First-Order Optimization Algorithm With Sophisticated Search Control for Convolutional Neural Networks

KYUNG SOO KIM[1] **AND YONG SUK CHOI**[2]

[1]Department of Computer Engineering, Kumoh National Institute of Technology, Gumi, Gyeongbuk 39177, Republic of Korea
[2]Department of Computer Science and Engineering, Hanyang University, Seoul 04763, Republic of Korea

Corresponding author: Yong Suk Choi (cys@hanyang.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korean Government [Ministry of Education (MOE)] under Grant 2022R1I1A3065378 and [Ministry of Science and ICT (MSIT)] under Grant (2018R1A5A7059549, 2020R1A2C1014037), and also supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government (MSIT) under Grant 2020-0-01373.

ABSTRACT As the performance of computing devices such as graphics processing units (GPUs) has improved dramatically, many deep neural network models, especially convolutional neural networks (CNNs), have been widely applied to various applications such as image classification, semantic segmentation, and object recognition. However, effective first-order optimization methods for CNNs have rarely been studied, although many CNN models have been successfully developed. Accordingly, this paper investigates various advanced adaptive solution search methods and proposes a new first-order optimization algorithm for CNNs called Adam-ASC. Our approach uses four sophisticated adaptive solution search methods to adjust its search strength in the complicated large-dimensional weight solution space spanned by a loss function. At the same time, we explain how they can be combined compensatively to form a complete optimizer with a detailed implementation. From the experiments, we found that our Adam-ASC can significantly improve the image recognition performance of practical CNNs in both the image classification and segmentation tasks. These experimental results show that the four fundamental methods of Adam-ASC and their compensative combination strategy play a crucial role in training CNNs by effectively finding their optimal weights.

INDEX TERMS Machine learning, deep learning, convolutional neural networks, optimization methods, gradient methods, image classification, image segmentation.

I. INTRODUCTION

Since the convolutional neural network (CNN) was developed in 1998 [1], it has shown remarkable performance in many image processing tasks, such as image classification [2], [3], image segmentation [4], [5], and object tracking & detection [6], [7], [8]. Accordingly, CNNs have been widely used as a basic model to handle complex image data in the real world [9], [10], [11], [12]. Such excellent ability in image processing tasks is caused by the distinctive properties of the CNN, which are different from the traditional multi-layer perceptron-based feedforward neural networks, such as convolutional filters and subsampling operations. In addition, modern CNNs are much more complicated and have extensive architectures to improve their performance when compared to the original CNNs.

Accordingly, many recent studies have focused on designing optimal CNN models for specific tasks to enhance their performance [13], [14], [15], [16]. Such research paradigms are mainly divided into two approaches: one is to increase the size of the model by stacking more layer blocks. The other is to densely connect each layer block using skip connections (or residual connections) [17], [18]. As a result, the latest CNNs have become more massive than the traditional ones and require more computation and memory, making it extremely difficult to use them on devices with limited

The associate editor coordinating the review of this manuscript and approving it for publication was Ines Domingues.

computational resources and memory. Therefore, it is essential to study the fundamental training and optimization mechanisms for CNNs to prevent the model size from increasing drastically and to improve its performance.

In general, most CNNs are trained using classical first-order optimization algorithms such as stochastic gradient descent (*SGD*) [19], *RMSProp* [20], and adaptive momentum (*Adam*) [21]. One of these common properties is that they require relatively less computation than second-order optimization algorithms, such as the Newton method [22], BFGS [23], and L-BFGS [24], by using gradient information when searching for an optimal weight in a solution space which is composed of many weights spanned by a loss function. As explained earlier, since a CNN consists of many weight tensors, the first-order optimization methods have been widely used to train most CNNs.

Among these first-order optimization algorithms, the Adam optimizer is one of the most popular methods for training CNNs. Unlike the existing SGD algorithm, the Adam optimizer controls an exponential moving average of historical gradients called *momentum*. By applying such adaptive momentum mechanisms, the Adam optimizer could effectively alleviate the slow convergence suffered by the SGD. Accordingly, many recent first-order optimization algorithms for training CNNs have been developed based on the Adam optimizer [25]. For example, *AdaBelief* [26], *diffGrad* [27], and *SAdam* [28] are typical Adam-based optimizers that have been successfully used to train many practical CNNs. However, as the structure of CNNs becomes more extensive and complicated than the previous ones, the existing Adam-based optimizers have also shown some critical weaknesses, such as insufficient convergence and unstable solution search ability, which often cause the trained CNNs to have unexpectedly low accuracy or insufficient learning ability [29].

To overcome the weaknesses of Adam-based optimizers, it is necessary to uncover various components that affect the search for optimal weights and study sophisticated methods with detailed solution search and control mechanisms for training CNNs. Accordingly, in this paper, we aim to improve the learning performance of CNNs by developing a new first-order optimization algorithm designed for CNNs, called *Adam-ASC*(**Adam**-based first-order optimization with **A**dvanced **S**earch **C**ontrol). We mainly focus on effectively searching the complicated solution space spanned by a loss function of CNNs by proposing four sophisticated adaptive solution search methods and combining these strategies. In addition, we experimentally verify how much our proposed method improves the learning performance of practical CNNs in image classification and segmentation tasks. From the experiments, we found that our Adam-ASC achieves better optimization performance and convergence ability than the existing first-order optimization algorithms when our proposed method is used to train practical CNNs such as ResNet [17], DenseNet [18], and FC-DenseNet [30].

Meanwhile, our study proposes a new research paradigm for CNNs, which is distinguished from the existing ones for the following reasons: First, our proposed method can effectively improve the learning ability of CNNs without oversizing the model or changing its architecture. Second, our approach can be used as a general optimizer to train any CNNs regardless of the target applications by simply replacing the existing optimizers. Third, our study is significantly challenging and worthwhile because it proposes the fundamental core techniques to effectively improve CNNs without excessive computation.

Furthermore, our proposed method has several novelties and significance when compared to the classical studies on first-order optimization algorithms used to train CNNs, as follows:

- First, we have developed several optimal solution search mechanisms that can significantly contribute to a stable solution search by preventing the excessive oscillation of the search trajectories around any local or global minima. In particular, our new search mechanisms are helpful in effectively analyzing the large-scale solution space by comprehensively utilizing much of the information observed from the current and historical gradients.
- Second, based on these new solution search mechanisms, we developed a new first-order optimization algorithm that further improves the performance of practical CNNs by exhaustively searching for the optimal weights in a large-dimensional weight solution space.
- Third, from the comprehensive experiments with practical CNN models, we found that the CNN models trained by our new optimization algorithm showed better performance than traditional optimizers in terms of training and testing accuracies. Thus, our study has remarkable significance in that Adam-ASC can be widely used as one of the fundamental methods for many studies to further improve the training performance of various CNN models.

This paper is organized as follows. Section II introduces several fundamental concepts of CNNs and their traditional optimization algorithms. In Section III, we present our proposed optimization method, called Adam-ASC, with their detailed principles. Then, we show a theoretical analysis to prove its convergence. In Section IV, we present the experimental results that validate the optimization performance of our proposed method for CNNs. In Section V, we provide a detailed discussion of the experimental results and various concerns related to Adam-ASC in detail. Finally, we discuss the advantages and disadvantages of our study and conclude this paper in Section VI.

II. RELATED WORKS

A. CONVOLUTIONAL NEURAL NETWORKS

A CNN is one of the neural network models proposed by Yann LeCun in 1998 [1]. The first CNN was designed under the influence of Neocognitron [31], which was developed by Kunihiko Fukushima in 1977 to improve the perceptive ability of any image data in computer vision applications. Unlike

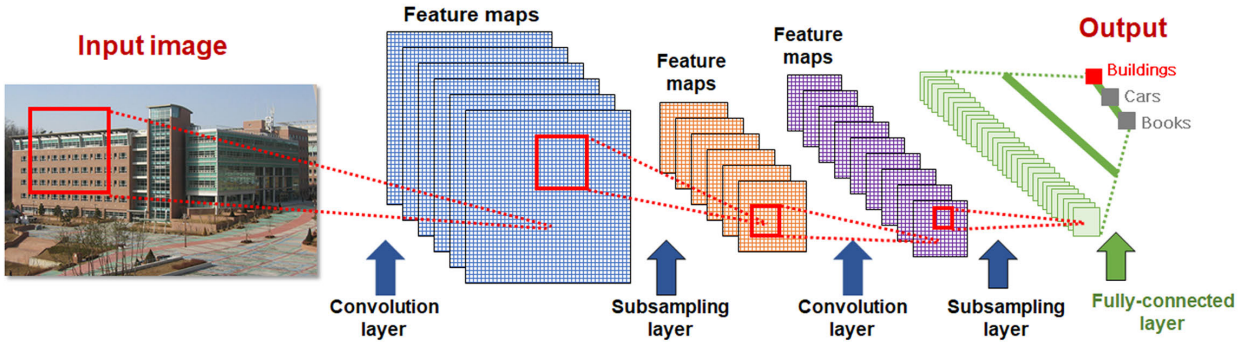

FIGURE 1. General architecture of the CNNs for image classification.

classical neural networks such as perceptron [32], CNNs have two critical properties, i.e., *invariance* and *equivariance* [33]. In this section, we briefly discuss the core properties of typical CNN models.

One of the most outstanding properties of the CNNs is "*invariance*," which means that the CNNs can preserve the positional information while extracting latent features from any given input data. This property is beneficial to process image data because each pixel contains its position and color information. A flattened neural network without the position invariance property has a relatively worse image perception ability than CNNs because it does not preserve the position information of each pixel involved in the input image data.

Meanwhile, another critical property of the CNNs is "*equivariance*," which means that the CNN can accurately recognize each feature information regardless of its positions. This property enables the CNN to extract various feature information from any image data accurately. Because of these properties, CNNs have been widely used as one of the standard models to handle complex data with arbitrary position information, such as image and video data.

Fig. 1 shows the general structure of a CNN, which includes many convolution layers, subsampling layers, and a fully connected layer. In the convolution layer, the feature information of an input image is extracted by a convolution filter [34]. Then, in the sub-sampling layer, the sub-sampling operation compresses the feature information extracted by the convolution filters. Such a series of convolution and sub-sampling operations make the CNNs accurately extract the core features of the input image, which are composed of multi-channels and various position information. The feature information extracted in the convolution and sub-sampling layers is passed to the fully-connected layer to produce its final output. Thus, the CNNs can effectively handle complex image data with multi-channel and position information, which is the essential property of the CNNs that distinguishes it from the existing multi-layer perceptron.

B. GRADIENT DESCENT AND ITS VARIATIONS TO TRAIN CNNS

Before explaining our proposed method, we need to consider what it means to train CNNs. As shown in Fig. 1, the CNN consists of many convolution and subsampling operations. Such operations compute linear combinations between the image data and the convolution & subsampling filters, which are modeled as weight tensors. Then, we can understand that the goal of training the CNN is to find the optimal convolution and subsampling filters to extract latent relationships between the input and output data as accurately as possible. As a result, this problem is an optimization problem to find the optimal weight tensors that constitute the optimal convolution and subsampling filters in the weight solution space spanned by the loss function.

Accordingly, many optimization methods have been actively studied to train CNNs effectively. Among them, the most popular optimization methods may be gradient descent and its variation algorithms [35], [36]. Let f be a CNN, $\boldsymbol{w}$ be a weight tensor of f, $<\boldsymbol{x}, \boldsymbol{y}>$ be a train and label sample pair, and $L\left(f\left(\boldsymbol{x}; \boldsymbol{w}\right), \boldsymbol{y}\right)$ be a loss function for f with a parameter $\boldsymbol{w}$. Then, we can derive a core formula of the gradient descent algorithm as

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha\, \boldsymbol{g}_t \tag{1}$$

where α is a learning rate that adjusts a degree of learning, and $\boldsymbol{g}_t$ is a gradient operator with respect to $\boldsymbol{w}^{(t)}$ of the loss function L evaluated at the step t, which is given by

$$\boldsymbol{g}_t \triangleq \nabla_{\boldsymbol{w}^{(t)}} \sum_{<\boldsymbol{x},\boldsymbol{y}>\in D} L\left(f\left(\boldsymbol{x}; \boldsymbol{w}^{(t)}\right), \boldsymbol{y}\right). \tag{2}$$

In other words, the gradient descent method searches an optimal weight tensor $\boldsymbol{w}^*$ step by step in the direction of the gradient of the loss function $\boldsymbol{g}_t$ in the solution space spanned by the loss function L. It implies that the quality of the found optimal weight $\boldsymbol{w}^*$ strongly depends on both the solution search direction and the search strength with the gradients observed at each training step t. In this case, searching the optimal weight by only depending on the direction of the gradient often leads to a wrong search direction, such as a local minimum or saddle points.

Fig. 2 shows an example surface of the loss function in the weight solution space. The x-axis and y-axis indicate the weight and loss function values, respectively. In Fig. 2, the following search direction is determined depending on the direction of the gradient $\boldsymbol{g}_t$ at the current weight $\boldsymbol{w}^{(t)}$. In this

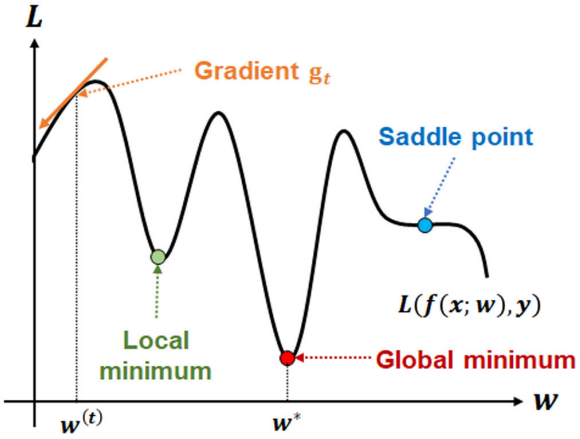

FIGURE 2. The local & global minima and the saddle point on the surface of the loss function.

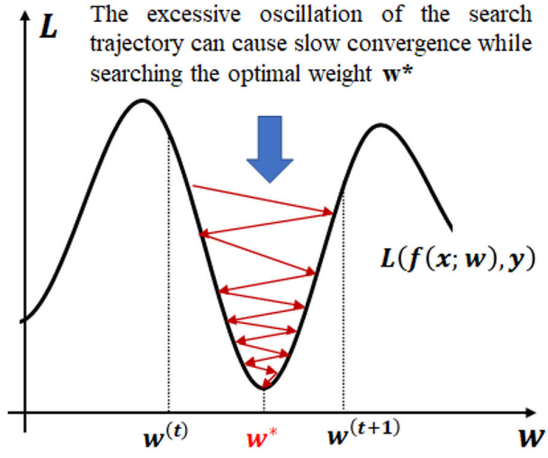

FIGURE 3. An example diagram of how the excessive movement of the search trajectory can cause slow convergence.

case, if $\boldsymbol{g}_t$ is in an unpromising direction, not an optimal weight $\boldsymbol{w}^*$, the following weight $\boldsymbol{w}^{(t+1)}$ will also be far from the optimal weight $\boldsymbol{w}^*$.

In addition, the strength of the search can also affect the performance of the solution search. For example, as shown in Fig. 3, if the learning rate α in (1) is set to an enormous value, the search trajectory will oscillate excessively around the optimal weight $\boldsymbol{w}^*$, eventually slowing down convergence. In fact, such phenomena are often observed in the classical gradient descent-based methods because their search direction is determined only by the gradient. Due to these characteristics, the gradient descent-based methods have several serious risks, such as falling into the local minima or suffering from unstable convergence near the local or global minima. Therefore, more sophisticated and stable solution search methods are needed to train large-scale CNNs effectively.

Accordingly, many optimization methods have been successfully proposed to overcome such weaknesses of the existing gradient descent-based methods. Significantly, the Adam optimizer [21] is one of the pioneering methods that solves many difficulties from which the traditional gradient-based methods had suffered by introducing adaptive first and second-order momentums. Algorithm 1 shows the pseudocode of the Adam optimizer. Unlike traditional gradient descent, the Adam optimizer uses the exponential moving average of historical gradients, called *momentum*, to determine the next search direction and adaptively controls the strength of the search. The Adam optimizer reduces the excessive oscillation of the search trajectory by using the historical gradient information to improve the optimization performance. The Adam optimizer showed better optimization ability for training various deep neural network models using CNNs than the existing optimization methods [21].

Algorithm 1 Adam Optimizer

Input: Initial weight tensor $\boldsymbol{w}_0$
Parameter: α, β_1, β_2
Output: Optimal weight tensor $\boldsymbol{w}^*$

Initialize $\boldsymbol{m}_0, \boldsymbol{s}_0$, randomly;
$t = 0$;
while(isConverged($\boldsymbol{w}_t, \boldsymbol{w}_{t-1}$) == False){
  $t = t + 1$;
  $\boldsymbol{g}_t = \nabla_{\boldsymbol{w}} L\left(f\left(\boldsymbol{x}; \boldsymbol{w}_t\right), \boldsymbol{y}\right)$;
  $\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\boldsymbol{g}_t$;
  $\boldsymbol{s}_{t+1} = \beta_2 \boldsymbol{s}_t + (1-\beta_2)\boldsymbol{g}_t^2$;
  $\hat{\boldsymbol{m}}_{t+1} = \boldsymbol{m}_{t+1}/(1-\beta_1^t)$;
  $\hat{\boldsymbol{s}}_{t+1} = \boldsymbol{s}_{t+1}/\sqrt{1-\beta_2^t}$;
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \hat{\boldsymbol{m}}_{t+1}/\hat{\boldsymbol{s}}_{t+1}$;
};
$\boldsymbol{w}^* = \boldsymbol{w}_{t+1}$;
return $\boldsymbol{w}^*$;

Thus, many recent optimization methods for training CNNs have been developed by further focusing on improving the Adam optimizer [25]. For example, AdaBelief [26], diffGrad [27], SAdam [28], and TAdam [37] are representative Adam-based optimizers. Algorithm 2 shows the pseudocode of AdaBelief, one of the most recent Adam-based advanced first-order optimizers. The general structure of AdaBelief is similar to that of the Adam optimizer. However, unlike the Adam optimizer, AdaBelief generates the second-order momentum by computing a difference between the current gradient and the updated first-order momentum. Such a method helps prevent the search strength from becoming excessively sensitive according to the values of the current gradient. As a result, AdaBelief could adjust the strength of the solution search further elaborately. Actually, AdaBelief showed better optimization performance with faster convergence than the existing methods, such as the Adam and RMSProp.

Nevertheless, the existing Adam-based optimizers still suffer from the problem of finding an approximate optimal

Algorithm 2 AdaBelief Optimizer

Input: Initial weight tensor $\boldsymbol{w}_0$
Parameter: α, β_1, β_2
Output: Optimal weight tensor $\boldsymbol{w}^*$

Initialize $\boldsymbol{m}_0$, $\boldsymbol{s}_0$, randomly;
$t = 0$;
while(isConverged($\boldsymbol{w}_t, \boldsymbol{w}_{t-1}$) == False){
  $t = t + 1$;
  $\boldsymbol{g}_t = \nabla_w L\left(f\left(\boldsymbol{x}; \boldsymbol{w}_t\right), \boldsymbol{y}\right)$;
  $\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\boldsymbol{g}_t$;
  $\boldsymbol{s}_{t+1} = \beta_2 \boldsymbol{s}_t + (1-\beta_2)\left(\boldsymbol{m}_{t+1} - \boldsymbol{g}_t\right)^2$;
  $\hat{\boldsymbol{m}}_{t+1} = \boldsymbol{m}_{t+1}/1-\beta_1^t$;
  $\hat{\boldsymbol{s}}_{t+1} = \boldsymbol{s}_{t+1}/\sqrt{1-\beta_2^t}$;
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha\hat{\boldsymbol{m}}_{t+1}/\hat{\boldsymbol{s}}_{t+1}$;
};
$\boldsymbol{w}^* = \boldsymbol{w}_{t+1}$;
return $\boldsymbol{w}^*$;

weight in the large-dimensional solution space. The representative reasons can be summarized as follows.

- First, the observed gradient is not necessarily guaranteed to lead every promising search direction to an optimal weight. Several gradients can sometimes lead to an unpromising direction, such as the saddle points or local minima, which are far from the optimal weights [38], [39].
- Second, the solution space spanned by a loss (or objective) function usually has a considerably complicated surface because its dimension is enormous [40]. Thus, we need more sophisticated and stable adaptive solution search methods than the classical ones.
- Third, when the gradient norm becomes extremely large, its solution search strength also increases excessively, slowing its convergence.

Such critical issues often cause severe problems for large-scale complicated CNNs because they involve complicated structures with many weights and complex connections. Therefore, our study aims to develop more sophisticated and stable optimization methods than the existing ones to effectively search for the approximate global optimal weight in the large-dimensional solution space.

III. SOPHISTICATED ADAPTIVE SOLUTION SEARCH CONTROL

In general, many conventional first-order optimization algorithms used to train DNNs have suffered from insufficient solution search abilities and slow convergence, making the solution search difficult. Several typical reasons for this are (i) unpromising search directions, (ii) large-dimensional complicated weight solution spaces spanned by a loss function, and (iii) unsophisticated search strength control methods. To address such issues effectively, we propose four sophisticated search control methods and show how they

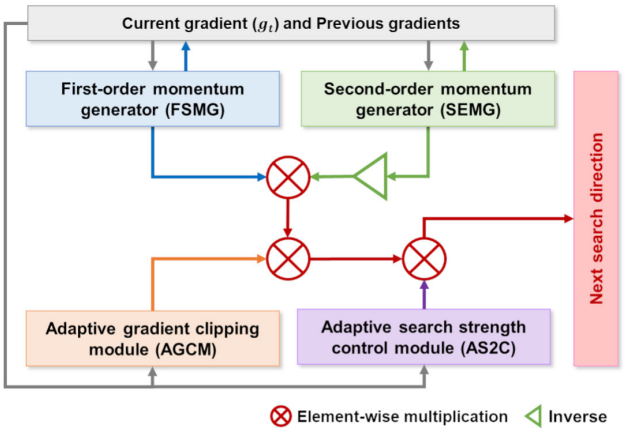

FIGURE 4. An overall architecture of Adam-ASC.

are coupled into an optimization algorithm with a detailed implementation.

Fig. 4 presents the overall architecture of our proposed optimization method. Our new approach consists of four core modules, which control the strength and direction of solution search adaptively and sophisticatedly in the solution space spanned by the loss function. Moreover, by compensatively combining them into one algorithm, we can quickly implement an improved Adam-based optimization algorithm to train CNNs effectively. The four core modules are listed below:

- First-order momentum generator (FSMG)
- Second-order momentum generator (SEMG)
- Adaptive gradient clipping module (AGCM)
- Adaptive search strength control module (AS2C)

In this section, we show how each module can improve the solution search ability with detailed explanations and formulas. Then, we propose our new first-order optimization algorithm for CNNs, i.e., Adam-ASC, with careful implementation and verify its convergence theoretically.

A. FIRST-ORDER MOMENTUM GENERATOR (FSMG)

Our first module aims to improve the computation of the first-order momentum to determine the subsequent search direction. Accordingly, we propose a robust first-order momentum generation method that minimizes unpromising gradients and adaptively controls their strength. As discussed in Section II, the gradient does not always indicate the promising search direction, i.e., an approximate global minimum. In particular, a gradient is significantly distinguished from the historical ones; it may head in any unpromising direction, called an *outlier gradient* [37], [41].

Fig. 5 shows the current first-order momentum $\boldsymbol{m}_t$ and two gradients ($\boldsymbol{g}_A$ and $\boldsymbol{g}_B$) in the two-dimensional solution space. $\boldsymbol{g}_A$ points to the global minimum. In this case, $\boldsymbol{g}_A$ makes the solution search direction to which the first-order momentum points more promising, which ultimately speeds up its convergence. On the other hand, $\boldsymbol{g}_B$, called the outlier gradient, leads to an unpromising direction far from the global

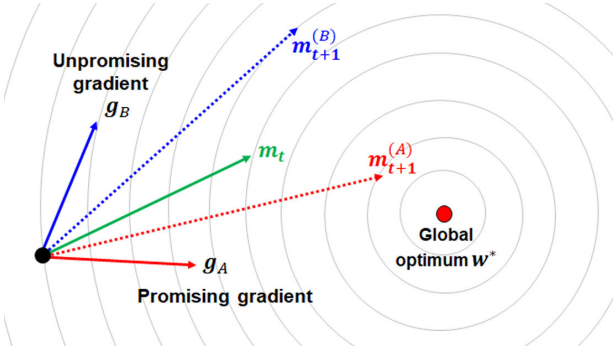

FIGURE 5. An influence of the promising and unpromising gradients when calculating the following first momentum at step t.

Algorithm 3 FSMG

Input: $\boldsymbol{m}, \boldsymbol{g}_t, \boldsymbol{Q}, \boldsymbol{q}, t$
Parameter: β_1, ϵ
Output: $\hat{\boldsymbol{m}}^{(new)}, \boldsymbol{m}^{(new)}, \boldsymbol{Q}^{(new)}, \boldsymbol{q}^{(new)}$

1. $\boldsymbol{\beta}_{1,t} = \boldsymbol{Q}/(\boldsymbol{Q}+\boldsymbol{q})$
2. $\boldsymbol{Q}^{(new)} = \left(2-\beta_1^{-1}\right)\boldsymbol{Q}+\boldsymbol{q}$
3. $\boldsymbol{v} = \mathbf{var}(\boldsymbol{g}_t)$ /* *variance of* $\boldsymbol{g}_t$ */
4. $\boldsymbol{q}^{(new)} = 2n\left(n+\sum_{j=1}^{n}\frac{(\boldsymbol{g}_{t,j}-\boldsymbol{m}_j)}{\boldsymbol{v}_j+\epsilon}\right)^{-1}$
5. $\boldsymbol{m}^{(new)} = \boldsymbol{\beta}_{1,t}\boldsymbol{m}+\left(1-\boldsymbol{\beta}_{1,t}\right)\boldsymbol{g}_t$
6. $\hat{\boldsymbol{m}}^{(new)} = \boldsymbol{m}^{(new)}/(1-\beta_1^t)$
7. return $\hat{m}^{(new)}, \boldsymbol{m}^{(new)}, \boldsymbol{Q}^{(new)}, \boldsymbol{q}^{(new)}$

optimum. Such an outlier gradient often brings on an imprecise and slow convergence because it continuously leads the first-order momentum to the unpromising direction. In other words, $\boldsymbol{m}_{t+1}^{(A)}$ and $\boldsymbol{m}_{t+1}^{(B)}$ present that the subsequent solution search direction $\boldsymbol{m}_{t+1}$ is strongly influenced by the current gradient $\boldsymbol{g}_t$.

From the examples described in Fig. 5, we can see that it is necessary to accurately detect the outlier gradients and minimize their influence when updating the first-order momentum. For this purpose, we introduce a new coefficient $\boldsymbol{\beta}_{1,t}$ as

$$\boldsymbol{\beta}_{1,t} = \boldsymbol{Q}_{t-1}/\left(\boldsymbol{Q}_{t-1}+\boldsymbol{q}_t\right). \tag{3}$$

Meanwhile, $\boldsymbol{Q}_{t-1}$ and $\boldsymbol{q}_{t-1}$ in (3) are incrementally updated by

$$\boldsymbol{Q}_t = \left(2-\beta_1^{-1}\right)\boldsymbol{Q}_{t-1}+\boldsymbol{q}_{t-1} \tag{4}$$

and

$$\boldsymbol{q}_t = 2n\left(n+\sum_{j=1}^{n}\left(\boldsymbol{g}_{t,j}-\boldsymbol{m}_{t,j}\right)/\left(\boldsymbol{v}_{t,j}+\epsilon\right)\right)^{-1} \tag{5}$$

where $\boldsymbol{g}_{t,j}$ and $\boldsymbol{m}_{t,j}$ indicate the jth entries in $\boldsymbol{g}_t$ and $\boldsymbol{m}_t$, respectively, and $\boldsymbol{v}_t$ is a variance of $\boldsymbol{g}_t$ [41].

Thus, the first-order momentum $\boldsymbol{m}_{t+1}$ is computed with the new coefficient shown in (3) by

$$\boldsymbol{m}_{t+1} = \boldsymbol{\beta}_{1,t}\boldsymbol{m}_t+\left(1-\boldsymbol{\beta}_{1,t}\right)\boldsymbol{g}_t. \tag{6}$$

Equation (5) shows that a difference between $\boldsymbol{g}_t$ and $\boldsymbol{m}_t$ is scaled by $\boldsymbol{v}_t+\epsilon$. In other words, if $\boldsymbol{g}_t$ is unpromising (i.e., it is an outlier gradient), $\boldsymbol{\beta}_{1,t}$ is further increased to minimize the influence of $\boldsymbol{g}_t$ in the following first-order momentum. As a result, the unpromising gradient $\boldsymbol{g}_t$ is minimized in the new first-order momentum shown in (6), which makes the following search direction more promising and robust.

By combining (4) – (6), we can implement the first-order momentum generator (FSMG) module, which is provided by Algorithm 3. The FSMG module takes the previous momentum $\boldsymbol{m}$ and two tensors $\boldsymbol{Q}$ and $\boldsymbol{q}$, the current gradient $\boldsymbol{g}_t$, and the current step t as its inputs. β_1 and ϵ are control parameters set by any user in advance. As outputs, the FSMG returns the new first-order momentum $\hat{m}^{(new)}$ and three updated tensors $\boldsymbol{m}^{(new)}$, $\boldsymbol{Q}^{(new)}$, and $\boldsymbol{q}^{(new)}$ to be used in the next step.

The FSMG is used as a base module to generate the first-order momentum in our new optimization algorithm, which will be shown in Section III-E.

B. SECOND-ORDER MOMENTUM GENERATOR (SEMG)

In Adam and its variants, the second-order momentum is used to adaptively adjust the strength of the first-order momentum. As shown in Algorithm 2, the typical second-order momentum is an exponential moving average of the squared gradients observed from the first step to the current step. However, such an ordinary second-order momentum often fails to accurately adjust the strength of the solution search because it strongly depends on the relative strength of each element in the gradient. To alleviate this weakness, AdaBelief [26], one of the state-of-the-art (SOTA) first-order optimizers, introduced a new second-order momentum using a squared error between the current gradient and the first-order momentum, which is formulated as

$$\boldsymbol{s}_{t+1} = \beta_2\boldsymbol{s}_t+(1-\beta_2)\left(\boldsymbol{m}_{t+1}-\boldsymbol{g}_t\right)^2 \tag{7}$$

where $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t+1}$ are the second-order momentums at the current step t and the following step $t+1$, and β_2 is a coefficient parameter for calculating the exponentially weighted average of $\left(\boldsymbol{m}_{t+1}-\boldsymbol{g}_t\right)^2$.

The new second-order momentum described in (7) is an exponential moving average of a squared error between the updated first-order momentum $\boldsymbol{m}_{t+1}$ and the current gradient $\boldsymbol{g}_t$. Accordingly, the dependence of the gradient $\boldsymbol{g}_t$ in the new second-order momentum $\boldsymbol{s}_{t+1}$ is further reduced when compared to the existing second-order momentum used in Adam. Nevertheless, the second-order momentum shown in (7) can be further improved by applying an adaptive noise term $0.01t^{-1}$, which is formulated as

$$\boldsymbol{s}_{t+1}^{*} = \boldsymbol{s}_{t+1}+0.01t^{-1}. \tag{8}$$

Equation (8) shows that the second-order momentum contains the adaptive noise term depending on the step variable t. We consider simple examples to analyze the effectiveness of

the noise term described in (8). In the first step ($t = 1$), the adaptive noise term 0.01 is added to $\boldsymbol{s}_{t+1}$. However, the magnitude of the adaptive noise term decreases monotonically as the training phase progresses, i.e., as t increases. In the initial training phases, the second-order momentum contains relatively more noise in its elements, which enhances the diversity of the solution searches. On the other hand, as the training progresses step by step, the influence of the noise in the second-order momentum is gradually reduced to zero. From (8), we can obtain a final second-order momentum by applying the bias correction method commonly used in Adam-based optimizers. Equation (9) represents the new second-order momentum with the adaptive noise and bias correction.

$$\hat{s}_{t+1} = \boldsymbol{s}^*_{t+1}/(1-\beta_2^t). \tag{9}$$

By using the new second-order momentum formulated in (9), we can expect to perform a more detailed and sophisticated search in the complicated solution space by introducing the additional noise term in the second-order momentum. Algorithm 4 shows how our second-order momentum generator (SEMG) is implemented. The SEMG takes four inputs $\boldsymbol{s}$, $\boldsymbol{g}_t$, $\boldsymbol{m}^{(new)}$, and t, with the parameter β_2. It then computes the new second-order momentum $\boldsymbol{s}^{(new)}$ with the adaptive noise $0.01t^{-1}$ and its bias correction $\hat{s}^{(new)}$ based on (7) – (9). Finally, the SEMG module returns $\boldsymbol{s}^{(new)}$ and $\hat{s}^{(new)}$, where $\boldsymbol{s}^{(new)}$ is used for its incremental update in the next step.

Algorithm 4 SEMG

Input: $\boldsymbol{s}, \boldsymbol{g}_t, \boldsymbol{m}^{(new)}, t$
Parameter: β_2
Output: $\hat{\boldsymbol{s}}^{(new)}, \boldsymbol{s}^{(new)}$

1. $\boldsymbol{s}^{(new)} = \beta_2 \boldsymbol{s} + (1-\beta_2)\left(\boldsymbol{m}^{(new)} - \boldsymbol{g}_t\right)^2$
2. $\boldsymbol{s}^* = \boldsymbol{s}^{(new)} + 0.01t^{-1}$
3. $\hat{\boldsymbol{s}}^{(new)} = \boldsymbol{s}^*/(1-\beta_2^t)$
4. return $\hat{s}^{(new)}$ and $\boldsymbol{s}^{(new)}$

C. ADAPTIVE GRADIENT CLIPPING MODULE (AGCM)

As explained in Section II, the first-order optimization methods search the approximate global optimum in the direction of the gradient. As shown in Fig. 6, a gradient with a large norm often makes its solution search strength extremely excessive. Such massive search strength causes its entire search trajectories to be longer, which slows down its convergence.

Accordingly, we introduce a new auxiliary momentum called the clipping momentum to effectively control such extreme search strength caused by gradients with large norms. Like the first-order momentum, the clipping momentum is an exponential moving average of the historical gradients. However, unlike the first- or second-order momentum, the clipping momentum accumulates the absolute value of each entry in the gradient. In other words, the clipping momentum is calculated as

$$\boldsymbol{c}_{t+1} = \beta_3 \boldsymbol{c}_t + (1-\beta_3)\left|\boldsymbol{g}_t\right|. \tag{10}$$

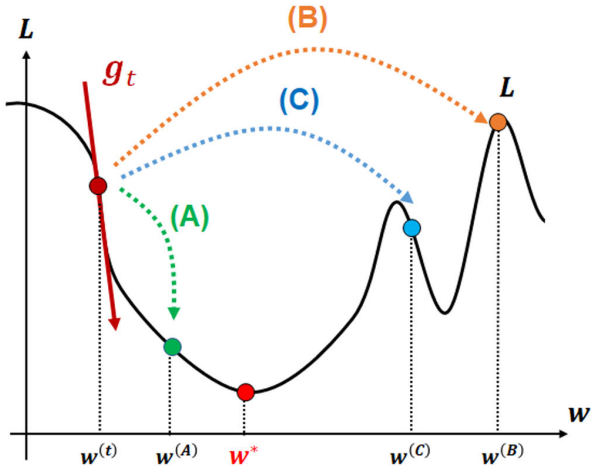

FIGURE 6. An example diagram illustrating the need for the gradient clipping mechanism.

where β_3 is a coefficient parameter to calculate the exponentially weighted average of $\left|\boldsymbol{g}_t\right|$. Then, a bias correction of (10) is computed by

$$\hat{\boldsymbol{c}}_{t+1} = \boldsymbol{c}_{t+1}/\left(1-\beta_3^{t+1}\right). \tag{11}$$

Thus, the clipping momentum maintains the average absolute strength of historical gradients by taking their exponentially moving average. Then, we compute an element-wise clipping rate tensor from (11) as

$$\boldsymbol{CR}_{t+1} = \min\left(\hat{\boldsymbol{c}}_{t+1}/\left(\left|\boldsymbol{m}_{t+1}\right|+\varphi\right), \boldsymbol{1}\right) \tag{12}$$

where $\boldsymbol{1}$ is a one tensor, $\min(\boldsymbol{x}, \boldsymbol{y})$ is an element-wise minimum function, and φ is a smoothing factor to prevent zero denominators.

In our proposed method, the clipping rate tensor $\boldsymbol{CR}_{t+1}$ adaptively adjust the search strength by multiplying it with the first-order momentum in an element-wise manner. If every entry in $\hat{c}_{t+1}$ is more than one in $\boldsymbol{m}_{t+1}$, its search strength is maintained because $\boldsymbol{CR}_{t+1} = \boldsymbol{1}$. Otherwise, its search strength is reduced as the ratio between $\hat{c}_{t+1}$ and $|\boldsymbol{m}_{t+1}|$, i.e., $\hat{\boldsymbol{c}}_{t+1}/(|\boldsymbol{m}_{t+1}| + \varphi)$. That is, when the magnitude of the clipping momentum is relatively larger than the absolute values of the first-order momentum, the clipping rate is set to one to prevent unreasonable search, such as the example cases (B) and (C) in Fig. 6. On the other hand, when the magnitude of the clipping momentum is relatively smaller than that of the first-order momentum, the clipping rate is set to a value less than one for more precise solution search.

Thus, we can derive the adaptive gradient clipping module (AGCM) as shown in Algorithm 5 from (10) – (12). In our proposed optimization method, the AGCM adaptively controls the strength of the solution search in each dimension. The detailed way to combine it with other modules is shown in the following sections.

Algorithm 5 AGCM

Input: $\boldsymbol{c}, \boldsymbol{g}_t, \boldsymbol{m}^{(new)}, t$
Parameter: β_3, φ
Output: $\boldsymbol{CR}, \boldsymbol{c}^{(new)}$

1. $\boldsymbol{c}^{(new)} = \beta_3 \boldsymbol{c} + (1-\beta_3)\left|\boldsymbol{g}_t\right|$
2. $\hat{\boldsymbol{c}} = \boldsymbol{c}^{(new)} / \left(1-\beta_3^{t+1}\right)$
3. $\boldsymbol{CR}^{(temp)} = \hat{\boldsymbol{c}} / \left(\left|\boldsymbol{m}^{(new)}\right| + \varphi\right)$
4. $\boldsymbol{CR} = \min\left(\boldsymbol{CR}^{(temp)}, \boldsymbol{1}\right)$
5. return $\boldsymbol{CR}$ and $\boldsymbol{c}^{(new)}$

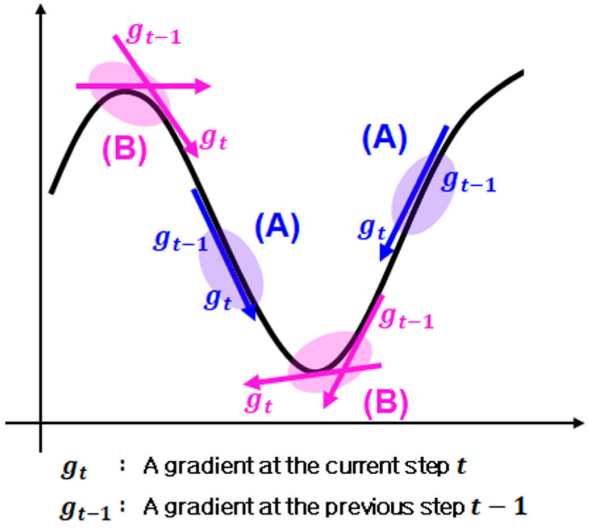

FIGURE 7. A method of measuring a degree of variation from the previous gradient to the current gradient at step t.

D. ADAPTIVE SEARCH STRENGTH CONTROL MODULE (AS2C)

In addition to the three methods described above, we can control the power of the solution search more elaborately by a detailed analysis of the observed gradients. In detail, we can adjust by considering a degree of gap between the previous and current gradients, i.e., $\left|\boldsymbol{g}_t - \boldsymbol{g}_{t-1}\right|$. Fig. 7 shows how our method sophisticatedly controls the strength of the solution search based on two successive gradients $\boldsymbol{g}_t$ and $\boldsymbol{g}_{t-1}$. As shown in case (A) of Fig. 7, suppose a degree of variation from $\boldsymbol{g}_{t-1}$ to $\boldsymbol{g}_t$ is small. In this case, it is reasonable to reduce the search strength to avoid passing the promising local (or global) optimum, which may be a global minimum. On the other hand, if its variation is large, as described in the case (B) of Fig. 7, a more active solution search can further accelerate its convergence. However, since the unrestricted enhancement of its convergence may cause the excessive oscillation of the search trajectory shown in Fig. 3, the strength of the solution search is maintained without reducing its strength. Such a search strength control mechanism can be implemented using a sigmoid function as follows.

$$\boldsymbol{\Gamma}_1 = \left(1 + \exp\left(-\left|\boldsymbol{g}_t - \boldsymbol{g}_{t-1}\right|\right)\right)^{-1}. \tag{13}$$

As shown in (13), each entry in $\boldsymbol{\Gamma}_1$ has a real value between 0.5 and 1. If any variation between two successive gradients is close to zero, each of the entries in $\boldsymbol{\Gamma}_1$ converges to 0.5. In this case, the strength of the solution search is reduced to half for further precise solution search around the current position. However, if the variation between two gradients is large, the entries in $\boldsymbol{\Gamma}_1$ become close to one to maintain the current search strength. Then, $\boldsymbol{\Gamma}_1$ is also multiplied by the first-order momentum on an element-by-element basis. Accordingly, the strength of each dimension is adaptively controlled according to the variations between the current and previous gradients.

Meanwhile, we introduce the warm-up strategy [42] as one of our search control mechanisms to increase the solution search strength as the training progresses gradually [41]. The warm-up strategy is formulated as

$$\gamma_2 = \sqrt{\frac{\rho_\infty (1-\beta_2)^t \left(\rho_t^2 - 6\rho_t + 8\right)}{\rho_t \left(\rho_\infty^2 - 6\rho_\infty + 8\right)}}^{1-\sum_{i=1}^{4} \delta_{\rho_t, i}} \tag{14}$$

where $\rho_\infty = 2/(1-\beta_2) - 1$, $\rho_t = \rho_\infty - 2t\beta_2^t \left(1-\beta_2^t\right)^{-1}$, and $\delta_{\rho_t, i}$ is a Kroneker delta.

Now, we can derive the adaptive search strength control (AS2C) module by coupling (13) and (14) into a function as described in Algorithm 6. In line 6, γ_2 is multiplied by all entries in $\boldsymbol{\Gamma}_1$ because $\boldsymbol{\Gamma}_1$ is a tensor and γ_2 is a scalar value. Accordingly, the output of AS2C, i.e., $\boldsymbol{\Gamma}$, is a tensor. In addition, AS2C has three control modes by setting two Boolean parameters, `isCG` and `isWS`, as follows: if `isCG=1` and `isWS=0`, the solution search control is performed only by (13). On the other hand, if `isCG=0` and `isWS=1`, the search strength is controlled by (14). Similarly, if `isCG=1` and `isWS=1`, both (13) and (14) control the solution search strength. Finally, "`isCG=0` and `isWS=0`" is not considered in our method because it indicates that AS2C is not working at all.

Algorithm 6 AS2C

Input: $\boldsymbol{g}_t, \boldsymbol{g}_{t-1}, t$
Parameter: β_2, isCG, isWS
Output: $\boldsymbol{\Gamma}$

1. $\boldsymbol{\Gamma}_1 = 1/\left(1 + \exp\left(-\left|\boldsymbol{g}_t - \boldsymbol{g}_{t-1}\right|\right)\right)$
2. $\rho_\infty = 2/(1-\beta_2) - 1$
3. $\rho_t = \rho_\infty - 2t\beta_2^t \left(1-\beta_2^t\right)^{-1}$
4. $V = 1 - \sum_{i=1}^{4} \delta_{\rho_t, i}$
5. $\gamma_2 = \left(\frac{\rho_\infty (1-\beta_2)^t \left(\rho_t^2 - 6\rho_t + 8\right)}{\rho_t \left(\rho_\infty^2 - 6\rho_\infty + 8\right)}\right)^{V/2}$
6. $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_1^{\text{isCG}} \times \gamma_2^{\text{isWS}}$
7. return $\boldsymbol{\Gamma}$

E. IMPLEMENTATION OF ADAM-ASC

Algorithm 7 shows a complete implementation of our new first-order optimizer, i.e., Adam-ASC. The overall architecture of Adam-ASC is similar to the Adam optimizer, as shown in Algorithm 1. However, our Adam-ASC incorporates more

Algorithm 7 Adam-ASC Optimizer

Input: An initial weight tensor $\mathbf{w}_0$
Parameter: α, β_1, β_2, β_3, ϵ, φ, isCG, isWS
Output: An optimal weight tensor $\boldsymbol{w}^*$

1. Initialize $\boldsymbol{m}_0, \boldsymbol{s}_0, \boldsymbol{c}_0, \boldsymbol{Q}_0, \boldsymbol{q}_0$
2. $t = 0$
3. **while**(isConverged($\boldsymbol{w}_t, \boldsymbol{w}_{t-1}$) == False){
4. $t = t + 1$
5. $\boldsymbol{g}_t = \nabla_{\boldsymbol{w}} L\left(f\left(\boldsymbol{x}; \boldsymbol{w}_t\right), \boldsymbol{y}\right)$
6. $\hat{\boldsymbol{m}}_{t+1}, \boldsymbol{m}_{t+1}, \boldsymbol{Q}_{t+1}, \boldsymbol{q}_{t+1} =$ **FSMG** $\left(\boldsymbol{m}_t, \boldsymbol{g}_t, \boldsymbol{Q}_t, \boldsymbol{q}_t, t; \beta_1, \epsilon\right)$
7. $\hat{\boldsymbol{s}}_{t+1}, \boldsymbol{s}_{t+1} =$ **SEMG**$(\boldsymbol{s}_t, \boldsymbol{g}_t, \boldsymbol{m}_{t+1}, t; \beta_2)$
8. $\boldsymbol{CR}, \boldsymbol{c}_{t+1} =$ **AGCM** $\left(\boldsymbol{c}_t, \boldsymbol{g}_t, \boldsymbol{m}_{t+1}, t; \beta_3, \varphi\right)$
9. $\boldsymbol{\Gamma} =$ **AS2C**$(\boldsymbol{g}_t, \boldsymbol{g}_{t-1}, t; \beta_2,$ isCG, isWS$)$
10. $\boldsymbol{\Psi}_{t+1} = \boldsymbol{\Gamma} \times \boldsymbol{CR} \times \frac{\hat{\boldsymbol{m}}_{t+1}}{\sqrt{\hat{\boldsymbol{s}}_{t+1}+\epsilon}}$
11. $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \times \boldsymbol{\Psi}_{t+1}$
12. }
13. $\boldsymbol{w}^* = \boldsymbol{w}_{t+1}$
14. **return** $\boldsymbol{w}^*$

powerful and sophisticated solution search control methods described in Sections III-B– III-D.

In detail, our proposed algorithm consists of four control modules, i.e., FSMG, SEMG, AGCM, and AS2C, which are described in Algorithms 3 – 6, respectively. First, in line 6, the FSMG computes the first-order momentum by minimizing the outlier gradient. Second, in line 7, the SEMG calculates the second-order momentum. Third, in line 8, the gradient clipping tensor is generated by the AGCM. Fourth, the AS2C function computes the detailed strength of the solution search in all the dimensions according to the control modes determined by the two parameters, "Courier New," in line 9. Fifth, our Adam-ASC determines the following solution search direction $\boldsymbol{\Psi}_{t+1}$ by compensatively combining $\hat{\boldsymbol{m}}_{t+1}$, $\hat{\boldsymbol{s}}_{t+1}$, $\boldsymbol{\Gamma}$, and $\boldsymbol{CR}$ into one term, at line 10. Finally, the following weight tensor $\boldsymbol{w}^{(t+1)}$ is computed by the gradient descent mechanism explained in (1).

As described in Algorithm 7, Adam-ASC is modularized into four core functions. It shows that we can quickly introduce additional functions to Adam-ASC or exclude some unnecessary functions depending on the applications or tasks. Therefore, our Adam-ASC can be conveniently used as a base optimizer to train CNNs in various applications.

F. THEORETICAL ANALYSIS

In order to prove the convergence of Adam-ASC, it is necessary to derive its upper regret bound $R(T)$. For this, we define several notations as follows [27], [37], [41], [43]. First, let $w_1, \ldots, w_T$ be the weight sequences found by Adam-ASC, $\hat{v}_1, \ldots, \hat{v}_T$ be the bias-corrected second-order momentums. Second, for the learning rate and coefficient values of the momentums, let $\alpha_t = \alpha/t$, $\beta_{1,t} = \bar{\beta_w}$, $\beta_{min} = \min\{\beta_{1,1}, \ldots, \beta_{1,T}\}$, and $\gamma = \bar{\beta_w}/\beta_2^{1/2}$. Third, for convenient proofs, we use a notation D_∞ as a bound diameter of f. Finally, we assume that a bounded gradient of f_t satisfies $\|g_{t,w}\|_2 \leq G$, and $\|g_{t,w}\|_\infty \leq G_\infty$ for $w_1, \ldots, w_T$. Then, we can derive the regret upper bound of Adam-ASC as follows.

Theorem 1: The upper regret bound of Adam-ASC is given by

$$\begin{aligned} R_T \leq & \frac{D_\infty^2}{\alpha_T\left(1-\bar{\beta_w}\right)} \sum_{i=1}^{n} \sqrt{\hat{v}_{T,i}} \\ & + \frac{D_\infty^2}{\left(1-\bar{\beta_w}\right)^2} \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{\beta_{1,t}\sqrt{\hat{v}_{t,i}}}{\alpha_t \eta_{t,i}} \\ & + \frac{\alpha\sqrt{1+\log T}}{\left(1-\bar{\beta_w}\right)^2 |\beta_{min}|\left(1-\gamma\right)\sqrt{\left(1-\beta_2\right)}} \sum_{i=1}^{n} \left|\left|g_{1:T,i_2}\right|\right|_2 \end{aligned} \tag{15}$$

where $\eta_t = \boldsymbol{\Gamma} \times \boldsymbol{CR}$ which is shown in Algorithm 7.

Proof. The detailed proofs of Theorem 1 are provided in our supplementary material, available on the web. □

From Theorem 1, we can prove that the weights $w_1, \ldots, w_T$, found by Adam-ASC over all training steps, converge to an optimal weight w^* as the training phase progresses by verifying $R(T)/T = 0$. Theorem 2 shows its detailed results as follows.

Theorem 2: The sequence of weights found by Adam-ASC, i.e., $w_1, \ldots, w_T$, converge to an optimal weight w^* as $T \to \infty$.

Proof. To prove that $w_1, \ldots, w_T$ converge to w^*, we assume that $w_1, \ldots, w_T$ satisfy $\|w_n - w_m\|_2 \leq D$ and $\|w_n - w_m\|_\infty \leq D_\infty$, $\forall n, m \in \{1, \ldots, T\}$. Then, Adam-ASC satisfies

$$\forall T > 1, R(T)/T = \mathrm{O}\left(1/\sqrt{T}\right). \tag{16}$$

From (16), we can prove that $w_1, \ldots, w_T$ converge to zero when $T \to \infty$ by taking its limitation as follows.

$$\lim_{T\to\infty} R(T)/T = 0. \tag{17}$$

Thus, we can prove that $w_1, \ldots, w_T$ converge to the optimal weight w^* as the training steps progresses. □

IV. EXPERIMENTS

To verify the performance and effectiveness of our proposed methods, we evaluated the practical CNNs trained by each of the ten first-order optimization algorithms involving Adam-ASC in the image classification and segmentation tasks. In this section, we explain the detailed configurations, experimental measures, and benchmark datasets used in our experiments. Then, we analyze all the experimental results in both image classification and segmentation tasks.

A. EXPERIMENTAL CONFIGURATIONS

First, to evaluate the optimization performance of Adam-ASC on practical CNNs, we adopted two typical application tasks in which CNNs have been most widely used, namely, the

(A) CIFAR-100 dataset (B) CAMVID dataset

FIGURE 8. Example sample images in the CIFAR-100 and CamVid benchmark datasets.

image classification and segmentation tasks. In addition, we adopted the Canadian Institute for Advanced Research (CIFAR)-100 [44] and the Cambridge-driving Labeled Video Database (CamVid) [45], [46] datasets as the benchmark datasets for the tasks, respectively. The CIFAR-100 dataset is one of the most famous benchmark datasets, which is widely used to evaluate the performance of CNNs in computer vision applications, particularly the image classification task. In detail, the CIFAR-100 dataset contains 60000 sample images in 100 classes, each containing 600 images. The CamVid dataset is one of the benchmark datasets for the image segmentation task and involves 700 images and 32 segment labels. Fig. 8 illustrates several example sample images in the CIFAR-100 and CamVid datasets.

Second, we adopted nine popular first-order optimization algorithms that have been widely used to train CNNs, namely, SGD [19], RMSProp [20], Adam [21], Yogi [47], Fromage [48], AdaBelief [26], diffGrad [27], AngularGrad [49], and SAdam [28], as the comparison models. In addition, we used ResNet [17] and DenseNet [18] in the image classification task and FC-DenseNet [30] in the image segmentation task as the basic CNN models to evaluate the optimization ability of Adam-ASC.

Third, we evaluated three versions of Adam-ASC, which were determined by adjusting the mode parameters "isCG" and "isWS". In the experiments, we refer to each of them as *Adam-ASC1* (isCG=0 AND isWS=1), *Adam-ASC2* (isCG=1 AND isWS=0), and *Adam-ASC3* (isCG=1 AND isWS=1), respectively. In addition, all the parameters of the optimizers evaluated in our experiments were set according to the guidelines suggested in their original papers. Table 1 lists the detailed parameter settings of our Adam-ASC and other compared optimizers.

Finally, based on the above configurations, we trained the base CNN models using Adam-ASC and other comparative optimizers. In the training phase, we used the cross-entropy function [35] to evaluate the degree of error between the ground truth (GT) values and the output values predicted by the trained CNN models. In addition, we evaluated the training/testing accuracies of the trained CNN models at each epoch. For this, the following accuracy measures were used as follows:

$$Accuracy_{train} = \frac{1}{|D_{train}|} \sum_{<x,y>\in D_{train}} \delta\left(\hat{y}, y\right) \tag{18}$$

TABLE 1. Parameter settings of the optimizers evaluated in our experiments.

Optimizers	Parameter settings
Adam-ASC	$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \beta_3 = 0.1, \varphi = 10^{-8}, \epsilon = 10^{-8}$
SGD [13]	$\alpha = 0.001$
RMSProp [14]	$\alpha = 0.01, \beta_1 = 0.99, \epsilon = 10^{-8}$
Adam [15]	$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
Yogi [41]	$\alpha = 0.01, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 0.001$
Fromage [42]	$\alpha = 0.01$
diffGrad [21]	$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
AdaBelief [20]	$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
AngularGrad [43]	$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
SAdam [22]	$\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

and

$$Accuracy_{test} = \frac{1}{|D_{test}|} \sum_{<x,y>\in D_{test}} \delta\left(\hat{y}, y\right) \tag{19}$$

where D_{train} and D_{test} are the training and test benchmark datasets, and $< x, y >$ is sample data with an input image x and its GT value y. In addition, $\delta\left(\hat{y}, y\right)$ is a Kronecker delta function that returns one if $\hat{y}$ is equal to y and zero otherwise. Using these performance metrics, we analyzed the convergence curves of the training and test accuracies of the CNNs trained by each optimizer across all epochs. The detailed experimental results are presented in the following sections.

B. PARAMETER SENSITIVITY ANALYSIS

1) PARAMETER SENSITIVITY ANALYSIS FOR β_3 AND φ

Before comparing the optimization performance between our proposed method and traditional ones, we first analyzed how a variation of the control parameters sensitively affects our method. As shown in Algorithm 7, Adam-ASC has a total of six control parameters, α, β_1, β_2, β_3, ϵ, and φ. Among them, α, β_1, β_2, and ϵ are standard control parameters used in most Adam-based optimizers. However, the parameters β_3 and φ are used only in Adam-ASC. In particular, they play a critical role in controlling the adaptive clipping momentum, which is essential in our proposed method. Accordingly, we first conducted the parameter sensitivity test for two control parameters, β_3 and φ, to evaluate how the train performance of our approach is affected when their values are varied.

Fig. 9 shows all the parameter sensitivity test results of our Adam-ASC according to variations of the settings of β_3 and φ. In Fig. 9, (a)-(c) present the results for Adam-ASC1, 2, and 3, respectively, and (d) describes their average test accuracies under each parameter setting. As shown in Fig. 9-(a), Adam-ASC1 showed the best test accuracies under

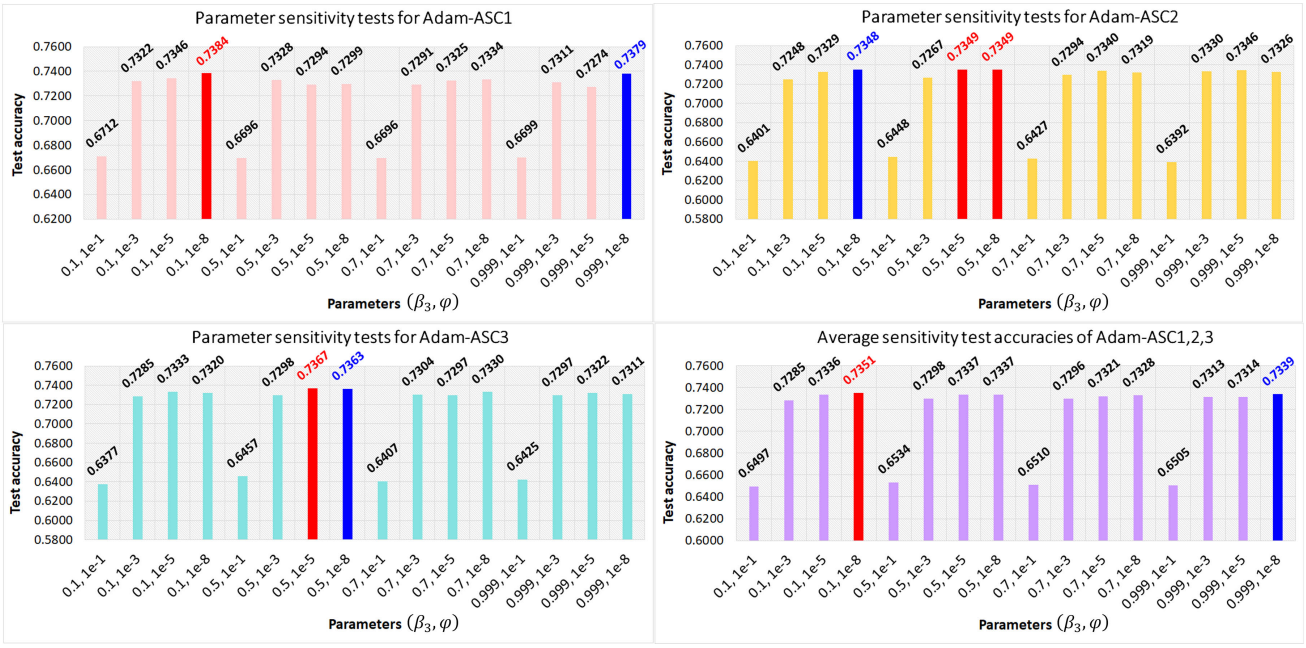

FIGURE 9. The parameter sensitivity test results of Adam-ASC for two control parameters β_3 and φ.

both $\beta_3 = 0.1$ and $\varphi = 1e-8$, i.e., 0.7384. Similarly, Adam-ASC1 achieved the second-best result, i.e., 0.7379, when both $\beta_3 = 0.999$ and $\varphi = 1e-8$. Meanwhile, as described in Fig. 9-(b), Adam-ASC2 presented the best results, i.e., 0.7349 test accuracy, when two parameters were set as ($\beta_3 = 0.5, \varphi = 1e-5$) or ($\beta_3 = 0.5, \varphi = 1e-8$).

Moreover, Adam-ASC2 with the parameter configuration ($\beta_3 = 0.1, \varphi = 1e-8$) also showed the most equivalent test accuracy to the best ones, i.e., 0.7348. Adam-ASC3 achieved the first and second-best test accuracies when its parameters were set to ($\beta_3 = 0.5, \varphi = 1e-5$) and ($\beta_3 = 0.5, \varphi = 1e-8$), respectively.

Meanwhile, we found an interesting fact from Fig. 9 that the test accuracies of Adam-ASC are worst when the parameter φ is set to 1e–1, regardless of the setting of β_3. The results imply that setting φ to a large value close to one could worsen the optimization performance of our Adam-ASC. As explained in Section III-C, φ is a Laplacian smoothing factor in the denominator to prevent zero denominators in (12). In this case, if φ is set to any significant value, its value can strongly affect the clipping rate. That is, the clipping rate can be lowered unnecessarily by a value of φ in the denominator. Such a low clipping rate can slow the solution search and convergence, resulting in low optimization performance. Accordingly, when φ is set to 0.1, the test accuracies were the worst, regardless of other parameter settings or the versions of Adam-ASC.

On the other hand, we also found that β_3 has little influence on the overall optimization performance of Adam-ASC regardless of its version. As described in (9), β_3 is used as a coefficient value to generate the clipping momentum. As β_3 is set to a larger value, the degree of accumulation of the current gradient $|\boldsymbol{g}_t|$ becomes smaller. Nevertheless, no significant differences between the results were found when the parameter was set to four values, i.e., 0.1, 0.5, 0.7, and 0.999. From the results, we can conclude that φ is more sensitive to its setting value than β_3, especially when its value is close to 1, such as $\varphi = 1e-1$. This indicates that it is reasonable to set the smoothing factor φ as small as possible. In addition, we can also see that the coefficient parameter in the gradient clipping momentum does not significantly affect the overall optimization performance of Adam-ASC.

Meanwhile, Fig. 9-(a) – (c) present the most appropriate parameter settings of Adam-ASC. Accordingly, we calculated the average test accuracies of three versions of Adam-ASC at each parameter set to find the most suitable for all versions of Adam-ASC as shown in Fig. 9-(d). As a result, we found that both $\beta_3 = 0.1$ and $\varphi = 1e-8$ are the most suitable for Adam-ASC because they achieved the best or good optimization results under the parameter settings without significant performance loss. Therefore, we adopted both $\beta_3 = 0.1$ and $\varphi = 1e-8$ as the primary parameter settings.

2) PARAMETER SENSITIVITY ANALYSIS FOR α, β_1 AND β_2

The learning rate α controls the strength of the solution search in the typical first-order optimizers with Adam-ASC. Accordingly, we analyzed the convergence performance of the test accuracies according to the variation of α. For this purpose, we set the other parameters except for α to be the same as those described in Section IV-A.

Fig. 10 shows the test accuracy curves of Adam-ASC1 – Adam-ASC3 when α is set to 0.1, 0.01,

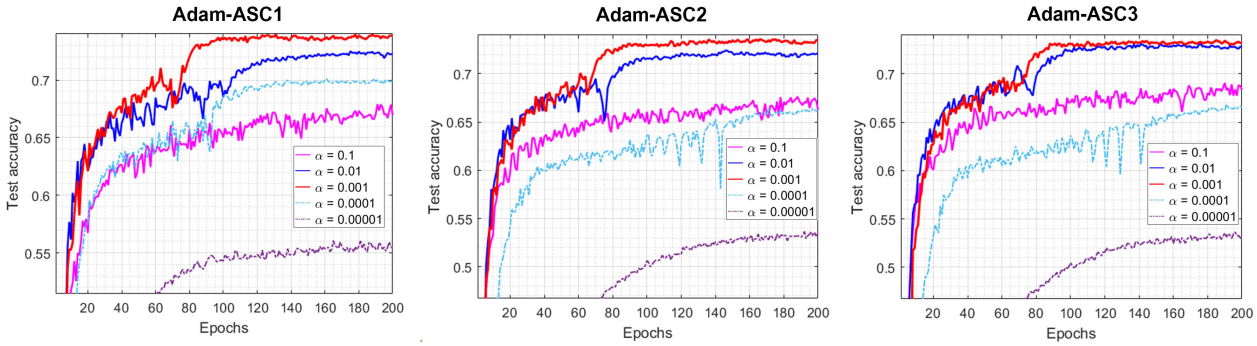

FIGURE 10. The test accuracy curves of Adam-ASC according to the values of the learning rate α.

TABLE 2. Parameter sensitivity test results of Adam-ASC for two control parameters β_1 and β_2.

Parameters		Test accuracies		
β_1	β_2	Adam-ASC1	Adam-ASC2	Adam-ASC3
0.5	0.5	0.5667	0.7333	0.5021
0.5	0.7	0.7311	0.7345	0.7304
0.5	0.999	0.7310	0.7342	0.7331
0.7	0.5	0.5681	0.7313	0.5040
0.7	0.7	0.7295	0.7300	0.7330
0.7	0.999	0.7374	0.7356	0.7310
0.9	0.5	0.5646	0.7336	0.5130
0.9	0.7	0.7315	0.7300	0.7361
0.9	0.999	0.7384	0.7348	0.7320

0.001, 0.0001, and 0.00001, respectively. When α was set to 0.001, Adam-ASC showed the best test accuracy curves regardless of its version. In detail, Adam-ASC1 with $\alpha = 0.001$ achieved the best convergence among five test accuracy curves. Similarly, Adam-ASC2 and Adam-ASC3 with $\alpha = 0.001$ also showed better convergence curves among all the experimental results. Meanwhile, when α was set to 0.00001, Adam-ASC showed the worst convergence performance: the slowest speed and the lowest test accuracy. In addition, when α was set to 0.1, Adam-ASC showed relatively lower test accuracies and convergence curves when compared to other convergence results. Thus, the convergence results shown in Fig. 10 indicate that setting α to too large or too small in Adam-ASC can significantly degrade its optimization ability.

Table 2 lists the test accuracies of Adam-ASC1 – Adam-ASC3 for the nine parameter settings of β_1 and β_2. Similar to the sensitivity tests for α, the other parameters, except for β_1 and β_2, were equally set to those shown in Table 1. We found that Adam-ASC1 and Adam-ASC3 showed relatively lower test accuracies when β_2 was set to 0.5, regardless of the values of β_1. On the other hand, Adam-ASC2 showed test accuracies similar to the results when β_2 was set to 0.7 or 0.999. As shown in (7), β_2 is a coefficient value used to update the second-order momentum. If β_2 is set to a tiny value, the next second-order momentum will contain almost no information from the historical momentum. In this case, the second-order momentum will be close to the current gradient, which may interfere with a correct solution search. On the other hand, if β_2 is set to a large value close to 1, the second-order momentum to be updated may be influenced more by the historical momentum than by the current gradient. Thus, it is recommended to set β_2 to a large value, at least greater than 0.7, to ensure the optimization ability of Adam-ASC1 and Adam-ASC3.

Meanwhile, Adam-ASC2 was relatively less sensitive than Adam-ASC1 and Adam-ASC3. As shown in Table 2, Adam-ASC2 showed almost similar test accuracies under all the parameter settings, regardless of the settings of β_1 and β_2. Unlike Adam-ASC1 and Adam-ASC3, Adam-ASC2 does not use the warm-up strategy in the AS2C function. In other words, we can understand that the warm-up strategy affects the parameter sensitivity for β_2 in Adam-ASC. These results indicate that it is necessary to sufficiently reduce the influence of the current gradient in the second-order momentum if the fast convergence is suppressed by the warm-up strategy in the early steps.

C. EXPERIMENTAL RESULTS IN IMAGE CLASSIFICATION TASKS WITH ResNet-18 AND ResNet-101

As explained in Section IV-A, we adopted ResNet as one of the base CNN models to evaluate the optimization performance of Adam-ASC for training the practical CNN models. ResNet is one of the most widely used practical CNNs. Unlike conventional CNNs, ResNet consists of many residual blocks with skip connections in each layer. The skip connection effectively prevents the reduction of the input signals as the number of layers increases. In fact, ResNet has shown better image recognition performance than traditional CNNs in various computer vision tasks. However, as more residual blocks and layers are involved in ResNet, its architecture becomes massive and complicated. Therefore, it is necessary to study more improved and stable optimization methods to handle complex CNNs such as ResNet effectively.

Analyzing the convergence curves of training accuracy helps to understand how fast the optimizer can reach the

highest training accuracy as the training phase progresses. Accordingly, we analyzed the training accuracy curves of ResNet-18 and ResNet-101 trained by Adam-ASC and other compared optimizers. These results are shown in Fig. 11 and 12. We provide detailed curve plots, as shown in Fig. 11, to compare the convergence results more clearly. The top plots in Fig. 11 describe the training accuracy curves generated by Adam-ASC and the traditional optimizers, i.e., SGD, RMSProp, and Adam. The bottom plots in Fig. 11 show the training accuracy curves for Adam-ASC and the recent optimizers, such as AdaBelief, diffGrad, and SAdam. In addition, the two plots on the right-hand side in Fig. 11 are zoomed in on the two plots on the left-hand side to show more detail of the convergence curves within specific epochs and accuracy ranges.

In detail, the left-hand side plots in Figs. 11 and 12 show the training accuracy curves of ResNet-18 and ResNet-101 trained by Adam-ASC, SGD, RMSProp, and Adam. Similarly, the right-hand side plots in Figs. 11 and 12 present the training accuracy curves generated by Adam-ASC and the recent optimizers, such as AdaBelief, diffGrad, and SAdam. The convergence plots show that ResNet-18 and ResNet-101 trained by our Adam-ASC have better convergence performance than the existing optimizers. Furthermore, we found that AdaBelief and Yogi had good convergence performance similar to that of Adam-ASC. Moreover, diffGrad and SAdam showed worse convergence than Adam-ASC. Finally, SGD had the worst training accuracy curves among all comparative optimizers.

We also analyzed the test accuracy curves of the ResNet-18 and ResNet-101 trained by the 12 optimizers with Adam-ASC. Their results are described in Figs. 13 and 14, respectively. We found that the test accuracies of Adam-ASC had the fastest convergence when compared to the other optimizers. In particular, the test accuracies of Adam-ASC increased dramatically after about 70 epochs and maintained the best performance until the last epoch. In detail, Fig. 13 shows that the test accuracies of ResNet-18 trained by Adam-ASC were stable between approximately 0.73 and 0.74 after 80 epochs. Similarly, we also found that the test accuracies of ResNet-101 trained by Adam-ASC were maintained between 0.74 and 0.76 after about 100 epochs from the results shown in Fig. 14. These results indicate that our Adam-ASC has better and more stable convergence abilities regarding the training and test accuracies than the existing optimizers.

Tables 3 and 4 describe the final test accuracies of ResNet-18/101 trained by our Adam-ASC and other comparative optimizers. The final test accuracies were evaluated at the final epoch, i.e., the 200th epoch. We found that our Adam-ASC had the highest test accuracies when compared to other optimizers. In detail, Adam-ASC1 achieved the best test accuracy, i.e., 0.7384 and 0.754 in ResNet-18 and ResNet-101, respectively. Similarly, Adam-ASC2 and Adam-ASC3 also recorded 0.7348/0.7442 and 0.732/0.7433 test accuracies in ResNet-18/101, the second and third-best results. Meanwhile, AdaBelief showed the highest test accuracies among the existing optimizers, i.e., 0.7114 and 0.7313, in ResNet-18 and ResNet-101, respectively. On the other hand, the conventional optimizers such as SGD, RMSProp, and Adam showed relatively lower test accuracies than those of our Adam-ASC and the latest methods such as AdaBelief, diffGrad, and SAdam. For example, Adam achieved test accuracies of 0.6808 and 0.7085 test accuracies in ResNet-18 and ResNet-101, respectively. Such experimental results indicate that our Adam-ASC can effectively train both ResNet-18 and ResNet-101 with significantly improved optimization performance compared to the existing optimizers.

As explained in Section III, Adam-ASC controls the solution search by compositely utilizing gradient clipping, the warm-up strategy, and advanced momentum computation methods. Accordingly, Adam-ASC showed a convergence speed similar to that of the existing optimizers, even though Adam-ASC achieves better test accuracies than them. One of the reasons for this phenomenon is the sophisticated solution search mechanisms of our Adam-ASC. Suppose the solution

TABLE 3. Test accuracy results and their ranks evaluated in ResNet-18.

Methods	Test accuracies	Ranks
Adam-ASC1	**0.7384**	**1**
Adam-ASC2	**0.7348**	**2**
Adam-ASC3	**0.7320**	**3**
SGD	0.5562	12
RMSProp	0.6128	10
Adam	0.6808	9
Yogi	0.6848	7
Fromage	0.6084	11
diffGrad	0.6886	6
AdaBelief	0.7114	4
AngularGrad	0.6843	8
SAdam	0.6968	5

TABLE 4. Test accuracy results and their ranks evaluated in ResNet-101.

Methods	Test accuracies	Ranks
Adam-ASC1	**0.7540**	**1**
Adam-ASC2	**0.7442**	**2**
Adam-ASC3	**0.7433**	**3**
SGD	0.4436	12
RMSProp	0.6265	11
Adam	0.7085	8
Yogi	0.6916	9
Fromage	0.6724	10
diffGrad	0.7246	5
AdaBelief	0.7313	4
AngularGrad	0.7194	7
SAdam	0.7227	6

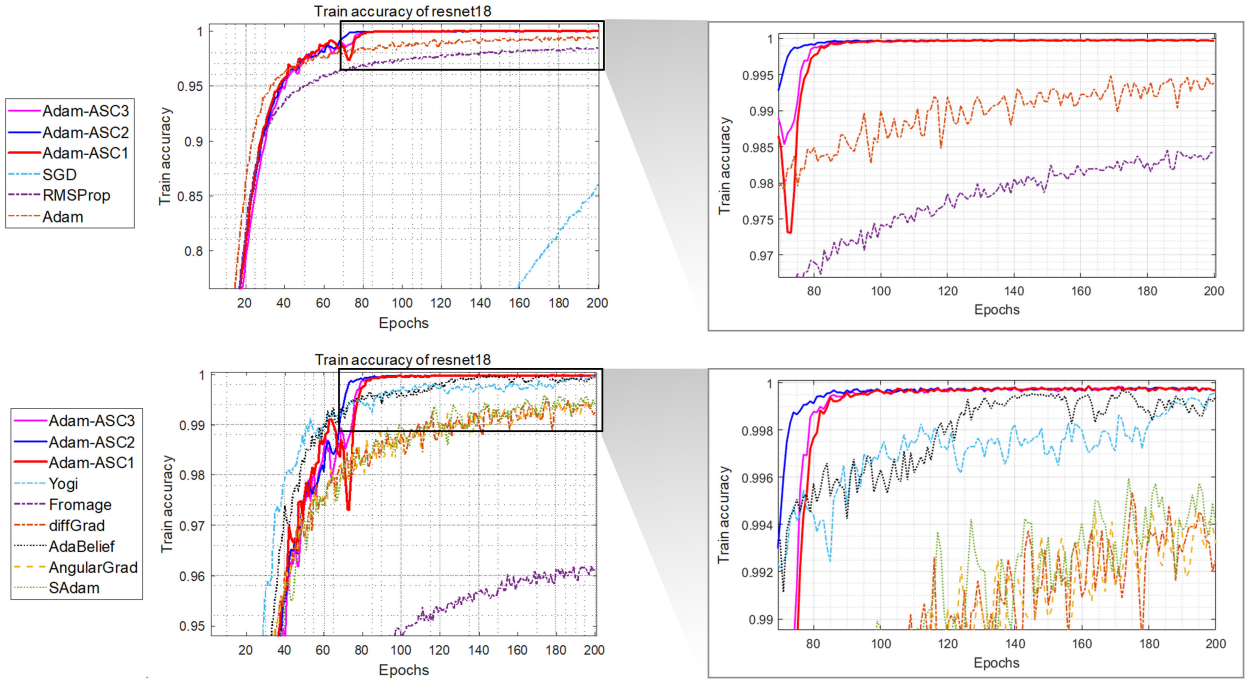

FIGURE 11. Training accuracy curves of ResNet-18 trained by Adam-ASC and the existing optimizers.

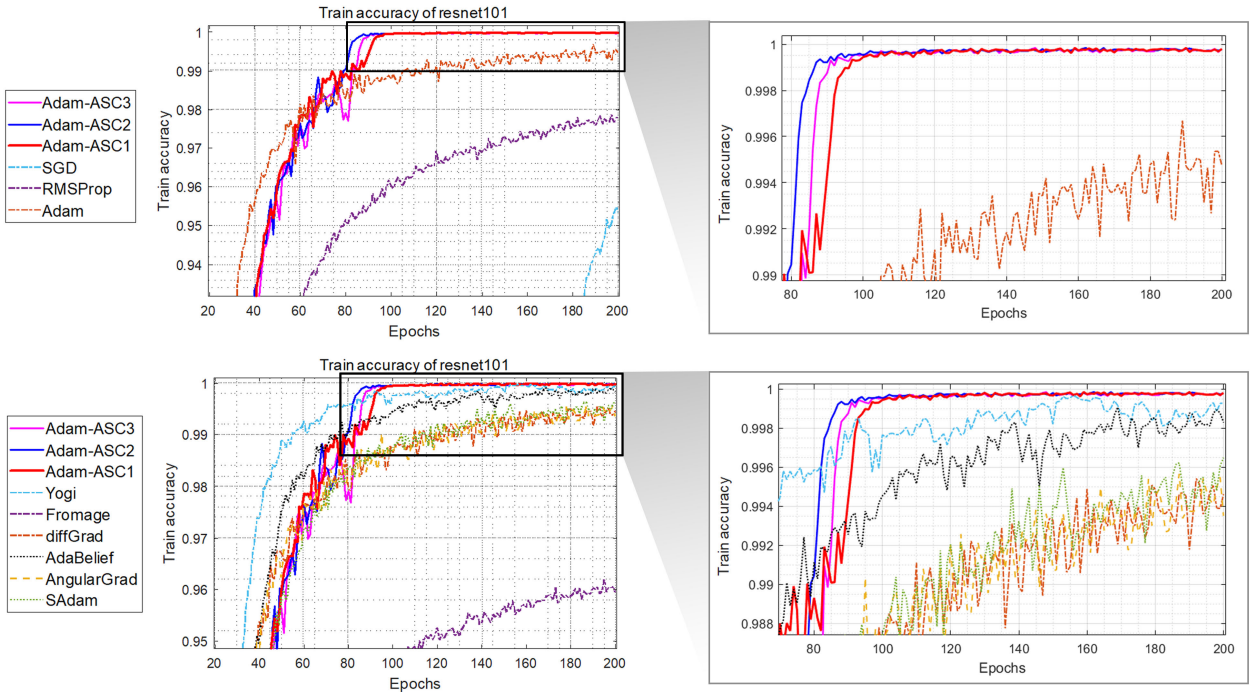

FIGURE 12. Training accuracy curves of ResNet-101 trained by Adam-ASC and the existing optimizers.

search speed becomes drastically fast. In this case, its search trajectory may oscillate excessively, as shown in Fig. 3, or it may converge prematurely to a local minimum [50]. This phenomenon often degrades the performance of trained CNN models by failing to find their approximate optimal weights in the solution space. Therefore, it is necessary to elaborately control its solution strength to prevent excessive oscillation of the search trajectory or premature convergence.

Such solution search mechanisms have the remarkable advantage that we can find better optimal weight solutions,

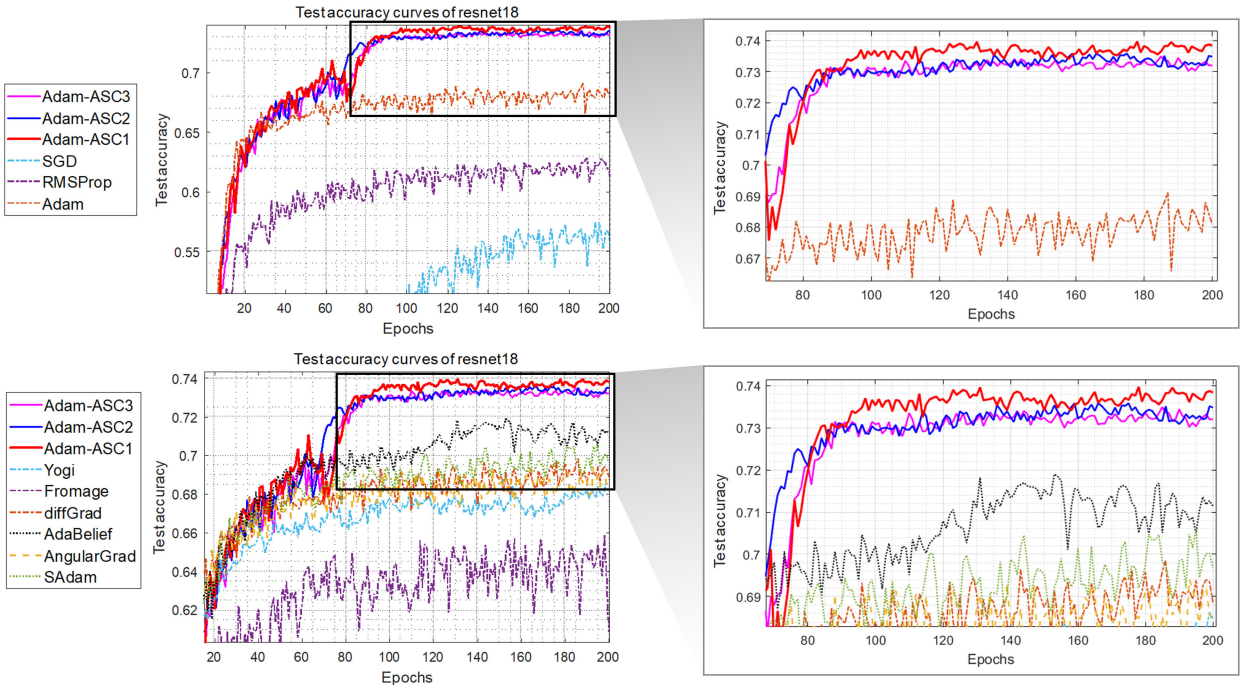

FIGURE 13. Test accuracy curves of ResNet-18 trained by Adam-ASC and the existing optimizers.

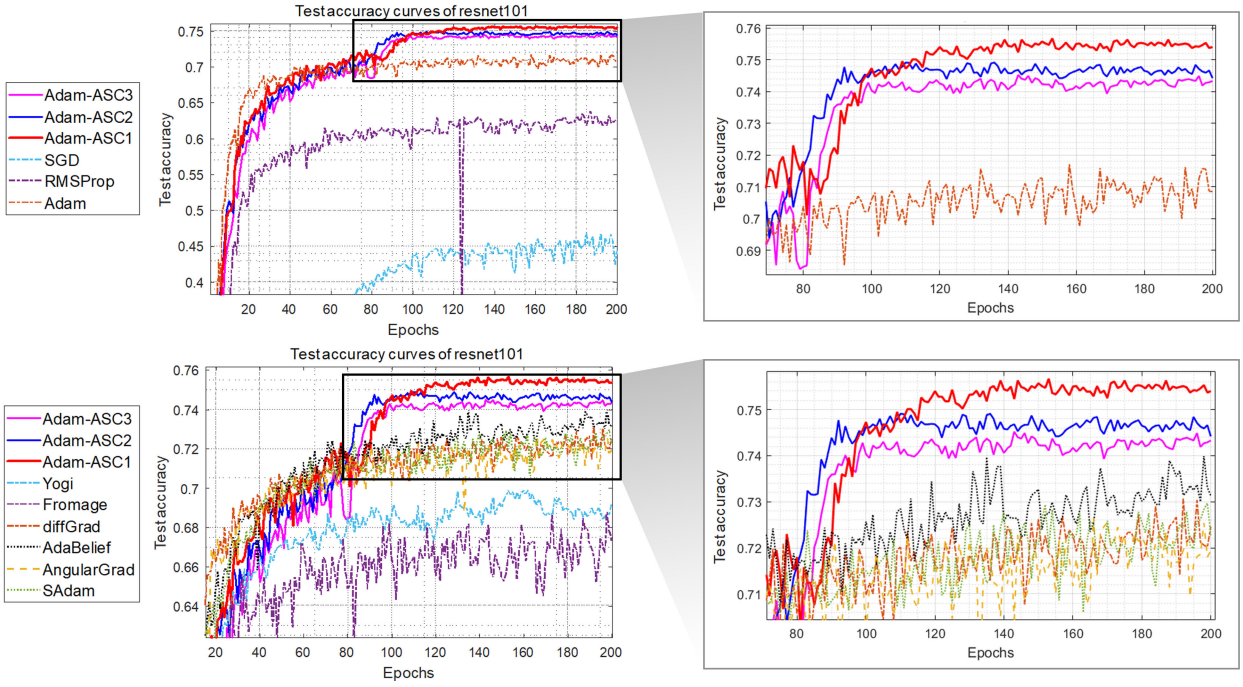

FIGURE 14. Test accuracy curves of ResNet-101 trained by Adam-ASC and the existing optimizers.

i.e., more accurate weight values of the trained CNN model, even though their convergence speed is relatively similar to existing optimizers. Accordingly, our Adam-ASC also uses exquisite solution search methods rather than an aggressive solution search mechanism. In fact, the detailed solution search and control mechanisms of Adam-ASC adaptively limit the strength of the solution search around any local minima to further improve the quality of the optimal weight to be found, although its convergence speed was not significantly improved. As a result, Adam-ASC achieved better

optimization capability in training the CNNs without sacrificing its convergence speed compared to the existing optimizers such as Adam, AdaBelief, and diffGrad.

D. EXPERIMENTAL RESULTS IN IMAGE CLASSIFICATION TASKS WITH DenseNet-121 AND DenseNet-169

As another base CNN model to evaluate the optimization ability of Adam-ASC in image classification tasks, we adopted both DenseNet-121 and DenseNet-169. DenseNet is one of the improved models of ResNet [18]. In detail, unlike the residual connections of ResNet, DenseNet consists of more complicated residual blocks, called dense block, which connects all the dense blocks. Accordingly, DenseNet becomes more complicated, which indicates that a more sophisticated optimization method is required to find its optimal weights in the complex solution space. In this experiment, we evaluated how our proposed optimization method can significantly contribute to improving the image classification ability of DenseNet.

The training accuracy curves of DenseNet-121 and DenseNet-169 are described in Figs. 15 and 16, respectively. Similar to the convergence curves of ResNet, DenseNet trained by Adam-ASC also showed the best training accuracy and convergence among the comparative optimizers. These results present that Adam-ASC has a robust training ability even though the size of DenseNet is larger than that of ResNet. On the other hand, the convergence curves of the other optimizers were also similar to the results shown in both Figs. 11 and 12.

Meanwhile, Figs. 17 and 18 illustrate the test accuracy curves of DenseNet-121 and DenseNet-169 trained by Adam-ASC and other optimizers, respectively. We found that our Adam-ASC had better test accuracies than other existing optimizers, similar to the results of ResNet-18/101. In particular, Adam-ASC still showed stable and robust test accuracy and convergence ability, even though DenseNet had more layers and complicated structures than ResNet. For example, in DenseNet-121, Adam-ASC1 approached a test accuracy of 0.74 at about the 60th epoch. After that, Adam-ASC1 maintained a stable accuracy of about 0.76 until the last epoch. Adam-ASC2 and Adam-ASC3 also showed good convergence of the test accuracy curves, although they are inferior to those of Adam-ASC1. In addition, Adam-ASC maintained test accuracies between 0.74 and 0.76 from approximately the 80th to the final epoch. On the other hand, the convergence of the test accuracies of the other optimizers was worse than that measured in ResNet-18/101. These results indicate that the solution search strategies of Adam-ASC are significantly stable and robust to the size and complexity of the CNN model.

Tables 5 and 6 describe the final test accuracies of DenseNet-121/169 trained by Adam-ASC and the other optimizers. Both tables present that the DenseNet models trained by Adam-ASC have the best test accuracies. In detail, from the results shown in Table 5, we can see that the DenseNet-121 models trained by Adam-ASC1 – Adam-ASC3 have test accuracies of 0.7649, 0.7614, and 0.7518, respectively. Likewise, Table 6 provides that the DenseNet-169 models trained by Adam-ASC1 – Adam-ASC3 achieved test accuracies of 0.7641, 0.7571, and 0.7623, which are better results when compared to the results of other optimizers. Meanwhile, AdaBelief and diffGrad showed good test accuracies among the existing optimizers, i.e., 0.7464/0.7486 and 0.7087/0.7111 in DenseNet-121/169, respectively. In addition, the test accuracies of SAdam and AngularGrad were also better than those of the conventional methods, such as SGD and RMSProp.

TABLE 5. Test accuracy results and their ranks evaluated in DenseNet-121.

Methods	Test accuracies	Ranks
Adam-ASC1	**0.7649**	**1**
Adam-ASC2	**0.7614**	**2**
Adam-ASC3	**0.7518**	**3**
SGD	0.4799	12
RMSProp	0.625	11
Adam	0.7067	8
Yogi	0.6964	9
Fromage	0.6816	10
diffGrad	0.7087	7
AdaBelief	0.7464	4
AngularGrad	0.7139	6
SAdam	0.7254	5

TABLE 6. Test accuracy results and their ranks evaluated in DenseNet-169.

Methods	Test accuracies	Ranks
Adam-ASC1	**0.7641**	**1**
Adam-ASC2	**0.7571**	**3**
Adam-ASC3	**0.7623**	**2**
SGD	0.5003	12
RMSProp	0.667	11
Adam	0.713	7
Yogi	0.6953	9
Fromage	0.6838	10
diffGrad	0.7111	8
AdaBelief	0.7486	4
AngularGrad	0.7184	6
SAdam	0.7272	5

Finally, we found that Adam-ASC1 achieved the best test accuracies in both DenseNet-121 and DenseNet-169, which were the same results as the experiments conducted on ResNet-18 and ResNet-101. In detail, Tables 3 – 6 show that Adam-ASC1 achieved the best rank in both four CNN models. Adam-ASC2 achieved the second-best rank in the three CNN models, except for DenseNet-169. Adam-ASC3 showed the second-best rank in DenseNet-169 and the third-best rank in the other models. These results indicate that the combination of (13), except (14), with FSMG, SEMG, and AGCM can significantly contribute to the search for an optimal weight in the large-dimensional solution

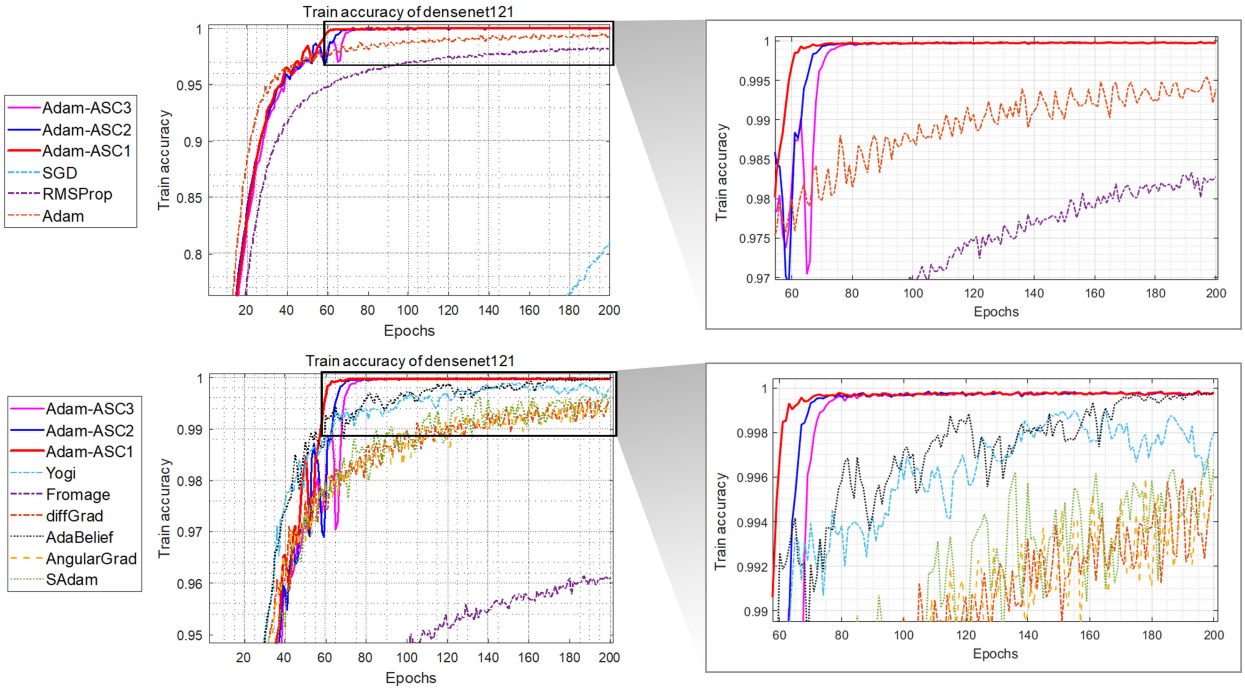

FIGURE 15. Training accuracy curves of DenseNet-121 trained by Adam-ASC and the existing optimizers.

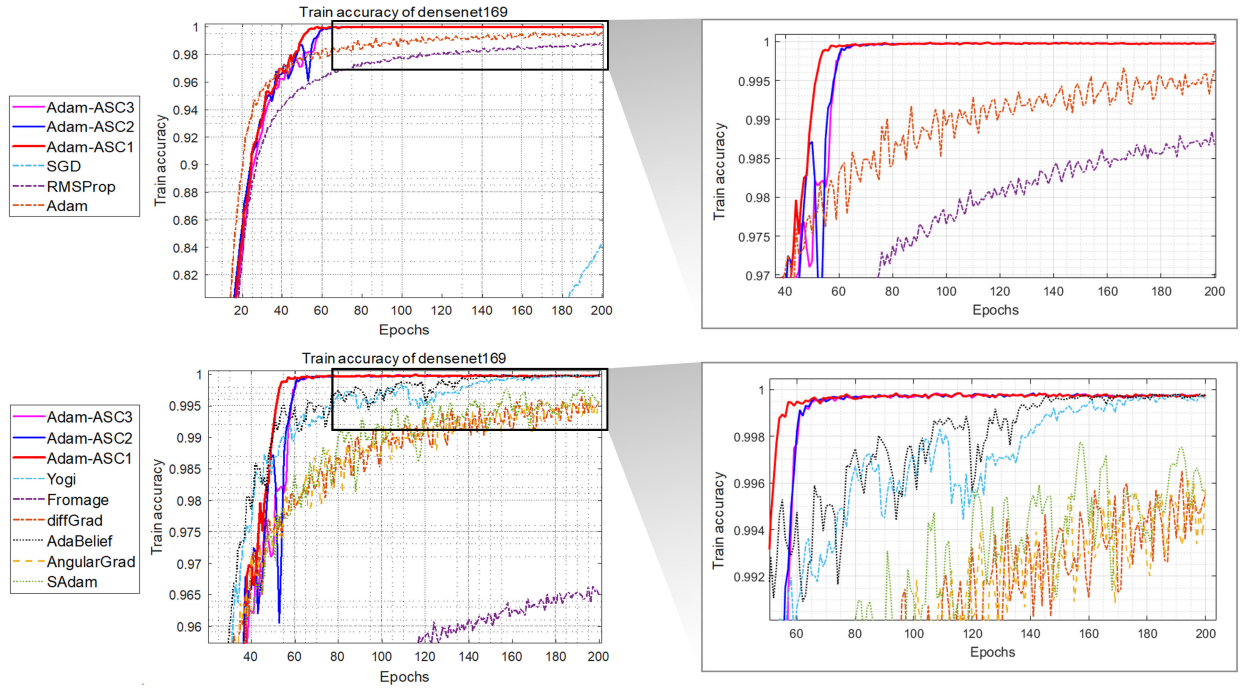

FIGURE 16. Training accuracy curves of DenseNet-169 trained by Adam-ASC and the existing optimizers.

space of DenseNet. On the other hand, using both (13) and (14) showed worse test accuracies than Adam-ASC1. Thus, Adam-ASC can achieve the best optimization performance in the image classification task by using only (14) without (13) in the AS2C function.

From the experimental results, we can conclude that our Adam-ASC has a more stable and effective optimization ability for CNNs with complex structures, such as ResNet and DenseNet, than the existing optimizers in image classification tasks.

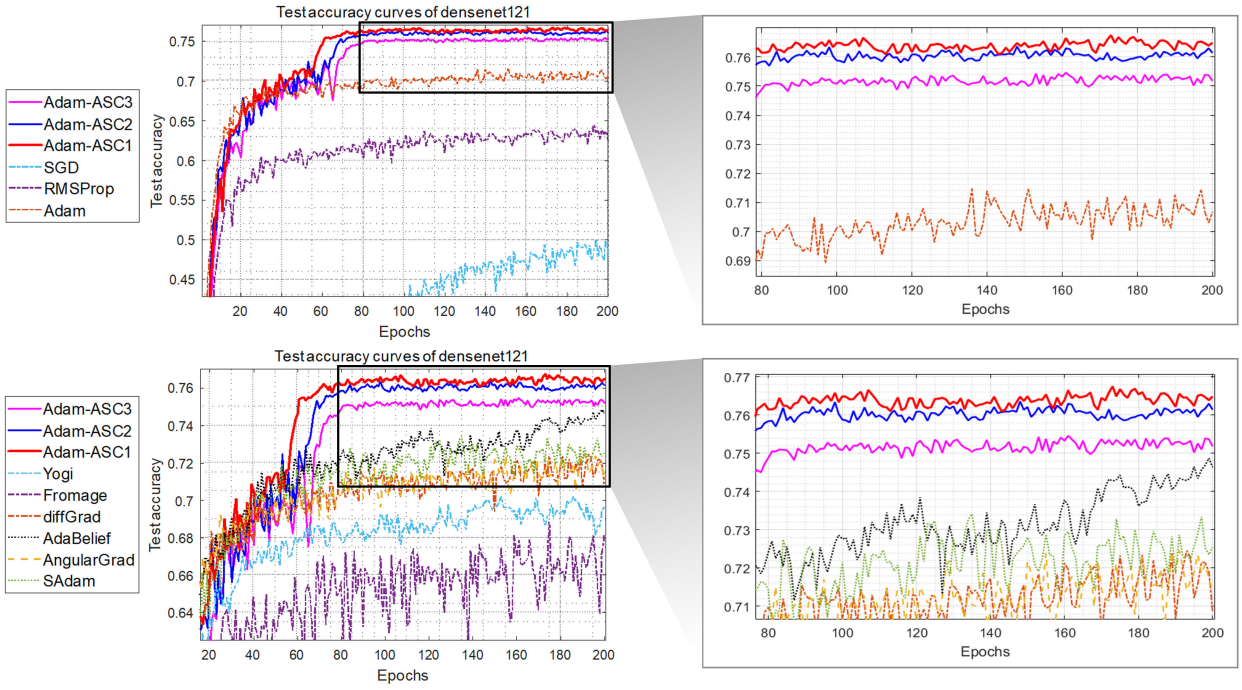

FIGURE 17. Test accuracy curves of DenseNet-121 trained by Adam-ASC and the existing optimizers.

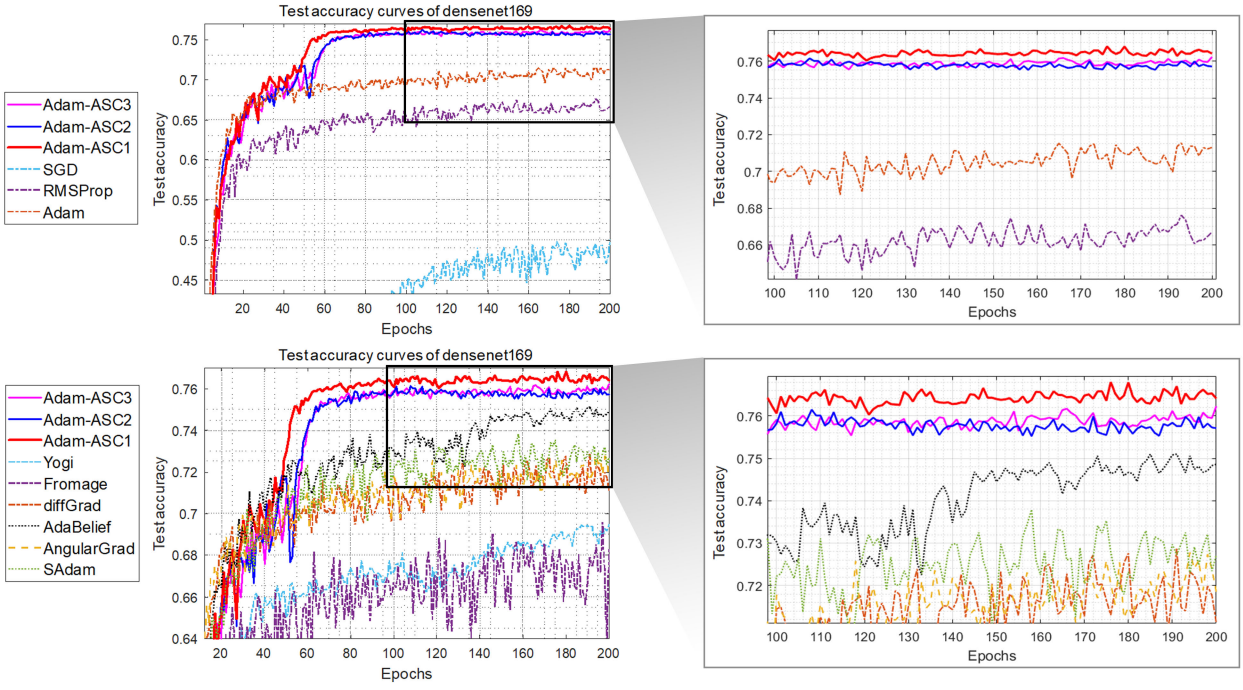

FIGURE 18. Test accuracy curves of DenseNet-169 trained by Adam-ASC and the existing optimizers.

E. VALIDATION LOSS ANALYSIS OF ADAM-ASC IN ResNet AND DenseNet

Fig. 19 shows the validation loss curves of ResNet-18 and 101 trained by Adam-ASC1 – Adam-ASC3, respectively. We found that the validation loss curves of ResNet-18 and ResNet-101 drastically decreased in the early epochs and then slightly increased from 1.5 to 2.0 loss values between about 20 – 60 epochs. These growth patterns were stably decelerated within 2.0 loss values. The validation loss curves of DenseNet-121/169 are presented in Fig. 20, respectively. The validation loss curves of DenseNet-121 and DenseNet-169 also showed similar patterns to those shown in Fig. 19.

Meanwhile, as explained in Sections IV-C and IV-D, the test accuracies observed in about 20 – 60 epochs were worse than those observed after 60 – 70 epochs in both ResNet and

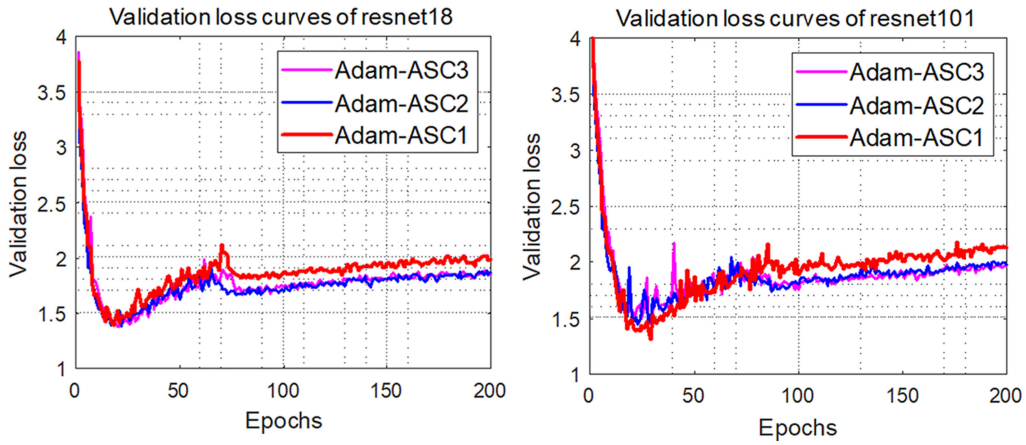

FIGURE 19. Validation curves of ResNet-18 and Resnet-101 trained by Adam-ASC1, Adam-ASC2, and Adam-ASC3.

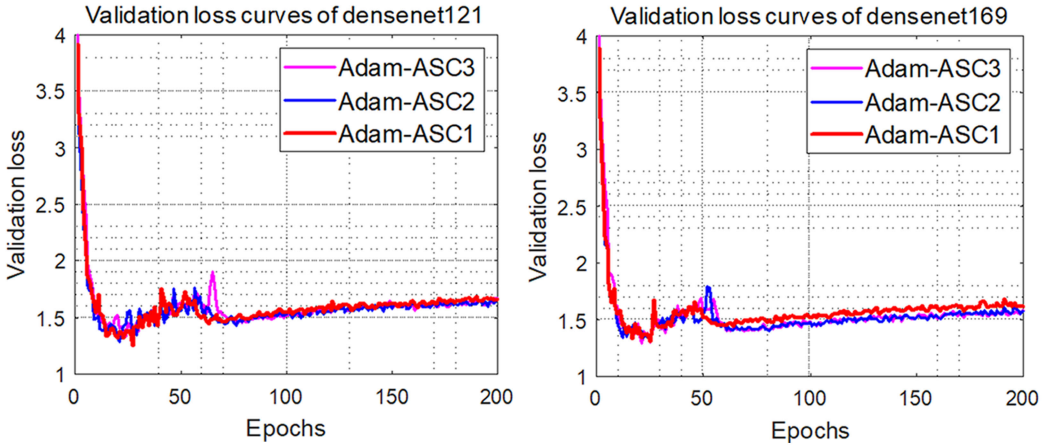

FIGURE 20. Validation loss curves of DenseNet-121 and DenseNet-169 trained by Adam-ASC1, Adam-ASC2, and Adam-ASC3.

DenseNet. A difference between the measures of the validation loss and test accuracy causes such a phenomenon. In our experiments, the cross-entropy used to compute the validation loss measures the degree of difference between the ground truth and the predicted outputs. On the other hand, the test accuracy of (19) measures a ratio of correctly classified samples among all samples. As a result, although the validation loss is not minimal, the test accuracy can be further improved. Thus, the CNN models trained by Adam-ASC can achieve and maintain stable prediction performance after training for several epochs.

F. EXPERIMENTAL RESULTS IN IMAGE SEGMENTATION TASKS WITH FC-DenseNet

In the previous sections, we evaluate the optimization performance of Adam-ASC in the image classification task. In addition, we conducted experiments to evaluate the optimization performance of our Adam-ASC in the image segmentation task. Unlike the image classification task, the image segmentation task requires more complex and massive CNN models because many computations are needed to segment each image elaborately. Accordingly, as explained in Section IV-A, we adopted the FC-DenseNet [30] as the basic CNN model for image segmentation. The FC-DenseNet is an enhanced DenseNet-based deep neural network for image segmentation tasks. The FC-DenseNet has a more complex and massive structure when compared to the ResNet and DenseNet used in the image classification task.

Accordingly, we evaluated the optimization performance of Adam-ASC and other compared optimizers in training the FC-DenseNet. We then analyzed their results in terms of train & test accuracy and convergence performance. These results are shown in Figs. 21 and 22. The left-hand side plot in Fig. 21 shows the training accuracy curves of the FC-DenseNet models trained by Adam-ASC and other optimizers. We found that Adam-ASC had fast and stable training convergence when compared to the classical optimizers. In particular, Adam-ASC1 achieved the best convergence with significant accuracies across all epochs.

Meanwhile, the right-hand side plot in Fig. 21 describes the validation loss curves of FC-DenseNet trained by Adam-ASC and other compared optimizers. Considering the training accuracy and validation loss curves, we found that the FC-DenseNet models trained by Adam-ASC can converge stably without any significant anomaly. Thus, our Adam-ASC

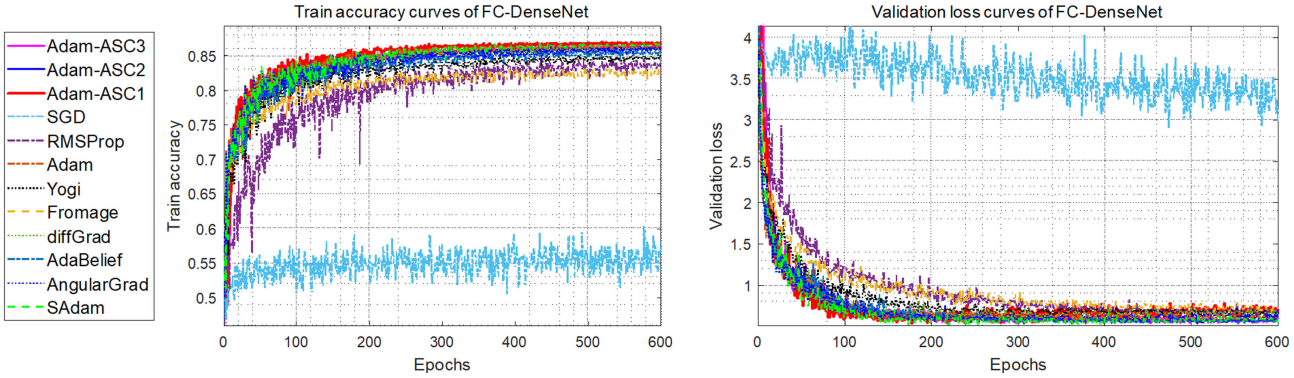

FIGURE 21. Training accuracy and validation loss curve plots of FC-DenseNet trained by Adam-ASC and the existing optimizers.

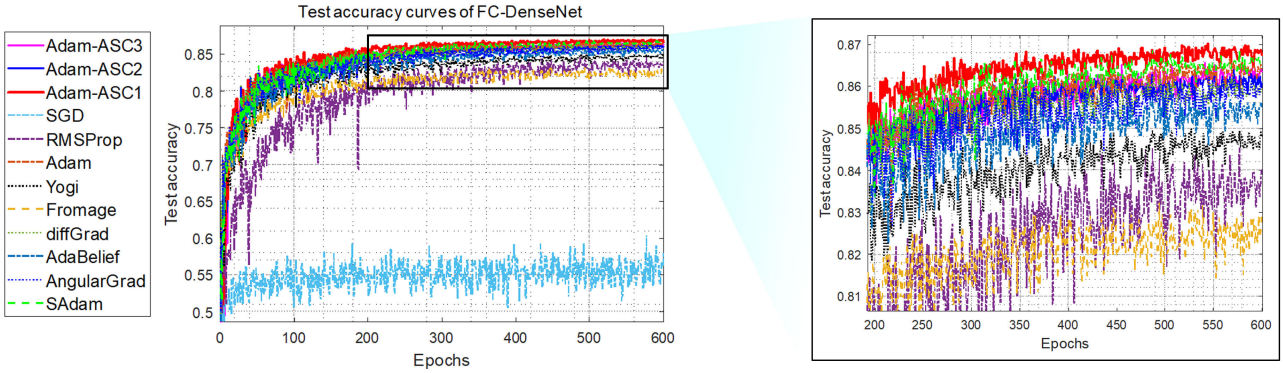

FIGURE 22. Test accuracy curves of DenseNet-121 trained by Adam-ASC and the existing optimizers.

can be effectively used as a basic optimizer for training CNNs in image classification and segmentation tasks.

We also analyzed the convergence curves of the test accuracies of the FC-DenseNet models trained by Adam-ASC and other existing optimizers. The results are shown in Fig. 22. The right-hand side plot in Fig. 22 is the zoomed plot of the left-hand side plot within the test accuracies between 0.8 and 0.87. The FC-DenseNet trained by Adam-ASC1 showed the best test accuracy curve when compared to the results of the other optimizers. In detail, Adam-ASC1 achieved a test accuracy of about 0.86 after about 280 epochs. Then, the test accuracy curve steadily maintained test accuracies of 0.86 – 0.87 until the last epoch. Meanwhile, the test accuracy curves of other existing optimizers showed worse convergence patterns than the curves of our Adam-ASC. However, among the existing optimizers, SAdam and Adam showed convergence curves almost similar to those of Adam-ASC1. Also, their test accuracy curves were slightly better than those of Adam-ASC2 and Adam-ASC3, even though they achieved worse results than Adam-ASC2 and Adam-ASC3. We also found that AdaBelief and diffGrad achieved worse convergence performance than the curves of Adam-ASC and Adam, even though they had the best test accuracy among the existing optimizers. These results indicate that our Adam-ASC has a better convergence ability than the existing optimizers, especially the conventional ones, such as RMSProp and Adam.

TABLE 7. Test accuracy results evaluated in FC-DenseNet.

Methods	Test accuracies	Ranks
Adam-ASC1	**0.8675**	**1**
Adam-ASC2	0.8606	5
Adam-ASC3	0.8636	4
SGD	0.5788	12
RMSProp	0.8401	10
Adam	**0.8642**	**3**
Yogi	0.8480	9
Fromage	0.8289	11
diffGrad	0.8594	7
AdaBelief	0.8549	8
AngularGrad	0.8595	6
SAdam	**0.8654**	**2**

Finally, Table 7 shows the final test accuracy results of Adam-ASC and other optimizers. We found that the latest optimization methods, except for SAdam, had lower test accuracies than Adam. AdaBelief and diffGrad achieved relatively poor performance compared to other methods, although they achieved promising results among the existing methods except for Adam-ASC in the image classification task. Such results show that a more aggressive solution search is needed to train FC-DenseNet instead of the detailed search methods. Nevertheless, Adam-ASC1 achieved promising optimization

performance with the best test accuracies, i.e., 0.8675 when compared to other methods. Our new solution search methods could significantly improve the optimization performance of the trained CNNs regardless of their tasks. However, Adam-ASC2 and Adam-ASC3 showed slightly worse accuracies than Adam and SAdam. As explained earlier, Adam-ASC2 finely controls the search strength according to the degree of variation of the gradients, and Adam-ASC3 performs the most sophisticated search strength control using (13) and (14). Nevertheless, Adam-ASC2 and Adam-ASC3 achieved better results than other existing optimizers is significant and promising.

Thus, synthetically considering all the experimental results in both tasks, we can conclude that an appropriate search strength control is more beneficial than too strict or too lax controls in complex real-world tasks.

V. DISCUSSION

A. COMPREHENSIVE DISCUSSION ABOUT THE EXPERIMENTAL RESULTS OF ADAM-ASC

The experimental results in both tasks show that the sophisticated solution search control methods of Adam-ASC could significantly contribute to finding the approximate optimal weights in the complicated solution spaces. In particular, we found several noteworthy facts from the experiments as follows. First, each task requires different search strategies, such as aggressive or sophisticated methods. As shown in Tables 3 – 6, the latest optimization methods involving our Adam-ASC showed better performance than those of the traditional methods, such as RMSProp and Adam, in the image classification task.

Meanwhile, as shown in Fig. 11 and 12, the training accuracy curves of Adam-ASC showed relatively slower convergence than those of several existing optimizers up to about 80 epochs. Such a phenomenon is caused by the sophisticated search control method of Adam-ASC. Unlike the existing optimization mechanism, Adam-ASC performs a sophisticated and stable solution search rather than the fast search. Accordingly, the convergence of Adam-ASC may be slower than the other methods in the early epochs. However, Adam-ASC showed better training accuracies as the training progressed after about 80 epochs. Such experimental results show that using the more sophisticated and stable solution search methods is better than the conventional search methods to train the general CNNs for the image classification task.

On the other hand, in the image segmentation task with FC-DenseNet, the latest optimization methods showed lower test accuracies than Adam. In particular, AdaBelief and diffGrad achieved relatively unsatisfactory performance compared to other methods, although they achieved promising results among the existing methods except for Adam-ASC in the image classification task. Such results indicate that a more aggressive solution search is needed to train FC-DenseNet instead of the detailed search methods. Nevertheless, Adam-ASC1 achieved promising optimization performance with the best test accuracies compared to other methods. These results show that our new solution search methods could significantly improve the optimization performance of the trained CNNs, regardless of their tasks. However, Adam-ASC2 and Adam-ASC3 showed slightly lower accuracies than the result of Adam. As explained earlier, Adam-ASC2 finely controls the search strength according to the degree of variation of the gradients, and Adam-ASC3 performs the most sophisticated search strength control using both (13) and (14).

Thus, considering all the experimental results in both tasks synthetically, we can conclude that the appropriate search strength control is more beneficial than any strictly detailed control in various practical tasks.

B. WHY ADAM-ASC CAN ACHIEVE BETTER OPTIMIZATION PERFORMANCE THAN OTHER METHODS

From the experimental results, we found that the CNN models trained by Adam-ASC showed the best test accuracies when compared to the CNNs trained by other existing optimizers in the image classification and segmentation tasks. Such results indicate that the various solution search mechanisms of Adam-ASC can significantly contribute to searching optimal weight solutions in the large-scale weight solution space. Especially, our Adam-ASC can achieve better optimization performance than classical optimizers for the following reasons:

First, Adam-ASC minimizes the influence of outlier gradients when computing the first-order momentum to determine the next search direction. As explained in Section III-B, the outlier gradient often distorts the overall search trajectory of the historical gradients, allowing search in any unpromising direction. However, our Adam-ASC evaluates how far the current gradient is from the previous first-order momentum and adjusts its weight value to be used to update the first-order momentum. As a result, our Adam-ASC has a stable search trajectory, which mainly prevents falling into any local minima and contributes to finding an approximate optimal solution in a stable manner.

Second, Adam-ASC applies an adaptive noise factor $0.001t^{-1}$ in its denominator when calculating the second-order momentum at each step, as shown in (8). The noise factor decreases from 0.001 to zero as the training step progresses. In other words, in the early steps, Adam-ASC slightly adjusts the second-order momentum by adding a noise term to it. Then, as the step progresses, the noise term becomes close to zero. This allows Adam-ASC to further enhance the variety of solution searches in the early stages of training.

Third, gradient clipping is helpful in preventing excessive movement of the search trajectory [35]. Our Adam-ASC uses the gradient clipping momentum to adaptively control the degree of movement of the solution search trajectory. The gradient clipping momentum is multiplied by the first-order momentum in an element-wise manner to control each dimension's search strength sophisticatedly. For example, suppose the first and third elements of the current

gradient have magnitudes greater than a threshold. In this case, the clipping mechanism may not encourage solution search in the first and third dimensions at all. Thus, Adam-ASC can maintain a stable solution search without being sensitive to the magnitude of the gradient.

Fourth, the AS2C function described in Section III-D mainly contributes to the sophisticated solution search in the large solution space. In detail, the AS2C function adaptively controls the strength of the solution search according to the degree of variation between the current and previous gradients. This method is particularly useful for in-depth analysis of the current gradient without any computational overhead. In addition, the warm-up strategy shown in the AS2C function also contributes to further improvement of the trained CNN models although it slightly slows down their convergence by adjusting the solution strength elaborately in the early training steps.

Finally, these mechanisms are compensatively combined with each other as shown in Algorithm 7. Accordingly, Adam-ASC can be equipped with a more sophisticated and robust ability to control the solution search more effectively in large-scale solution spaces. As a result, the CNN models trained with Adam-ASC also have more powerful performance with higher image classification and segmentation accuracy than those trained with conventional optimization methods.

C. CONDITIONS THAT WEAKEN OPTIMIZATION PERFORMANCE OF ADAM-ASC

Meanwhile, Adam-ASC can have relatively worse optimization performance than the existing optimizers under several parameter setting conditions. As shown in Section IV-B, the control parameters of Adam-ASC, i.e., β_1, β_2, β_3, and φ, affect its optimization performance. Among them, when β_2 and φ, Adam-ASC showed worse test accuracies, i.e., the CNN models trained by Adam-ASC with these parameter settings achieved the worst optimization performance. In other words, the solution search ability of Adam-ASC is weakened when the coefficient value of the second-order momentum is too small or the noise factor parameter is too large. Such weaknesses of Adam-ASC under some parameter settings can be explained by analyzing the roles and characteristics of the parameters.

The coefficient parameter β_2 controls the degree of update of the current second-order momentum $\boldsymbol{s}_{t+1}$ as described in (7). Accordingly, if β_2 is set to any small value, the degree of participation of $\boldsymbol{s}_t$ is exponentially reduced when computing the new second-order momentum $\boldsymbol{s}_{t+1}$. In the actual experiments, we found that Adam-ASC has a relatively worse optimization performance when $\beta_2 = 0.5$. One of the most important reasons for using the second-order momentum is to sophisticatedly control the search strength by dividing the first-order momentum by it in an element-wise manner. Thus, if β_2 is set to a small value, the second-order momentum $\boldsymbol{s}_{t+1}$ can hardly include the previous second-order momentum $\boldsymbol{s}_t$, which causes the next search direction to be biased toward unpromising ones. Thus, we suggest setting β_2 to any large value such that $0.9 \leq \beta_2 < 1$, e.g., $\beta_2 = 0.999$.

Meanwhile, the parameter φ is used to prevent the denominator in (12) from becoming zero by adding a small value to the first-order momentum $|\boldsymbol{m}_{t+1}|$ in the denominator. In this case, if φ is set to any large value, the clipping momentum $\boldsymbol{CR}_{t+1}$ may be inaccurately calculated by the denominator with φ. In detail, $\boldsymbol{CR}_{t+1}$ approaches the one vector $\mathbf{1}$ as φ increases excessively. Actually, the parameter sensitivity test results for φ show that Adam-ASC with $\varphi = 0.1$ has the worst optimization performance, regardless of the settings of other parameters. On the other hand, when φ is less than 0.001, Adam-ASC showed stable test accuracies. Such results indicate that φ should be set to a small value such that $\varphi \leq 0.001$ to improve the optimization ability of Adam-ASC.

D. COMPUTATIONAL COMPLEXITY AND OPTIMIZATION ABILITY OF ADAM-ASC

The main factor affecting the computational complexity of the optimizers is the process of computing the gradient or Hessian matrix at each step to determine the direction of the solution search and its strength in the large-dimensional solution space. As explained in Section II, the second-order optimizers have quadratic computational complexity in terms of the dimension of the solution space due to the computation of the Hessian matrix. Accordingly, there are many limitations to using it as a base optimization method to train the optimal weight values of the large deep neural network models. In contrast, the first-order optimizers compute the gradients at each step and search for the optimal weights based on the gradients. Compared to computing the Hessian matrix in the second-order optimizers, the calculation of the gradient requires linear computational complexity, which is much less expensive. For these reasons, first-order optimizers are widely used to train practical deep neural network models [35].

Our Adam-ASC is a typical first-order optimizer because it also performs optimal solution searches based on the gradient computed at each step. In fact, Adam-ASC computes the gradient only once per step, just like the other first-order optimizers. In other words, Adam-ASC has the same computational complexity as traditional first-order optimizers such as SGD, Adam, and AdaBelief, i.e., linear complexity with respect to the dimension of the solution space.

Meanwhile, the various operations used in Adam-ASC to control the optimal solution search are performed by referring only to the gradient information computed at each step, which does not significantly affect its computational complexity. Actually, Adam-ASC calculates the gradient only once at each step and then does not compute any additional gradients. Accordingly, Adam-ASC does not sacrifice its convergence performance, despite having slightly more operations than the traditional optimizers such as SGD or Adam.

The results of the experiments in Section IV show that these analyses are reasonable. For example, as described in Algorithm 1, Adam performs an optimal solution search in

fewer operations than Adam-ASC. However, the solution search performance of Adam was inferior to that of Adam-ASC, and the final test accuracies of the CNNs trained by Adam were lower than the results of Adam-ASC. Furthermore, AdaBelief and diffGrad, which have slightly fewer operations than Adam-ASC, also showed relatively low optimization capabilities when compared to the results of Adam-ASC.

In other words, although Adam-ASC performs slightly more operations than the existing first-order optimizers such as Adam and AdaBelief, it has a more stable and robust ability to search for the optimal weights without significantly slowing down its convergence speed. These results indicate that various operations used in the core functions of Adam-ASC can mainly contribute to improving the image perception performance of the trained CNNs without incurring an excessive computational overhead.

VI. CONCLUSION

In this paper, we studied various adaptive strategies to find the optimal weight effectively in Adam-based first-order optimization algorithms. Then, we proposed a new Adam-based first-order optimization method based on these strategies, called Adam-ASC. In detail, we first developed four sophisticated and stable solution search control methods. Then, we implemented a new first-order optimizer to train CNNs, i.e., Adam-ASC, by compensatively coupling them into one. As explained in Section III, Adam-ASC was designed based on significantly intuitive and reasonable mathematical mechanisms. Accordingly, Adam-ASC can be conveniently applied as a fundamental optimizer to train the CNN models by replacing the existing optimizer. Moreover, we have mathematically verified that Adam-ASC can converge after the training phase is sufficiently progressed.

Our comprehensive experiments with practical CNNs show that Adam-ASC has remarkable optimization performance with the highest test accuracies in image classification tasks using the CIFAR-100 benchmark dataset. Moreover, Adam-ASC showed promising performance in training the FC-DenseNet in the image segmentation task using the CamVid dataset. Such experimental results indicate that our Adam-ASC is significant because it proposes new optimization methods for training CNNs and shows a promising research paradigm.

Nevertheless, our study still has several areas for improvement. As explained in Section IV, the optimization performance of Adam-ASC was only evaluated on limited tasks and applications, such as image classification and semantic segmentation. Accordingly, in the future, we will conduct more experiments on different tasks and applications, such as deep language models [51], [52], and deep reinforcement learning [53], [54]. By carefully conducting such additional studies, we expect that our Adam-ASC will be used as a fundamental optimizer to train various deep neural network models involving CNNs.

APPENDIX

We provide supplementary materials to explain the detailed proof of Theorem 1 shown in Section III-F. The supplementary file is accessible online.

REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[2] G. Li, Q. Fang, L. Zha, X. Gao, and N. Zheng, "HAM: Hybrid attention module in deep convolutional neural networks for image classification," *Pattern Recognit.*, vol. 129, Sep. 2022, Art. no. 108785.
[3] T. Kaur and T. K. Gandhi, "Deep convolutional neural networks with transfer learning for automated brain image classification," *Mach. Vis. Appl.*, vol. 31, no. 3, p. 20, Mar. 2020.
[4] H. Liu, W. Chu, and H. Wang, "Automatic segmentation algorithm of ultrasound heart image based on convolutional neural network and image saliency," *IEEE Access*, vol. 8, pp. 104445–104457, 2020.
[5] S. Niyas, S. J. Pawan, M. A. Kumar, and J. Rajan, "Medical image segmentation with 3D convolutional neural networks: A survey," *Neurocomputing*, vol. 493, pp. 397–413, Jul. 2022.
[6] S. Zou, C. Li, H. Sun, P. Xu, J. Zhang, P. Ma, Y. Yao, X. Huang, and M. Grzegorzek, "TOD-CNN: An effective convolutional neural network for tiny object detection in sperm videos," *Comput. Biol. Med.*, vol. 146, Jul. 2022, Art. no. 105543.
[7] Q. Chen, Z. Zhang, Y. Lu, K. Fu, and Q. Zhao, "3-D convolutional neural networks for RGB-D salient object detection and beyond," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 13, 2022, doi: 10.1109/TNNLS.2022.3202241.
[8] D. P. Carrasco, H. A. Rashwan, M. Á. García, and D. Puig, "T-YOLO: Tiny vehicle detection based on YOLO and multi-scale convolutional neural networks," *IEEE Access*, vol. 11, pp. 22430–22440, 2023.
[9] M. Chicchon, H. Bedon, C. R. Del-Blanco, and I. Sipiran, "Semantic segmentation of fish and underwater environments using deep convolutional neural networks and learned active contours," *IEEE Access*, vol. 11, pp. 33652–33665, 2023.
[10] R. A. Hazarika, D. Kandar, and A. K. Maji, "A deep convolutional neural networks based approach for Alzheimer's disease and mild cognitive impairment classification using brain images," *IEEE Access*, vol. 10, pp. 99066–99076, 2022.
[11] S.-J. Hong, S. Park, C.-H. Lee, S. Kim, S.-W. Roh, N. I. Nurhisna, and G. Kim, "Application of X-ray imaging and convolutional neural networks in the prediction of tomato seed viability," *IEEE Access*, vol. 11, pp. 38061–38071, 2023.
[12] S. M. Zainab, K. Khan, A. Fazil, and M. Zakwan, "Foreign object debris (FOD) classification through material recognition using deep convolutional neural network with focus on metal," *IEEE Access*, vol. 11, pp. 10925–10934, 2023.
[13] D.-L. Nguyen, M. D. Putro, and K.-H. Jo, "Driver behaviors recognizer based on light-weight convolutional neural network architecture and attention mechanism," *IEEE Access*, vol. 10, pp. 71019–71029, 2022.
[14] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.
[15] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. Int. Conf. Eng. Technol. (ICET)*, Aug. 2017, pp. 1–6.
[16] I. D. Apostolopoulos and T. A. Mpesiana, "COVID-19: Automatic detection from X-ray images utilizing transfer learning with convolutional neural networks," *Phys. Eng. Sci. Med.*, vol. 43, no. 2, pp. 635–640, Jun. 2020.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[19] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Statist.*, 2010, pp. 177–186.
[20] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning—Lecture 6a: Overview of mini-batch gradient descent," Dept. Comput. Sci., Toronto Univ., Toronto, ON, Canada, 2012. Accessed: Jun. 30, 2023. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[22] Y. Nesterov, *Lectures on Convex Optimization*. Berlin, Germany: Springer, 2018.
[23] J. D. Head and M. C. Zerner, "A Broyden—Fletcher—Goldfarb—Shanno optimization procedure for molecular geometries," *Chem. Phys. Lett.*, vol. 122, no. 3, pp. 264–270, Dec. 1985.
[24] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, nos. 1–3, pp. 503–528, Aug. 1989.
[25] S. H. Haji and A. M. Abdulazeez, "Comparison of optimization techniques based on gradient descent algorithm: A review," *PalArch's J. Archaeol. Egypt/Egyptol.*, vol. 18, no. 4, pp. 2715–2743, 2021.
[26] J. T. Zhuang, T. Tang, Y. F. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. S. Duncan, "AdaBelief optimizer: Adapting step-sizes by the belief in observed gradients," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, 2020, pp. 18795–18806.
[27] S. R. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, and B. B. Chaudhuri, "DiffGrad: An optimization method for convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4500–4511, Nov. 2020.
[28] G. Wang, S. Lu, W. Tu, and L. Zhang, "SAdam: A variant of Adam for strongly convex functions," 2019, *arXiv:1905.02957*.
[29] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020.
[30] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1175–1183.
[31] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.
[32] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.
[33] O. S. Kayhan and J. C. van Gemert, "On translation invariance in CNNs: Convolutional layers can exploit absolute spatial location," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14262–14273.
[34] Y. Mao, Z. He, Z. Ma, X. Tang, and Z. Wang, "Efficient convolution neural networks for object tracking using separable convolution and filter pruning," *IEEE Access*, vol. 7, pp. 106466–106474, 2019.
[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[36] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.
[37] W. E. L. Ilboudo, T. Kobayashi, and K. Sugimoto, "Robust stochastic gradient descent with student-t distribution based first-order momentum," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1324–1337, Mar. 2022.
[38] Y. Zhang, Y. Li, Z. Zhang, T. Luo, and Z.-Q. J. Xu, "Embedding principle: A hierarchical structure of loss landscape of deep neural networks," 2021, *arXiv:2111.15527*.
[39] M. Wang, W. Fu, X. He, S. Hao, and X. Wu, "A survey on large-scale machine learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2574–2594, Jun. 2022.
[40] K. S. Kim and Y. S. Choi, "Cooperative coevolutionary algorithm with resource allocation strategies to minimize unnecessary computations," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 108013.
[41] K. S. Kim and Y. S. Choi, "HyAdamC: A new Adam-based hybrid optimization algorithm for convolution neural networks," *Sensors*, vol. 21, no. 12, p. 4054, Jun. 2021.
[42] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–13.
[43] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," 2019, *arXiv:1904.09237*.
[44] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, Apr. 2009.
[45] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2008, pp. 44–57.
[46] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009.
[47] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 31, 2018, pp. 9815–9825.
[48] J. Bernstein, A. Vahdat, Y. Yue, and M.-Y. Liu, "On the distance between two neural networks and the stability of learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, 2020, pp. 21370–21381.
[49] S. K. Roy, M. E. Paoletti, J. M. Haut, S. R. Dubey, P. Kar, A. Plaza, and B. B. Chaudhuri, "AngularGrad: A new optimization technique for angular convergence of convolutional neural networks," 2021, *arXiv:2105.10190*.
[50] D. Liu, W. Ding, Z. S. Dong, and W. Pedrycz, "Optimizing deep neural networks to predict the effect of social distancing on COVID-19 spread," *Comput. Ind. Eng.*, vol. 166, Apr. 2022, Art. no. 107970.
[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 30, 2017, pp. 6000–6010.
[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol. (NAACL-HLT)*, 2019, pp. 4171–4186.
[53] Y. Ansari, S. Yasmin, S. Naz, H. Zaffar, Z. Ali, J. Moon, and S. Rho, "A deep reinforcement learning-based decision support system for automated stock market trading," *IEEE Access*, vol. 10, pp. 127469–127501, 2022.
[54] N. Le, V. S. Rathour, K. Yamazaki, K. Luu, and M. Savvides, "Deep reinforcement learning in computer vision: A comprehensive survey," *Artif. Intell. Rev.*, vol. 55, no. 4, pp. 2733–2819, Apr. 2022.

KYUNG SOO KIM was born in Incheon, South Korea, in 1988. He received the B.E. degree in computer education from Mokpo National University, Jeonnam, South Korea, in February 2011, and the Ph.D. degree in electronics and computer engineering from Hanyang University, Seoul, South Korea, in August 2020. He was a Postdoctoral Researcher with the Center for Computational Social Science, Hanyang University, from September 2020 to February 2022. Since March 2022, he has been an Assistant Professor with the Department of Computer Engineering, Kumoh National Institute of Technology, Gumi, Gyeongbuk, South Korea. His research interests include optimization algorithms for machine learning, evolutionary computation, nonlinear optimization algorithms, and computational intelligence theory.

YONG SUK CHOI was born in Busan, South Korea, in 1969. He received the B.S., M.S., and Ph.D. degrees in computer science from Seoul National University, Seoul, South Korea, in 1993, 1995, and 2000, respectively. He joined Hanyang University, Seoul, in 2001, after working for the Telecommunication Research Laboratory, Samsung Electronics Company, from 1997 to 2001. He is currently a Professor with the Department of Computer Science and Engineering, Hanyang University. He authorizes two books, more than 125 articles, and more than 25 inventions. His research interests include deep learning algorithms, text understandings and summarization, large language models, image transaction, visual question and answering, and multi-modal AI.