



Make ℓ_1 regularization effective in training sparse CNN

Juncai He¹ · Xiaodong Jia² · Jinchao Xu¹ · Lian Zhang¹  · Liang Zhao³

Received: 24 August 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Compressed Sensing using ℓ_1 regularization is among the most powerful and popular sparsification technique in many applications, but why has it not been used to obtain sparse deep learning model such as convolutional neural network (CNN)? This paper is aimed to provide an answer to this question and to show how to make it work. Following Xiao (J Mach Learn Res 11(Oct):2543–2596, 2010), We first demonstrate that the commonly used stochastic gradient decent and variants training algorithm is not an appropriate match with ℓ_1 regularization and then replace it with a different training algorithm based on a regularized dual averaging (RDA) method. The RDA method of Xiao (J Mach Learn Res 11(Oct):2543–2596, 2010) was originally designed specifically for convex problem, but with new theoretical insight and algorithmic modifications (using proper initialization and adaptivity), we have made it an effective match with ℓ_1 regularization to achieve a state-of-the-art sparsity for the highly non-convex CNN compared to other weight pruning methods without compromising accuracy (achieving 95% sparsity for ResNet-18 on CIFAR-10, for example).

Keywords Sparse optimization · ℓ_1 regularization · Dual averaging · CNN

1 Introduction

This paper is devoted to the training of sparse deep neural networks. In the many successful applications of deep learning [18], the number of weights in most of the relevant models is often much more than the number of data available (c.f. [9, 13,

✉ Jinchao Xu
xu@math.psu.edu

¹ Department of Mathematics, Pennsylvania State University, University Park, PA 16802, USA

² Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA

³ State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, Beijing 100190, China

34]). It is therefore of great theoretical and practical interests to develop numerical methods to reduce such weight redundancy and hence compress the network models. The aim of this paper is to study sparse training algorithms for a special class of deep neural networks, namely convolutional neural networks (CNN).

As summarized in [4], roughly speaking, there are four major different methods that have been developed for compressing neural network models: (1) network pruning and sharing, (2) low-rank factorization, (3) transferred/compact convolutional filters and, (4) knowledge distillation. In particular, the network pruning is the most popular compressing method due to its good compatibility and competitive performance and it is also the one that the current paper focuses on.

Among the many possible approaches for network pruning, the widely used compressed sensing with ℓ_1 regularization [3, 5] appears to be an obvious choice. One natural step is to first add a proper multiple of the ℓ_1 norm of the weights to a standard loss function and then train the resulting model with the most commonly used training algorithms such as SGD [29]. But this approach, as observed in [10], does not give satisfactory sparse results for CNN models. Another approach [2, 17] is to zero out the weights under a threshold at each iteration by using a proximal SGD (Prox-SGD). As explained in Sect. 3, this approach is slightly more efficient than the above method, but still generates very limited sparsity due to its decaying soft-thresholding parameter.

Perhaps due to the aforementioned non-satisfactory performances of SGD when applied to ℓ_1 regularization, no reports can be found in the literature on any successful application of compressed sensing technique with ℓ_1 regularization to deep neural networks. Such a situation is, however, different in the context of convex optimization such as logistic regressions. Xiao [37] successfully developed a special compressed sensing technique for convex machine learning models. In this work, he also observed that the SGD type method is not effective when used with ℓ_1 regularization. Instead he turns to the simple dual averaging method (SDA) [32], which is specifically designed for convex optimization problems. By combining SDA with ℓ_1 regularization, Xiao [37] developed the regularized dual averaging (RDA) method and obtained very satisfactory sparse solutions of convex stochastic regularized problems.

One natural question is if the idea in [37] can be generalized to deep neural networks that are often highly non-convex. But, with an extensive literature search, we have not yet found any works that discuss such a generalization. In fact, we could not find any works that use SDA type of methods for the training of any machine learning models that are not convex. Given the fact that SDA method is originally designed for convex problems naturally, SDA is not expected to work and has never been applied for non-convex problems, not to mention non-convex problems together with ℓ_1 regularization.

Despite of these historic developments, we report in this paper that SDA, with some appropriate modification, can also be made highly effective with ℓ_1 regularization to obtain sparse convolutional neural networks. Our work is motivated by a critical observation that we made and report in this paper: SDA can be interpreted as a perturbation of SGD! Since SGD is a good training algorithm for CNN, we expect that SDA is potentially also a good training algorithm for CNN. Furthermore, we

demonstrate that SDA can be combined with a soft-thresholding operator in the forward–backward splitting form to obtain an RDA algorithm for training sparse CNN.

With careful theoretical analysis and extensive numerical experiments, we find that the effectiveness of our RDA method depends crucially on two important techniques, namely (1) proper initialization, and (2) adaptive sparse retraining. The first one is the key for RDA to work with CNN, and the second one further improves both sparsity and accuracy. Consequently, our RDA training process can be described as a two-step pipeline: (1) train CNN by RDA with a specific initialization, and (2) apply adaptive sparse retraining. These two steps lead to state-of-the-art performance of RDA to achieve high sparsity for CNN without compromising accuracy in comparison with other weight pruning methods.

The remainder of this paper is organized as follows. In Sect. 3, we briefly review SGD, SDA, Prox-SGD and RDA, then provide a comparison of these methods to explain why RDA performs better than the other methods. We describe in Sect. 4 two techniques that are essential in utilizing RDA. Following the detailed implementation listed in Sect. 5, we show the numerical results of different methods and compare them with some existing work. In Sect. 6, we summarize our results.

2 Related works

Recently, there have been many discussions in the literature on the value of network pruning. Liu et al. [24] reviews various pruning methods and proposes that the value of network pruning is to search good architectures. Mittal et al. [30] shows that a randomly pruned network has comparable performance to the original one due to its plasticity. Zhu and Suyog [39] argues that pruned large sparse models outperform small-dense models, although their memory footprints are almost the same, and hence indicates that network pruning is meaningful in practice.

In general, network pruning includes individual weight pruning and structured pruning. The earliest examples of individual weight pruning methods are Optimal Brain Damage [19] and Optimal Brain Surgeon [11]. Recently, Han et al. [10] presents a general three-step pipeline: training, pruning and fine-tuning. Typically, individual weight pruning can only guarantee the sparsity of weight matrices, but does not necessarily lead to compression and speedup without the support of specific hardware and libraries. A three-stage pipeline is proposed by Han et al. [9] to reduce the storage and energy required to run the networks. The first stage is based on the individual weight pruning in [10], followed by quantization and Huffman coding stages to reduce the storage.

Structured pruning, on the other hand, aims to prune the filters or channels. Filters can be pruned based on their corresponding ℓ_1 norm [21]. Similarly, some other methods prune filters based on the information of output channels [14, 15, 25]. Group sparsity is also widely used in the pruning process after training. Wen et al. [36] proposes a group sparsity strategy including filter-wise, channel-wise, shape-wise and depth-wise structured sparsity. Alvarez and Mathieu [1] makes use of a group regularizer on the neurons of the fully connected layers. Liu et al. [23] utilizes the scaling factors in BN layers as a metric to prune filters. Huang and Naiyan [16]

selects sparse structures by imposing sparsity constraints on the outputs of specific structures, such as neurons, groups or residual blocks.

Compressed sensing with ℓ_1 regularization has been successfully used in many applications [7, 26]. As an important technique, ℓ_1 regularization is also adopted in machine learning fields to obtain sparse model in specific learning problems. In past few years, numerous algorithms are designed to find solutions of regularized convex optimization problems. Among them, the Prox-SGD method, also known as FOBOS [6] in forward–backward splitting form [22] has been used in deep learning, for example in [16]. As noticed in [38], one drawback of Prox-SGD is that the thresholding parameters will decay in the training process, which results in unsatisfactory sparsity. Thus Xiao [38] developed the RDA method to obtain more sparse solution while keeping the accuracy, and further established the convergence of his RDA method for convex problems. But Xiao’s RDA method has not yet been applied in deep learning thus far.

3 Algorithms using ℓ_1 regularization

In this section, we first briefly review SGD and SDA as training algorithms for deep learning, and prove that SDA can be viewed as a perturbation of SGD. We then introduce Prox-SGD and RDA for ℓ_1 regularized problems. In the equivalent forward–backward splitting form, these two algorithms can be viewed as iteratively using SGD or SDA with soft-thresholding. Finally, we explain why RDA is much more effective than Prox-SGD for obtaining sparsity, which motivates its use to a sparse training algorithm for deep learning.

3.1 SGD and SDA

Consider a classification problem. Let $z = (x, y)$ be an input-output pair of data, such as a picture and its corresponding label. Let w be weights in the model, and $f(w, z)$ be the loss function corresponding to z and w . Our aim is to solve the optimization problem

$$\min_w \left\{ \frac{1}{n} \sum_{z \in Z} f(w, z) \right\}, \quad (1)$$

where $Z = \{z_1, z_2, \dots, z_n\}$ is the dataset.

SGD is a commonly used algorithm for solving (1). The major step in SGD with mini-batch can be represented as:

$$w_{t+1} = w_t - \eta_t g_t, \quad (2)$$

with $g_t = \frac{1}{m} \sum_{z \in X_t} \nabla_w f(w_t, z)$ on mini-batch $X_t \subset Z$. In another form, w_{t+1} can be interpreted as follows

$$w_{t+1} = \arg \min_w \left\{ g_t^T w + \frac{1}{2\eta_t} \|w - w_t\|_2^2 \right\}. \quad (3)$$

Intuitively, the empirical loss function in (1) is replaced with its first-order approximation, then we have $g_t^T w$ on X_t . And regularization term $\frac{1}{2} \|w - w_t\|_2^2$, which uses w_t as moving proximal center, is added to control the distance between w_{t+1} and w_t . Generally speaking, for convex functions, η_t can be taken as $\frac{1}{\alpha\sqrt{t}}$ in [31] with hyperparameter α . And in real application of CNN, we use the strategy as discussed in Sect. 5.2.

SDA can be understood as solving a different subproblem at each time step with respect to the SGD form in (3). As shown in [32], SDA is primal-dual type method since, it generates a feasible approximation to the optimum of an appropriately formulated dual problem. Specifically, the update scheme of SDA we consider here is

$$w_{t+1} = \arg \min_w \left\{ \bar{g}_t^T w + \frac{1}{2\xi_t} \|w - w_c\|_2^2 \right\}. \quad (4)$$

where $\bar{g}_t = \frac{1}{t} \sum_{\tau=1}^t g_\tau$. Unlike SGD, the original loss function in (1) is approximated by $\frac{1}{t} \sum_{\tau=1}^t g_\tau^T w$, a linear function obtained by averaging all previous stochastic gradient g_τ . This sequence corresponds to the support functions $g_\tau^T w$ in the dual space. Also, the second term establishes a dynamically updated scale between the primal and dual spaces. The regularization term $\frac{1}{2} \|w - w_c\|_2^2$ is strongly convex and uses w_c as fixed proximal center, which is different from SGD. According to RDA in [38], $w_c = 0$ if we apply the ℓ_1 regularization term. Thus, we will take $w_c = 0$ in the rest of our paper. And $\{\xi_t\}$ is a nonnegative and nondecreasing sequence which determines the convergence rate. Here, following the idea in [32], ξ_t is chosen to be $\frac{\sqrt{t}}{\alpha}$.

Originally, the SDA method was designed for solving convex optimization problems because it was first inspired by convex combination of linear functions. Some comparison and connections between SDA and SGD are discussed in [27, 28]. We also show the underlying relation between SGD and SDA with a concise lemma below.

Lemma 1 *The SDA method is equivalent to the following perturbed SGD method:*

$$w_{t+1} = (1 - \epsilon_t)w_t - \gamma_t g_t \quad (5)$$

where

$$\epsilon_t = \frac{1}{t + \sqrt{t^2 - t}} < \frac{1}{2t - 1},$$

$$\text{and } \gamma_t = \frac{\xi_t}{t} = \frac{1}{\alpha\sqrt{t}}.$$

Proof The update scheme of (4) can be rewritten as

$$\begin{aligned}
w_{t+1} &= -\xi_t \bar{g}_t \quad (w_c = 0 \text{ in (4)}) \\
&= -\frac{\xi_t}{t} \sum_{\tau=1}^t g_\tau \\
&= -\gamma_t \sum_{\tau=1}^t g_\tau.
\end{aligned} \tag{6}$$

Then SDA can be expanded recursively as

$$\begin{aligned}
w_{t+1} &= -\gamma_t \sum_{\tau=1}^t g_\tau \\
&= -\gamma_t \left(\sum_{\tau=1}^{t-1} g_\tau + g_t \right) \\
&= \frac{\gamma_t}{\gamma_{t-1}} \left(-\gamma_{t-1} \sum_{\tau=1}^{t-1} g_\tau \right) - \gamma_t g_t \\
&= \frac{\gamma_t}{\gamma_{t-1}} w_t - \gamma_t g_t \\
&= (1 - \epsilon_t) w_t - \gamma_t g_t,
\end{aligned}$$

where

$$\epsilon_t = \frac{1}{t + \sqrt{t^2 - t}} < \frac{1}{2t - 1}. \tag{7}$$

This finishes the proof. $\square$

Thus SDA can be viewed as a perturbation of SGD, since, as either t is sufficiently large

$$1 - \epsilon_t = \frac{\gamma_t}{\gamma_{t-1}} = \sqrt{1 - \frac{1}{t}} \approx 1, \tag{8}$$

and $\gamma_t = \frac{\xi_t}{t} = \frac{1}{\alpha\sqrt{t}}$. From the lemma above, SDA may potentially have similar efficiency with SGD in solving non-convex problems, even applied to deep learning fields.

Lemma 2 *Let w_t and $\tilde{w}_t$ be the sequences generated by SGD and SDA respectively. Then*

$$w_t - \tilde{w}_t \rightarrow 0$$

as $t \rightarrow \infty$, in some appropriate sense.

3.2 ℓ_1 regularization, sparsity and algorithms

A natural idea to obtain a sparse CNN model is to add an ℓ_1 regularization term to the loss function, which is a well-known technique in compressed sensing [5]. In other words, we hope to achieve the sparsity by solving the following regularized problem

$$\min_w \left\{ \phi(w) = \frac{1}{n} \sum_{z \in \mathcal{Z}} f(w, z) + \lambda \|w\|_1 \right\}, \quad (9)$$

where λ is a hyper-parameter which controls the sparsity of solution. Despite the fact that there is no rigorous theory to prove the sparsity for the solution of such a complex model (9), numerical soft-thresholding introduced by the ℓ_1 norm may generate sparsity at the cost of accuracy. That is to say, an appropriate training algorithm with ℓ_1 regularization may achieve sparsity with acceptable accuracy. Naturally, we have the following two strategies for solving the above problem:

- *Prox-SGD* add the ℓ_1 regularization into (3), which will be discussed in Sect. 3.3.
- *RDA* add the ℓ_1 regularization into (4), which will be discussed in Sect. 3.4.

Before these two algorithms are introduced, the soft-thresholding operator related to ℓ_1 regularization defined as entry-wised form

$$(\text{soft}(x, \delta))^{(i)} = \text{sgn}(x^{(i)}) \max \{|x^{(i)}| - \delta, 0\}, \quad (10)$$

where i is the index of element. Numerically speaking, we can conclude from the definition of the soft-thresholding operator that the larger the parameter δ , the more sparse the solution we will be.

3.3 Prox-SGD: applying ℓ_1 directly to SGD

Adding the regularization term $\lambda \|w\|_1$ to subproblem (3) directly gives prox-SGD as:

$$w_{t+1} = \arg \min_w \left\{ g_t^T w + \frac{1}{2\eta_t} \|w - w_t\|_2^2 + \lambda \|w\|_1 \right\}. \quad (11)$$

With some simple induction, Prox-SGD can be written in the forward-backward splitting (FOBOS [6]) scheme

$$\begin{aligned} w_{t+\frac{1}{2}} &= w_t - \eta_t g_t, \\ w_{t+1} &= \arg \min_w \left\{ \frac{1}{2\eta_t} \|w - w_{t+\frac{1}{2}}\|_2^2 + \lambda \|w\|_1 \right\}, \end{aligned} \quad (12)$$

where the forward step is a single step of SGD, and the backward step is equivalent to a soft-thresholding operator working on $w_{t+\frac{1}{2}}$ with parameter $\eta_t \lambda$. The learning rate $\eta_t = \frac{1}{\alpha \sqrt{t}}$ to obtain reasonable convergence rate in convex problem.

3.4 RDA by Xiao: applying ℓ_1 in a different way

Regularized dual averaging (RDA) is originally designed for convex online learning and stochastic optimization problems [38]. However, RDA can also be understood as SDA with an additional ℓ_1 regularization. Based on the analysis in Lemma 1 connecting of SDA and SGD, and the success of SGD in non-convex optimization, we hope that RDA may also work for non-convex problems, especially for CNN models.

Similar to Prox-SGD, RDA is obtained from adding $\lambda\|w\|_1$ to subproblem (4), and it also requires $w_c = 0$. The update scheme takes the form

$$w_{t+1} = \arg \min_w \left\{ \bar{g}_t^T w + \frac{1}{2\xi_t} \|w\|_2^2 + \lambda \|w\|_1 \right\}. \quad (13)$$

We can clearly see the underlying relation between Prox-SGD and SGD with soft-thresholding from the forward–backward splitting form. The following induction

$$\begin{aligned} w_{t+1} &= \arg \min_w \left\{ \bar{g}_t^T w + \frac{1}{2\xi_t} \|w\|_2^2 + \lambda \|w\|_1 \right\} \\ &= \arg \min_w \left\{ \frac{1}{2\xi_t} \|w + \xi_t \bar{g}_t\|_2^2 + \lambda \|w\|_1 \right\}, \end{aligned} \quad (14)$$

gives us the forward–backward splitting of RDA,

$$\begin{aligned} w_{t+\frac{1}{2}} &= -\xi_t \bar{g}_t, \\ w_{t+1} &= \arg \min_w \left\{ \frac{1}{2\xi_t} \|w - w_{t+\frac{1}{2}}\|_2^2 + \lambda \|w\|_1 \right\}, \end{aligned} \quad (15)$$

where $\xi_t = \frac{\sqrt{t}}{\alpha}$ to obtain the best convergence rate in the convex case [38], and α is hyper-parameter. From (15), one can see that the forward step is actually SDA's single step and the backward step is the soft-thresholding operator working on $w_{t+\frac{1}{2}}$ with the parameter $\lambda\xi_t = \frac{\lambda\sqrt{t}}{\alpha}$ as presented in (10).

The final algorithms of Prox-SGD and RDA for CNN with ℓ_1 regularization term can be found in Algorithms 1 and 2.

Algorithm 1 Prox-SGD (Directly applying ℓ_1 to SGD)

Input: a dataset Z and a loss function $\frac{1}{n} \sum_{z \in Z} f(w, z) + \lambda \|w\|_1$ where w is a vector of the weights.

Initialization: initialize w_0 with the standard method.

for $t = 1$ **to** T **do**

Select a mini-batch X_t from the dataset.

Compute $g_t = \frac{1}{m} \sum_{z \in X_t} \nabla_w f(w_t, z)$.

Update w_{t+1} with Prox-SGD in element-wised form:

$$w_{t+1}^{(i)} = \begin{cases} w_t^{(i)} - \eta_t(g_t^{(i)} + \lambda), & w_t^{(i)} - \eta_t g_t^{(i)} > \eta_t \lambda, \\ 0, & |w_t^{(i)} - \eta_t g_t^{(i)}| \leq \eta_t \lambda, \\ w_t^{(i)} - \eta_t(g_t^{(i)} - \lambda), & w_t^{(i)} - \eta_t g_t^{(i)} < -\eta_t \lambda, \end{cases} \quad (16)$$

where where i is the index of the elements.

end for

Algorithm 2 RDA (Applying ℓ_1 in a different way)

Input: a dataset Z and a loss function $\frac{1}{n} \sum_{z \in Z} f(w, z) + \lambda \|w\|_1$ where w is a vector of the weights.

Initialization: randomly choose w_1 as introduced in §4.1, and set $\bar{g}_0 = 0$.

for $t = 1$ **to** T **do**

Select a mini-batch X_t from the dataset.

Compute $g_t = \frac{1}{m} \sum_{z \in X_t} \nabla_w f(w_t, z)$.

Update

$$\bar{g}_t = \frac{t-1}{t} \bar{g}_{t-1} + \frac{1}{t} g_t.$$

Update w_{t+1} with RDA in element-wise form:

$$w_{t+1}^{(i)} = \begin{cases} -\xi_t(\bar{g}_t^{(i)} + \lambda), & \bar{g}_t^{(i)} < -\lambda, \\ 0, & |\bar{g}_t^{(i)}| \leq \lambda, \\ -\xi_t(\bar{g}_t^{(i)} - \lambda), & \bar{g}_t^{(i)} > \lambda. \end{cases} \quad (17)$$

where i is the index of the elements.

end for

3.5 Comparison of Prox-SGD and RDA

The soft-thresholding of Prox-SGD and RDA are quite different.

- In Algorithm 2, we have

$$w_{t+1}^{(i)} = 0, \text{ if } |\bar{g}_t^{(i)}| \leq \lambda, \quad (18)$$

where the criterion to zero out $w_{t+1}^{(i)}$ only depends on a constant λ .

- In Algorithm 1, we have

$$w_{t+1}^{(i)} = 0, \text{ if } \left| w_t^{(i)} - \eta_t g_t^{(i)} \right| \leq \eta_t \lambda, \quad (19)$$

where $\eta_t = \frac{1}{\alpha\sqrt{t}}$, thus the criterion in this case depends on $\frac{1}{\alpha\sqrt{t}}\lambda$, which approaches to 0 as t goes to infinity.

Considering that $w_t^{(i)} - \eta_t g_t^{(i)}$ will converges to certain point which many not be zero in (19), we cannot expect significant sparsity in Algorithm 1 since $\eta_t \lambda$ will approach to 0. However, the right hand term (thresholding value) in (18) will keep constant as in RDA, which may produce a better sparsity. Similar discussions can also be found in [38].

Furthermore, from the formulation of the regularized problem (9), one can see that there is a trade-off between the accuracy and the regularization term, which can be concluded as too large regularization term controlled by λ can weaken the effect of the loss function. In other words, increasing regularization term λ will decrease the accuracy of the model. Thus, it is necessary to make use of an algorithm which can produce a sparse solution with small λ . As our analysis above shows, RDA has a good balance between sparsity and accuracy.

4 Two techniques for RDA in CNN

In this section, we introduce two techniques, the initialization and the adaptive sparse retraining method. The first one is essential for RDA to work, and the second one gives much improvement to the results given by RDA.

4.1 Initialization

In the original paper [38], the theoretical analysis requires that $w_c = 0$ and $w_1 = \arg \min_w \|w\|_1 = 0$ as an initialization. Such an initialization is also shown to work very well numerically for convex problem studied in [38]. Let us examine now how this initialization technique would work for a typical CNN model, such as VGG [35], ResNet [13] which we will test in this paper. We note that a typical CNN model can be written as:

$$f(w;x) = S(Wf_{\text{CNN}}(\theta;x) + b), \quad (20)$$

where $S(y) = \text{Softmax}(y) := \left(\frac{e^{y_i}}{\sum e^{y_i}} \right)$ and $f_{\text{CNN}}(\theta;x)$ stands for the main CNN structure except for the fully connected layer with Softmax. Let $w = \{W, b, \theta\}$, where $\{W, b\}$ are the parameters for last fully connected layer with Softmax and θ represents all parameters in the main structure of CNN models. One simple but important observation is that all those CNN models satisfying the following property

$$f_{\text{CNN}}(0;x) = 0, \quad (21)$$

as long as the underly activation function satisfies

$$\sigma(0) = 0. \quad (22)$$

This property is satisfied by $\sigma(x) = \text{ReLU}(x) := \max\{0, x\}$. That is to say, if all parameters are chosen as zero, the output of the main structure of a general CNN model will always equal to zero. Thus, for a general CNN model $f(w; x)$ as in (20) with property (21), we will have

$$\begin{aligned} \left. \frac{\partial f^{(i)}(w; x)}{\partial W^{(j,k)}} \right|_{w=0} &= \frac{\partial}{\partial W^{(j,k)}} \left[s \left(\sum_p W^{(i,p)} f_{\text{CNN}}^{(p)}(0; x) + b^{(i)} \right) \right] \Big|_{w=0} \\ &= S'(b^{(i)}) \delta_{ij} f_{\text{CNN}}^{(k)}(0; x) = 0, \quad \forall i, j, k. \end{aligned} \quad (23)$$

That is to say

$$\left. \frac{\partial f(w; x)}{\partial W} \right|_{w=0} = 0. \quad (24)$$

Furthermore,

$$\begin{aligned} \left. \frac{\partial f^{(i)}(w; x)}{\partial \theta^{(j)}} \right|_{w=0} &= \frac{\partial}{\partial W^{(j,k)}} \left[s \left(\sum_p W^{(i,p)} f_{\text{CNN}}^{(p)}(\theta; x) + b^{(i)} \right) \right] \Big|_{w=0} \\ &= S'(b^{(i)}) \sum_p W^{(i,k)} \left. \frac{\partial f_{\text{CNN}}^{(p)}(\theta; x)}{\partial \theta^j} \right|_{w=0} = 0, \end{aligned} \quad (25)$$

for all i, j as $W = 0$. That indicates that

$$\left. \frac{\partial f(w; x)}{\partial \theta} \right|_{w=0} = 0 \quad (26)$$

Considering the observations (24) and (26) for zero initialization in CNN, we have the next proposition.

Proposition 1 *The RDA method with $w_1 = 0$ cannot converge for CNN with activation function σ satisfying $\sigma(0) = 0$.*

As a result, non-zero initialization is a necessary condition in all gradient-based training algorithm including RDA for CNN. Thus we propose to initialize w_1 via some random strategies as discussed later in this subsection. Actually, this modification will not influence the convergence of the algorithm. As proven in Theorem 1, the convergence rate for convex problems based on this modification is still $\mathcal{O}(\frac{1}{\sqrt{t}})$ when $\xi_t = \mathcal{O}(\sqrt{t})$.

Theorem 1 *Assume the loss function $f(w, z)$ in the problem (9) is convex and there exists an optimal solution w^* to the problem (9) with $\Psi(w) = \lambda \|w\|_1$ that satisfies*

$\frac{1}{2}\|w^\star\|_2^2 \leq D^2$ for some $D > 0$. In addition, we assume that we have the next bound for the randomly chosen w_1 :

$$\Psi(w_1) = \lambda\|w_1\|_1 \leq Q. \quad (27)$$

Let the sequences $\{w_t\}_{t \geq 1}$ be generated by Algorithm 2, and assume $\|g_t\|_2 \leq G$ for some constant G . Then the expected cost $\mathbf{E}\phi(\bar{w}_t)$ converges to $\phi^\star$ with rate $\mathcal{O}(\frac{1}{\sqrt{t}})$

$$\mathbf{E}\phi(\bar{w}_t) - \phi^\star = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right), \quad (28)$$

with $\bar{w}_t = \frac{1}{t} \sum_{\tau=1}^t w_\tau$ and $\phi^\star = \phi(w^\star)$.

By adding the extra assumption that for the bound of w_1 as in (27), we can then prove the above result by following Xiao's work in [38] with difference of an extra coefficient in $\mathcal{O}(\frac{1}{\sqrt{t}})$ which is related to Q . The only difference between the original RDA and RDA used in Algorithm 2 is that the former one takes $w_1 = \arg \min_w \|w\|_1 = 0$ as initialization whereas the latter one allows us to choose w_1 randomly. This is a small modification in algorithm and proof but it plays a crucial role in applying RDA to CNN as discussed in the beginning of this section.

In particular, when the activation function is ReLU, the weights in CNN are usually initialized with a uniform or a normal distribution [8, 12, 20]. For RDA, we propose to initialize the weights with a uniform distribution $\mathcal{U}(-b, b)$, where

$$b = \sqrt{\frac{s}{n}}. \quad (29)$$

For a convolutional layer, $n = k^2c$ is the size of the filter, where c is the number of input channels and k is the width of the filter. For a fully connected layer, n is the dimension of the input vector. In both cases, s is a scalar to increase the weights (e.g. [12] proposes to choose $s = 6$).

Since f is non-linear, the effect of initialization on g_1 , the gradient of w_1 , is not that clear. Assuming that f is a linear function, then g_1 is scaled in the same way as w_1 . Since with a thresholding (ignoring the initial learning rate $\eta_1 = 1$), g_1 becomes the value of w_2 , the initial value should not be too small, nor should it be too large because of the exploding gradient problem [33], as shown in Tables 1 and 2. Here we have the next definition for sparsity of CNN models:

$$\text{Sparsity} = \frac{\text{the number of zero weights}}{\text{the number of all weights}},$$

for all tables referred later.

Finally, we listed some good choices for s in Table 3.

Table 1 Different initialization scalars on ResNet-18, CIFAR-10 with RDA

$\sqrt{s}$	TOP-1	TOP-5	Sparsity
1, 2	10.00	50.00	N/A
3	85.52	99.24	0.98
4	86.72	99.45	0.97
5	90.03	99.44	0.95
10	90.67	89.50	0.94
100	91.41	99.58	0.84
1000	90.36	99.62	0.63
10,000	71.80	97.94	0.34
20,000	68.06	97.39	0.99

This table shows TOP-1 and TOP-5 accuracy on validation dataset. All models are trained for 120 epochs

(*TOP-1 accuracy* is the conventional accuracy, which means that the model answer (the one with the highest probability) must be exactly the expected answer. *TOP-5 accuracy* means that any of your model that gives 5 highest probability answers that must match the expected answer)

Table 2 Different initialization scalars on ResNet-18, CIFAR-100 with RDA

$\sqrt{s}$	TOP-1	TOP-5	Sparsity
1	63.67	87.85	0.91
2	66.90	88.53	0.60
5	65.47	88.09	0.60
10	65.54	88.21	0.42
15	64.22	87.53	0.43
25	63.06	88.10	0.50
30	62.75	86.80	0.42
50	64.48	87.14	0.38
100	60.00	86.14	0.36

This table shows TOP-1 and TOP-5 accuracy on validation dataset. All models are trained for 120 epochs

Table 3 Suitable $\sqrt{s}$ for different models and datasets

Dataset	Model	$\sqrt{s}$
CIFAR-10	ResNet-18	10
	VGG-16bn	20
	VGG-19bn	10
CIFAR-100	ResNet-18	2
	VGG-16bn	60
	VGG-19bn	40
ImageNet	ResNet-18	2

ImageNet represents ILSVRC2012

Table 4 ASR helps improve both the sparsity and the accuracy

Model	RDA		RDA (ASR)	
	TOP-1	Sparsity	TOP-1	Sparsity
ResNet-18	91.34	0.87	93.47	0.95
VGG-16bn	93.07	0.92	93.24	0.94
VGG-19bn	92.65	0.74	93.02	0.90

This table shows TOP-1 accuracy on validation dataset, and the sparsity of weights. The dataset is CIFAR-10

Table 5 Compare RDA and prox-SGD for ResNet-18 on CIFAR-10 with 120 epochs

Method	TOP-1	TOP-5	λ	α	Sparsity
prox-SGD	89.80	99.40	10^{-5}	0.8	0.03
RDA	91.41	99.69	10^{-6}	1.0	0.84

RDA achieves better accuracy and sparsity

4.2 Adaptive sparse retraining (ASR)

Fine-tuning is a widely used technique that retrains a pruned model, since the pruning method often decreases the accuracy. This is equivalent to fix the weights to be pruned to zero in the original model, and only update the remaining weights.

During our retraining step, we fix the zero weights and update the remaining weights. If there are newly trained zero weights, they will also be fixed. Thus, in retraining, once a weight becomes zero, it will never be updated. This can be viewed as a stronger fine-tuning, and we call this method adaptive sparse retraining, where the optimization method we use is the same as that used in the first phase. This technique helps improve both the accuracy and sparsity of a model, as shown in Table 4.

5 Experiments

In this section, we compare the results of RDA and other methods. All results of RDA are based on the two techniques introduced in Sect. 4. All accuracies are of the validation dataset. The implementation is listed as follows.

All experiments are carried out with PyTorch (pytorch.org) on TITAN V GPU. For Prox-SGD, we use the same strategy with SGD for initialization and take learning rate as in Algorithm 1. The total epoch number for Prox-SGD as reported in Table 5 is 120. This number of epochs is reasonable because, first, the accuracy reaches the highest point in the end, and second, due to the decreasing threshold of Prox-SGD, there should not be too many training epochs, otherwise there will be no sparsity in the end as discussed in Sect. 3.5.

For RDA, filters, and weights as well as bias in fully connected layers, are initialized with uniform distribution introduced in Sect. 4.1. Weights in batch



Fig. 1 An example of the first 120 epochs of loss and accuracy curves for different methods, and the sparsity curve of RDA on ResNet-18, CIFAR-10

normalization are initialized with default settings in PyTorch (the mean is set to a 0-vector, and the variant is set to a 1-vector). In all experiments, the training mini-batch size is 128¹. Models are all first trained by RDA for 2400 epochs, and then RDA with ASR for 1200 epochs. Furthermore, in Sect. 5.2, we have reduced the number of epochs to 300 on CIFAR-10 and CIFAR-100 with ResNet-18 by tuning the parameter α and λ .

ResNet-18 is based on [13]. VGG-16bn and VGG-19bn are based on [35], and both are implemented with batch normalization.

5.1 Numerical results

We first compare RDA and prox-SGD for ResNet-18 on CIFAR-10 with both 120 epochs as shown in Table 5. One can see that RDA performs much better than prox-SGD, and achieves a sparsity of 84%, which can be further improved to 95%

¹ In the original paper [38], RDA is proposed as an online learning algorithm, which takes one input at each time.

Table 6 RDA on different models and CIFAR-10

MODEL	TOP-1	TOP-5	λ	α	Sparsity
ResNet-18	93.47	99.69	10^{-6}	1.0	0.95
VGG-16bn	93.24	99.52	10^{-6}	1.0	0.94
VGG-19bn	93.02	99.34	10^{-5}	1.0	0.90

RDA works well on different CNN models

Table 7 RDA on ResNet-18 and different datasets

Dataset	TOP-1	TOP-5	λ	α	Sparsity
MNIST	99.63	100.00	10^{-6}	0.1	0.95
CIFAR-10	93.47	99.69	10^{-6}	1.0	0.95
CIFAR-100	72.29	89.94	10^{-8}	0.09	0.56
ImageNet	64.93	84.92	10^{-8}	0.1	0.36

RDA works well on different datasets

Table 8 To compare RDA with the three-step pipeline in [10], we adapt the implementation in [24] where the model is first trained by SGD with 160 epochs, then pruned according to the sparsity, and finally fine tuned with 40 epochs to retrieve the performance (denoted as Model (Han))

Dataset	Model	TOP-1	Sparsity
CIFAR-10	ResNet-18 (RDA)	93.47	0.95
	ResNet-18 (Han)	93.95	0.95
	VGG-16bn (RDA)	93.24	0.94
	VGG-16bn (Han)	93.55	0.94
	VGG-19bn (RDA)	93.02	0.90
	VGG-19bn (Han)	93.60	0.90
CIFAR-100	ResNet-18 (RDA)	72.29	0.56
	ResNet-18 (Han)	74.67	0.56
	VGG-16bn (RDA)	69.04	0.67
	VGG-16bn (Han)	73.56	0.67
	VGG-19bn (RDA)	67.46	0.48
	VGG-19bn (Han)	72.52	0.48

The results of RDA are comparable to [10]

by ASR, as explained in Sect. 4.2. We have analyzed why RDA could be better than prox-SGD in Sect. 3, and the experiments support our claim. Figure 1 depicts an example of the sparsity changing during training.

For RDA itself, we show the results on ResNet-18, VGG-16bn and VGG-19bn, CIFAR-10 in Table 6. One can see that RDA performs well on all models tested. In Sect. 3, we have shown that SDA is a perturbation of SGD, and based on SDA, RDA keeps its general optimization ability on different models.

Table 7 shows the results of ResNet-18 on CIFAR-10, CIFAR-100 and ImageNet (ILSVRC2012). In general, RDA performs well on different datasets. For ImageNet, the typical accuracy of SGD for ResNet should be around 69%. In

Table 9 RDA with adaptive α helps speed up the training process

Epochs	α	λ	TOP-1	Sparsity
[1, 100]	1	10^{-5}	92.19	0.9257
[101, 200]	0.2	10^{-5}	93.38	0.9422
[201, 300]	0.05	10^{-5}	93.13	0.9645

The dataset is CIFAR-10 and the network is ResNet-18

Table 10 RDA with adaptive α helps speed up the training process. The dataset is CIFAR-100 and the network is ResNet-18

Epochs	α	λ	TOP-1	Sparsity
[1, 100]	0.28	10^{-6}	68.2	0.7455
[101, 200]	0.21	10^{-7}	71.25	0.6842
[201, 300]	0.08	10^{-6}	72.67	0.7782

some sense, ResNet-18 could lack the redundancy to be sparse while maintaining satisfactory accuracy. A larger model may help improve the performance.

Table 8 compares RDA with the three-step pipeline in [10], based on the source code provided by [24]. Han et al. [10] proposes the pipeline to compress CNN models, where the first step is training a model, the second is pruning a given percentage of weights in the trained model, and the third is fine tuning it. We compare the two methods based on the same sparsity, i.e. if a model trained by RDA has sparsity 0.95, then the model trained by SGD will be pruned 95% weights and then fine tuned. One can see that the results of RDA are comparable to [10]. This shows RDA is a powerful sparse optimization method for CNN.

5.2 Additional heuristic techniques

The numerical results presented in the previous subsections show that RDA works well in CNN. Next, we present some heuristic techniques that help improve the performance of RDA. In training algorithms like SGD and RDA, when the iteration step t gets large, the learning rate becomes too small to lead to any significant update of the weights at each step. In order to solve this problem, we developed some heuristic strategy for parameter turning for RDA. For example, we modify the parameter α and λ after training appropriate number of epochs, which can speed up the training process significantly according to our investigation on CIFAR-10 and CIFAR-100 with ResNet-18 shown in Tables 9 and 10.

By trial and error, $\{10^{-5}, 10^{-6}, 10^{-7}\}$ is a suitable search space for the parameter λ , and for the parameter α , it should be decreasing during the training process. These numerical experiments reveal the possibility to speed up and improve RDA with suitable adaptive parameter strategies. How to automatically find a proper adaptive parameter on different datasets and networks by theoretical analysis and more parameters tuning is still under further investigation.

6 Concluding remarks

In contrary to the common perception that the SDA method of [32] should only work for convex optimization problem for which the SDA was originally designed, in this paper, we manage to make this method as an effective training algorithm for the highly non-convex CNN. In particular, by combining it with ℓ_1 regularization, we develop the corresponding RDA method that proves to be very effective to obtain sparse CNN models without compromising generalization accuracy. The theoretical foundation of this approach is based on a critical observation we make, namely the SDA method (with a slight modification) is equivalent to a small perturbation of the SGD method if the learning rate is chosen appropriately. While our work is motivated by [38] for convex optimization problem, we find that the effectiveness of our RDA method depend crucially on proper initialization and adaptive sparse retraining. Preliminary numerical experiments show that our new method can be used to train sparse CNN with performances comparable to the state-of-the-art weight pruning methods [10]. We further provide theoretical justification of this method for convex optimization problems and analysis of the effectiveness of different choices of hyper-parameters in the algorithm.

Acknowledgements This work was partially supported by the Penn State and Peking University Joint Center for Computational Mathematics and Applications, the Beijing International Center for Mathematical Research from Peking University, and the Verne M. William Professorship Fund from Penn State University. The research of L. Zhao and L. Zhang was also supported by the China Scholarship Council (for visiting Penn State) and by HKUST16301218 Hong Kong RGC Competitive Earmarked Research Grant (for visiting Penn State), respectively. The authors wish to thank Drs. Lin Xiao and Liang Yang for helpful suggestions and discussions.

References

1. Alvarez, J.M., Salzmann, M.: Learning the number of neurons in deep networks. In: Advances in Neural Information Processing Systems, pp. 2270–2278 (2016)
2. Bertsekas, D.P.: Incremental proximal methods for large scale convex optimization. *Math. Program.* **129**, 163 (2011)
3. Candès, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory* **52**(2), 489–509 (2006)
4. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks (2017). arXiv preprint [arXiv:1710.09282](https://arxiv.org/abs/1710.09282)
5. Donoho, D.L.: Compressed sensing. *IEEE Trans. Inf. Theory* **52**(4), 1289–1306 (2006)
6. Duchi, J., Singer, Y.: Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.* **10**(Dec), 2899–2934 (2009)
7. Eldar, Y.C., Kutyniok, G.: Compressed Sensing: Theory and Applications. Cambridge University Press, Cambridge (2012)
8. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
9. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding (2015). arXiv preprint [arXiv:1510.00149](https://arxiv.org/abs/1510.00149)
10. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, pp. 1135–1143 (2015)

11. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: optimal brain surgeon. In: *Advances in Neural Information Processing Systems*, pp. 164–171 (1993)
12. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034 (2015)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
14. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: *International Conference on Computer Vision (ICCV)*, vol. 2 (2017)
15. Hu, H., Peng, R., Tai, Y.-W., Tang, C.-K.: Network trimming: a data-driven neuron pruning approach towards efficient deep architectures (2016). arXiv preprint [arXiv:1607.03250](https://arxiv.org/abs/1607.03250)
16. Huang, Z., Wang, N.: Data-driven sparse structure selection for deep neural networks (2017). arXiv preprint [arXiv:1707.01213](https://arxiv.org/abs/1707.01213)
17. Langford, J., Li, L., Zhang, T.: Sparse online learning via truncated gradient. *J. Mach. Learn. Res.* **10**(2), 777–801 (2009)
18. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436 (2015)
19. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: *Advances in Neural Information Processing Systems*, pp. 598–605 (1990)
20. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backprop. In: Montavon, G., Orr, G., Müller, K.R. (eds.) *Neural Networks: Tricks of the Trade*, pp. 9–48. Springer, Berlin (2012)
21. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets (2016). arXiv preprint [arXiv:1608.08710](https://arxiv.org/abs/1608.08710)
22. Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.* **16**(6), 964–979 (1979)
23. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2755–2763. IEEE (2017)
24. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning (2018). arXiv preprint [arXiv:1810.05270](https://arxiv.org/abs/1810.05270)
25. Luo, J.-H., Wu, J., Lin, W.: Thinet: a filter level pruning method for deep neural network compression (2017). arXiv preprint [arXiv:1707.06342](https://arxiv.org/abs/1707.06342)
26. Lustig, M., Donoho, D., Pauly, J.M.: Sparse MRI: the application of compressed sensing for rapid MR imaging. *Magn. Reson. Med. Off. J. Int. Soc. Magn. Reson. Med.* **58**(6), 1182–1195 (2007)
27. McMahan, B.: Follow-the-regularized-leader and mirror descent: equivalence theorems and 11 regularization. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 525–533 (2011)
28. McMahan, H.B.: A survey of algorithms and analysis for adaptive online learning. *J. Mach. Learn. Res.* **18**(1), 3117–3166 (2017)
29. Mine, H., Fukushima, M.: A minimization method for the sum of a convex function and a continuously differentiable function. *J. Optim. Theory Appl.* **33**(1), 9–23 (1981)
30. Mittal, D., Bhardwaj, S., Khapra, M.M., Ravindran, B.: Recovering from random pruning: on the plasticity of deep convolutional neural networks (2018)
31. Nemirovsky, A.S., Yudin, D.B.: *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York (1983)
32. Nesterov, Y.: Primal-dual subgradient methods for convex problems. *Math. Program.* **120**(1), 221–259 (2009)
33. Pascanu, R., Mikolov, T., Bengio, Y.: Understanding the exploding gradient problem (2012). CoRR [arXiv:abs/1211.5063](https://arxiv.org/abs/1211.5063)
34. Pratt, L.Y.: Comparing biases for minimal network construction with back-propagation. In: *International Conference on Neural Information Processing Systems*, pp. 177–185 (1988)
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations* (2015)
36. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: *Advances in Neural Information Processing Systems*, pp. 2074–2082 (2016)
37. Xiao, L.: Dual averaging method for regularized stochastic learning and online optimization. In: *Advances in Neural Information Processing Systems*, pp. 2116–2124 (2009)

38. Xiao, L.: Dual averaging methods for regularized stochastic learning and online optimization. *J. Mach. Learn. Res.* **11**(Oct), 2543–2596 (2010)
39. Zhu, M., Gupta, S.: To prune, or not to prune: exploring the efficacy of pruning for model compression (2017). arXiv preprint [arXiv:1710.01878](https://arxiv.org/abs/1710.01878)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.