Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 5790608, 11 pages
https://doi.org/10.1155/2021/5790608

Research Article

AdaCN: An Adaptive Cubic Newton Method for Nonconvex Stochastic Optimization

Yan Liu, Maojun Zhang, Zhiwei Zhong, and Xiangrong Zeng

School of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Xiangrong Zeng; zengxrong@foxmail.com

Received 4 July 2021; Revised 30 September 2021; Accepted 15 October 2021; Published 10 November 2021

Academic Editor: Paolo Gastaldo

In this work, we introduce AdaCN, a novel adaptive cubic Newton method for nonconvex stochastic optimization. AdaCN dynamically captures the curvature of the loss landscape by diagonally approximated Hessian plus the norm of difference between previous two estimates. It only requires at most first order gradients and updates with linear complexity for both time and memory. In order to reduce the variance introduced by the stochastic nature of the problem, AdaCN hires the first and second moment to implement and exponential moving average on iteratively updated stochastic gradients and approximated stochastic Hessians, respectively. We validate AdaCN in extensive experiments, showing that it outperforms other stochastic first order methods (including SGD, Adam, and AdaBound) and stochastic quasi-Newton method (i.e., Apollo), in terms of both convergence speed and generalization performance.

1. Introduction

Stochastic gradient descent (SGD) [1] is the workhorse method for nonconvex stochastic optimization in machine learning, particularly for training deep neural networks (DNNs). During the last decades, many accelerated first order variants of SGD are widely used due to their simplicity and versatility, including the accelerated SGD (ASGD) methods using Nesterov scheme [2], momentum [3] and heavy-ball method [4], and the adaptive methods such as AdaGrad [5], AdaDelta [6], RMSProp [7], and Adam [8]. Recently, Adam has become the default optimization method for many deep learning applications because of its rapid convergence speed and relatively insensitive choices of hyperparameters [9], and it also has engendered an ever-growing list of modifications, such as AdamW [10], NosAdam [11], AMSGrad [12], AdaBound [13], Radam [14], Sadam [15], and Adax [16], to name a few. The main difference between the ASGD methods and the adaptive methods is the former scales the gradients in different directions uniformly while the latter uses adaptively element-wisely scaled learning rates, which usually causes that the latter is able to converge faster and less sensitive to the learning rate than the former. However, it has been observed that the adaptive methods may converge to bad/suspicious local optima, leading to worse generalization ability than the ASGD methods [17], or fail to converge because of unstable and extreme learning rates [13].

The abovementioned methods, belonging to the stochastic first order method family, only use gradient information and do not consider the curvature of the loss landscape, thereby leading to their suboptimal behavior in algorithmic iterations. Instead, existing stochastic second order methods can capture and exploit the curvature properties of the loss landscape by incorporating both gradient and Hessian information. For example, stochastic Newton methods are typical ones that adopt exact stochastic Hessians. However, computing the full Hessian for training large-scale DNNs is prohibitively expensive, and thus it is necessary to approximate it or avoid directly computing it in algorithmic iterations. According to the way of approximating the stochastic Hessian matrix, stochastic second order methods for training large-scale DNNs can be broadly categorized into two branches: the stochastic quasi-Newton (SQN) methods approximate the Hessian as a series sum of first order information from prior iterations, such as AdaQN

[18], SdLBFGS [19], and Apollo [20]; the stochastic second order Hessian-free methods compute the Hessian-vector product exactly through an efficient procedure proposed in [21], such as AdaHessian [22] which approximates the Hessian diagonal using Hutchinson's method based on the Hessian-vector product, which is significantly more costly than SQN methods.

Recently, based on a classic method in the nonstochastic setting, the cubic regularized Newton method [23], the stochastic adaptive regularization methods using cubics (SARC) [24–26] are proposed to address relatively small-scale nonconvex stochastic optimization problems, and they find the minimizer of a local second order Taylor approximation with a cubic regularization term at each iteration. It is observed that the SARC methods are able to escape saddle points more efficiently, leading to better generalization performance than most of the abovementioned stochastic first order and second order methods [23, 27] in relatively small-scale machine learning problems. More recently, a variant of SARC combining with negative curvature (SANC) is proposed in [28] with even better generalizability since a direction of negative curvature also benefits escaping strict saddle points [29, 30]. Unlike previous SARC methods, the SANC uses independent sets of data points of consistent size over all iterations to attain stochastic gradient and Hessian estimators, making it more practical than SARC. However, SARC and SANC use a Krylov subspace method to iteratively solve a cubic regularized Newton subproblem and use a trust region-like scheme to determine if an iteration is successful or not and update only when it is successful, which hinders their applications in large-scale nonconvex stochastic optimization (in the sense of large datasets and/or high-dimensional parameter spaces). Therefore, these existing cubic regularized Newton methods are not suitable for training large-scale DNNs.

In this work, we develop a novel adaptive cubic Newton method, AdaCN, for large-scale nonconvex stochastic optimization, and it can inherit the superiority of SARC and SANC methods (i.e., great generalizability), but tackle their aforementioned challenges (i.e., unsuitable for training large-scale DNNs). AdaCN is designed for nonconvex stochastic optimization through dynamically capturing the curvature of the loss function by diagonally approximated Hessian and the norm of difference between previous two estimates. It only requires at most first order gradients and updates with linear complexity for both time and memory, thus it is quite suitable for large-scale nonconvex stochastic optimization problems. In order to reduce the variance introduced by the stochastic nature of the problem, AdaCN hires the first and second moment to implement an exponential moving average on iteratively calculated gradients and approximated Hessians, respectively. Therefore, these moments are able to not only accelerate convergence speed but also smooth the noisy curvature information and get an approximation to the global curvature information, avoiding misleading local gradient and Hessian information which can be catastrophic. The superiority of AdaCN can be illustrated with two simple 2D functions (one is convex and the other is nonconvex, and these two functions give hints to local behavior of optimizers in deep learning), as shown in Figure 1, where we show the trajectories of different optimizers. As can be seen, AdaCN can converge much faster than stochastic first order methods such as SGD and Adam and stochastic quasi-Newton method Apollo. Furthermore, we will experimentally show the superiority of AdaCN through image classification tasks: LeNet [31] on Mnist and VGG11 [32], ResNet34 [33], and DenseNet121 [34] on CIFAR10 and CIFAR100 [35] dataset; and language modeling task: LSTM [36] on Penn Treebank [37] dataset.

Notation: we use italics letters α, β to denote scalars, bold lowercase letters $\mathbf{x}$, $\mathbf{y}$ to denote vectors, and bold uppercase letters $\mathbf{A}$, $\mathbf{B}$ to denote matrices. For vectors, we use $\|\cdot\|$ to denote the $l2$-norm.

2. Formulation of Adaptive Cubic Newton Method

2.1. Problem Statement. In this paper, we consider the following stochastic optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = E_{\xi\sim\mathcal{D}}[f(\mathbf{x};\xi)], \tag{1}$$

where f is a continuously differentiable function and possibly nonconvex, $\mathbf{x}\in\mathbb{R}^d$ is the parameter to be optimized, and E denotes the expectation with respect to $\mathbf{x}$, a random variable with the distribution $\mathcal{D}$. A special case of (1) that arises frequently in supervised machine learning is the empirical risk minimization (ERM) problem [38]:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x};\xi_i), \tag{2}$$

where $f_i\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ is the loss function corresponding to the i-th data instance, and n is the number of data samples that is assumed to be extremely large.

2.2. Newton Update from Cubically Regularized Model. We begin with a stochastic Newton (SN) method. At a high level, we sample two independent mini-batches S_g and S_B at each iteration, and the stochastic gradient and Hessian estimators, say, $\mathbf{g}_k$ and $\mathbf{B}_k$, can be defined as

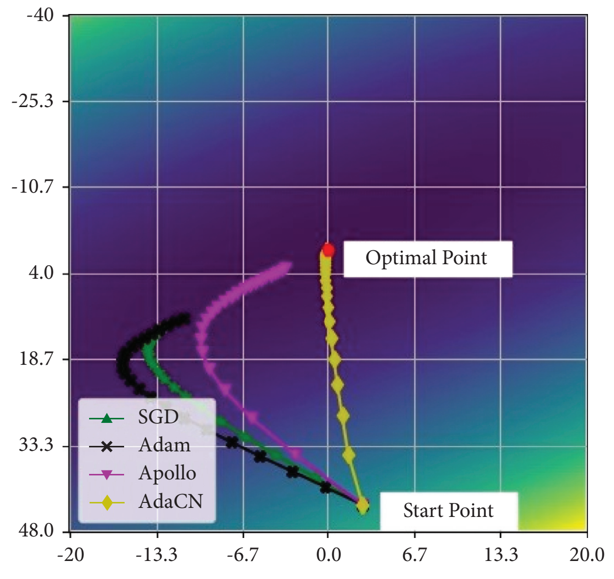
$$\begin{aligned}\mathbf{g}_k &= \frac{1}{|S_g|}\sum_{i\in S_g}\nabla f_i(\mathbf{x};\xi_i),\\ \mathbf{B}_k &= \frac{1}{|S_B|}\sum_{i\in S_B}\nabla^2 f_i(\mathbf{x};\xi_i).\end{aligned} \tag{3}$$

In each iteration of the SN method, gradient descent finds the minimizer of a local second order Taylor expansion

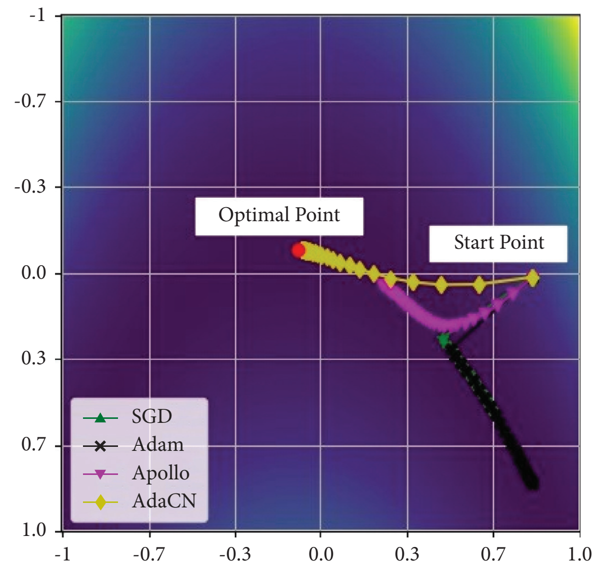
$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}_k) + \mathbf{g}_k^T(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T\mathbf{B}_k(\mathbf{x}-\mathbf{x}_k), \tag{4}$$

and its corresponding Newton update is shown as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{B}_k^{-1}\mathbf{g}_k. \tag{5}$$



(a)



(b)

Figure 1: Trajectories of SGD, Adam, Apollo, and AdaCN on a convex function (a) and a nonconvex function (b), respectively. Parameters of two experiments are set as follows: the learning rates of AdaCN, Apollo, Adam, and SGD are set to 2×10^{-3}, 2×10^{-3}, 3×10^{-2}, and 5×10^{-4}, respectively. For SGD and Apollo, $\beta = 0.9$, whereas for AdaCN and Adam, $\beta_1 = 0.9$ and $\beta_2 = 0.9$. The model is trained for 2.5×10^3 epochs. (a) Loss function is $f(x, y) = (x + y)^2 + (x - y)^2/10$. (b) $f(x, y) = (x - 1)^2 + 100(y - x^2)^2$.

For large-scale stochastic optimization problems, the stochastic Hessian $\mathbf{B}_k$ is properly approximated. For example, AdaQN [18], SdLBFGS [19], and Apollo [20] (belonging to stochastic quasi-Newton methods) approximate $\mathbf{B}_k$ as a series sum of first order information from prior iterations, while AdaHessian [22] (one of the stochastic second order Hessian-free methods) approximates the Hessian diagonal using Hutchinson's method [39] based on the Hessian-vector product.

A new principled variant of the SN method that could enjoy global convergence guarantees is proposed in [23], and it finds the minimizer of the following cubically regularized second order approximation of f with $\mathbf{g}_k$, $\mathbf{B}_k$ and a sufficient large $\rho > 0$:

$$\begin{aligned}\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}_k) + g_k^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T \\ B_k(\mathbf{x} - \mathbf{x}_k) + \frac{\rho}{6}\|\mathbf{x} - \mathbf{x}_k\|^3,\end{aligned} \tag{6}$$

where $\mathbf{x}_k$ is the value at k-th iteration. By first order optimality conditions, we set the derivative of the objective to zero, which immediately yields

$$\mathbf{g}_k + \mathbf{B}_k(\mathbf{x} - \mathbf{x}_k) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{x}_k\|(\mathbf{x} - \mathbf{x}_k) = 0, \tag{7}$$

which is a nonlinear system and can be approximated by a linear one as follows:

$$\mathbf{g}_k + \mathbf{B}_k(\mathbf{x} - \mathbf{x}_k) + \frac{\rho}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|(\mathbf{x} - \mathbf{x}_k) = 0, \tag{8}$$

yielding a novel Newton update:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{B}_k + \frac{\rho}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \cdot \mathbf{I}\right)^{-1}\mathbf{g}_k, \tag{9}$$

where I represents the identity matrix with the same size as $\mathbf{B}_k$. Comparing (5) with (9) additionally makes use of the norm of difference between previous two estimates, leading to better performance since it captures more curvature information.

2.3. Updating $\mathbf{B}_k$.

The stochastic Hessian $\mathbf{B}_k$ can be updated based on the weak secant equation [40, 41]:

$$\begin{aligned}\mathbf{B}_k &= \arg\min_{\mathbf{B}}\|\mathbf{B} - \mathbf{B}_{k-1}\|, \\ \text{s.t. } \mathbf{s}_k^T\mathbf{B}s_k &= \mathbf{s}_k^T\mathbf{y}_k \,(\text{weak secant equation}),\end{aligned} \tag{10}$$

where $\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$ and $\mathbf{y}_k = \mathbf{g}_k - \mathbf{g}_{k-1}$. The solution of the above problem with Frobenius matrix based on the variational technique in [42] is given by

$$\mathbf{B}_k = \mathbf{B}_{k-1} + \frac{\mathbf{s}_k^T\mathbf{y}_k - \mathbf{s}_k^T\mathbf{B}_{k-1}\mathbf{s}_k}{\|\mathbf{s}_k\|_4^4 + \varepsilon}\mathrm{Diag}(\mathbf{s}_k^2), \tag{11}$$

where $\mathrm{Diag}(\mathbf{v})$ is the diagonal matrix with diagonal elements from vector $\mathbf{v}$. Further, with rectifying operation which guarantees the positive-definiteness, the updated cubic Newton becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{D}_k^{-1}\mathbf{g}_k, \tag{12}$$

where

$$\mathbf{D}_k = \max\left(\mathrm{abs}\left(\mathbf{B}_k + \frac{\rho}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \cdot \mathbf{I}\right), \theta \cdot \mathbf{I}\right), \tag{13}$$

```
     Require: $n_g$ //Mini-batch size
     Require: $\eta$ //Stepsize
     Require: $\beta_1, \beta_2 \in [0,1)$//Parameters of exponential moving average
     Require: $\varepsilon, \rho, \theta$//Positive parameters
     Require: $\mathbf{x}_0, \mathbf{g}_0, \mathbf{B}_0, \mathbf{m}_0, \mathbf{V}_0$//Initialize variables as zero vectors or zero matrices
     Require: $k \longleftarrow 0$//Initialize timestep
(1)  While $\mathbf{x}_k$ not converged do
(2)    $k \longleftarrow k+1$
(3)    sample $S_g \longleftarrow \{\xi_i\}_{i=1}^{n_g}$
(4)    $\mathbf{g}_k \longleftarrow (1/|S_g|)\sum_{i\in S_g}\nabla f_i(\mathbf{x}, \xi_i)$//Stochastic gradient at timestep $k$
(5)    $\mathbf{s}_k \longleftarrow \mathbf{x}_k - \mathbf{x}_{k-1}$
(6)    $\mathbf{y}_k \longleftarrow \mathbf{g}_k - \mathbf{g}_{k-1}$
(7)  $\mathbf{B}_k \longleftarrow \mathbf{B}_{k-1} + ((\mathbf{s}_k^T\mathbf{y}_k - \mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k)/\|\mathbf{s}_k\|_4^4 + \varepsilon)\mathrm{Diag}(\mathbf{s}_k^2)$//Update diagonal Hessian
(8)    $\mathbf{D}_k \longleftarrow \max(|\mathbf{B}_k + (\rho/2)\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \cdot \mathbf{I}|, \theta)$
(9)    $\mathbf{m}_k \longleftarrow \beta_1\mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k$
(10)   $\mathbf{V}_k \longleftarrow \beta_2\mathbf{V}_{k-1} + (1-\beta_2)\mathbf{D}_k \odot \mathbf{D}_k$
(11)   $\widehat{m}_k \longleftarrow \mathbf{m}_k/(1-\beta_1^k)$
(12)   $\widehat{V}_k \longleftarrow \mathbf{V}_k/(1-\beta_2^k)$
(13)   $\mathbf{x}_{k+1} \longleftarrow \mathbf{x}_k - \eta\widehat{V}_k^{-(1/2)}\widehat{m}_k$
(14) end while
(15) return $\mathbf{x}_{k+1}$
```

Algorithm 1: AdaCN.

where θ is a positive parameter, and the cost of computing $\mathbf{D}$ is marginal since $\mathbf{B}_k$ is diagonal, abs$(\mathbf{V})$ takes absolute values of all the elements of the matrix $\mathbf{V}$. It is worth mentioning that, according to equations (5) and (12), $\max(\cdot,\cdot)$ operation enables $\mathbf{D}_k$ to prevent the step size from becoming arbitrary large since there exists zero value in $\mathbf{B}_k$.

2.4. Moments for $\mathbf{g}_k$ and $\mathbf{D}_k$. In this paper, we adopt the moments for $\mathbf{g}_k$ and $\mathbf{D}_k$, given by

$$\begin{aligned}\mathbf{m}_k &\longleftarrow \beta_1\mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k,\\ \mathbf{V}_k &\longleftarrow \beta_2\mathbf{V}_{k-1} + (1-\beta_2)\mathbf{D}_k \odot \mathbf{D}_k,\end{aligned} \tag{14}$$

where $\odot$ denotes the elementwise multiplication, and $\beta_1, \beta_2 \in (0,1)$ are the first and second moment hyperparameters that are also used in Adam [8] and its many variants. The moments are further bias corrected as

$$\begin{aligned}\widehat{m}_k &\longleftarrow \frac{\mathbf{m}_k}{(1-\beta_1^k)},\\ \widehat{V}_k &\longleftarrow \frac{\mathbf{V}_k}{(1-\beta_2^k)}.\end{aligned} \tag{15}$$

Using the first and second moment amounts to carrying out an exponential moving average on iteratively updated stochastic gradients and approximated stochastic Hessians plus the norm of difference between previous two estimates, which can smooth the noisy curvature information and get an approximation to the global curvature information, avoiding misleading local gradient and Hessian information which can be catastrophic.

To summarize, the complete algorithm of AdaCN is given in Algorithm 1, in which at most first order gradients are required, and $\mathbf{B}_k$, $\mathbf{D}_k$, $\mathbf{V}_k$, and $\widehat{V}_k$ are all diagonal. Therefore, AdaCN updates with linear complexity for both time and memory.

3. Experiments

In this section, we access the performance of AdaCN on learning tasks: image classification and language modeling, comparing with stochastic first order optimizers such as Adam [8], SGD [1], AdaBound [13], and stochastic second order optimizer Apollo [20], listed in Table 1. For image classification, we investigate models by LeNet [31] on Mnist and VGG11 [32], ResNet34, and DenseNet121 [34] on CIFAR10 and CIFAR100 [35], while for language modeling, LSTM [36] on Penn Treebank [37] is tested. Moreover, the robustness to hyperparameters is tested, through comparing AdaCN and Apollo with different values of ε and learning rate. The results are shown in the following subsections.

3.1. Experiments Setup. We perform a careful hyperparameter tuning in experiments as follows. AdaCN: we set $\theta = 1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and $\rho = 5 \times 0.9^{200-k}$ at k-th iteration. The learning rate $\eta = 0.4$ for Mnist dataset, 0.2 for CIFAR datasets, and 30 for Penn Treebank dataset. SGD: the momentum is set to 0.9, while the learning rate is searched among $\{a \times 10^b\}$ where $a \in \{1,2,3,4,5,6,7,8,9\}$ and $b \in \{-3,-2,-1,0,1\}$. Adam, AdaBound, and Apollo: the learning rate is searched as SGD, and other parameters are set as their own default values in the literature.

3.2. Image Classification. At first, we evaluate the convergence and generalization of AdaCN on image classification. We use LeNet on Mnist and VGG11, ResNet34, and DenseNet121 on CIFAR10 and CIFAR100 dataset.

Table 1: Summaries of the settings used in experiments.

Task	Dataset	Network type	Architecture	Optimizer
Image classification	Mnist CIFAR10 CIFAR100	Convolutional neural network	LeNet VGG11, ResNet34, DenseNet121	AdaCN, Apollo, Adam, SGD, AdaBound
Language modeling	Penn Treebank	Recurrent	1, 2, 3-layer LSTM	

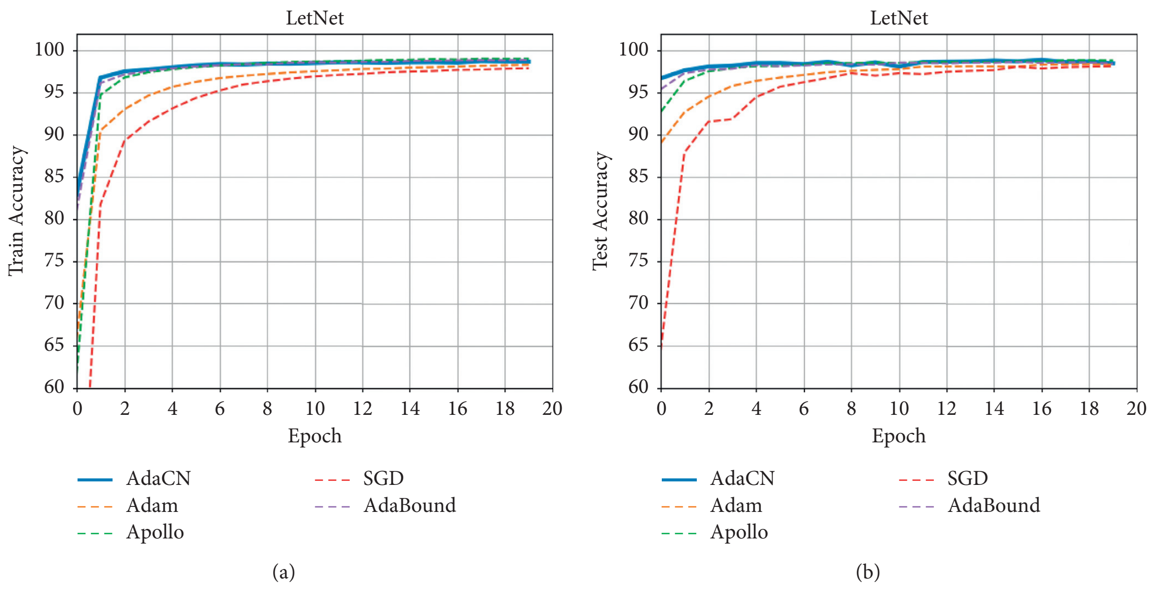


Figure 2: The curves of train and test accuracy on Mnist. (a) Train Accuracy. (b) Test Accuracy.

Table 2: Test accuracy of LeNet on Mnist.

Model	Adam	SGD	AdaBound	Apollo	AdaCN
LeNet	0.984	0.981	0.987	0.988	**0.989**

The best result is shown in bold.

Mnist. Results on Mnist are shown as Figure 2: the curves of train and test accuracy on Mnist and Table 2, from which we can see that AdaCN achieves the best convergence speed and classification accuracy.

CIFAR10. We report the results on CIFAR10 in Figure 3 and Table 3. For all three network architectures, AdaCN obviously outperforms other optimizers with comparable convergence speed and best classification accuracy.

CIFAR100. Results on CIFAR100 are shown in Figure 4 and Table 4. For VGG11, AdaCN is better than Adam and Apollo, but worse than SGD and AdaBound in terms of convergence speed and classification accuracy. For ResNet34 and DenseNet121, AdaCN achieves the best classification accuracy.

3.3. Language Modeling. On language modeling, we experiment with 1, 2, 3-layer LSTM model on Penn Treebank dataset. As Figure 5 and Table 5 show, AdaCN can also keep fastest convergence speed and achieve the lowest perplexity among the optimizers for 1, 2, 3-layer LSTM.

Finally, we explore the effects of the hyperparameters including ε and learning rates on the performance of AdaCN, respectively. The results on CIFAR10 dataset are reported in Figures 6 and 7, where the values of ε are ranging from 10^{-3} to 10^{-7} in a log-scale grid on ResNet34 and learning rate ranging from 5×10^{-3} to 2×10^{-1} on VGG11, respectively. As can be seen, the test accuracies of AdaCN are above 95% for all values of ε, while Apollo achieves an accuracy consistently below 95%. Moreover, the results validate the robustness of AdaCN to ε and learning rate.

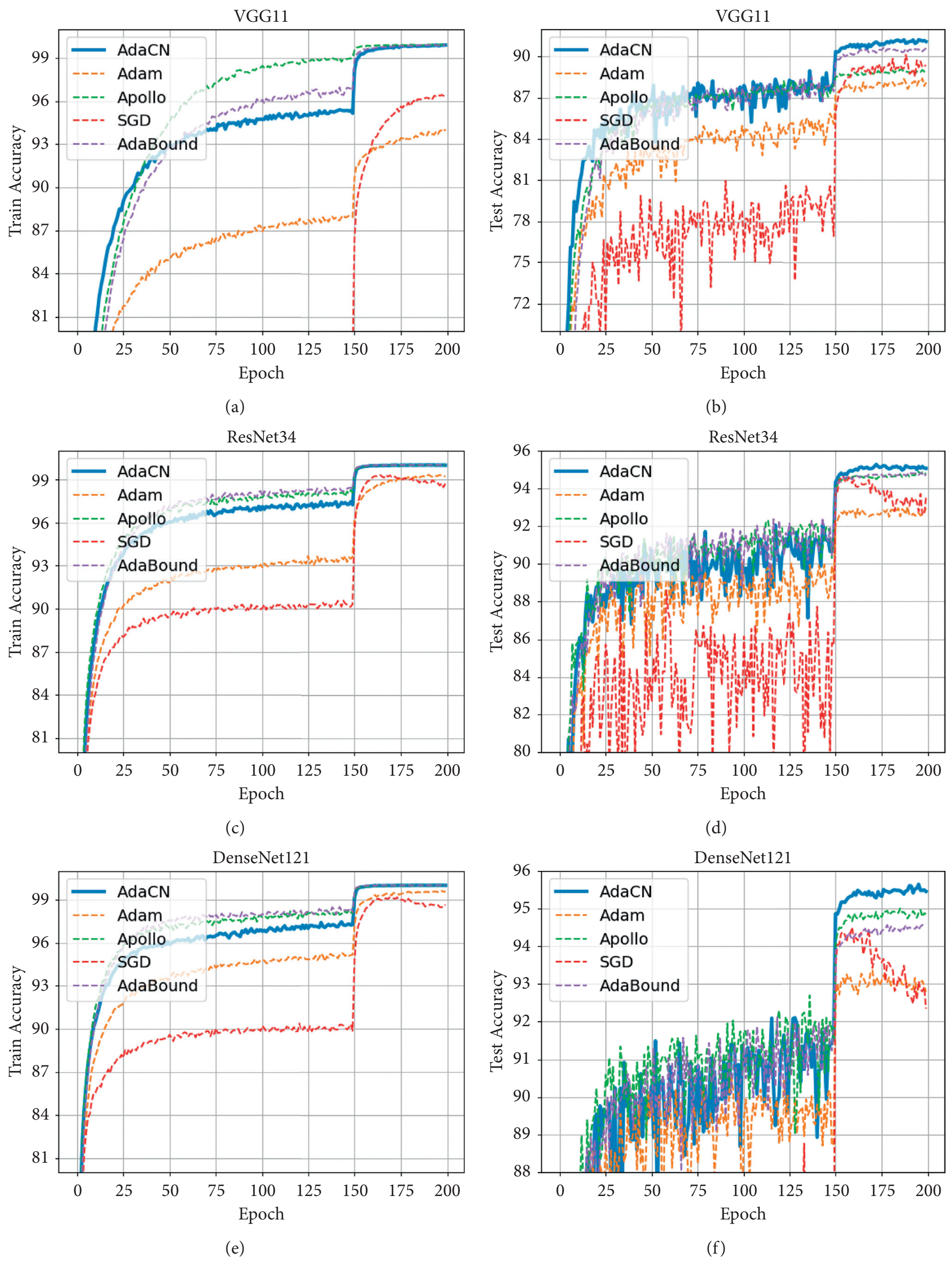
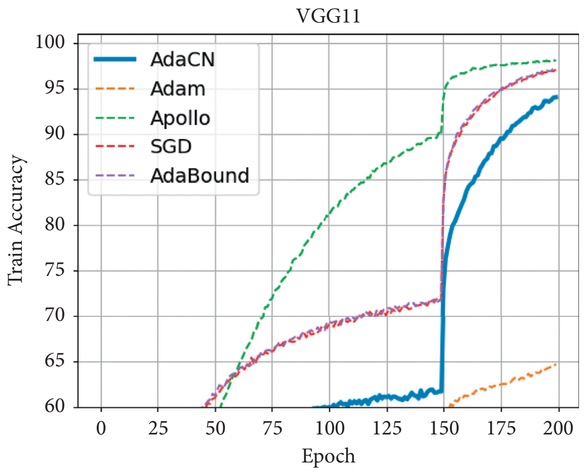


Figure 3: The curves of train and test accuracy on CIFAR10. (a) Train accuracy of VGG11 on CIFAR10. (b) Test accuracy of VGG11 on CIFAR10. (c) Train accuracy of ResNet34 on CIFAR10. (d) Test accuracy of ResNet34 on CIFAR10. (e) Train accuracy of DenseNet121 on CIFAR10. (f) Test accuracy of DenseNet121 on CIFAR10.

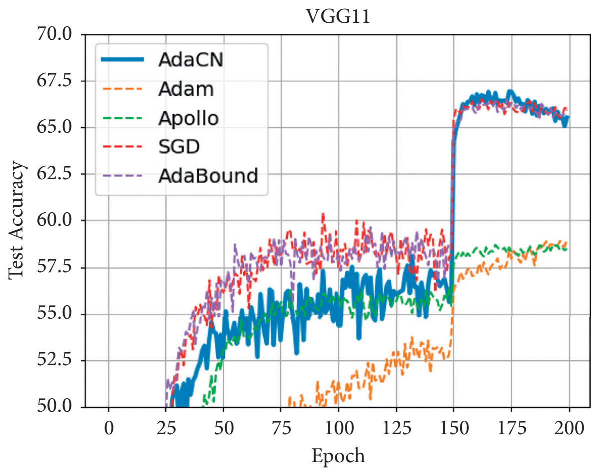
Table 3: Test accuracy of VGG11, ResNet34, and DenseNet121 on CIFAR10.

Model	Adam	SGD	AdaBound	Apollo	AdaCN
VGG11	88.40	89.72	90.60	88.90	**91.34**
ResNet34	93.02	93.62	94.75	94.79	**95.15**
DenseNet121	92.79	93.07	94.58	94.85	**95.52**

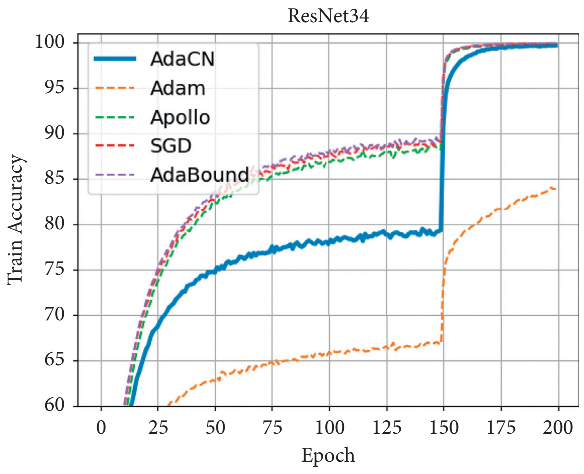
The best results are shown in bold.



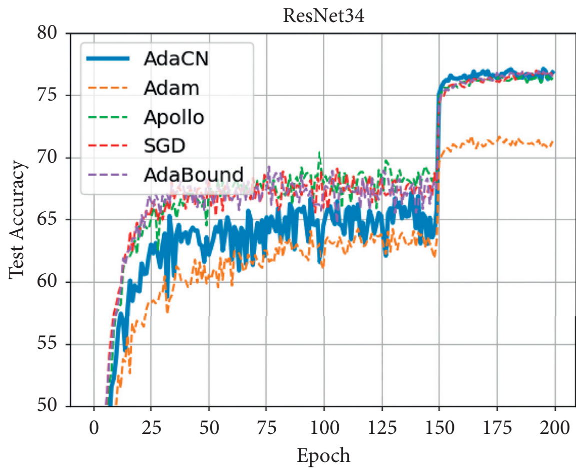
(a)



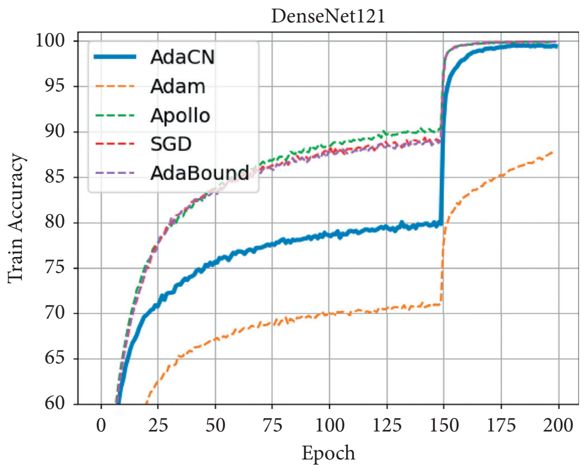
(b)



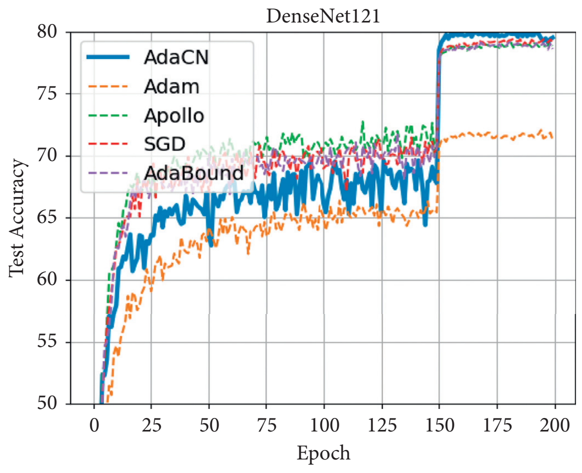
(c)



(d)



(e)



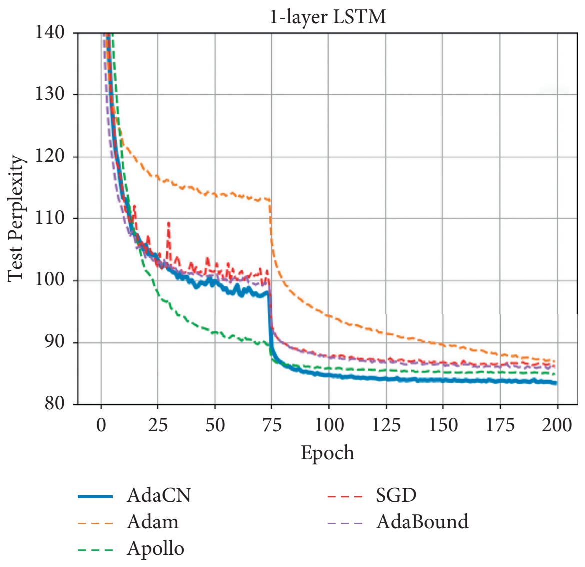
(f)

Figure 4: The curves of train and test accuracy on CIFAR100. (a) Train accuracy of VGG11 on CIFAR100. (b) Test accuracy of VGG11 on CIFAR100. (c) Train accuracy of ResNet34 on CIFAR100. (d) Test accuracy of ResNet34 on CIFAR100. (e) Train accuracy of DenseNet121 on CIFAR100. (f) Test accuracy of DenseNet121 on CIFAR100.

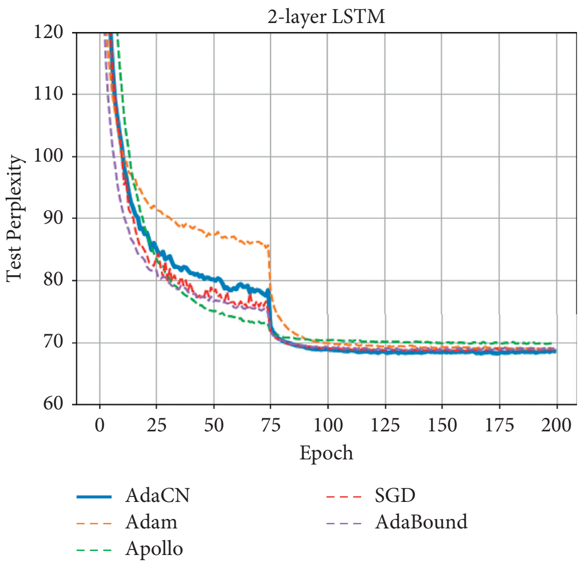
Table 4: Test accuracy of VGG11, ResNet34, and DenseNet121 on CIFAR100.

Model	Adam	SGD	AdaBound	Apollo	AdaCN
VGG11	58.96	**66.01**	65.92	58.50	65.51
ResNet34	71.39	76.98	76.88	76.42	**77.11**
DenseNet121	71.85	79.35	79.00	79.01	**79.50**

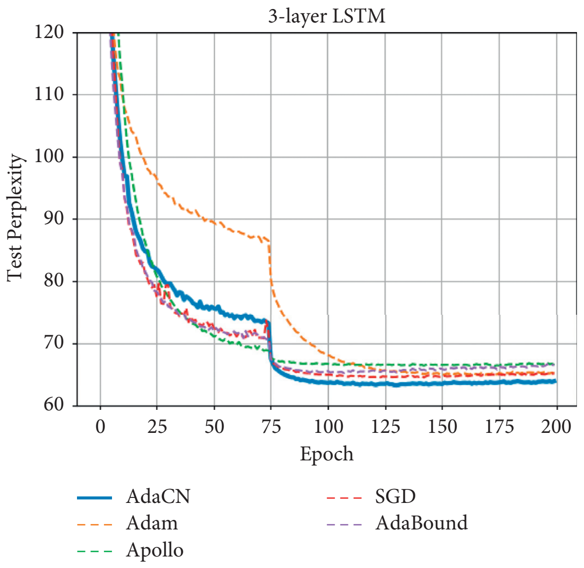
The best results are shown in bold.



(a)



(b)



(c)

Figure 5: The curves of test perplexity on Penn Treebank for 1, 2, 3-layer LSTM. Lower is better. (a) Test perplexity for 1-layer LSTM. (b) Test perplexity for 2-layer LSTM. (c) Test perplexity for 3-layer LSTM.

Table 5: Test accuracy of 1, 2, 3-layer LSTM on Penn Treebank dataset.

Model	Adam	SGD	AdaBound	Apollo	AdaCN
1-layer LSTM	86.86	86.14	85.68	84.90	**83.40**
2-layer LSTM	68.81	68.54	68.58	69.71	**68.17**
3-layer LSTM	64.91	64.50	65.18	66.37	**63.26**

The best results are shown in bold.

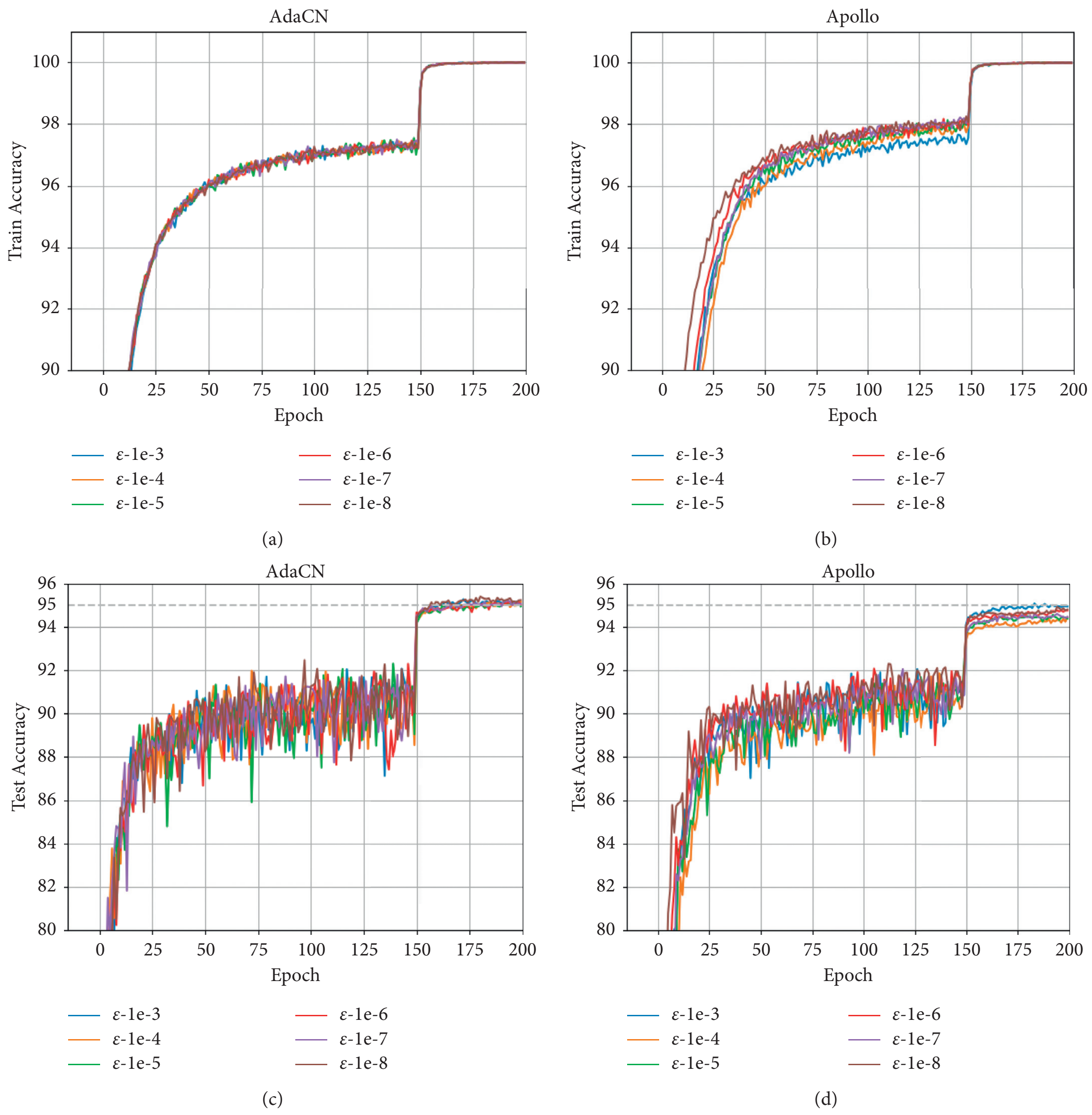


Figure 6: The curves of train and test accuracy of ResNet34 on CIFAR10 with respect to different values of ε. (a) Train accuracy of AdaCN on CIFAR10. (b) Train accuracy of Apollo on CIFAR10. (c) Test accuracy of AdaCN on CIFAR10. (d) Test accuracy of Apollo on CIFAR10.

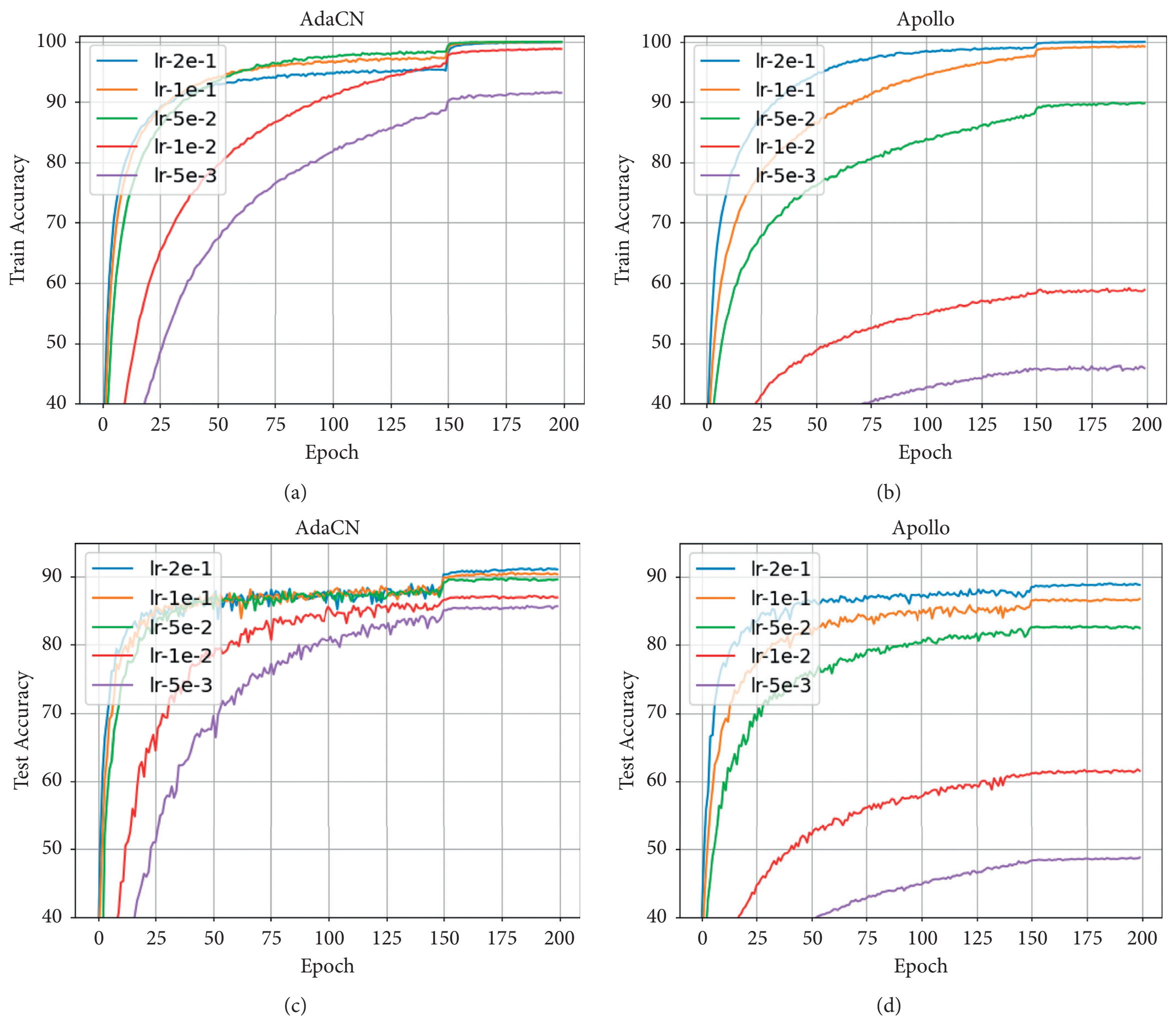


Figure 7: The curves of train and test accuracy of VGG11 on CIFAR10 with respect to different values of learning rate. As can be seen, AdaCN is more robust to learning rate. (a) Train accuracy of AdaCN on CIFAR10. (b) Train accuracy of Apollo on CIFAR10. (c) Test accuracy of AdaCN on CIFAR10. (d) Test accuracy of Apollo on CIFAR10.

4. Conclusion

We have proposed AdaCN, a novel, efficient, and effective adaptive cubic Newton method for nonconvex stochastic optimization. This method is designed for large-scale nonconvex stochastic optimization problems which are the core of state-of-the-art deep learning literature. Experimental results on image classification tasks and language modeling task demonstrate the superiority of AdaCN, in terms of convergence speed and generalization performance.

Data Availability

The data used to support the findings of this study are open datasets which could be found in general websites, and the datasets are also freely available.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported in part by the Neural Science Foundation of Hunan Province (Grant No. 2019JJ50746) and National Nature Science Foundation of China (Grant No. 61602494).

References

[1] H. Robbins and S. Monro, “A stochastic approximation method,” *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.
[2] Y. Nesterov, “A method of solving a convex programming problem with convergence rate o (1/k̂2),” *Soviet mathematics-Doklady*, vol. 269, 1983.
[3] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, no. 3, pp. 1139–1147, 2013.
[4] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[5] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[6] M. D. Zeiler, "Adadelta: an adaptive learning rate method," p. 5701, 2012, http://arxiv.org/abs/1212.5701.

[7] A. Graves, "Generating sequences with recurrent neural networks," 2013, http://arxiv.org/abs/1308.1250850.

[8] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.

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

[10] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 2019.

[11] H. Huang, C. Wang, and B. Dong, "Nostalgic adam: weighting more of the past gradients when designing the adaptive learning rate," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pp. 2556–2562, Macao, China, August 2019.

[12] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," 2019, http://arxiv.org/abs/1904.09237.

[13] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," in *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 2019.

[14] L. Liu, H. Jiang, P. He, W. Chen, J. G. X. Liu, and J. Han, "On the variance of the adaptive learning rate and beyond," 2019, http://arxiv.org/abs/1908.03265.

[15] G. Wang, S. Lu, W. Tu, and L. Zhang, "Sadam: a variant of adam for strongly convex functions," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pp. 2556–2562, Macao, China, August 2019.

[16] W. Li, Z. Zhang, X. Wang, and P. Luo, "Adax: adaptive gradient descent with exponential long term memory," 2020, http://arxiv.org/abs/2004.09740.

[17] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2017*, pp. 4148–4158, Long Beach, CA, USA, December 2017.

[18] N. S. Keskar and A. S. Berahas, "Adaqn: an adaptive quasi-newton algorithm for training rnns," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2017*, pp. 4148–4158, Long Beach, CA, USA, December 2017.

[19] X. Wang, S. Ma, D. Goldfarb, and W. Liu, "Stochastic quasi-newton methods for nonconvex stochastic optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 927–956, 2017.

[20] X. Ma, "Apollo: an adaptive parameter-wise diagonal quasi-newton method for nonconvex stochastic optimization," 2020, http://arxiv.org/abs/2009.13586.

[21] B. A. Pearlmutter, "Fast exact multiplication by the Hessian," *Neural Computation*, vol. 6, no. 1, pp. 147–160, 1994.

[22] Z. Yao, A. Gholami, S. Shen, K. Keutzer, and M. W. Mahoney, "Adahessian: an adaptive second order optimizer for machine learning," 2020, http://arxiv.org/abs/2006.00719.

[23] Y. Nesterov and B. T. Polyak, "Cubic regularization of newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006.

[24] N. Tripuraneni, M. Stern, C. Jin, J. Regier, and M. I. Jordan, "Stochastic cubic regularization for fast nonconvex optimization," 2017, http://arxiv.org/abs/1711.02838.

[25] J. M. Kohler and A. Lucchi, "Sub-sampled cubic regularization for non-convex optimization," 2017, http://arxiv.org/abs/1705.05933.

[26] E. H. Bergou, Y. Diouane, and S. Gratton, "A line-search algorithm inspired by the adaptive cubic regularization framework and complexity analysis," *Journal of Optimization Theory and Applications*, vol. 178, no. 3, pp. 885–913, 2018.

[27] P. Xu, F. Roosta, and M. W. Mahoney, "Second-order optimization for nonconvex machine learning: an empirical study," in *Proceedings of the 2020 SIAM International Conference on Data Mining*, Cincinnati, OH, USA, May 2020.

[28] S. Park, S. H. Jung, and P. M. Pardalos, "Combining stochastic adaptive cubic regularization with negative curvature for nonconvex optimization," *Journal of Optimization Theory and Applications*, vol. 184, 2020.

[29] S. J. Reddi, M. Zaheer, S. Sra et al., "A generic approach for escaping saddle points," in *Proceedings of the International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, vol. 84, pp. 1233–1242, Playa Blanca, Lanzarote, Canary Islands, Spain, April 2018.

[30] Y. Xu, J. Rong, and T. Yang, "First-order stochastic algorithms for escaping from saddle points in almost linear time," in *Proceedings of the Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 5535–5545, Montreal, Canada, December 2018.

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

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference 180 on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, Las Vegas, NV, USA, June 2016.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Honolulu, HI, USA, July 2017.

[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, Elsevier, Amsterdam, Netherlands, 2009.

[36] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, vol. 9, pp. 473–479, Denver, CO, USA, December 1996.

[37] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: the penn -treebank, Comput," *Linguistics*, vol. 19, pp. 313–330, 1993.

[38] V. N. Vapnik, *Statistical Learning Theory*, pp. 99–106, John Wiley & Sons, Hoboken, NJ, USA, 1998.

[39] H. Avron and S. Toledo, "Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix," *Journal of the ACM*, vol. 58, no. 2, p. 8, 2011.

[40] J. L. Nazareth, "If quasi-newton then why not quasi-cauchy," *SIAG/OPT Views-and-News*, vol. 6, pp. 11–14, 1995.

[41] J. E. Dennis and A. H. Wolkowicz, "Sizing and least-change secant methods," *SIAM Journal on Numerical Analysis*, vol. 30, no. 5, 1993.

[42] M. Zhu, J. L. Nazareth, and H. Wolkowicz, "The quasi-cauchy relation and diagonal updating," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1192–1204, 1999.