# BALANCE IS ESSENCE: ACCELERATING SPARSE TRAINING VIA ADAPTIVE GRADIENT CORRECTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Despite impressive performance on a wide variety of tasks, deep neural networks require significant memory and computation costs, which prohibits their application in resource-constrained scenarios. Sparse training is one of the most common techniques to reduce these costs, however, the sparsity constraints add difficulty to the optimization, resulting in an increase in training time and instability. In this work, we aim to overcome this problem and achieve space-time co-efficiency. To accelerate and stabilize the convergence of sparse training, we analyze the gradient changes and develop an adaptive gradient correction method. Specifically, we approximate the correlation between the current and previous gradients, which is used to balance the two gradients to obtain a corrected gradient. Our method can be used with most popular sparse training pipelines under both standard and adversarial setups. Theoretically, we prove that our method can accelerate the convergence rate of sparse training. Extensive experiments on multiple datasets, model architectures, and sparsities demonstrate that our method outperforms leading sparse training methods by up to **5.0%** in accuracy given the same number of training epochs, and reduces the number of training epochs by up to **52.1%** to achieve the same accuracy.

## 1 INTRODUCTION

With the development of deep neural networks (DNNs), there is a trend towards larger and more intensive computational models to enhance task performance. Despite of the good performance, such large models consume a considerable amount of energy and produce a large amount of carbon footprint. It is also not applicable when memory or computational resources are limited. As a result, there are efforts in research to find more resource-efficient ways to train DNNs while maintaining results comparable to the state of the art (Yu & Li, 2021; Rock et al., 2021; Leite & Xiao, 2021).

Sparse training (Dettmers & Zettlemoyer, 2019; Ishikawa, 1996) is one of the most popular class of methods to improve efficiency from the spatial aspect and is receiving increasing attention. During sparse training, certain proportions of weights are set to zero to save space. The sparse patterns (which weights are zero) are updated iteratively with two popular training paradigms, namely pruning (Guo et al., 2016; Ullrich et al., 2017; Carreira-Perpinán & Idelbayev, 2018) and dynamic sparse training (Mocanu et al., 2018; Bellec et al., 2018; Frankle & Carbin, 2019; Mostafa & Wang, 2019; Dettmers & Zettlemoyer, 2019; Evci et al., 2020; Jayakumar et al., 2020; Liu et al., 2021; Özdenizci & Legenstein, 2021; Schwarz et al., 2021). Our goal is to find a sparse neural network with comparable or even higher performance compared to the dense model.

However, sparse training can bring some side effects to the training process, especially in the case of high sparsity (e.g. 99% weights are zero). First, sparsity can increase the variance of stochastic gradients, which will guide the model in a sub-optimal direction and hence slow convergence (Alistarh et al., 2017; Hoefler et al., 2021; Graesser et al., 2022). Second, it can also increase training instability (that is, a noisy trajectory of test accuracy w.r.t. iterations) (Bartoldson et al., 2020), which requires additional time to compensate for the decrease in model accuracy and further leads to slow convergence. In addition, the need to consider model robustness during sparse training is highlighted (Özdenizci & Legenstein, 2021). Therefore, the key question remains to be answered to apply sparse training widely: how to simultaneously improve convergence speed and stabilize sparse training in both standard and adversarial setups.

In this work, we propose an **a**daptive **g**radi**en**t correc**t**ion (AGENT) method to accelerate and stabilize sparse training under both standard and adversarial setups, which is compatible with most popular sparse training methods. Existing gradient correction methods, such as variance reduction, usually assume that previous gradients and current gradients are highly correlated, and therefore a large constant amount of previous gradients is added to correct the gradient. (Chen et al., 2019; Chatterji et al., 2018; Dubey et al., 2016). However, this assumption does not hold in sparse training, and we find that a balance between the current and previous gradients is critical. In our AGENT, we adaptively change the weights of the current and previous gradient based on their correlations to prevent adding too many low-correlation previous gradients. To more accurately approximate the full gradient, especially during the adversarial setup where additional bias can occur (see Section 4.2), we design a scaling parameter on the weights of the two gradients, which can further control the amount of previous gradients added and guarantee the balance and acceleration. Theoretically, we prove that our method can accelerate the convergence rate of sparse training. In contrast to previous efforts of sparse training acceleration which mainly focus on structured sparse patterns (Hubara et al., 2021; Chen et al., 2021), our method is compatible with both unstructured ans structured sparse training pipelines. Overall, our contributions can be summarized as follows:

- We develop an **a**daptive **g**radi**en**t correc**t**ion (AGENT) method for sparse training to achieve time efficiency and reduce training instability from an optimization perspective, which can be incorporated into any SGD-based sparse training pipeline.
- We analyze the gradient changes under sparsity constraints and multiply the previous gradients by an adaptive weight and a scaling parameter to achieve fine-grained control over the balance of current and previous gradients. In theory, we demonstrate that our method can accelerate the convergence rate of sparse training.
- We perform extensive experiments on multiple benchmark datasets, model architectures, and sparsities. In both standard and adversarial setups, our method improves the accuracy by up to **5.0%** given the same number of epochs and reduces the number of epochs up to **52.1%** to achieve the same performance compared to the leading sparse training methods.

## 2 Related Work

### 2.1 Sparse Training

Interest in sparse DNNs has been on the rise recently, especially when dealing with resource constraints. The goal is to achieve comparable performance with sparse weights to satisfy the constraints. An influential pipeline called pruning encourages many workflows (Guo et al., 2016; Ullrich et al., 2017; Carreira-Perpinán & Idelbayev, 2018). Recently, dynamic sparse training has emerged and been followed by various studies, where sparse weights are maintained in the training process. Various pruning and growth criteria are proposed, such as weight/gradient magnitude, random selection, and weight sign (Mocanu et al., 2018; Bellec et al., 2018; Frankle & Carbin, 2019; Mostafa & Wang, 2019; Dettmers & Zettlemoyer, 2019; Evci et al., 2020; Jayakumar et al., 2020; Liu et al., 2021; Özdenizci & Legenstein, 2021; Zhou et al., 2021b; Schwarz et al., 2021).

However, the aforementioned studies focus on improving the performance of sparse training, while neglecting the side effect of sparse training. Sparsity not only increases gradient variance, thus delaying convergence (Alistarh et al., 2017; Hoefler et al., 2021; Graesser et al., 2022), but also leads to training instability (Bartoldson et al., 2020). Additionally, sparse training can also exacerbate models' vulnerability to adversarial samples, which is a weaknesses of DNNs (Özdenizci & Legenstein, 2021). In this paper, we focus on dynamic sparse training. In general, our method can be applied to any SGD-based sparse training pipelines.

### 2.2 Accelerating Training

Studies have been conducted in recent years on how to achieve time efficiency in DNNs, and one popular direction is to obtain a more accurate gradient estimates to update the model (Gorbunov et al., 2020), such as variance reduction. SGD is the most common training method, where one uses small batches of data to approach the full gradient. In standard training, the batch estimator is unbiased, but can have a large variance and misguide the model, leading to studies on variance

reduction (Johnson & Zhang, 2013; Xiao & Zhang, 2014; Shang et al., 2018; Zou et al., 2018; Chen et al., 2019; Gorbunov et al., 2020). While adversarial training brings bias in the gradient estimation (Li et al., 2020), and we need to face the bias-variance tradeoff when doing gradient correction. A shared idea is to balance the gradient noise with a less-noisy old gradient (Nguyen et al., 2017; Fang et al., 2018; Chen et al., 2019). Some other momentum-based methods have a similar strategy of using old information (Cutkosky & Orabona, 2019; Chayti & Karimireddy, 2022) However, all the above work considers only the acceleration in non-sparse case.

Acceleration is more challenging in sparse training, and previous research on it has focused on structured sparse training (Hubara et al., 2021; Chen et al., 2021; Zhou et al., 2021a). First, sparse training will induce larger variance (Hoefler et al., 2021). In addition, some key assumptions associated with gradient correction methods do not hold under sparsity constraint. In the non-sparse case, the old and new gradients are assumed to be highly correlated, so we can collect a large amount of knowledge from the old gradients (Chen et al., 2019; Chatterji et al., 2018; Dubey et al., 2016). However, sparsity tends to lead to lower correlations, and this irrelevant information can be harmful, making previous methods no longer applicable to sparse training and requiring a finer balance between new and old gradients. Furthermore, the structured sparsity pattern is not flexible enough, which can lead to lower model accuracy. In contrast, our method accelerates sparse training from an optimization perspective and is compatible with both unstructured and structured sparse training pipelines.

## 3 Preliminaries: Stochastic Variance Reduced Gradient

Stochastic variance reduced gradient (SVRG) (Johnson & Zhang, 2013; Allen-Zhu & Hazan, 2016; Dubey et al., 2016) is a widely-used gradient correction method, which has been followed by many studies (Zou et al., 2018; Baker et al., 2019; Chen et al., 2019). Specifically, after each epoch of training, we evaluate the full gradients $\widetilde{\boldsymbol{g}}$ based on $\widetilde{\boldsymbol{\theta}}$ at that time and store them for later use. In the next epoch, the batch gradient estimate on $\mathbf{B}_t$ is updated using the stored old gradients via Eq. (1).

$$\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t) = \frac{1}{n}\sum_{i\in\mathbf{B}_t}\left(\boldsymbol{g}_i(\boldsymbol{\theta}_t) - \boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})\right) + \widetilde{\boldsymbol{g}} \tag{1}$$

where $\boldsymbol{g}_i(\boldsymbol{\theta}_t) = \nabla G(\mathbf{x}_i|\boldsymbol{\theta}_t)$, $G(\boldsymbol{\theta}_t) = (\sum_{i=1}^{N} G(\mathbf{x}_i|\boldsymbol{\theta}_t))/N$ is the loss function, $\widetilde{\boldsymbol{g}} = \nabla G(\widetilde{\boldsymbol{\theta}})$, $\boldsymbol{\theta}_t$ is the current parameters, $n$ is the number of samples in each mini-batch data, and $N$ is the total number of samples. SVRG successfully accelerates many training tasks in the non-sparse case, but does not work well in sparse training, which is similar to many other gradient correction methods.

## 4 Method

We propose an **a**daptive **g**radi**en**t correc**t**ion (AGENT) method and integrate it with recent sparse training pipelines to achieve accelerations and improve training stability. Specifically, to accomplish the goal, our AGENT filters out less relevant information and obtains a well-controlled and time-varying amount of knowledge from the old gradients. Our method overcomes the limitations of previous acceleration methods such as SVRG (Allen-Zhu & Hazan, 2016; Dubey et al., 2016; Elibol et al., 2020), and successfully accelerates and stabilizes sparse training. We will illustrate each part of our method in the following sections. Our AGENT method is outlined in Algorithm 1.

### 4.1 Adaptive Control over Old Gradients

In AGENT, we designed an adaptive addition of old gradients to new gradients to filter less relevant information and achieve a balance between new and old gradients. Specifically, we add an adaptive weight $c_t \in [0, 1]$ to the old gradient as shown in Eq. (2), where we use $\boldsymbol{g}_{\text{new}} = \frac{1}{n}\sum_{i\in\mathbf{B}_t}\boldsymbol{g}_i(\boldsymbol{\theta}_t)$ and $\boldsymbol{g}_{\text{old}} = \frac{1}{n}\sum_{i\in\mathbf{B}_t}\boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})$ to denote the gradient on current parameters $\boldsymbol{\theta}_t$ and previous parameters $\widetilde{\boldsymbol{\theta}}$ for a random subset $\mathbf{B}_t$, respectively. When the old and new gradients are highly correlated, we need a large $c$ to get more useful information from the old gradient. Conversely, when the relevance is low, we need a smaller $c$ so that we do not let irrelevant information corrupt the new gradient.

$$\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t) = \frac{1}{n}\sum_{i\in\mathbf{B}_t}\left(\boldsymbol{g}_i(\boldsymbol{\theta}_t) - c_t\cdot\boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})\right) + c_t\cdot\widetilde{\boldsymbol{g}} = \boldsymbol{g}_{\text{new}} - c_t\cdot\boldsymbol{g}_{\text{old}} + c_t\cdot\widetilde{\boldsymbol{g}}. \tag{2}$$

A suitable $c_t$ should effectively reduce the variance of $\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t)$. To understand how $c_t$ influence the variance of updated gradient, we decompose the variance of $\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t)$ in Eq. (3) with some abuse of notation, where the variance of updated gradient is a quadratic function of $c_t$. For simplicity, considering the case where $\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t)$ is a scalar, the optimal $c_t^*$ will be in the form of Eq. (3). As we can see, $c_t^*$ is not closed to 1 when the new gradient is not highly correlated with the old gradient. Since low correlation between $\boldsymbol{g}_{\text{new}}$ and $\boldsymbol{g}_{\text{old}}$ is more common in sparse training, directly setting $c_t = 1$ in previous methods is not appropriate and we need to estimate adaptive weights $c_t^*$. In support of this claim, we include a discussion and empirical analysis in the Appendix B.6 to demonstrate that as sparsity increases, the gradient changes faster, leading to lower correlations between $\boldsymbol{g}_{\text{new}}$ and $\boldsymbol{g}_{\text{old}}$.

$$\text{Var}(\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t)) = \text{Var}(\boldsymbol{g}_{\text{new}}) + c_t^2 \cdot \text{Var}(\boldsymbol{g}_{\text{old}}) - 2c_t \cdot \text{Cov}(\boldsymbol{g}_{\text{new}}, \boldsymbol{g}_{\text{old}}), \quad c_t^* = \frac{\text{Cov}(\boldsymbol{g}_{\text{new}}, \boldsymbol{g}_{\text{old}})}{\text{Var}(\boldsymbol{g}_{\text{old}})}. \tag{3}$$

We find it impractical to compute the exact $c_t^*$ and thus propose an approximation algorithm for it to obtain a balance between the new and old gradient. There are two challenges to calculate the exact $c_t^*$. On the one hand, to approach the exact value, we need to calculate the gradients on every batch data, which is too expensive to do it in each iteration. On the other hand, the gradients are often high-dimensional and the exact optimal $c_t^*$ will be different for different gradients. Thus, inspired by Deng et al. (2020), we design an approximation algorithm that makes good use of the loss information and leads to only a small increase in computational effort. More specifically, we estimate $c_t^*$ according to the changes of loss as shown in Eq. (4) and update $\widehat{c}_t^*$ adaptively before each epoch using momentum. Loss is a scalar, which makes it possible to estimate the shared correlation for all current and previous gradients. In addition, the loss is intuitively related to gradients and the correlation between losses can give us some insights into that of the gradients (some empirical analyses are included in the Appendix B.7).

**Algorithm 1** Adaptive Gradient Correction

**Input:** $\widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_0$, epoch length $m$, step size $\eta_t$, $c_0 = 0$, scaling parameter $\gamma$, smoothing factor $\alpha$
**for** $t = 0$ **to** $T-1$ **do**
  **if** $t \mod m = 0$ **then**
    $\widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_t$
    $\widetilde{\boldsymbol{g}} = (\sum_{i=1}^{N} \nabla G(\mathbf{x}_i|\widetilde{\boldsymbol{\theta}}))/N$
    **if** $t > 0$ **then**
      Calculate $\hat{c}_t^*$ via Eq. (4)
      $c_t = (1-\alpha)c_{t-1} + \alpha\hat{c}_t^*$
    **end if**
  **else**
    $c_t = c_{t-1}$
  **end if**
  Sample a mini-batch data $\mathbf{B}_t$ with size $n$
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \Big( \frac{1}{n} \sum_{i \in \mathbf{B}_t} \big( \boldsymbol{g}_i(\boldsymbol{\theta}_t) - \gamma c_t \cdot \boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}}) \big) + \gamma c_t \cdot \widetilde{\boldsymbol{g}} \Big)$
**end for**

$$\widehat{c}_t^* = \frac{\text{Cov}(G(\mathbf{B}|\boldsymbol{\theta}_t), G(\mathbf{B}|\widetilde{\boldsymbol{\theta}}))}{\text{Var}(G(\mathbf{B}|\widetilde{\boldsymbol{\theta}}))}, \tag{4}$$

where $\mathbf{B}$ denotes a subset of samples used to estimate the gradients.

### 4.2 Additional Scaling Parameter is Important

To guarantee successful acceleration in sparse and adversarial training, we further propose a scaling strategy that multiplies the estimated $c_t^*$ by a small scaling parameter $\gamma$. There are two main benefits of using a scaling parameter. First, the scaling parameter $\gamma$ can reduce the bias of the gradient estimates in adversarial training (Li et al., 2020). In standard training, the batch gradient estimator is an unbiased estimator of the full gradient. However, in adversarial training, we perturb the mini-batch of samples $\mathbf{B}_t$ into $\bar{\mathbf{B}}_t$. The old gradients $\boldsymbol{g}_{\text{old}}$ are calculated on batch data $\bar{\mathbf{B}}_t$, but the stored old gradients $\widetilde{\boldsymbol{g}}$ are obtained from the original data including $\mathbf{B}_t$, which makes $\mathbb{E}[\boldsymbol{g}_{\text{old}} - \widetilde{\boldsymbol{g}}]$ unequal to zero. Consequently, as shown in Eq. (5), the corrected estimator for full gradients will no longer be unbiased. It may have a small variance but a large bias, resulting in poor performance. Therefore, we propose a scaling parameter $\gamma$ between 0 and 1 to reduce the bias from

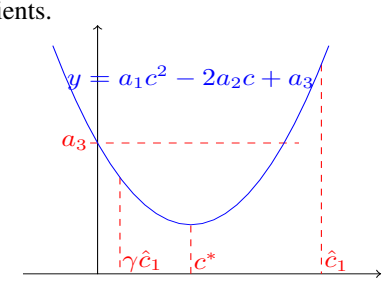



Figure 1: Illustration of how the scaling parameter $\gamma = 0.1$ ensures the acceleration in the face of worst-case estimate of $c_t^*$. The blue curve is a quadratic function, representing the relationship between $c_t$ and the variance. $c^*$ is the optimal value. $\hat{c}_1$ is a poor estimate making the variance larger than $a_3$ (variance in SGD). $\gamma\hat{c}_1$ can reduce the variance.

$c_t(\boldsymbol{g}_{\text{old}} - \widetilde{\boldsymbol{g}})$ to $\gamma c_t(\boldsymbol{g}_{\text{old}} - \widetilde{\boldsymbol{g}})$.

$$\mathbb{E}[\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t)] = \mathbb{E}[\boldsymbol{g}_{\text{new}} - c_t(\boldsymbol{g}_{\text{old}} - \widetilde{\boldsymbol{g}})] \neq \mathbb{E}[\boldsymbol{g}_{\text{old}}] = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{g}_i(\boldsymbol{\theta}_t). \tag{5}$$

Second, the scaling parameter $\gamma$ guarantees that the variance can still be reduced in the face of worst-case estimates of $c_t^*$ to accelerate the training. The key idea is illustrated in Figure 1, where x and y axis correspond to the weight $c_t$ and the gradient variance, respectively. The blue curve is a quadratic function that represents the relationship between $c_t$ and the variance. Suppose the true optimal is $c^*$, and we make an approximation to it. In the worst case, this approximation may be as bad as $\hat{c}_1$, making the variance even larger than $a_3$ (variance in SGD) and slowing down the training. Then, if we replace $\hat{c}_1$ with $\gamma\hat{c}_1$, we can reduce the variance and accelerate the training.

## 5 Theoretical Justification

Theoretically, we provide a convergence analysis for our AGENT and compare it to SVRG (Reddi et al., 2016). We use $G(.)$ to denote the loss function and $\boldsymbol{g}$ to denote the gradient. Our proof is based on Assumptions 1-2, and detailed derivation is included in Appendix A.

**Assumption 1.** *The loss function $G : \mathbb{R}^n \to \mathbb{R}$ is L-smooth, i.e., for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, it satisfies $||\nabla G(\boldsymbol{x}) - \nabla G(\boldsymbol{y})|| \leq L||\boldsymbol{x} - \boldsymbol{y}||$.*

**Assumption 2.** *The gradient of G is $\sigma$-bounded, i.e., $||\nabla G_i(\boldsymbol{x})|| \leq \sigma$ for all $i \in [N]$ and $\boldsymbol{x} \in \mathbb{R}^n$.*

**Our convergence analysis framework** outlines four steps:

- We first show that a proper selection of $c_t$ will lead to a smaller variance of gradient estimate.
- Next, we show the convergence rate of one arbitrary training epoch.
- We then extend the one-epoch results and analyze the convergence rate for the whole epoch.
- After obtaining the convergence rate, we bring it to the real case of sparse learning and find that our method indeed yields a tighter bound.

Given Assumptions 1-2, we follow the analysis framework above and establish Theorem 1 to show the convergence rate of our AGENT:

**Theorem 1.** *Under Assumptions 1-2, with proper choice of step size $\eta_t$ and $c_t$, the gradient $\mathbb{E}[||g(\boldsymbol{\theta}_{\boldsymbol{\pi}})||^2]$ using AGENT after $T$ training epochs can be bounded by:*

$$\mathbb{E}[||g(\boldsymbol{\theta}_{\boldsymbol{\pi}})||^2] \leq \frac{(G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*))LN^{\alpha}}{Tn\nu} + \frac{2\kappa\mu^2\sigma^2}{N^{\alpha}m\nu}$$

*where $\boldsymbol{\theta}_{\pi}$ is sampled uniformly from $\{\{\boldsymbol{\theta}_t^s\}_{t=0}^{m-1}\}_{s=0}^{T-1}$, $N$ denotes the data size, $n$ denotes the mini-batch size, $m$ denotes the epoch length, $\boldsymbol{\theta}_0$ is the initial point and $\boldsymbol{\theta}_*$ is the optimal solution, $\nu, \mu, \kappa, \alpha > 0$ are constants depending on $\eta_t$ and $c_t$, $N$ and $n$.*

In regard to Theorem 1, we make the following remarks to justify the acceleration from our AGENT:

**Remark 1.** *(Faster Gradient Change Speed) An influential difference between sparse and dense training is the gradient change speed, which is reflected in Assumption 1 (L-smooth). Typically, $L$ in sparse training will be larger than $L$ in dense training.*

**Remark 2.** *(First Term Analysis) In Theorem 1, the first term in the bound of our AGENT measures the error introduced by deviations from the optimal parameters, which goes to zero when the number of epochs $T$ reaches infinity. However, in real sparse training applications, $T$ is finite and this term is expanded due to the increase of $L$ in sparse training, which implies that the optimization under sparse constraints is more challenging.*

**Remark 3.** *(Second Term Analysis) In Theorem 1, the second term measures the error introduced by the noisy gradient and the finite data. Since $\sigma^2$ is relatively small and $N$ is large in DNNs training, the second term is negligible or much smaller compared to the first term when $T$ is finite.*

From the above analysis, we can compare the bounds of AGENT and SVRG and find that in the case of sparse training, an appropriate choice of $c_t$ can make the bound for our AGENT tighter than the bound for SVRG by well-corrected gradients.

**Remark 4.** *(Comparison with SVRG) Under Assumptions 1-2, the gradient* $\mathbb{E}[||g(\boldsymbol{\theta_\pi})||^2]$ *using SVRG after* $T$ *training epochs can be bounded by (Reddi et al., 2016):*

$$\mathbb{E}[||g(\boldsymbol{\theta_\pi})||^2] \leq \frac{(G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*))LN^\alpha}{Tn\nu^*}.$$

*This bound is of a similar form to the first term in Theorem 1. Since the second term of Theorem 1 is negligible or much smaller than the first one, we only need to compare the first term. With a proper choice of* $c_t$*, the variance of* $\hat{\boldsymbol{g}}(\boldsymbol{\theta}_t)$ *will decrease, which leads to a smaller* $\nu$ *for AGENT than* $\nu^*$ *for SVRG. Thus, AGENT can bring a smaller first term compared to SVRG, which indicates that AGENT effectively reduces the error due to the deviations and has a tighter bound compared to SVRG.*

## 6 EXPERIMENTS

We add our AGENT to three recent sparse training pipelines, namely BSR-Net (Özdenizci & Legenstein, 2021), RigL (Evci et al., 2020) and ITOP (Liu et al., 2021). BSR-Net is a new sparse training pipeline that updates connections by Bayesian sampling and also includes adversarial setups. RigL is also a popular dynamic sparse training paradigm which uses weight and gradient magnitudes to learn the connections. ITOP is another recent pipeline for dynamic sparse training, which uses sufficient and reliable parameter exploration to achieve in-time over-parameterization. Detailed information about the dataset, model architectures, and other training and evaluation setups is provided below.

**Datasets & Model Architectures:** The datasets we use include CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and ImageNet-2012 (see Appendix) (Russakovsky et al., 2015). For model architectures, we use VGG-16 (Simonyan & Zisserman, 2015), ResNet-18, ResNet-50 (He et al., 2016), and Wide-ResNet-28-4 (Zagoruyko & Komodakis, 2016).

**Training Settings:** For sparse training, we choose two sparsity levels, namely 90% and 99%. For BSR-Net, we consider both standard and adversarial setups. In RigL and ITOP, we focus on standard training. In standard training, we only use the original data to update the parameters. For adversarial part, we use the perturbed data with two popular objective (AT and TRADES) (Madry et al., 2018; Zhang et al., 2019; Özdenizci & Legenstein, 2021). Following Özdenizci & Legenstein (2021), we evaluate robust accuracy against PGD attacks with random starts using 50 iterations ($\text{PGD}^{50}$).

### 6.1 CONVERGENCE SPEED & STABILITY COMPARISONS

We compare the convergence speed by two criteria, including (a) **the test accuracy** at the same number of pass data (epoch) and (b) **the number of pass data (epoch)** required to achieve the same test accuracy, which is widely used to compare the speed of optimization algorithms (Allen-Zhu & Hazan, 2016; Chatterji et al., 2018; Zou et al., 2018; Cutkosky & Orabona, 2019).

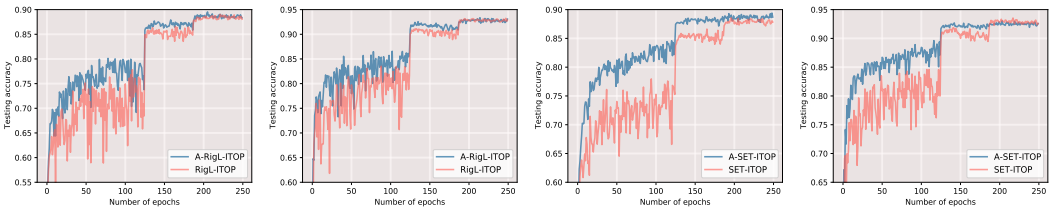
**For BSR-Net-based results using criterion (a)**, Table 1 lists the accuracies on both clean and adversarial samples after 20, 40, 70, 90, 140, and 200 epochs of training, where the higher accuracies are bolded. Sparse VGG-16 are learned on CIFAR-10 in both standard and adversarial sutups. For the standard setup, we only present the clean accuracy. As we can see, our method maintains higher clean and robust accuracies for almost all training epochs and setups which demonstrates the successful acceleration from our method. In particular, for limited time periods like 20 epochs, our A-BSR-Net usually shows dramatic improvements with clean accuracy as high as **11.4%**, indicating a significant reduction in early search time. In addition, considering the average accuracy improvement over the 6 time budgets, our method outperforms BSR-Net in accuracy by upto **5.0%**.

**For ITOP-based results using criterion (a)**, as shown in Figure 2, the blue curves (A-RigL-ITOP and A-SET-ITOP) are always higher than the red curves (RigL-ITOP and SET-ITOP), indicating faster training when using our AGENT. In addition, we can see that the red curves experience severe up and down fluctuations, especially in the early stages of training. In contrast, the blue curves are more stable all the settings, which indicates AGENT is effective in stabilizing the sparse training.

**For BSR-Net-based results using criterion (b)**, Figure 3 depicts the number of training epochs required to achieve certain accuracy. We can see that the blue curves (A-BSR-Net) are always lower

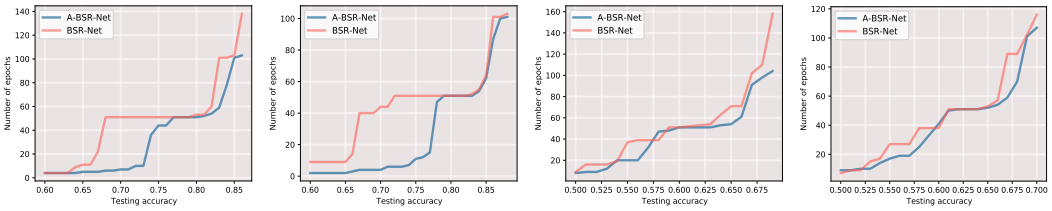
Table 1: Testing accuracy (%) of BSR-Net-based models. Sparse VGG-16 are learned in standard and adversarial setups. Results are presented as clean/robust accuracy (%). For the same number of training epochs, our method has higher accuracy compared to BSR-Net in almost all cases.

| | | 90% Sparsity | | 99% Sparsity | |
|---|---|---|---|---|---|
| | | BSR-Net | Ours | BSR-Net | Ours |
| AT | 20-th | 55.0 (1.59)/**38.2** | **63.6 (1.31)**/37.3 | 49.8 (1.46)/31.0 | **56.4 (1.39)/31.4** |
| | 40-th | 62.2 (1.88)/**39.2** | **64.9 (0.81)**/37.9 | 54.1 (1.72)/33.9 | **57.7 (0.39)/34.5** |
| | 70-th | 73.1 (0.39)/37.8 | **75.1 (0.27)/45.2** | 64.7 (0.30)/34.9 | **66.0 (0.23)/39.4** |
| | 90-th | 73.2 (0.29)/33.6 | **74.1 (0.25)/44.8** | 63.7 (0.25)/35.8 | **65.8 (0.24)/39.8** |
| | 140-th | 76.7 (0.27)/**46.5** | **77.4 (0.26)**/43.8 | 68.4 (0.20)/40.8 | **69.8 (0.14)/41.2** |
| | 200-th | 76.6 (0.25)/43.3 | **78.1 (0.24)/44.6** | 69.0 (0.15)/**42.2** | **70.7 (0.06)**/42.0 |
| TRADES | 20-th | 62.0 (0.82)/33.3 | **65.0 (0.61)/37.6** | 55.7 (0.76)/25.5 | **57.6 (0.45)/31.6** |
| | 40-th | 65.4 (0.97)/35.3 | **66.0 (0.34)/37.2** | **60.6 (0.69)**/28.9 | 58.4 (0.34)/**33.4** |
| | 70-th | 73.4 (0.52)/34.8 | **73.5 (0.33)/45.4** | 66.3 (0.35)/33.5 | **67.3 (0.30)/39.0** |
| | 90-th | 73.0 (0.36)/36.8 | **73.6 (0.28)/44.8** | 66.2 (0.33)/31.7 | **67.5 (0.24)/39.1** |
| | 140-th | 76.4 (0.25)/45.1 | **76.8 (0.25)/46.3** | **70.0 (0.29)**/38.2 | 69.9 (0.21)/**41.5** |
| | 200-th | 75.6 (0.23)/**47.2** | **77.0 (0.24)**/46.2 | 70.8 (0.19)/39.3 | **70.9 (0.25)/41.2** |
| Standard | 20-th | 70.4 (2.50)/0.0 | **81.8 (0.62)**/0.0 | 60.6 (1.26)/0.0 | **69.8 (1.45)**/0.0 |
| | 40-th | 77.6 (1.39)/0.0 | **82.4 (0.47)**/0.0 | 62.6 (2.47)/0.0 | **73.7 (0.36)**/0.0 |
| | 70-th | 86.8 (0.78)/0.0 | **89.7 (0.38)**/0.0 | 79.7 (0.72)/0.0 | **83.7 (0.24)**/0.0 |
| | 90-th | 87.6 (0.63)/0.0 | **89.3 (0.22)**/0.0 | 80.5 (0.55)/0.0 | **83.9 (0.42)**/0.0 |
| | 140-th | 91.7 (0.44)/0.0 | **92.5 (0.06)**/0.0 | 85.7 (0.42)/0.0 | **86.9 (0.07)**/0.0 |
| | 200-th | 91.8 (0.23)/0.0 | **92.6 (0.12)**/0.0 | 85.8 (0.12)/0.0 | **87.1 (0.25)**/0.0 |




(a) VGG-C(RigL) (b) ResNet-34(RigL) (c) VGG-C(SET) (d) ResNet-34(SET)

Figure 2: Testing accuracy for ITOP-based models at 99% sparsity on CIFAR-10. A-RigL-ITOP and A-SET-ITOP (blue curves) converge faster than RigL-ITOP and SET-ITOP (pink curves).



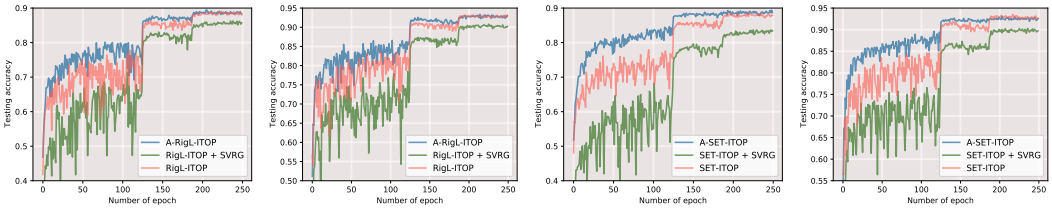

(a) VGG-16, Standard (b) WRN-28-4, Standard (c) VGG-16, AT (d) WRN-28-4, AT

Figure 3: Number of training epochs required to achieve the accuracy at 99% sparsity. Our A-BSR-Net (blue curves) need less time to achieve the accuracy compared to BSR-Net (pink curves).

than the red curves (BSR-Net), and on average our method reduces the number of training epochs by up to **52.1%**, indicating faster training when using our proposed A-BSR-Net.

### 6.2 Final Accuracy Comparisons

In addition, we compare the final accuracy after sufficient training. RigL-based results on CIFAR-10/100 are shown in Table 2. Our method A-RigL tends to be the best in almost all the scenarios.




(a) VGG-C (b) ResNet-34 (c) VGG-C (d) ResNet-34

Figure 4: Testing accuracy for ITOP-based models at 99% sparsity on CIFAR-10. SVRG (green curves) slows down the training. Our AGENT (blue curves) accelerates the training.

For BSR-Net-based results in Table 3, we compare our A-BSR-Net with BSR-Net on SVHN using VGG-16 and WideResNet-28-4 (WRN-28-4), and our method is often the best again. This shows that our AGENT can accelerate sparse training while maintaining or even improving the accuracy.

### 6.3 Comparison with Other Gradient Correction Methods

We also compare our AGENT with SVRG (Baker et al., 2019), a popular gradient correction method in the non-sparse case. The presented ITOP-based results are based on sparse (99%) VGG-C and ResNet-34 on CIFAR-10. Figure 4 (a)-(b) show the testing accuracy of A-RigL-ITOP (blue), RigL-ITOP (red), and RigL-ITOP+SVRG (green) at different epochs. We can see that the yellow curve for RigL-ITOP+SVRG is often lower than the other two curves, indicating that model convergence is slowed down by SVRG. As for the blue curve for our A-RigL-ITOP, it is always on the top of the green curve for RigL-ITOP, indicating a successful acceleration. The SET-ITOP-based results depicted in Figure 4 (c)-(d) show a similar pattern. This demonstrates that SVRG does not work for sparse training, while AGENT overcomes its limitations.

Table 2: Final accuracy (%) of RigL-based models. AGENT + RigL (A-RigL) maintains or even improves the accuracy.

| | | Dense | 90% | 99% |
|---|---|---|---|---|
| CIFAR-10 | A-RigL | **95.2 (0.24)** | **95.0 (0.21)** | **93.1 (0.25)** |
| | RigL | 95.0 (0.26) | 94.2 (0.22) | 92.5 (0.33) |
| CIFAR-100 | A-RigL | 72.9 (0.19) | **72.1 (0.20)** | **66.4 (0.14)** |
| | RigL | **73.1 (0.17)** | 71.6 (0.26) | 66.0 (0.19) |

### 6.4 Ablation Studies

We demonstrate the importance of each component in our framework by removing it one by one and comparing the results. Specifically, we consider examining the contribution of the time-varying weight $c_t$ of old gradients and the scaling parameter $\gamma$. The term "Fixed $c_t$" corresponds to fixing weight $c_t = 0.1$ during training, and "No $\gamma$" represents a direct use of $\hat{c}_t^*$ in Eq. (4) and the momentum scheme without adding the scaling parameter.

Table 3: Final accuracy (%) of BSR-Net-based models on SVHN with adversarial training objectives (TRADES). Our AGENT maintains or even improves the accuracy.

| | | BSR-Net | Ours |
|---|---|---|---|
| 90% | VGG-16 | 89.4 (0.29) | **94.4 (0.25)** |
| | WRN-28-4 | 92.8 (0.24) | **95.5 (0.23)** |
| 99% | VGG-16 | 86.4 (0.25) | **90.9 (0.26)** |
| | WRN-28-4 | 89.5 (0.22) | **92.2 (0.19)** |

Table 4 shows the clean and robust accuracies of standard and adversarial (AT or TRADES) training on CIFAR-10 using VGG-16 under different number of training epochs. In the adversarial training, we can see that "No $\gamma$" is poorly learned and has the worst results. While our method outperforms "Fix $c_t$" and "No $\gamma$" in almost all cases, especially in highly sparse tasks. For standard training, "No $\gamma$" can learn some information, but still performs worse than the other two methods. "Fix $c_t$" provides similar convergence speed as our method, while ours tends to have a better final score.

From the above discussion, both the adaptive update of $c_t$ and the multiplication of the scaling parameter $\gamma$ are important for the acceleration. On the one hand, the traditional way of setting $c_t = 1$ is not desirable in sparse training. Fixing it as a smaller value, such as 0.1, sometimes can work in standard training. But updating $c_t$ adaptively with loss-dependent information usually provides some benefits. These benefits become more significant in sparse and adversarial training which are more

Table 4: Ablation Studies: testing accuracy (%) comparisons with Fixed $c$ and No $\gamma$ on sparse VGG-16. Results are presented as clean/robust accuracy (%). For the same number of training epochs, our method has higher accuracy compared to Fixed $c$ and No $\gamma$ in almost all cases.

| | | 90% Sparsity | | | 99% Sparsity | | |
|---|---|---|---|---|---|---|---|
| | | Fixed $c_t$ | No $\gamma$ | Ours | Fixed $c_t$ | No $\gamma$ | Ours |
| AT | 20-th | 54.1/36.2 | 28.6/20.1 | **63.6/37.3** | 10.0/10.0 | 10.0/10.0 | **56.4/31.4** |
| | 40-th | 58.9/37.1 | 20.4/13.0 | **64.9/37.9** | 10.0/10.0 | 10.0/10.0 | **57.7/34.5** |
| | 70-th | 66.8/41.6 | 19.9/14.7 | **75.1/45.2** | 10.0/10.0 | 10.0/10.0 | **66.0/39.4** |
| | 90-th | 67.7/43.3 | 21.8/15.6 | **74.1/44.8** | 10.0/10.0 | 10.0/10.0 | **65.8/39.8** |
| | 140-th | 71.4/43.4 | 20.0/12.1 | **77.4/43.8** | 10.0/10.0 | 10.0/10.0 | **69.8/41.2** |
| | 200-th | 71.7/43.0 | 20.5/9.5 | **78.1/44.6** | 10.0/10.0 | 10.0/10.0 | **70.7/42.0** |
| TRADES | 20-th | 62.6/35.2 | 38.5/21.8 | **65.0/37.6** | 54.5/31.2 | 35.2/21.6 | **57.6/31.6** |
| | 40-th | 65.0/**38.0** | 34.7/20.2 | **66.0**/37.2 | 56.0/30.5 | 21.5/10.0 | **58.4/33.4** |
| | 70-th | **73.9**/44.5 | 28.8/18.4 | 73.5/**45.4** | 62.5/36.8 | 18.8/16.2 | **67.3/39.0** |
| | 90-th | **75.1**/44.4 | 25.8/15.9 | 73.6/**44.8** | 63.9/37.4 | 16.9/15.9 | **67.5/39.1** |
| | 140-th | 76.7/**46.5** | 28.6/14.1 | **76.8**/46.3 | 65.5/39.0 | 19.7/14.4 | **69.9/41.5** |
| | 200-th | 76.8/46.1 | 30.7/12.7 | **77.0/46.2** | 70.3/38.5 | 20.1/13.2 | **70.9/41.2** |
| Standard | 20-th | 80.9/0.0 | 70.6/0.0 | **81.8**/0.0 | **73.7**/0.0 | 51.8/0.0 | 69.8/0.0 |
| | 40-th | **83.3**/0.0 | 68.0/0.0 | 82.4/0.0 | **74.9**/0.0 | 55.2/0.0 | 73.7/0.0 |
| | 70-th | **90.2**/0.0 | 77.3/0.0 | 89.7/0.0 | **84.1**/0.0 | 65.9/0.0 | 83.7/0.0 |
| | 90-th | **89.8**/0.0 | 77.8/0.0 | 89.3/0.0 | 80.5/0.0 | 67.8/0.0 | **83.9**/0.0 |
| | 140-th | 92.4/0.0 | 80.7/0.0 | **92.5**/0.0 | **87.2**/0.0 | 71.9/0.0 | 86.9/0.0 |
| | 200-th | 92.1/0.0 | 78.6/0.0 | **92.6**/0.0 | 86.4/0.0 | 70.0/0.0 | **87.1**/0.0 |

challenging and of great value. On the other hand, we recommend adding a scaling parameter $\gamma$ (like 0.1) to $c_t$ to avoid increasing the variance and reduce the potential bias in adversarial training.

### 6.5 Scaling Parameter Setting

The choice of scaling parameters is important and can be seen as a hyper-parameter tuning process. We check different values from 0 to 1 and find that setting $\gamma = 0.1$ is a good choice to accelerate the training effectively. If setting $\gamma$ close to 1, we will not be able to completely avoid the increase in variance, which leads to performance drop, similar to "No $\gamma$" in Table 4. If $\gamma$ is set too small, such as 0.01, the weight of the old gradients will be too small and the old gradients will have limited influence on the model update, which will return to SGD's slowdown and training instability. More detailed experimental results using different scaling parameters $\gamma$ are included in the Appendix.

## 7 Discussion and Conclusion

We develop an adaptive gradient correction (AGENT) method for sparse training to achieve the time efficiency and reduce training instability from an optimization perspective, which can be incorporated into any SGD-based sparse training pipeline and work in both standard and adversarial setups. To achieve a fine-grained control over the balance of current and old gradients, we use loss information to analyze gradient changes, and add an adaptive weight on the old gradients. In addition, we design a scaling parameter to reduce the bias of the gradient estimator introduced by the adversarial samples and improve the worst case of the adaptive weight estimate. Experiment results on multiple datasets, model architectures, and sparsities demonstrate that our method outperforms state-of-the-art sparse training methods in terms of accuracy by up to **5.0%** and reduces the number of training epochs by up to **52.1%** for the same accuracy achieved.

A number of methods can be employed to reduce the FLOPs in our AGENT. Similar to SVRG, our AGENT increases the training FLOPs in each iteration due to the extra forward and backward used to compute the old gradients. To reduce the FLOPs, the first method is to use sparse gradients (Elibol et al., 2020), which effectively reduces the cost of backward in sparse training and can be easily applied to our method. The second method is parallel computing Allen-Zhu & Hazan (2016). Since the additional forward and backward over the old model parameters are fully parallelizable, we can view it as doubling the mini-batch size. Third, we can follow the idea of SAGA (Defazio et al., 2014) by storing gradients for each single sample. By this way, we do not need extra forward and backward steps, saving the computation. However, it requires extra memory to store the gradients.

## REFERENCES


Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30, 2017.

Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pp. 699–707. PMLR, 2016.

Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Control variates for stochastic gradient mcmc. *Statistics and Computing*, 29(3):599–615, 2019.

Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33: 20852–20864, 2020.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *International Conference on Learning Representations (ICLR)*, 2018.

Miguel A Carreira-Perpinán and Yerlan Idelbayev. "learning-compression" algorithms for neural net pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8532–8541, 2018.

Niladri Chatterji, Nicolas Flammarion, Yian Ma, Peter Bartlett, and Michael Jordan. On the theory of variance reduction for stochastic gradient monte carlo. In *International Conference on Machine Learning*, pp. 764–773. PMLR, 2018.

El Mahdi Chayti and Sai Praneeth Karimireddy. Optimization with access to auxiliary information. *arXiv preprint arXiv:2206.00395*, 2022.

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. Pixelated butterfly: Simple and efficient sparse training for neural network models. *arXiv preprint arXiv:2112.00029*, 2021.

Changyou Chen, Wenlin Wang, Yizhe Zhang, Qinliang Su, and Lawrence Carin. A convergence analysis for a class of practical variance-reduction stochastic gradient mcmc. *Science China Information Sciences*, 62(1):1–13, 2019.

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Wei Deng, Qi Feng, Georgios Karagiannis, Guang Lin, and Faming Liang. Accelerating convergence of replica exchange stochastic gradient mcmc via variance reduction. *arXiv preprint arXiv:2010.01084*, 2020.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Kumar Avinava Dubey, Sashank J Reddi, Sinead A Williamson, Barnabas Poczos, Alexander J Smola, and Eric P Xing. Variance reduction in stochastic gradient langevin dynamics. *Advances in neural information processing systems*, 29:1154–1162, 2016.

Melih Elibol, Lihua Lei, and Michael I Jordan. Variance reduction with sparse gradients. *arXiv preprint arXiv:2001.09623*, 2020.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in Neural Information Processing Systems*, 31, 2018.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pp. 680–690. PMLR, 2020.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 7766–7792. PMLR, 2022.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances In Neural Information Processing Systems*, pp. 1379–1387, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 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.

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

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in Neural Information Processing Systems*, 34, 2021.

Masumi Ishikawa. Structural learning with forgetting. *Neural networks*, 9(3):509–521, 1996.

Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

Clayton Frederick Souza Leite and Yu Xiao. Optimal sensor channel selection for resource-efficient deep activity recognition. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021)*, pp. 371–383, 2021.

Yan Li, Ethan Fang, Huan Xu, and Tuo Zhao. Implicit bias of gradient descent based adversarial training on separable data. In *International Conference on Learning Representations*, 2020.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pp. 6989–7000. PMLR, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *In International Conference on Learning Representations (ICLR)*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *In International Conference on Learn- ing Representations (ICLR)*, 2018.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pp. 4646–4655. PMLR, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pp. 2613–2621. PMLR, 2017.

Ozan Özdenizci and Robert Legenstein. Training adversarially robust sparse networks via bayesian connectivity sampling. In *International Conference on Machine Learning*, pp. 8314–8324. PMLR, 2021.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pp. 314–323. PMLR, 2016.

Johanna Rock, Wolfgang Roth, Mate Toth, Paul Meissner, and Franz Pernkopf. Resource-efficient deep neural networks for automotive radar interference mitigation. *IEEE Journal of Selected Topics in Signal Processing*, 15(4):927–940, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Jonathan Schwarz, Siddhant Jayakumar, Razvan Pascanu, Peter E Latham, and Yee Teh. Powerpropagation: A sparsity inducing weight reparameterisation. *Advances in Neural Information Processing Systems*, 34:28889–28903, 2021.

Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *arXiv preprint arXiv:2002.10509*, 2020.

Fanhua Shang, Kaiwen Zhou, Hongying Liu, James Cheng, Ivor W Tsang, Lijun Zhang, Dacheng Tao, and Licheng Jiao. Vr-sgd: A simple stochastic variance reduction method for machine learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):188–202, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Interna- tional Conference on Learning Representations (ICLR)*, 2015.

Varun Sundar and Rajat Vadiraj Dwaraknath. [reproducibility report] rigging the lottery: Making all tickets winners. *arXiv preprint arXiv:2103.15767*, 2021.

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *CoRR, abs/1702.04008*, 2017.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Rong Yu and Peichun Li. Toward resource-efficient federated learning in mobile edge computing. *IEEE Network*, 35(1):148–155, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *British Machine Vision Conference*, 2016.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482. PMLR, 2019.

Xiao Zhou, Weizhong Zhang, Zonghao Chen, Shizhe Diao, and Tong Zhang. Efficient neural network training via forward and backward propagation sparsification. *Advances in Neural Information Processing Systems*, 34:15216–15229, 2021a.

Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3599–3608, 2021b.

Difan Zou, Pan Xu, and Quanquan Gu. Subsampled stochastic variance-reduced gradient langevin dynamics. In *International Conference on Uncertainty in Artificial Intelligence*, 2018.

# A APPENDIX: THEORETICAL PROOF OF CONVERGENCE RATE

In this section, we provide a detailed proof for the convergence rate of our AGENT method. We start with some assumptions on which we will give some useful lemmas. Then, we will establish the convergence rate of our AGENT method based on these lemmas.

## A.1 ALGORITHM REFORMULATION

We reformulate our Adaptive Gradient Correction (AGENT) into a math-friendly version that is shown in Algorithm 2.

**Algorithm 2** Adaptive Gradient Correction

**Input:** Initialize $\boldsymbol{\theta}_0^0$ and $c_{-1} = 0$, set the number of epochs $S$, epoch length $m$, step sizes $h_t$, scaling parameter $\gamma$, and smoothing factor $\alpha$
**for** $s = 0$ **to** $S - 1$ **do**
  $\widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_0^s$
  $\widetilde{g} = (\sum_{i=1}^{N} \nabla G(\mathbf{x}_i; \widetilde{\boldsymbol{\theta}}))/N$
  Calculate $\widehat{c}_s^*$ via Eq. (4)
  $\widetilde{c}_s = (1-\alpha)\widetilde{c}_{s-1} + \alpha\widehat{c}_s^*$
  $c_s = \gamma\widetilde{c}_s$
  **for** $t = 0$ **to** $m - 1$ **do**
    Sample a mini-batch data $\mathbf{B}_t$ with size $n$
    $\boldsymbol{\theta}_{t+1}^s = \boldsymbol{\theta}_t^s - \eta_t\left(\frac{1}{n}\sum_{i\in\mathbf{B}_t}\left(g_i(\boldsymbol{\theta}_t^s) - c_s \cdot g_i(\widetilde{\boldsymbol{\theta}})\right) + c_s \cdot \widetilde{g}\right)$
  **end for**
  $\boldsymbol{\theta}_0^{s+1} = \boldsymbol{\theta}_m^s$
**end for**
**Output:** Iterates $\boldsymbol{\theta}_\pi$ chosen uniformly random from $\{\{\boldsymbol{\theta}_t^s\}_{t=0}^{m-1}\}_{s=0}^{S-1}$

## A.2 ASSUMPTIONS

**L-smooth**: A differentiable function $G : \mathbb{R}^n \to \mathbb{R}$ is said to be L-smooth if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is satisfies $||\nabla G(\boldsymbol{x}) - \nabla G(\boldsymbol{y})|| \leq L||\boldsymbol{x} - \boldsymbol{y}||$. And an equivalent definition is for. all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$:

$$-\frac{L}{2}||\boldsymbol{x} - \boldsymbol{y}||^2 \leq G(x) - G(y) - \langle \nabla G(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{y}\rangle \leq \frac{L}{2}||\boldsymbol{x} - \boldsymbol{y}||^2$$

**$\boldsymbol{\sigma}$-bounded**: We say function G has a $\sigma$-bounded gradient if $||\nabla G_i(\boldsymbol{x})|| \leq \sigma$ for all $i \in [N]$ and $\boldsymbol{x} \in \mathbb{R}^n$

## A.3 ANALYSIS FRAMEWORK

Under the above assumptions, we are ready to analyze the convergence rate of AGENT in **Algorithm 2**. To introduce the convergence analysis more clearly, we provide a brief analytical framework for our proof.

- First, we need to show that the variance of our gradient estimator is smaller than that of minibatch SVRG under proper choice of $c_s$. Since the gradient estimator of both AGENT and minibatch SVRG are unbiased estimators in standard training, we only need to show that our bound $E[||\mathbf{u}_t||^2]$ is smaller than minibatch SVRG. (See in Lemma 1)
- Based on above fact, we next apply the Lyapunov function to prove the convergence rate of AGENT in one arbitrary epoch. (See in Lemma 3)
- Then, we extend our previous results to the entire epoch (from 0 to $S$-th epoch) and derive the convergence rate of the output $\theta_\pi$ of **Algorithm 2**. (See in Lemma 4)

- Finally, we compare the convergence rate of our AGENT with that of minibatch SVRG. Setting the parameters in Lemma 4 according to the actual situation of sparse learning, we obtain a bound that is more stringent than minibatch SVRG.

### A.4 LEMMA

We first denote step length $\eta_t = N \cdot h_t$. Since we mainly focus on a single epoch, we drop the superscript $s$ and denote $\mathbf{u}_t = \frac{1}{n}\sum_{i\in\mathbf{B}_t}\left(g_i(\boldsymbol{\theta}_t) - c\cdot g_i(\widetilde{\boldsymbol{\theta}})\right) + c\cdot\widetilde{g}$ which is the gradient estimator in our algorithm and $\boldsymbol{\tau}_t = \frac{1}{n}\sum_{i\in\mathbf{B}_t}\left(g_i(\boldsymbol{\theta}_t) - c\cdot g_i(\widetilde{\boldsymbol{\theta}})\right)$, then lines the update procedure in **Algorithm 2** can be replaced with $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \cdot \mathbf{u}_t$

#### A.4.1 LEMMA 1

For the $\mathbf{u}_t$ defined above and function G is a L-smooth, $\lambda$ - strongly convex function with $\sigma$-bounded gradient, then we have the following results:

$$\mathbb{E}\left[||\mathbf{u}_t||^2\right] \leq 2\mathbb{E}\left[||g(\boldsymbol{\theta}_t)||^2\right] + \frac{4c^2L^2}{n}\mathbb{E}\left[||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right] + \frac{4(1-c)^2}{n}\sigma^2 \tag{6}$$

*Proof*:

$$\begin{aligned}
\mathbb{E}\left[||\mathbf{u}_t||^2\right] &= \mathbb{E}\left[||\boldsymbol{\tau}_t + c\cdot\widetilde{g}||^2\right] = \mathbb{E}\left[||\boldsymbol{\tau}_t + c\cdot\widetilde{g} - g(\boldsymbol{\theta}_t) + g(\boldsymbol{\theta}_t)||^2\right] \\
&\leq 2\mathbb{E}\left[||g(\boldsymbol{\theta}_t)||^2\right] + 2\mathbb{E}\left[||\boldsymbol{\tau}_t - \mathbb{E}(\boldsymbol{\tau}_t)||^2\right] \leq 2\mathbb{E}\left[||g(\boldsymbol{\theta}_t)||^2\right] + \frac{2}{n}\mathbb{E}\left[\boldsymbol{\tau}_t^2\right] \\
&= 2\mathbb{E}\left[||g(\boldsymbol{\theta})||^2\right] + \frac{2}{n}\mathbb{E}\left[||c(g_i(\boldsymbol{\theta}_t) - g_i(\tilde{\boldsymbol{\theta}})) + (1-c)g_i(\boldsymbol{\theta}_t)||^2\right] \\
&\leq 2\mathbb{E}\left[||g(\boldsymbol{\theta})||^2\right] + \frac{4}{n}\mathbb{E}\left[||c(g_i(\boldsymbol{\theta}_t) - g_i(\tilde{\boldsymbol{\theta}}))||^2\right] + \frac{4(1-c)^2}{n}\mathbb{E}\left[||g_i(\boldsymbol{\theta}_t)||^2\right] \\
&\leq 2\mathbb{E}\left[||g(\boldsymbol{\theta})||^2\right] + \frac{4c^2L^2}{n}\mathbb{E}\left[||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right] + \frac{4(1-c)^2}{n}\sigma^2
\end{aligned}$$

The first and third inequality are because $||a+b||^2 \leq 2||a||^2 + 2||b||^2$ , the second inequality follows the $\mathbb{E}\left[||\boldsymbol{\tau} - E\left[\boldsymbol{\tau}\right]||^2\right] \leq \mathbb{E}\left[||\boldsymbol{\tau}||^2\right]$ and the last inequality follows the L-smoothness and $\sigma$-bounded of function $G_i$.

**Remark 5.** *Compared with the gradient estimator of minibatch SVRG, the bound of* $\mathbb{E}[||\boldsymbol{u}_t||^2]$ *is smaller when L is large,* $\sigma$ *is relative small and* c *is properly chosen.*

#### A.4.2 LEMMA 2

$$\mathbb{E}\left[G(\boldsymbol{\theta}_{t+1})\right] \leq \mathbb{E}\left[G(\boldsymbol{\theta}_t) + \eta_t||g(\boldsymbol{\theta}_t)||^2 + \frac{L\eta^2}{2}||\mathbf{u}_t||^2\right] \tag{7}$$

*Proof*:

By the L-smoothness of function G, we have

$$\mathbb{E}\left[G(\boldsymbol{\theta}_{t+1})\right] \leq \mathbb{E}\left[G(\boldsymbol{\theta}_t) + \langle g(\boldsymbol{\theta}_t), \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\rangle + \frac{L}{2}||\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t||^2\right]$$

By the update procedure in algorithm 2 and unbiasedness, the right hand side can further upper bounded by

$$\mathbb{E}\left[G(\boldsymbol{\theta}_t) + \eta_t||g(\boldsymbol{\theta}_t)||^2 + \frac{L\eta_t^2}{2}||\mathbf{u}_t||^2\right]$$

### A.4.3 LEMMA 3

For $b_t, b_{t+1}, \zeta_t > 0$ and $b_t$ and $b_{t+1}$ have the following relationship

$$b_t = b_{t+1}(1 + \eta_t\zeta_t + \frac{4c^2\eta_t^2L^2}{n}) + 2\frac{c^2\eta_t^2L^3}{n}$$

and define

$$\Phi_t \coloneqq \eta_t - \frac{b_{t+1}\eta_t}{\zeta_t} - \eta_t^2L - 2b_{t+1}\eta_t^2$$

$$\Psi_t \coloneqq \mathbb{E}\left[G(\boldsymbol{\theta}_t) + b_t||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right] \tag{8}$$

$\eta_t$, $\zeta_t$ and $b_{t+1}$ can be chosen such that $\Phi_t > 0$.Then the $x_t$ in Algorithm 1 have the bound:

$$\mathbb{E}[||g(\boldsymbol{\theta}_t)||^2] \leq \frac{\Psi_t - \Psi_{t+1} + \frac{2(L\eta_t^2 + 2b_{t+1}\eta_t^2)(1-c)^2}{n}\sigma^2}{\Phi_t}$$

*Proof*:
We apply Lyapunov function

$$\Psi_t = \mathbb{E}\left[G(\boldsymbol{\theta}_t) + b_t||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right]$$

Then we need to bound $||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||$

$$\begin{aligned}
\mathbb{E}\left[||\boldsymbol{\theta}_{t+1} - \tilde{\boldsymbol{\theta}}||^2\right] &= \mathbb{E}\left[||\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t + \boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right] \\
&= \mathbb{E}\left[||\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t||^2 + ||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2 + 2\langle\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t, \boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}\rangle\right] \\
&= \mathbb{E}\left[\boldsymbol{\eta_t^2}||\mathbf{u}_t||^2 + ||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right] - 2\eta_t\mathbb{E}\left[\langle g(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}\rangle\right] \\
&\leq \mathbb{E}[\eta_t^2||\mathbf{u}_t^{s+1}||^2 + ||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2] + 2\eta_t\mathbb{E}\left[\frac{1}{2\zeta_t}||g(\boldsymbol{\theta}_t)|| + \frac{\zeta_t}{2}||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2\right]
\end{aligned} \tag{9}$$

The third equality due to the unbiasedness of the update and the last inequality follows Cauchy-Schwarz and Young's inequality. Plugging Equation (6), Equation (7) and Equation (9) into Equation (8), we can get the following bound:

$$\begin{aligned}
\Psi_{t+1} &\leq \mathbb{E}\left[G(\boldsymbol{\theta}_t)\right] + \left(b_{t+1}(1 + \eta_t\zeta_t + \frac{4c^2\eta_t^2L^2}{n}) + \frac{2c^2\eta_t^2L^3}{n}\right)\mathbb{E}[||\boldsymbol{\theta}_t - \tilde{\boldsymbol{\theta}}||^2] \\
&\quad - (\eta_t - \frac{b_{t+1}\eta_t}{\zeta_t} - L\eta_t^2 - 2b_{t+1}\eta_t^2)\mathbb{E}\left[||g(\boldsymbol{\theta}_t)||^2\right] + 4(\frac{L\eta_t^2}{2} + b_{t+1}\eta_t^2)\frac{(1-c)^2}{n}\sigma^2 \\
&= \Psi_t - (\eta_t - \frac{b_{t+1}\eta_t}{\zeta_t} - L\eta_t^2 - 2b_{t+1}\eta_t^2)\mathbb{E}\left[||g(\boldsymbol{\theta}_t)||^2\right] + 4(\frac{L\eta_t^2}{2} + b_{t+1}\eta_t^2)\frac{(1-c)^2}{n}\sigma^2
\end{aligned}$$

### A.4.4 LEMMA 4

Now we consider the effect of epoch and use $s$ to denote the epoch number. Let $b_m^s = 0$, $\eta_t^s = \eta$, $\zeta_t^s = \zeta$ and $b_t^s = b_{t+1}^s(1 + \eta\zeta + \frac{4c_s^2\eta L^2}{n}) + 2\frac{c_s^2\eta^2L^2}{n}$, $\Phi_t^s = \eta - \frac{b_{t+1}^s\eta}{\zeta_t} - \eta^2L - 2b_{t+1}^s\eta^2$ Define $\phi \coloneqq \min_{t,s}\Phi_t^s$. Then we can conclude that:

$$\mathbb{E}[||g(\boldsymbol{\theta_\pi})||^2] \leq \frac{G(\theta_0) - G(\theta_*)}{T\phi} + \sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \frac{2(L + 2b_{t+1}^s)(1-c_s)^2\eta^2\sigma^2}{Tn\phi}$$

*Proof*:

Under the condition of $\eta_t^s = \eta$, we apply telescoping sum on **Lemma 3**, then we will get:

$$\sum_{t=1}^{m-1} \mathbb{E}[||g(\boldsymbol{\theta}_t^s)||^2] \leq \frac{\Psi_0^s - \Psi_m^s}{\phi} + \sum_{t=0}^{m-1} \frac{2(L + 2b_{t+1}^s)(1-c_s)^2\eta^2\sigma^2}{n\phi}$$

From previous definition, we know $\Psi_0^s = G(\tilde{\boldsymbol{\theta}}^s)$, $\Psi_m^s = G(\tilde{\boldsymbol{\theta}}^{s+1})$ and plugging into previous equation, we obtain:

$$\sum_{t=1}^{m-1} \mathbb{E}[||g(\boldsymbol{\theta}_t^s)||^2] \leq \frac{G(\tilde{\boldsymbol{\theta}}^s) - G(\tilde{\boldsymbol{\theta}}^{s+1})}{\phi} + \sum_{t=0}^{m-1} \frac{2(L + 2b_{t+1}^s)(1-c_s)^2\eta^2\sigma^2}{n\phi}$$

Take summation over all the epochs and using the fact that $\tilde{\boldsymbol{\theta}}^0 = \theta_0$, $G(\tilde{\boldsymbol{\theta}}^S) \leq G(\boldsymbol{\theta}_*)$ we immediately obtain:

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=1}^{m-1} \mathbb{E}[||g(\boldsymbol{\theta}_t^s)||^2] \leq \frac{G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*)}{\phi} + \sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \frac{2(L + 2b_{t+1}^s)(1-c_s)^2\eta^2\sigma^2}{Tn\phi} \tag{10}$$

### A.5 THEOREM

#### A.5.1 THEOREM 1

Define $\xi_s = \sum_{t=0}^{m-1}(L + 2b_{t+1}^s)$ and $\xi := \min_s \xi_s$. Let $\eta = \frac{\mu n}{LN^\alpha} \quad (0 < \mu < 1) \quad and \quad (0 < \alpha \leq 1)$, $\zeta = \frac{L}{N^{\alpha/2}}$ and $m = \frac{N^{\frac{3\alpha}{2}}}{\mu n}$. Then there exists constant $\nu, \mu, \alpha, \kappa > 0$ such that $\phi \geq \frac{n\nu}{LN^\alpha}$ and $\xi \leq \kappa L$. Then $\mathbb{E}[||g(\boldsymbol{\theta_\pi})||^2]$ can be future bounded by:

$$\mathbb{E}[||g(\boldsymbol{\theta_\pi})||^2] \leq \frac{(G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*))LN^\alpha}{Tn\nu} + \frac{2\kappa\mu^2\sigma^2}{N^\alpha\nu m}$$

*Proof*:

By applying summation formula of geometric progression on the relation $b_t^s = b_{t+1}^s(1 + \eta_t\zeta_t + \frac{4c_s^2\eta_t^2L^2}{n}) + 2\frac{c_s^2\eta_t^2L^3}{n}$, we have $b_t^s = \frac{2c_s^2\eta^2L^3}{n}\frac{(1+\omega_s)^{m-t}-1}{\omega_s}$ where:

$$\omega_s = \eta\zeta + \frac{4c_s^2\eta^2L}{n} = \frac{\mu n}{N^{\frac{3\alpha}{2}}} + \frac{4c_s^2\mu^2 n}{N^{2\alpha}} \leq \frac{(4c_s^2+1)\mu n}{N^{\frac{3\alpha}{2}}}$$

This bound holds because $\mu \leq 1$ and $N \geq 1$ and thus $\frac{4c_s^2\mu^2 n}{N^{2\alpha}} = \frac{4c_s^2\mu n}{N^{\frac{3\alpha}{2}}} \times \frac{\mu}{N^{\frac{\alpha}{2}}} \leq \frac{4c_s^2\mu n}{N^{\frac{3\alpha}{2}}}$. And using this bound, we obtain:

$$
\begin{aligned}
b_0^s &= \frac{2\eta^2 c_s^2 L^3}{n}\frac{(1+\omega_s)^m-1}{\omega_s} = \frac{2\mu^2 n c_s^2 L}{N^{2\alpha}}\frac{(1+\omega_s)^m-1}{\omega_s} \\
&\le \frac{2\mu n c_s^2 L((1+\omega_s)^m-1)}{N^{\frac{\alpha}{2}}(4c_s+1)} \le \frac{2\mu n c_s^2 L((1+\frac{(4c_s^2+1)\mu n}{N^{\frac{3\alpha}{2}}})^{\frac{N^{\frac{3\alpha}{2}}}{\mu n}}-1)}{N^{\frac{\alpha}{2}}(4c_s^2+1)} \\
&\le \frac{2\mu n c_s^2 L(e^{\frac{1}{4c_s^2+1}}-1)}{N^{\frac{\alpha}{2}}(4c_s^2+1)}
\end{aligned}
$$

The last inequality holds because $(1+\frac{1}{x})^x$ is a monotone increasing function of $x$ when $x>0$. Thus $(1+\frac{(4c_s^2+1)\mu n}{N^{\frac{3\alpha}{2}}})^{\frac{N^{\frac{3\alpha}{2}}}{\mu n}} \le e^{\frac{1}{4c_s^2+1}}$ in the third inequality. And we can obtain the lower bound for $\phi$

$$
\phi = \min_{t,s}\Phi_t^s \ge \min_s(\eta - \frac{b_0^s\eta}{\zeta} - \eta^2 L - 2b_0^s\eta^2) \ge \frac{n\nu}{LN^\alpha}
$$

The first inequality holds since $b_s^t$ is a decrease function of $t$. Meanwhile, the second inequality holds because there exist uniform constant $\nu$ such that $\nu \ge \mu(1-\frac{b_0^s\eta}{\zeta} - L\eta - b_0^s\eta)$.

**Remark 6.** *In practice, $b_0^s \approx 0$ because both $\gamma$ and $c_s$ is both smaller than 0.1 which leads to $\mu(1-\frac{b_0^s}{\zeta} - L\eta - b_0^s\eta) \approx \mu(1-L\eta)$ and this value is usually much bigger than the $\nu^*$ in the bound of minibatch SVRG.*

We need to find the upper bound for $\xi$

$$
\begin{aligned}
\xi_s &= \sum_{t=0}^{m-1}(L+2b_{t+1}^s) = mL + 2\sum_{t=0}^{m-1} b_{t+1}^s \\
&= mL + 2\sum_{t=0}^{m-1}\frac{2c_s^2\eta^2L^3}{n}\frac{(1+\omega_s)^{m-t}-1}{\omega_s} \\
&= mL + \frac{2c_s^2\eta^2L^3}{n\omega_s}[\frac{(1+\omega_s)^{m+1}-(1+\omega_s)}{\omega_s} - m] \\
&\le mL + \frac{2c_s^2\eta^2L^3}{n}[\frac{1+\omega_s}{\omega_s^2}(e^{\frac{1}{4c_s^2+1}}-1) - m] \\
&\le mL + \frac{2c_s^2LN^\alpha}{n}(1+\frac{\mu n}{N^{3\alpha/2}})(e^{\frac{1}{4c_s^2+1}}-1) - \frac{2c_s^2\mu^2 nmL}{N^{2\alpha}} \\
&= L[(1-\frac{2c_s^2\mu^2 nL}{N^{2\alpha}})m + \frac{2c_s^2N^\alpha}{n}(1+\frac{\mu n}{N^{3\alpha/2}})(e^{\frac{1}{4c_s^2+1}}-1)]
\end{aligned}
$$

The reason why the first inequality holds is explained before and the second inequality holds because $\frac{1+x}{x^2}$ is a monotone decreasing function of $x$ when $x>0$, $\omega_s = \frac{\mu n}{N^{\frac{3\alpha}{2}}} + \frac{4c_s^2\mu^2 n}{N^{2\alpha}} \le \frac{\mu n}{N^{\frac{3\alpha}{2}}}$ and $\eta = \frac{\mu n}{LN^\alpha}$. Then $\xi = \max_s \xi_s \le \kappa L$ where $\kappa \ge \max_s((1-\frac{2c_s^2\mu^2 nL}{N^{2\alpha}})m + \frac{2c_s^2N^\alpha}{n}(1+\frac{\mu n}{N^{3\alpha/2}})(e^{\frac{1}{4c_s^2+1}}-1))$. When $c_s \approx 0$, $(1-\frac{2c_s^2\mu^2 nL}{N^{2\alpha}})m + \frac{2c_s^2N^\alpha}{n}(1+\frac{\mu n}{N^{3\alpha/2}})(e^{\frac{1}{4c_s^2+1}}-1) \approx m$.

Now we obtain the lower bound for $\phi$ and upper bound for $\xi$, plugging them into equation (10), we will have:

$$\begin{aligned}
\mathbb{E}[||g(\boldsymbol{\theta_\pi})||^2] &\leq \frac{G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*)}{\phi} + \sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\frac{2(L+2b_{t+1}^s)(1-c_s)^2\eta^2\sigma^2}{Tn\phi} \\
&\leq \frac{(G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*))LN^\alpha}{Tn\nu} + \sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\frac{2(L+2b_{t+1}^s)\eta^2\sigma^2}{Tn\phi} \\
&\leq \frac{(G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*))LN^\alpha}{Tn\nu} + \sum_{s=0}^{S-1}(\frac{2\eta^2\sigma^2}{Tn\phi})\sum_{t=0}^{m-1}(L+2b_{t+1}^s) \\
&\leq \frac{(G(\boldsymbol{\theta}_0) - G(\boldsymbol{\theta}_*))LN^\alpha}{Tn\nu} + \frac{2\kappa\mu^2\sigma^2}{N^\alpha\nu m}
\end{aligned}$$

### A.6 Real Case Analysis for Sparse Training

#### A.6.1 CIFAR-10/100 dataset

In our experiments, we apply both SVRG and AGENT on CIFAR-10 and CIFAR-100 dataset with $\eta = 0.1$, $\gamma = 0.1$, batch size $m = 128$ and in total 50000 training sample. Under this parameter setting, $\nu$ and $\nu^*$in **Theorem 1** and **Remark 4** are about 0.1 and 0.06, respectively. While $\frac{2\kappa\mu^2\sigma^2}{N^\alpha\nu m}$ is around $10^{-5}$ which is negligible so we know AGENT should have a tighter bound than SVRG in this situation which matches with the experimental results show in $Figure$ 6.

#### A.6.2 SVHN dataset

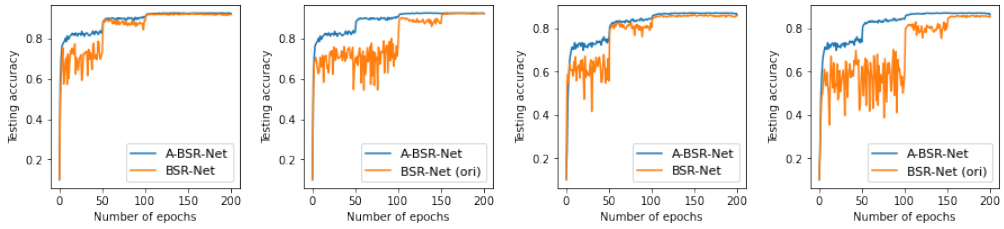
Meanwhile, in SVHN dataset, we train our model with parameters: $\eta = 0.1$, $\gamma = 0.1$, batch size $m = 573$ and sample size $N = 73257$. $\nu$, $\nu^*$ equal 0.4 and 0.06 respectively and $\frac{2\kappa\mu^2\sigma^2}{N^\alpha\nu m}$ is around $10^{-4}$. Although the second term in **Theorem 1** is bigger. Since $\nu$ here is a lot bigger than $\nu^*$ which lead to the first term in **Theorem 1** much smaller than that of **Remark 4**. So we still obtain a more stringent bound compared with SVRG which also meets with the outcome presented in $Figure$ 9.

# B ADDITIONAL EXPERIMENTAL RESULTS

We summarize additional experimental results for the BSR-Net-based Özdenizci & Legenstein (2021), RigL-based Evci et al. (2020), and ITOP-based Liu et al. (2021) models.

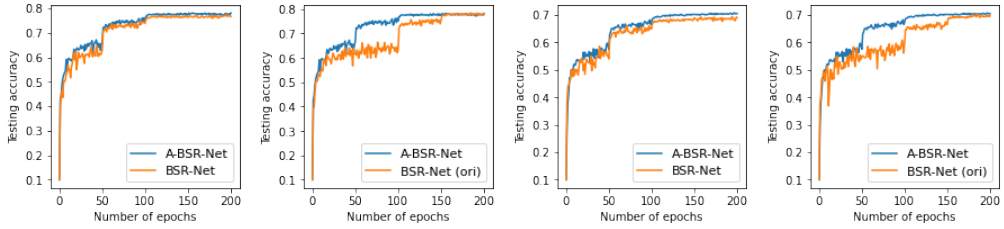
## B.1 ACCURACY COMPARISONS IN DIFFERENT EPOCHS

Aligned with the main manuscript, we compare the accuracy for a given number of epochs to compare both the speed of convergence and training stability. We first show BSR-Net-based results in this section. Since our approach has faster convergence and does not require a long warm-up period, the dividing points for the decay scheduler are set to the 50th and 100th epochs. In the manuscript, we also use this schedule for BSR-Net for an accurate comparison. In the Appendix, we include the results using its original schedule. BSR-Net and BSR-Net (ori) represent the results learned using our learning rate schedule and original schedule in Özdenizci & Legenstein (2021), respectively. As shown in Figures 5, 6, 7, 8, 9, 10, 11, the blue curves (A-BSR-Net) are always higher than the yellow curves and also much smoother than yellow curves (BSR-Net and BSR-Net (ori)), indicating faster and more stable training when using our proposed A-BSR-Net.



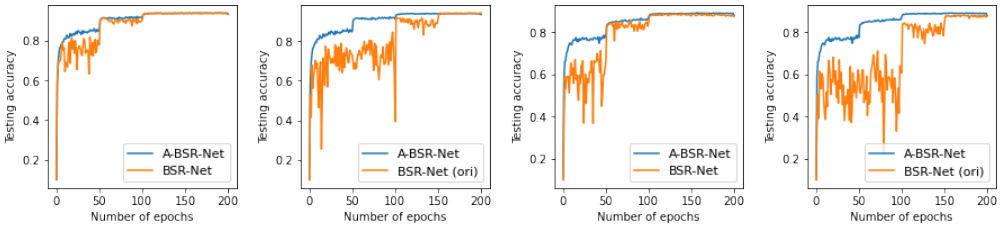

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 5: Comparisons (accuracy given the number of epochs) with BSR-Net Özdenizci & Legenstein (2021). We evaluate sparse networks (99% or 90%) learned with natural training on CIFAR-10 using VGG-16.



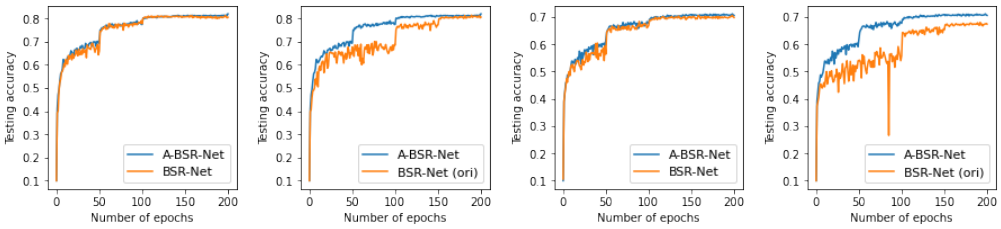

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 6: Comparisons (accuracy given the number of epochs) with BSR-Net Özdenizci & Legenstein (2021). We evaluate sparse networks (99% or 90%) learned with adversarial training (objective: AT) on CIFAR-10 using VGG-16.



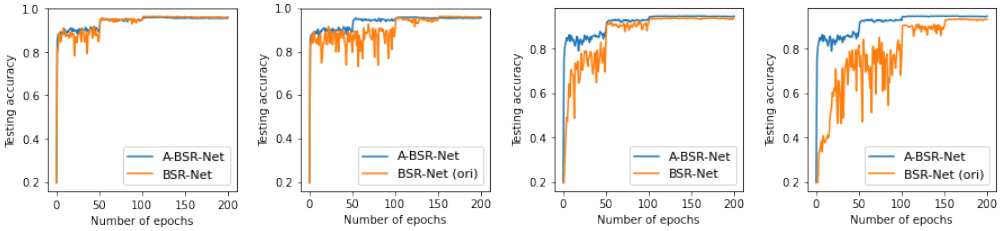

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 7: Comparisons (accuracy given the number of epochs) with BSR-Net Özdenizci & Legenstein (2021). We evaluate sparse networks (99% or 90%) learned with natural training on CIFAR-10 using Wide-ResNet-28-4.



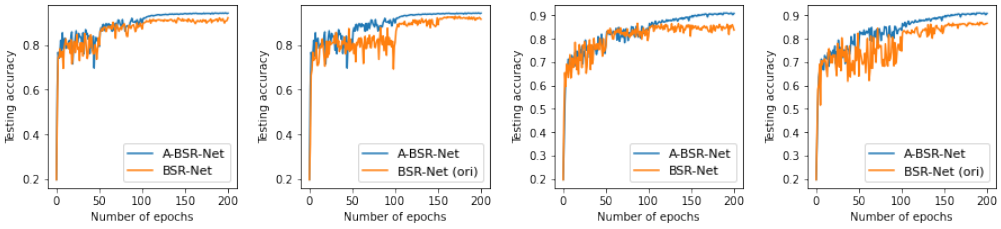

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 8: Comparisons (accuracy given the number of epochs) with BSR-Net Özdenizci & Legenstein (2021). We evaluate sparse networks (99% or 90%) learned with adversarial training (objective: AT) on CIFAR-10 using Wide-ResNet-28-4.



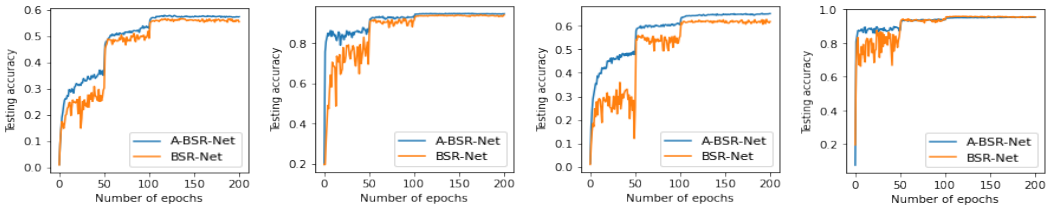

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 9: Comparisons (accuracy given the number of epochs) with BSR-Net Özdenizci & Legenstein (2021). We evaluate sparse networks (99% or 90%) learned with natural training on SVHN using VGG-16.



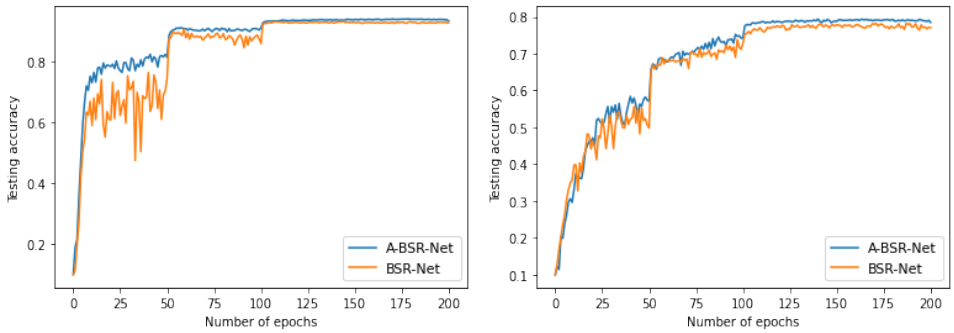

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 10: Comparisons (accuracy given the number of epochs) with BSR-Net Özdenizci & Legenstein (2021). We evaluate sparse networks (99% or 90%) learned with adversarial training (objective: TRADES) on SVHN using VGG-16.




(a) CIFAR-100,VGG-16 (b) SVHN,VGG-16 (c) CIFAR-100,WRN-28-4 (d) SVHN,WRN-28-4

Figure 11: Training curve (accuracy given number of epochs) of BSR-Net-based models (Özdenizci & Legenstein, 2021). Sparse networks (99%) are learned in standard setups on (a) CIFAR-100 using VGG-16, (b) SVHN using VGG-16, (c) CIFAR-100 using WRN-28-4, (d) SVHN using WRN-28-4.




(a) Standard (b) Adversarial (AT)

Figure 12: Training curve (required epochs to reach given accuracy) of BSR-Net-based models (Özdenizci & Legenstein, 2021). Dense networks are learned in standard and adversarial setups on CIFAR-10 using VGG-16.

In Figure 12, we also compare the convergence speed without sparsity. We show a BSR-Net-based result, where dense network is learned by adversarial training (AT) and standard training on CIFAR-10 using VGG-16. The blue curve of our A-BSR-Net tends to be above the yellow curve of BSR-Net, indicating successful acceleration. This demonstrates the broad applicability of our method.

Then, we also show ITOP-based results on ImageNet-2012. As shown in Figure 13, the red and blue curve represent AGENT + RigL-ITOP and RigL-ITOP on 90% sparse ResNet-50, respectively. We can see that the red curve is more stable than the blue curve, which shows the stable effect of our AGENT on large data sets and is a slightly different manifestation of the strengths of our AGENT. If we use SVRG in this case, we will not only fail to train stably, but also slow down the training speed. In contrast, our AGENT can solve the limitation of SVRG.

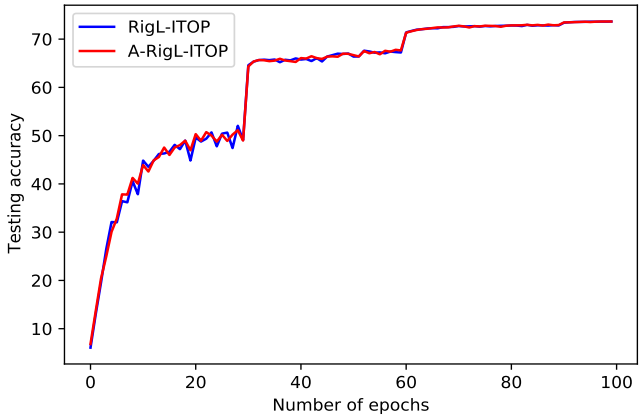
For other sparsity levels, we can expect advantages of our AGENT, in terms of acceleration or stability. Moreover, we can expect more significant speedups at different sparsity levels with more hyperparameter tuning, as the speedups are guaranteed by theoretical proofs.

### B.2 Number of Training Epoch Comparisons

We also compare the number of training epochs required to reach the same accuracy in BSR-Net-based results. In Figures 14, 15, 16, 17, 18, 19, 20, the blue curves (A-BSR-Net) are always lower than yellow curves (BSR-Net and BSR-Net (ori)), indicating faster convergence of A-BSR-Net.

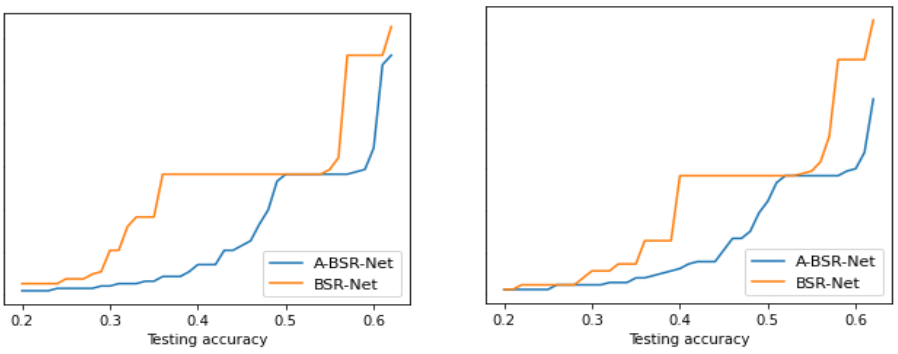
### B.3 Scaling Parameter Setting

The choice of the scaling parameter $\gamma$ is important to the acceleration and can be seen as a hyper-parameter tuning process. We experiment with different values of $\gamma$ and find that setting $\gamma = 0.1$




(a) Standard

Figure 13: Training curve (required epochs to reach given accuracy) of ITOP-based models (Liu et al., 2021). Sparse networks are learned in standard setup on ImageNet-2012 using ResNet-50.



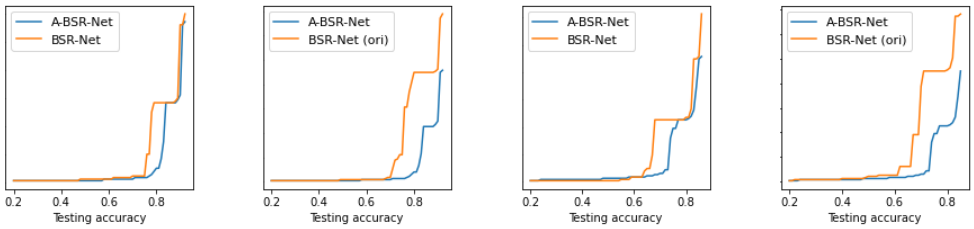

(a) Wide-ResNet-28-4 (b) ResNet-18

Figure 14: Comparisons (required hours to reach given accuracy. We evaluate sparse networks (99%) learned with natural training on CIFAR-100 using (a) Wide-ResNet-28-4, and (b) ResNet-18.

is a good choice for effective acceleration of training. The presented results are based on sparse networks (99%) learned with adversarial training (objective: AT) on CIFAR-10 using VGG-16.

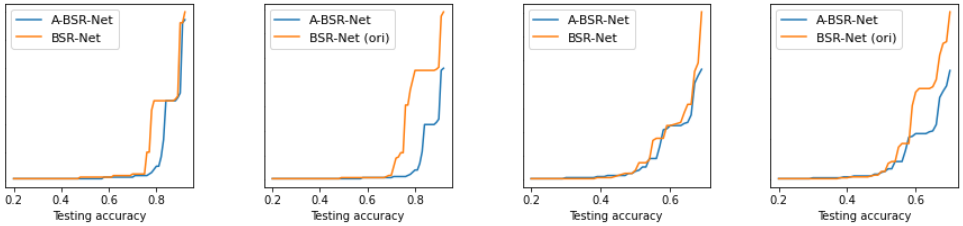
As shown in Figure 21 (a), we compare the training curves (testing accuracy at different epochs) A-BSR-Net ($\gamma = 0.1$), A-BSR-Net ($\gamma = 0.5$), and BSR-Net. The yellow curve for A-BSR-Net ($\gamma = 0.5$) collapses after around 40 epochs training, indicating a model divergence. The reason is that if setting $\gamma$ close to 1, e.g., like 0.5, we will not be able to completely avoid the increase in variance. The increase in variance will lead to a decrease in performance, which is similar to "No $\gamma$" in section 5.4 of the manuscript.

As shown in Figure 21 (b), we compare the training curves (testing accuracy at different epochs) A-BSR-Net ($\gamma = 0.1$), A-BSR-Net ($\gamma = 0.01$), and BSR-Net. The yellow curve for A-BSR-Net ($\gamma = 0.01$) is below the blue curve for A-BSR-Net ($\gamma = 0.1$), indicating a slower convergence speed. The reason is that if $\gamma$ is set small, such as 0.01, the weight of the old gradients will be small. Thus, the old gradients will have limited influence on the updated direction of the model, which tends to slow down the convergence and sometimes can lead to more training instability.



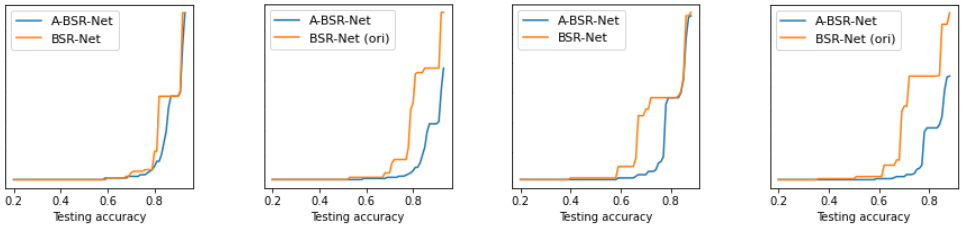

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 15: Comparisons (required hours to reach given accuracy. We evaluate sparse networks (99% or 90%) learned with natural training on CIFAR-10 using VGG-16.



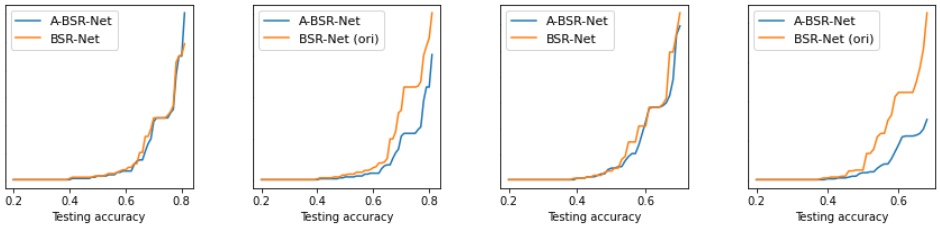

(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 16: Comparisons (required hours to reach given accuracy). We evaluate sparse networks (99% or 90%) learned with adversarial training (objective: AT) on CIFAR-10 using VGG-16.




(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 17: Comparisons (required hours to reach given accuracy). We evaluate sparse networks (99% or 90%) learned with natural training on CIFAR-10 using Wide-ResNet-28-4.

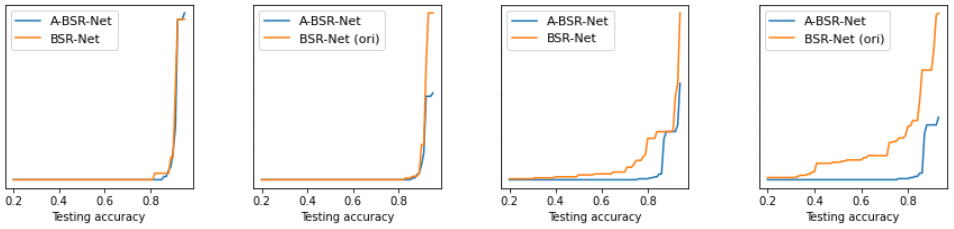



(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 18: Comparisons (required hours to reach given accuracy). We evaluate sparse networks (99% or 90%) learned with adversarial training (objective: AT) on CIFAR-10 using Wide-ResNet-28-4.

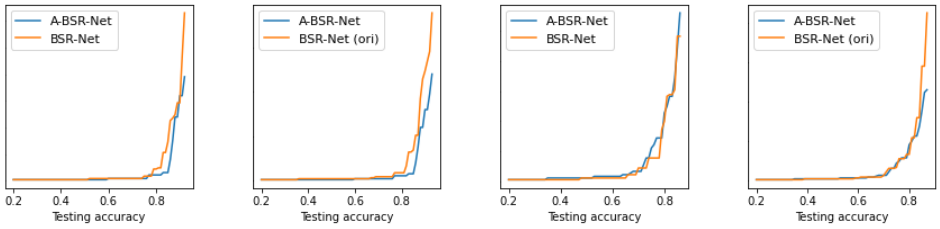
### B.4 Other Variance Reduction Method Comparisons

We also include more results about comparison between our ADSVRG and stochastic variance reduced gradient (SVRG) Baker et al. (2019); Chen et al. (2019); Zou et al. (2018), a popular variance reduction method in non-sparse case, to show the limitations of previous methods.




(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 19: Comparisons (required hours to reach given accuracy). We evaluate sparse networks (99% or 90%) learned with natural training on SVHN using VGG-16.




(a) 90% Sparsity (b) 90% Sparsity (c) 99% Sparsity (d) 99% Sparsity

Figure 20: Comparisons (required hours to reach given accuracy). We evaluate sparse networks (99% or 90%) learned with adversarial training (objective: TRADES) on SVHN using VGG-16.

### B.4.1 BSR-Net-based Results

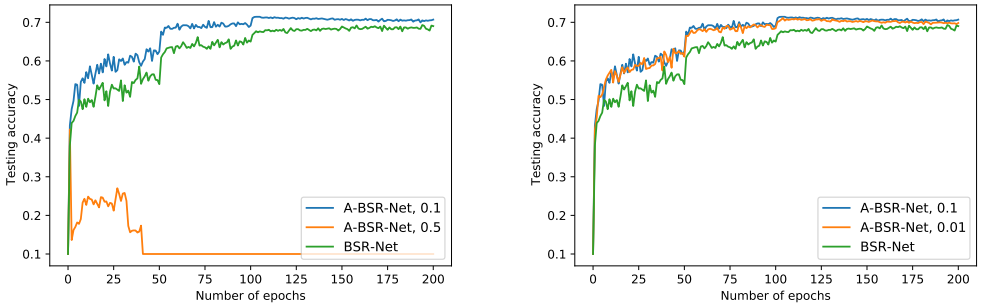
The presented results are based on sparse networks (99%) learned with adversarial training (objective: AT) on CIFAR-10 using VGG-16. As presented in Figure 22, we show the training curves (testing accuracy at different epochs)of A-BSR-Net, BSR-Net, and BSR-Net using SVRG. The yellow curve for BSR-Net using SVRG rises to around 0.4 and then rapidly decreases to a small value around 0.1, indicating a model divergence. This demonstrates that SVRG does not work for sparse training. As for the blue curve for our A-BSR-Net, it is always above the green curve for BSR-Net, indicating a successful acceleration.

### B.4.2 RigL-based Results

The presented results are based on sparse networks (90%) learned with standard training on CIFAR-100 using ResNet-50. As presented in Figure 23, we show the training curves (testing accuracy at different epochs) of A-RigL, RigL, and RigL using SVRG. The yellow curve for RigL using SVRG is always below the other two curves, indicating a slower model convergence. This demonstrate that SVRG does not work for sparse training. As for the blue curve for our A-RigL, it is always on the top of the green curve for RigL, indicating that the speedup is successful.

## B.5 Final Accuracy Comparisons

We also provide additional BSR-Net-based results for the final accuracy comparison. In addition to the BSR-Net and A-BSR-Net in the manuscript, we also include HYDRA in the appendix, which is also a SOTA sparse and adversarial training pipeline. The results are trained on SVHN using VGG-16 and WideResNet-28-4 (WRN-28-4). The final results for BSR-Net and HYDRA are obtained from Özdenizci & Legenstein (2021) using their original learning rate schedules. As shown in Table 5, it is encouraging to note that our method tends to be the best in all cases when given clean test samples. In terms of the robustness, our A-BSR-Net beats HYDRA in most cases, while experience a performance degradation compared to BSR-Net.




(a) Scaling parameter = 0.1 or 0.5 (b) Scaling parameter = 0.1 or 0.01

Figure 21: Comparisons (testing accuracy given the number of epochs) with different scaling parameters in BSR-Net-based models Özdenizci & Legenstein (2021). We evaluate sparse networks (99%) learned with adversarial training (objective: AT) on CIFAR-10 using VGG-16. (a) scaling parameter = 0.1 or 0.5, (b) scaling parameter = 0.1 or 0.01.

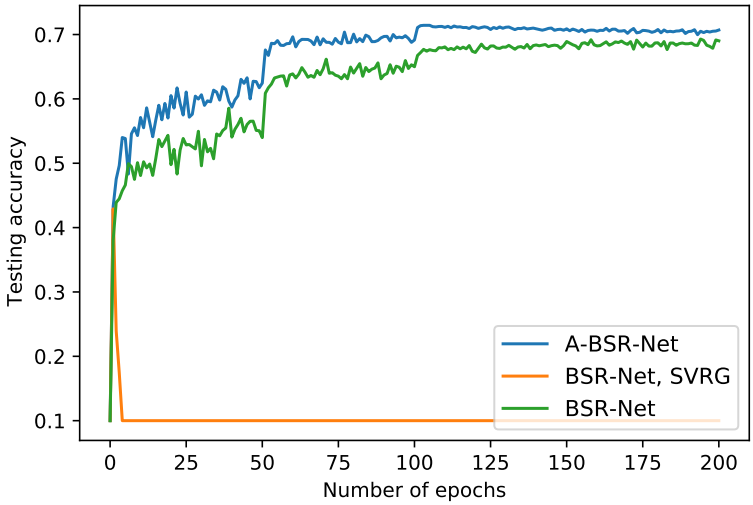



Figure 22: Comparisons (testing accuracy given the number of epochs) with different variance reduction methods in BSR-Net-based models Özdenizci & Legenstein (2021). We evaluate sparse networks (99%) learned with adversarial training (objective: AT) on CIFAR-10 using VGG-16.

Table 5: Comparisons the BSR-Net Özdenizci & Legenstein (2021) and HYDRA Sehwag et al. (2020). Evaluations of sparse networks learned with robust training objectives (TRADES) on SVHN using VGG-16 and WideResNet-28-4. Evaluations are after full training (200 epochs) and presented as clean/robust accuracy (%). Robust accuracy is evaluated via $\text{PGD}^{50}$ with 10 restarts $\epsilon = 8/255$.

| | | BSR-NET | HYDRA | OURS |
|---|---|---|---|---|
| 90% SPARSITY | VGG-16 | 89.4/**53.7** | 89.2/52.8 | **94.4**/51.9 |
| | WRN-28-4 | 92.8/**55.6** | 94.4/43.9 | **95.5**/46.2 |
| 99% SPARSITY | VGG-16 | 86.4/**48.7** | 84.4/47.8 | **90.9**/47.9 |
| | WRN-28-4 | 89.5/**52.7** | 88.9/39.1 | **92.2**/51.1 |

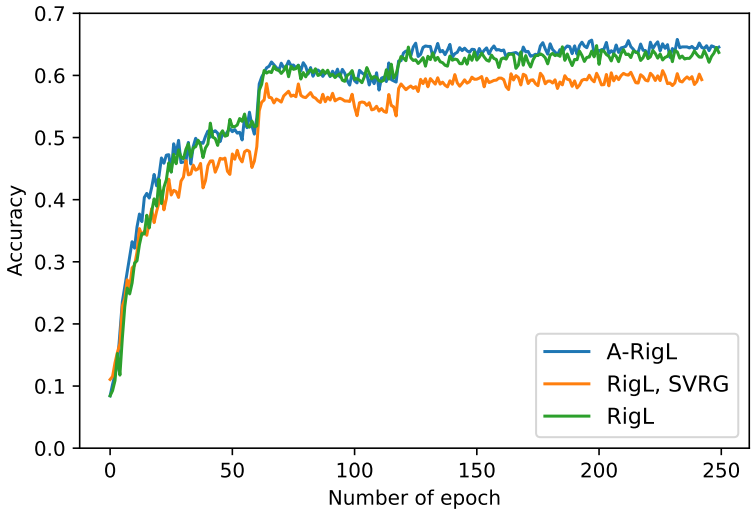



Figure 23: Comparisons (testing accuracy given the number of epochs) with different variance reduction methods in RigL-based models Evci et al. (2020). We evaluate sparse networks (90%) learned with standard training on CIFAR-100 using ResNet-50.

### B.6 Gradient Change Speed & Sparsity Level

In sparse training, when there is a small change in the weights, the gradient changes faster than in dense training, and this phenomenon can be expressed as a low correlation between the current and previous gradients, making the existing variance reduction methods ineffective.

We first demonstrate this lower correlation from an intuitive point of view. Considering the weights on which the current and previous gradients were calculated, there are three cases to be discussed in sparse training when the masks of current and previous gradients are different. First, if current weights are pruned, we do not need to consider their correlation because we do not need to update the current weights using the corresponding previous weights. Second, if current weights are not pruned but previous weights are pruned, the previous weights are zero and the difference between two weights is relatively large, leading to a lower relevance. Third, if neither the current nor the previous weights are pruned, which weights are pruned can still change significantly, leading to large changes in the current and previous models. Thus, the correlation between the current and previous gradients of the weights will be relatively small. Thus, it is not a good idea to set $c = 1$ directly in sparse training which can even increase the variance and slow down the convergence.

When the masks of the current and previous gradients are the same, the correlation still tends to be weaker. As we know, $c_t^* = \frac{\mathrm{Cov}(g_{\text{new}}, g_{\text{old}})}{\mathrm{Var}(g_{\text{old}})}$. Even if $\mathrm{Cov}(g_{\text{new}}, g_{\text{old}})$ does not decrease, the variance $\mathrm{Var}(g_{\text{old}})$ increases in sparse training, leading to a decrease in $c_t^*$.

Apart from the analysis above, we also do some experiments to demonstrate that the gradient changes faster as the sparsity increases. To measure the rate of change, our experiments are described below. We begin with fully-trained checkpoints from ResNet-50 on CIFAR-100 with RigL and SET at 0%, 50%, 80%, 90%, and 95% sparsity. We calculate and store the gradient of each weight on all training data. Then, we add Gaussian perturbations (std = 0.015) to all the weights and calculate the gradients again. Lastly, we calculate the correlation between the gradient of the new perturbed weights and the old original weights.

As we know, there is always a difference between the old and new weights. If the gradients become very different after adding some small noise to the weights, the new and old gradients will tend to have smaller correlations. If the gradients do not change a lot after adding some small noise, the old and new gradients will have a higher correlation. Thus, we add Gaussian noise to the weights to simulate the difference between the new and old gradients. As shown in Table 6, the correlation de-

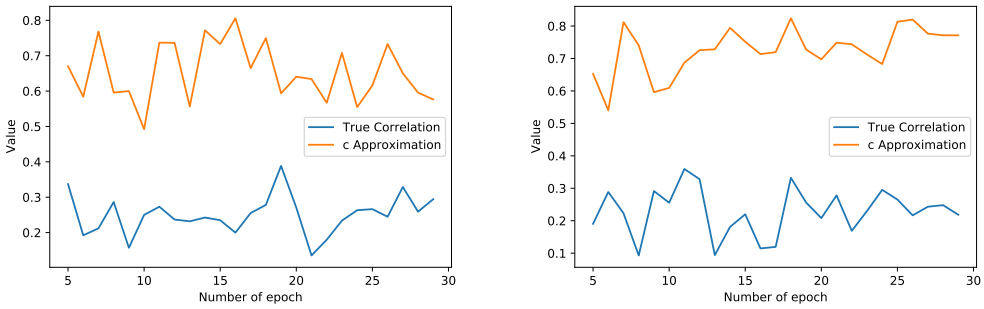
creases with increasing sparsity, which indicates a weaker correlation in sparse training and supports our claim.

Table 6: Correlation between the gradient of the new perturbed weights and the old original weights from ResNet-50 on CIFAR-100 produced by RigL and SET at different sparsity including 0%, 50%, 80%, 90%, 95%, 99%.

| SPARSITY | 0% | 50% | 80% | 90% | 95% |
|---|---|---|---|---|---|
| RESNET-50, CIFAR-100 (RIGL) | 0.6005 | 0.4564 | 0.3217 | 0.1886 | 0.1590 |
| RESNET-50, CIFAR-100 (SET) | 0.6005 | 0.4535 | 0.2528 | 0.1763 | 0.1195 |

### B.7 COMPARISON BETWEEN TRUE CORRELATION & OUR APPROXIMATION

In this section, to test how well our approximation estimates the true optimum c, we empirically compare the approximation $c^*$ in Eq. (4) (in the main manuscript) and the correlation between gradient of current weights and gradient of previous epoch weights. As shown in Figure 24, the yellow and blue curves represent the approximation $c^*$ and the correlation, respectively. The two curves tend to have similar up-and-down patterns, and the yellow curves usually have a larger magnitude. This suggests that our c approximation captures the dynamic patterns of the correlation. For the larger magnitude, it can be matched by our scaling parameter.



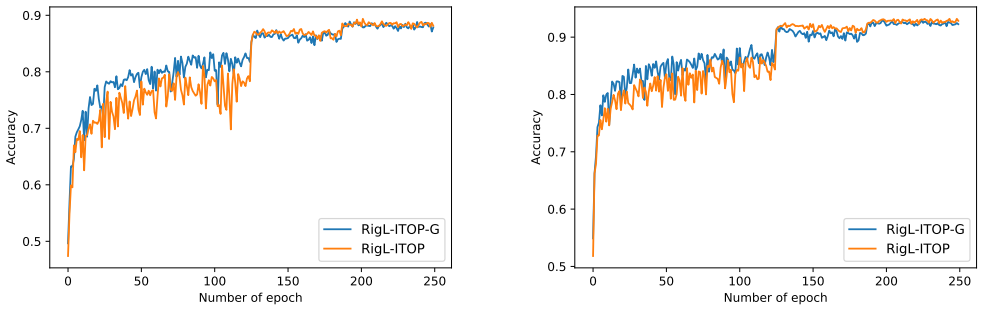

(a) 90% (b) 99%

Figure 24: Comparisons between the approximation $c^*$ and correlation between gradient of current weights and gradient of previous epoch weights. We evaluate sparse networks learned with RigL-based standard training on CIFAR-10 using ResNet-50 with (a) 90% sparsity and (b) 99% sparsity.

### B.8 VARIANTS OF RIGL

RigL is one of the most popular dynamic sparse training pipeline which uses weight magnitude for pruning and gradient magnitude for growing. Our method adaptively updates the new batch gradient using the old storage gradient which usually has less noise. As a result, the variance of the new batch gradient is reduced, leading to fast convergence. Currently, we only use gradients with corrected variance in weight updates. A natural question is how does it perform if we also use this variance-corrected gradient for weight growth in RigL.

We do some experiments in RigL-based models trained on CIFAR-10. As shown in Figure 25, the blue curves (RigL-ITOP-G) and yellow curves (RigL-ITOP) correspond to the weight growth with and without the variance-corrected gradient, respectively. We can see that in the initial stage, the blue curves are higher than the yellow curves. But after the first learning rate decay, they tend to be lower than the yellow curves. This suggests that weight growth using a variance-corrected gradient at the beginning of training can help the model improve accuracy faster. However, this may lead to a slight decrease in accuracy in the later training stages. This may be due to the fact that some variance in the gradient can help the model explore local regions better and find better masks as the model approaches its optimal point.



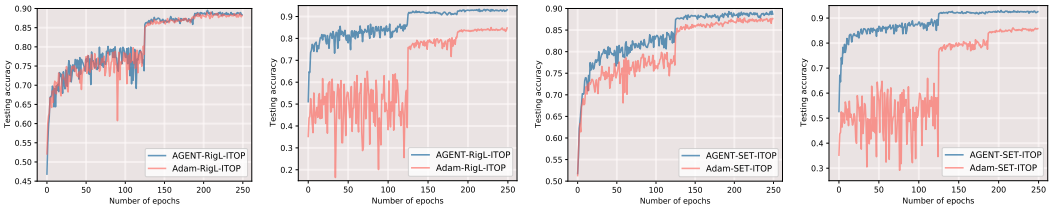

(a) VGG-C (b) ResNet-34

Figure 25: Comparisons (testing accuracy given the number of epochs) between weight growth with (RigL-ITOP-G) and without (RigL-ITOP) variance-corrected gradient Liu et al. (2021). We evaluate sparse networks (99%) learned with standard training on CIFAR-10 using (a) VGG-C and (b) ResNet-34.

### B.9 Comparison with Other Adaptive Gradient Method

We add experiments to compare our AGENT with Adam (Kingma & Ba, 2014). and find that our method has a faster convergence rate compared to Adam. As shown in Figure 26, our AGENT (blue curve) is usually higher than Adam (pink curve), indicating that our AGENT has better acceleration.

The main reason is that our AGENT is designed for sparse training and is tailored to the characteristics of sparse training. When old information is used to correct the gradients, the main problem is the reduced correlation between the old and new gradients. Therefore, our AGENT approximates the correlation and adds an adaptive weight to the old gradient to establish a balance between the old and new gradients. However, previous adaptive gradient methods, such as Adam, are not designed for sparse training. Although they also provide adaptive gradients, their adaptivity is different and does not take correlation into account.




(a) VGG-C(RigL) (b) ResNet-34(RigL) (c) VGG-C(SET) (d) ResNet-34(SET)

Figure 26: Testing accuracy for ITOP-based models at 99% sparsity on CIFAR-10. AGENT-RigL-ITOP and AGENT-SET-ITOP (blue curves) converge faster than Adam-RigL-ITOP and Adam-SET-ITOP (pink curves).

### B.10 Comparison with Reducing Learning Rate

To demonstrate the design of the scaling parameter $\gamma$, we compare our AGENT with "Reduce LR", where we remove the scaling parameter $\gamma$ from AGENT and set the learning rate to 0.1 times the original one. As shown in Table 7, reducing the learning rate can lead to a comparable convergence rate in the early stage. However, it slows down the later stages of training and leads to sub-optimal final accuracy. The reason is that it reduces both signal and noise, and therefore does not improve the signal-to-noise ratio or speed up the sparse training.

The motivation of $\gamma$ is to avoid introducing large variance due to error in approximating $c_t$ and bias due to the adversarial training. The true correlation depends on many factors such as the dataset, architecture, and sparsity. In some cases, it can be greater or smaller than 10%. For the value of

$\gamma$, it is a hyperparameter and we can choose different values for different settings. In our case, for simplicity, we choose $\gamma = 0.1$ for all the settings, and find that it works well and accelerates the convergence. If we tune the value of $\gamma$ for different settings according to their corresponding correlations, it is possible to obtain faster convergence rates.

Table 7: Testing accuracy (%) of SET-ITOP-based models for AGENT (ours) and "Reduce LR". Sparse VGG-C and ResNet-34 are learned in standard setups.

| EPOCH | 20 | 80 | 130 | 180 | 240 |
|---|---|---|---|---|---|
| REDUCE LR (VGG-C, SET-ITOP) | **76.5** | 81.3 | 84.6 | 85.5 | 85.5 |
| AGENT (VGG-C, SET-ITOP) | 76.1 | **81.5** | **87.6** | **87.1** | **88.6** |
| REDUCE LR (RESNET-34, SET-ITOP) | 81.4 | **85.9** | 89.3 | 89.5 | 89.8 |
| AGENT (RESNET-34, SET-ITOP) | **83.0** | 85.6 | **92.0** | **92.3** | **92.5** |

### B.11 COMPARISON WITH MOMENTUM-BASED METHODS

The momentum-based approach works well in general, but it still suffers from optimization difficulties due to sparsity constraints. For example, in our baseline SGD, following the original code base, we have also added momentum to the optimizer. However, as shown in the pink curves in Figure 2, it still has training instability and convergence problems. The reason is that they do not take into account the sparse and adversarial training characteristics and cannot provide an adaptive balance between old and new information.

Our method AGENT is designed for sparse and adversarial training and can establish a finer control over how much information we should get from the old to help the new. To demonstrate the importance of this fine-grained adaptive balance, we do ablation studies in Section 6.4. In "Fixed $c_t$", we set $c_t = 0.1$ and test the convergence rate without the adaptive control. We find that the adaptive balance (ours) outperforms "Fixed $c_t$" in almost all cases, especially in adversarial training. For standard training, "Fix $c_t$" provides similar convergence rates to our method, while ours tends to have better final scores.

## C ADDITIONAL DETAILS ABOUT EXPERIMENT SETTINGS

### C.1 IMPLEMENTATIONS

**In BSR-Net-based results**, aligned with the choice of Özdenizci & Legenstein (2021), the gradients for all models are calculated by SGD with momentum and decoupled weight decay (Loshchilov & Hutter, 2019). All models are trained for 200 epochs with a batch size of 128.

**In RigL-based results**, we follow the settings in Evci et al. (2020); Sundar & Dwaraknath (2021). We train all the models for 250 epochs with a batch size of 128, and parameters are optimized by SGD with momentum.

**In ITOP-based results**, we follow the settings in Liu et al. (2021). For CIFAR-10 and CIFAR-100, we train all the models for 250 epochs with a batch size of 128. For ImageNet-2012, we train all the models for 100 epochs with a batch size of 64. Parameters are optimized by SGD with momentum.

### C.2 LEARNING RATE

Aligned with popular sparse training methods (Evci et al., 2020; Özdenizci & Legenstein, 2021; Liu et al., 2021), we choose piecewise constant decay schedulers for learning rate and weight decay. In our A-BSR-Net, we use the 50th and 100th epochs as the dividing points of our learning rate decay scheduler. The reason is that our approach has faster convergence and doesn't require a long warm-up period. In the evaluation shown in the manuscript, we also use this scheduler for BSR-Net for a more accurate and fair comparison.

### C.3 INITIALIZATION (BSR-NET-BASED RESULTS)

Consistent with Özdenizci & Legenstein (2021), we also choose Kaiming initialization to initialize the network weights He et al. (2015)

### C.4 BENCHMARK DATASETS (BSR-NET-BASED RESULTS)

For a fair comparison, we choose the same benchmark datasets as Özdenizci & Legenstein (2021). Specifically, we use CIFAR-10 and CIFAR-100 Krizhevsky et al. (2009) and SVHN Netzer et al. (2011) in our experiments. Both CIFAR-10 and CIFAR-100 datasets include 50, 000 training and 10, 000 test images. SVHN dataset includes 73, 257 training and 26, 032 test samples.

### C.5 DATA AUGMENTATION

We follow a popular data augmentation method used in Özdenizci & Legenstein (2021); He et al. (2016). In particular, we randomly shift the images to the left or right, crop them back to their original size, and flip them in the horizontal direction. In addition, all the pixel values are normalized in the range of [0, 1].

## D SPARSE TRAINING METHOD DESCRIPTION

### D.1 BAYESIAN SPARSE ROBUST TRAINING

Bayesian Sparse Robust Training (BSR-Net) Özdenizci & Legenstein (2021) is a Bayesian Sparse and Robust training pipeline. Based on a Bayesian posterior sampling principle, a network rewiring process simultaneously learns the sparse connectivity structure and the robustness-accuracy trade-off based on the adversarial learning objective. More specifically, regarding its mask update, it prunes all negative weights and grows new weights randomly.

## E LIMITATIONS OF OUR ADAPTIVE GRADIENT CORRECTION METHOD

### E.1 EXTRA FLOPS

Similar to SVRG, our ADSVRG increases the training FLOPs in each iteration due to the extra forward and backward used to compute the old gradients.

However, the true computation difference can be smaller and the GPU-based runining time of SVRG will not be affected that much. For example, in the adversarial setting, we need additional computations to generate the adversarial samples, which is time-consuming and only needs to be done once in each iteration of our AVR and SGD. For BSR-Net, we empirically find that the ratio of time required for each iteration of our AVR and SGD is about 1.2.

There are also several methods to reduce the extra computation caused by SVRG. The first approach is to use the sparse gradients proposed by M Elibol (2020) Elibol et al. (2020). It can effectively reduce the computational cost of SVRG and can be easily applied to our method. The second approach is suggested by Allen-Zhu and Hazan (2016) Allen-Zhu & Hazan (2016). The extra cost on computing batch gradient on old model parameters is totally parallelizable. Thus, we can view SVRG as doubling the mini-batch size. Third, we can follow the idea of SAGA Defazio et al. (2014) and store gradients for individual samples. By this way, we do not need the extra forward and backward step and save the computation. But it requires extra memory to store the gradients.

In the main manuscript, we choose to compare the convergence speed of our ADSVRG and SGD for the same number of pass data (epoch), which is widely used as a criterion to compare SVRG-based optimization and SGD (Allen-Zhu & Hazan, 2016; Chatterji et al., 2018; Zou et al., 2018; Cutkosky & Orabona, 2019). A comparison in this way in this way can demonstrate the accelerating effect of the optimization method and provide inspiration for future work.

### E.2 SCALING PARAMETER TUNING

In our adaptive variance reduction method (AVR), we add an additional scaling parameter $\gamma$ which need to be adjusted. We find that setting $\gamma = 0.1$ is a good choice for BSR-Net, RigL, and ITOP. However, it can be different for other different sparse training pipelines.

### E.3 ROBUST ACCURACY DEGRADATION

For the final accuracy results of BSR-Net-based models, there is a small decrease in the robustness accuracy after using our AVR. It is still an open question how to further improve the robust accuracy when using adaptive variance reduction in sparse and adversarial training.