Learning Adversarially Robust Sparse Networks via Weight Reparameterization

Chenhao Li[1, 2], **Qiang Qiu**[1], **Zhibin Zhang**[1], **Jiafeng Guo**[1], **Xueqi Cheng**[1]

[1] CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{lichenhao19b, qiuqiang, zhangzhibin, guojiafeng, cxq}@ict.ac.cn

Abstract

Although increasing model size can enhance the adversarial robustness of deep neural networks, in resource-constrained environments, there exist critical sparsity constraints. While the recent robust pruning technologies show promising direction to obtain adversarially robust sparse networks, they perform poorly with high sparsity. In this work, we bridge this performance gap by reparameterizing network parameters to simultaneously learn the sparse structure and the robustness. Specifically, we introduce Twin-Rep, which reparameterizes original weights into the product of two factors during training and performs pruning on the reparameterized weights to satisfy the target sparsity constraint. Twin-Rep implicitly adds the sparsity constraint without changing the robust training objective, thus can enhance robustness under high sparsity. We also introduce another variant of weight reparameterization for better channel pruning. When inferring, we restore the original weight structure to obtain compact and robust networks. Extensive experiments on diverse datasets demonstrate that our method achieves state-of-the-art results, outperforming the current sparse robust training method and robustness-aware pruning method. Our code is available at https://github.com/UCAS-LCH/Twin-Rep.

Introduction

Deep neural networks (DNNs) have achieved extraordinary performance in various tasks. However, they are highly vulnerable to adversarial examples crafted by adding well-designed perturbations to natural examples (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2015). Such fragility becomes an obstacle to using DNNs in safety-critical environments such as face recognition (Deng et al. 2019) and autonomous driving (Bojarski et al. 2016). Numerous studies have proposed various adversarial defense techniques for such threats. To date, adversarial training that relies on training the network with adversarial examples has been found to be the most effective (Goodfellow, Shlens, and Szegedy 2015; Madry et al. 2018; Zhang et al. 2019; Wang et al. 2019).

While adversarial training can significantly improve the adversarial robustness of neural networks, it requires a larger network capacity to achieve high robustness than that of natural training (Nakkiran 2019; Madry et al. 2018). The required large network capacity limits the use in resource-constrained environments, highlighting a need to consider robustness under sparsity constraints. Unfortunately, despite a plethora of work on compressed model performance on clean data, there have been only a few studies on the robustness of compressed models under adversarial attacks.

Some early works have tried to generate both sparse and robust networks (Rakin et al. 2019; Ye et al. 2019; Gui et al. 2019) by directly combining network pruning and adversarial training techniques. These works inherit heuristic pruning algorithms developed for natural training (Han et al. 2015), which drops connections in neural networks with the lowest weight. However, these pruning methods incur poor performance under adversarial scenarios when a relatively high ratio is set. Specifically, since adversarially trained networks generally demand more non-zero parameters than naturally trained networks (Rakin et al. 2019; Ye et al. 2019), the traditional penalty-based paradigm suffers from a contradiction between robustness and sparsity. For example, training with a strong penalty achieves high sparsity but hurts robustness as the parameters deviate from the optima, while a weak penalty maintains the performance but results in a denser network, hence great pruning damage.

In this paper, we propose to address the above contradiction by decoupling network architecture between training and inference. Specifically, we construct the training-time structure by decomposing the original weights into two matrices/tensors, which are merged by element-wise multiplication during inference. Since the magnitude of the two factors is small, the trained network will have enough sparsity to achieve low pruning damage when merging. At the same time, we increase the trainable parameters, which helps to improve robustness under high sparsity. Note that we optimize the parameters together with the robust objective, and thus it is the same efficiency as the traditional training manner. The parameters are restored to the original structure during inference, so the model size and computational effort do not increase.

In short, our proposed technique overcomes the contradiction between robustness and sparsity that the traditional penalty-based paradigm suffers. On the one hand, the multiplication of two factors implicitly increases network sparsity, thus reducing pruning damage under high pruning ratios. On the other hand, the doubled parameters are only learned by adversarial loss during training, which has enough ability to achieve high robustness.

Contributions of our work are summarized as follows:

- We develop a novel weight reparameterization technique, which implicitly adds the sparse limit, by reparameterizing original parameters into two element-wise multiplied factors. We also introduce a variant of the proposed weight reparameterization approach for channel sparsity, which shows a promising direction to develop more reparameterization methods for better robustness under different sparsity constraints.
- We combine network pruning with weight reparameterization during adversarial training, achieving both ultra-high compression ratio and robustness to perturbed images. Moreover, we show that our method is compatible with both unstructured and structured pruning.
- We conduct extensive experiments on both white-box and black-box attacks, of which the results show that our approach achieves state-of-the-art performance. Notably, at a very high pruning ratio of 99%, the proposed method achieves significant gains up to 4.9% and 4.7% on CIFAR-10 with VGG-16 models, compared to the two strongest baselines, respectively.

Related Work

Adversarial Robustness

Over the past decade, a large number of defenses have been proposed to mitigate the effect of adversarial examples. To date, adversarial training that relies on harnessing adversarial examples has been demonstrated to be the most effective (Athalye, Carlini, and Wagner 2018). Madry et al. (2018) proposed a min-max formulation for training adversarially robust models using empirical risk minimization. They generated adversarial examples via projected gradient descent (PGD) to sufficiently approximate the inner maximization step so that the subsequent adversarial training yielded robust models. Following this, Zhang et al. (2019) quantified the trade-off between accuracy and robustness and proposed TRADES loss to achieve better robustness:

$$\mathcal{L}_{\mathrm{TRADES}} = \mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}, x, y) + \beta D_{\mathrm{KL}}(p(y|x, \boldsymbol{\theta})||p(y|\widetilde{x}, \boldsymbol{\theta})), \tag{1}$$

where β is the robustness-accuracy trade-off parameter and $\widetilde{x}$ is obtained by maximizing the Kullback-Leibler (KL) Divergence $D_{\mathrm{KL}}(p(y|x, \boldsymbol{\theta})||p(y|\widetilde{x}, \boldsymbol{\theta}))$. Later Wang et al. (2019) investigated the distinctive influence of misclassified and correctly classified examples on the robustness of adversarial training. They proposed MART loss, which explicitly differentiates misclassified examples during training. Moreover, Carmon et al. (2019) proposed a robust self-training (RST) scheme to train robust models with pseudo-labeled additional data samples. They revealed that adversarial robustness could significantly benefit from additional data information, even without precise labels.

While adversarial training improves network robustness to adversarial examples, it requires a significantly larger network capacity to obtain strong adversarial robustness than correctly classifying natural examples only (Madry et al. 2018). The need for a larger network capacity makes it challenging to achieve robustness and sparsity simultaneously.

Neural Network Pruning

Sparsification of neural networks is becoming increasingly important under the requirement to deploy such models to resource-limited devices. Pruning is widely used to eliminate the weight redundancy in over-parameterized DNNs by removing weights with less importance (Collins and Kohli 2014; Han et al. 2015; Louizos, Welling, and Kingma 2018). One such highly successful approach uses an iterative pruning and retraining pipeline to reduce the damage from pruning (Han et al. 2015; Guo, Yao, and Chen 2016). In addition to this pipeline, network pruning can be performed with training (Bellec et al. 2018; Lin et al. 2017) or before training (Frankle and Carbin 2019; Lee, Ajanthan, and Torr 2019). Such network pruning methods are mainly designed for natural training to obtain storage and computational savings with almost undamaged performance.

More closely related to our work is a recently emerging method to train compressed neural networks with reparameterized structures (Ding et al. 2021). They inserted a compactor after the original convolution layer, and equivalently merged them after training and pruning. By only adding penalty gradients to compactors, their approach avoided the confrontation between the objective function and penalty loss. Despite the standard natural training effectiveness, this work did not consider robust-training objectives.

Adversarial Robustness and Sparsity

Recent studies grew interested in exploring adversarial robustness under sparsity constraints. Guo et al. (2018) theoretically and experimentally demonstrated that adversarial robustness can be improved with an appropriate sparsity, whereas a very large sparsity tends to be harmful. Wang et al. (2018) derived similar conclusions that adversarial robustness decreases with high model sparsity. Instead of training robust and sparse networks, these works mainly explored the relationship between robustness and sparsity. Subsequently, several studies tried to compress robust networks using a magnitude-based pruning heuristic. Ye et al. (2019) and Gui et al. (2019) proposed a concurrent adversarial training and weight pruning scheme by solving a constrained optimization problem with an alternating direction method of multipliers (ADMM). While ADMM-based pruning techniques are successful in natural training, they incur a significant performance drop when combined with adversarial training. Rakin et al. (2019) applied the l_1 weight penalty in adversarial training to produce robust sparse models. However, their approach does not generalize to very sparse models. From an adversarial defense perspective, Madaan, Shin, and Hwang (2020) proposed a Bayesian framework to prune highly vulnerable features for better robustness. Chen et al. (2022) introduced static and dynamic sparsity in robust training to mitigate the robust generalization gap. Nevertheless, these approaches do not aim to obtain competitive robustness at high pruning ratios.

Recent two works constitute state-of-the-art methods for achieving robust networks at high sparsity. Sehwag et al. (2020) introduced HYDRA, where a robustness-aware mask was learned based on a robustly pre-trained network. HYDRA exploited the mask learning approach by Ramanujan et al. (2020) to search robust sub-networks in a pre-trained robust, dense network. While their approach can achieve very high model compression when integrated with various robust training objectives, necessitating a pre-trained dense, robust network and the fine-tuning step introduces an additional computational overload. Later, Özdenizci and Legenstein (2021) proposed a robust sparse training method that trained a robust network under strict sparse connectivity constraints throughout training. This dynamic sparsity approach achieves similar performance to HYDRA and costs less training time. However, it is inapplicable for structure pruning.

Methodology

In this section, we discuss details of the proposed weight reparameterization method, named Twin-Rep, which doubles the trainable parameters while implicitly introducing sparsity conditions. Then we introduce a variant of the proposed weight reparameterization approach for better robustness under channel sparsity constraints. Finally, we combine them with a heuristic pruning algorithm to achieve compact and robust networks.

Problem Formulation

We first formalize the problem for concurrent adversarial training and model compression. To make robust networks compact, we can state the problem as:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\widetilde{x}\in\mathcal{B}(x,\epsilon)} \mathcal{L}(\boldsymbol{\theta}, \widetilde{x}, y) \right] \quad s.t. ||\boldsymbol{\theta}||_0 \le k, \quad (2)$$

where adversarial examples $\widetilde{x}$ are defined within the l_∞-norm ball around samples x: $\mathcal{B}(x,\epsilon) := \{\widetilde{x} : ||\widetilde{x} - x||_\infty \le \epsilon\}$, with a perturbation strength of $\epsilon > 0$. The optimization for robustness is composed of inner maximization and outer minimization problems. We acquire the adversarial data $\widetilde{x}$ in inner maximization while improving the robustness of models in outer minimization. We can use cross-entropy loss or other robust training objectives as loss function. k defines the sparsity constraint on the network parameters.

Weight Reparameterization

In natural training with only benign examples, long-standing standard pruning methods use heuristic algorithms that prune connections with the lowest weight magnitude (Han et al. 2015). These methods use sparsity-inducing regularization to force some weights to converge close to zero, thus reducing damage from pruning. However, these methods perform poorly when integrated with robust training techniques. Since adversarially trained networks generally demand more non-zero parameters (See Figure 1) (Rakin et al. 2019; Ye et al. 2019), the traditional penalty-based paradigm faces a dilemma: a strong regularization hurts the training performance while a mild regularization cannot achieve high pruning ratios.

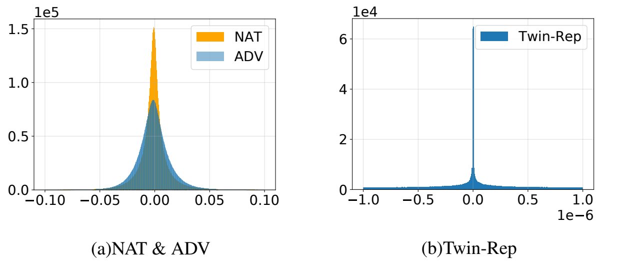

Figure 1: Weight distribution of ResNet-18 models trained with (a) Original weights with natural training and adversarial training (b) Reparameterized weights with adversarial training (merged).

Instead of explicitly applying a strong weight penalty, we implicitly add a sparsity constraint through weight reparameterization. Specifically, we perform a simple mapping on learnable network parameters $\boldsymbol{\theta}$. We reparametrize network parameters to the element-wise product of two matrices/tensors. During the training process, we use the parameters as follow:

$$\boldsymbol{\theta} = \boldsymbol{W}_1 \odot \boldsymbol{W}_2, \quad (3)$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ have the same shape as original parameters and are both used to fit the objective function. Notably, a mild regularization may bring positive effects and the state-of-the-art image classification considers DNNs trained with weight decay, we also apply weight decay on both of the reparametrized weights:

$$\begin{aligned} &(\boldsymbol{W}_1 - \lambda\boldsymbol{W}_1) \odot (\boldsymbol{W}_2 - \lambda\boldsymbol{W}_2) \\ &= \boldsymbol{W}_1 \odot \boldsymbol{W}_2 - 2\lambda\boldsymbol{W}_1 \odot \boldsymbol{W}_2 + \lambda^2\boldsymbol{W}_1 \odot \boldsymbol{W}_2 \\ &= \boldsymbol{\theta} - 2\lambda\boldsymbol{\theta} + \lambda^2\boldsymbol{\theta}, \end{aligned} \quad (4)$$

The third term on the right side of the equation can be ignored as the weight decay factor λ is small (Such as $2e-4$). Therefore, it is equivalent to adding weight decay to the original weights, which usually brings positive effects on performance and generalization.

Considering the empirical observation that weights are distributed over a small range, most products of the reparameterized weights will be very small. Therefore, our weight reparameterization method implicitly increases network sparsity. Figure 1 shows the weight distributions, where Twin-Rep leads to a significantly sparse model. Such sparse weights lead to little damage from pruning, even at high pruning ratios.

Additional sparsity constraints inevitably affect model performance in the training process, especially with such high sparsity. However, our approach doubles the trainable parameters, which helps reach higher robustness, thus achieving robust models at high compression rates.

Channel-Rep: Another Form of Weight Reparameterization

We also design another form of weight reparemeterization for better channel sparsity, named Channel-Rep. Let D be the number of output channels. Channel-Rep adds a $D \times D$ matrix and fuses the two weights using matrix multiplication:

$$\boldsymbol{\theta} = \boldsymbol{W}_1 \boldsymbol{W}_2, \tag{5}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{D \times D}$ and $\boldsymbol{W}_2$ has the same shape as original parameters. Channel-Rep will not cause an extremely sparse weight distribution like Twin-Rep does. Note that the linear combination of convolution weights is equivalent to the linear combination of pre-activation output features. Therefore, the matrix multiplication weight fusion operation aggregates the information of all channels. This property allows the remaining channels to contain more information, thus Channel-Rep can still extract sufficient features after pruning most output channels (or filters).

Weight Reparameterization Based Adversarial Training and Pruning

For obtaining robust networks, we minimize the adversarial loss without adding any term to the objective function (For the convenience of description, we take Twin-Rep as an example):

$$\min_{\boldsymbol{W}_1 \odot \boldsymbol{W}_2} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\widetilde{x} \in \mathcal{B}(x,\epsilon)} \mathcal{L}(\boldsymbol{W}_1 \odot \boldsymbol{W}_2, \widetilde{x}, y) \right], \tag{6}$$

The inner maximization is solved by projected gradient descent (PGD) (Madry et al. 2018), which presents a powerful adversarial attack as follows:

$$\widetilde{x}^{t+1} = \prod_{\mathcal{B}(x,\epsilon)} (\widetilde{x}^t + \alpha \cdot sign(\nabla_{\widetilde{x}^t} \mathcal{L}_{\text{PGD}}(\boldsymbol{W}_1 \odot \boldsymbol{W}_2, \widetilde{x}^t, y))), \tag{7}$$

where α is the step size, $sign(\cdot)$ returns the sign of the vector, Π is the projection operator on l_∞ norm-ball $B(x,\epsilon)$ around x with radius ϵ for each example, and $\widetilde{x}^t$ denotes the adversarial example at the t-th PGD step.

We optimize reparameterized weights using SGD just like the traditional training process. Then we perform network pruning after several training epochs, as important connections have been highlighted in this period (You et al. 2019). For unstructured pruning, we select top-k weights with the highest absolute values of $\boldsymbol{W}_1 \odot \boldsymbol{W}_2$ and set others to zero. And for structured pruning, we measure the importance of output channels using the l_2 norm. The metric values of convolution layers are defined as follows:

$$||\boldsymbol{\theta}_{i:}||_2 = \sqrt{\sum_{c=1}^{C} \sum_{p=1}^{K} \sum_{q=1}^{K} \boldsymbol{\theta}_{i,c,p,q}^2}, \quad \forall 1 \le i \le D, \tag{8}$$

where C and D are the number of input and output channels, K is the kernel size. The metric for linear layers can be computed similarly. We calculate the metric values for every output channel and sort them in ascending order. We select channels with the highest top-k values and zero out others. For the rest training phase, we only update selected weights or channels. While predicting, the two factors are multiplied together to restore the original parameters. The training and pruning procedure is outlined in Algorithm 1.

Algorithm 1: Pipeline of adversarial network pruning

```
Input: Dataset $\mathcal{D}$, neural network $f_\theta$ with reparameterized weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, PGD iterations $K$, perturbation strength $\epsilon$ and step size $\alpha$, learning rate $\eta$, weight decay $\lambda$, pruning ratio $p\%$, total iterations $T$, pruning time $\tau$.
Output: a pruned robust network $f_{\boldsymbol{\theta}}$
for $t = 1$ to $T$ do
    Sample a mini-batch $(\boldsymbol{x}, \boldsymbol{y}) \subset \mathcal{D}$
    for $k = 1$ to $K$ do
        Generate adversarial samples using PGD:
        $\widetilde{\boldsymbol{x}}^{k+1} = \Pi_\epsilon(\widetilde{\boldsymbol{x}}^k + \alpha \cdot sign(\nabla_{\widetilde{\boldsymbol{x}}^t} \mathcal{L}_{\text{PGD}}(f, \widetilde{\boldsymbol{x}}^t, \boldsymbol{y})))$
    end for
    Compute robust loss $\mathcal{L}$ on $(\widetilde{\boldsymbol{x}}, \boldsymbol{y})$
    $\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 - \eta(\nabla_{\boldsymbol{W}_1} \mathcal{L}(f, \widetilde{\boldsymbol{x}}, \boldsymbol{y}) + \lambda \boldsymbol{W}_1)$,
    $\boldsymbol{W}_2 \leftarrow \boldsymbol{W}_2 - \eta(\nabla_{\boldsymbol{W}_2} \mathcal{L}(f, \widetilde{\boldsymbol{x}}, \boldsymbol{y}) + \lambda \boldsymbol{W}_2)$
    if $t = \tau$ then
        Prune $p\%$ parameters with smallest magnitude or prune $p\%$ output channels with smallest norm.
    end if
end for
Merge parameters $\boldsymbol{\theta} = \boldsymbol{W}_1 \odot \boldsymbol{W}_2$ and output network $f_{\boldsymbol{\theta}}$
```

Experiments

Experimental Setup

Datasets and Models. In our main evaluations, we evaluate the robustness of VGG-16 (Simonyan and Zisserman 2015) and Wide-ResNet-28-4 (Zagoruyko and Komodakis 2016) networks on CIFAR-10 (Krizhevsky et al. 2009) and SVHN (Netzer et al. 2011) image classification datasets as most of the baseline works report their results under this setting. In what follows, we also report robust accuracy with variants of ResNet (He et al. 2016) models.

Baselines. We compare against following adversarial robustness and compression baselines:

- **ADMM:** The method combines adversarial training and weight pruning by solving a constrained optimization problem with ADMM. It prunes connections that have the lowest weight magnitude (Ye et al. 2019).
- **HYDRA:** HYDRA performs an architecture search for a robust network with the desired pruning ratio with the least performance drop. It formulates pruning as an empirical risk minimization problem and exploits a learning approach to discover efficient sub-networks (Sehwag et al. 2020).
- **BCS:** It is an end-to-end robust sparse networks training method via Bayesian connectivity sampling. The approach trains a robust network under strict connectivity constraints throughout training (Özdenizci and Legenstein 2021).

Adversarial Training Settings: In our main comparisons with current state-of-the-art methods, we train networks using the TRADES (Zhang et al. 2019) robust loss with the

		VGG-16						WideResNet-28-4					
		90% Sparsity			99% Sparsity			90% Sparsity			99% Sparsity		
		HYDRA	BCS	**Ours**	HYDRA	BCS	**Ours**	HYDRA	BCS	**Ours**	HYDRA	BCS	**Ours**
CIFAR-10	Clean	80.5	80.9	**81.4**	73.2	74.0	**77.8**	83.7	**84.8**	83.7	75.6	**76.9**	75.9
	FGSM	55.6	55.3	**57.1**	46.5	46.5	**52.5**	61.1	60.0	**61.1**	51.0	49.5	**51.4**
	PGD^{50}	50.0	49.6	**51.4**	41.9	42.3	**46.8**	55.6	54.0	**55.7**	47.4	45.1	**47.6**
	PGD^{100}	49.9	49.5	**51.4**	41.8	42.1	**46.6**	55.5	53.9	**55.7**	47.3	44.9	**47.6**
	B&B$_\infty$	48.1	47.7	**49.6**	39.1	40.0	**44.3**	53.8	52.2	**53.9**	45.2	42.9	**45.3**
	AA$_\infty$	45.5	45.0	**47.1**	37.2	37.4	**42.1**	51.7	49.8	**51.7**	42.8	40.2	**42.9**
SVHN	Clean	89.2	89.4	**89.6**	84.4	86.4	**88.3**	**94.4**	92.8	90.9	88.9	**89.5**	89.3
	FGSM	63.1	**64.5**	63.5	57.1	58.4	**61.5**	**88.8**	70.0	72.9	**74.3**	63.1	69.4
	PGD^{50}	52.8	53.7	**54.4**	47.8	48.7	**51.8**	43.9	55.6	**56.3**	39.1	52.7	**55.8**
	PGD^{100}	52.4	53.3	**54.3**	47.5	48.3	**51.7**	38.3	55.1	**56.0**	36.5	52.4	**54.9**
	B&B$_\infty$	48.9	49.8	**50.6**	43.7	45.0	**48.3**	36.5	52.1	**54.7**	32.3	49.9	**51.3**
	AA$_\infty$	45.5	44.9	**46.1**	38.8	40.8	**44.2**	30.6	47.0	**53.9**	26.7	45.8	**45.8**

Table 1: Comparisons with HYDRA and BCS under white-box attacks. All models are trained with TRADES adversarial loss. We also used pseudo-labeled additional data samples on CIFAR-10 dataset for consistency with the compared methods.

trade-off parameter $\beta = 6$, which is consistent with baseline works. We implemente RST (Carmon et al. 2019) using the 500K pseudo-labeled data shared by the authors. We use the PGD^{10} attack with random starts to craft adversarial examples in the training process, with a maximum perturbation $\epsilon = 8/255$ and a step size $2/255$.

Attacking Manner: We evaluate the robustness on both white-box attacks and black-box attacks.

- *White-box Attacks:* We follow the conventional settings (Madry et al. 2018) for l_∞-norm bounded white box robustness evaluations. Perturbation budget for all datasets and adversarial attacks is $\epsilon = 8/255$. We take evaluations with a wider range of attacks: fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2015), PGD^{50}, PGD^{100} (Madry et al. 2018), the Brendel & Bethge attack (B&B$_\infty$) (Brendel et al. 2019) and the AutoAttack benchmark (AA$_\infty$) (Croce and Hein 2020).
- *Black-box Attacks:* The model architecture and parameters are unknown to the attacker in real scenarios. Hence we consider black-box threats where the attacker only has access to send limited queries to the model. We evaluate on Square Attack, which is a powerful query-based black-box threat (Andriushchenko et al. 2020).

Implementation Details. Optimization for all models is performed using SGD with momentum 0.9 and an initial learning rate of 0.1, which is divided by 10 at the 70-th and 85-th epoch. All models are trained for 100 epochs with a batch size of 128, and we perform pruning at the 30-th epoch. Network weights ($\boldsymbol{W}_1$ and $\boldsymbol{W}_2$) are initialized via Kaiming initialization (He et al. 2015). Besides, the weight decay factor is set to 2×10^{-4}.

Robustness Evaluation and Analysis

Comparisons with current state-of-the-art methods. We compare our approach with current state-of-the-art robustness-aware neural network pruning HYDRA and Bayesian-based sparse adversarial training method BCS. For fair comparisons, we particularly report comparisons for VGG-16 and WideResNet-28-4 models on CIFAR-10 and SVHN datasets as indicated in their papers. In Table 1, we evaluate the robustness of all models under several white-box attacks, as well as the AutoAttack ensemble benchmark. Overall, our method achieves better robustness under most settings and several consistent observations can be drawn: (1) Our weight-reparameterization pruning method performs much better when 99% of parameters are pruned, achieving gains up to 4.5 and 3.4 percentage points in robust accuracy under PGD^{100} attack with VGG-16, for CIFAR-10 and SVHN datasets, respectively. (2) Compared with the performance on clean samples, our method obtains more consistent robust improvement. This phenomenon is consistent with our claim that increasing the trainable parameters benefits robustness under high sparsity. (3)Notably, our method obtains state-of-the-art robustness scores under the AutoAttack benchmark, indicating its stronger comprehensive defense capability. As we adopted common adversarial training hyperparameter settings on all models and datasets, which may result in insignificant performance gains on some models, such as WideResNet-28-4. Actually, setting the weight decay to 5×10^{-4} can obtain better performance.

Network	ADMM	HYDRA	BCS	**Ours**
VGG-16	34.1	37.9	42.1	**46.3**
ResNet-18	36.1	41.6	44.8	**47.1**
ResNet-34	41.5	44.4	44.7	**48.8**
ResNet-50	42.2	45.3	46.9	**50.1**
WRN-34-10	44.6	51.1	49.9	**53.4**

Table 2: Comparisons of robust accuracy on CIFAR-10 at 99% sparsity for standard adversarial training.

Improved robustness across architectures. We further

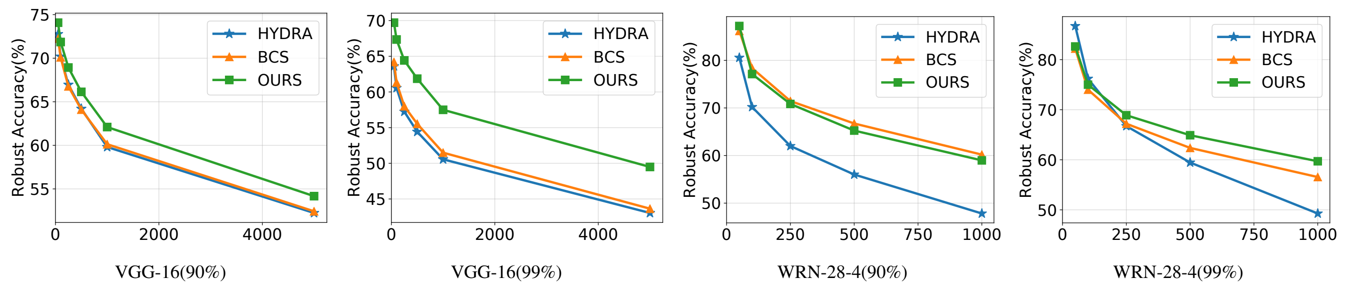

Figure 2: Black-box Square Attack evaluations with limited queries. We display robust accuracy with VGG-16 models on CIFAR-10 and WideResNet-28-4 models on SVHN. Sub-labels indicate the architecture (sparsity) for each evaluation.

take comparison with more network structures. For completeness, we also compare with earlier ADMM-based robust pruning method (Ye et al. 2019). Since Ye et al. (2019) focused on the standard adversarial training technique, we performed evaluations with the same technique for a fair comparison. Table 2 shows the robustness at 99% sparsity along with five network architectures on CIFAR-10. Our approach achieves superior results across all of them, outperforming the three compared methods on all architectures. The results show the scalability of our approach to different architectures under high compression.

Evaluating black-box robustness. We evaluate blockbox robustness for Twin-Rep compared to HYDYA and BCS. We test the query-based attack proposed by (Andriushchenko et al. 2020), using l_∞-norm perturbation limited with radius $\epsilon = 8/255$. Figure 2 represents robustness results, where the query access is limited to 5,000 for CIFAR-10 and 1,000 for SVHN. Again, the proposed approach achieves better robustness than other baselines at 99% sparsity. Compared with the results at 90% sparsity, our method achieves a larger gap with baselines at 99% sparsity. One possible explanation for this phenomenon is that our approach has minimized adversarial loss on extremely sparse weights, considering the weight distribution of the networks.

Comparison with robust pruning of original weights. We further conduct comparisons with pruning on original weights under two settings:

- Pruning with training: We perform pruning at the 30th epoch in the total 100 epochs training process.
- Pruning after pre-training (PT): We pre-train networks from scratch for 100 epochs and prune them without any fine-tuning process.

Figure 3 shows that our weight-reparameterized networks achieve stable robustness at high pruning ratios, while the performance drops significantly with original weights. This observation is consistent with our claim that our method can implicitly increase model sparsity while training for robustness, thus overcoming the contradiction between robustness and sparsity that the traditional penalty-based paradigm suffers. Notably, even without fine-tuning, our method maintains robustness when pruning 99% connections of a pretrained model, indicating it can train a network with both robustness and sparsity.

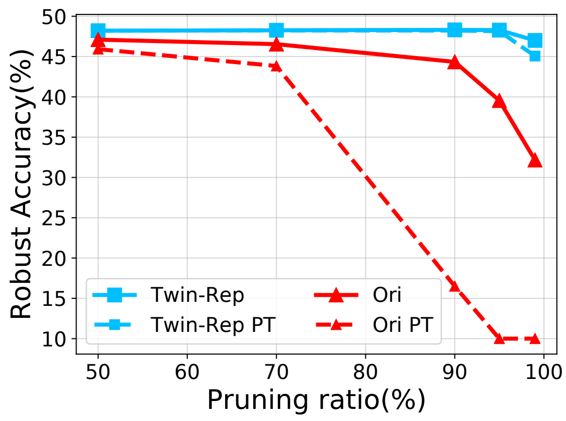

Figure 3: Comparison of robust accuracy with pruning on original weights and Twin-Rep weights. We display results with VGG-16 on CIFAR-10 under different pruning ratios.

Superior performance across datasets for both unstructured pruning and structured pruning. We further evaluate the performance of Twin-Rep and its variant Channel-Rep on more complicated datasets. Table 3 displays that both Twin-Rep and Channel-Rep achieve better robustness at high pruning ratios. We observe that Twin-Rep achieves higher accuracy for unstructured pruning while Channel-Rep performs better for structured pruning. Models trained with Twin-Rep are extremely sparse and suffer less damage from irregular pruning, thus perform much better at 99% sparsity. And Channel-Rep mixes information from all output channels, thus the remaining filters can extract abundant information.

Delving Deeper into Weight Reparameterization

Fusing weights before training. The weight distribution at initialization will be sparser than regular initialization if we fuse $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ in the Twin-Rep structure before training. However, after adversarial training, the weight distribution as well as performance becomes similar to networks without reparameterization. This phenomenon highlights the importance of the Twin-Rep structure for maintaining sparsity during adversarial training. We present more detailed results in Appendix C.

Dataset	Settings	base	unstructured pruning			structured pruning		
		0%	70%	90%	99%	50%	60%	70%
CIFAR-10	Baseline	**81.5/51.1**	80.2/49.1	78.0/48.2	71.2/40.5	78.0/47.4	74.6/43.7	69.7/37.1
	Channel-Rep	81.6/50.1	80.8/49.4	79.8/48.2	73.3/44.0	**79.1/48.6**	**78.9/47.8**	**77.4/47.7**
	Twin-Rep	81.2/50.0	79.5/**49.6**	**78.8/49.3**	**76.2/47.1**	77.8/48.1	76.7/47.5	74.5/45.9
CIFAR-100	Baseline	**52.0/27.2**	51.3/26.5	48.8/25.6	41.2/20.6	50.1/24.9	46.5/22.5	37.3/18.5
	Channel-Rep	51.3/26.7	51.0/26.4	49.3/25.7	43.4/21.2	50.6/25.2	**49.6/24.4**	**47.7/23.5**
	Twin-Rep	51.8/27.0	**51.9/26.9**	**51.3/26.7**	**49.3/24.9**	**51.0/25.2**	48.2/23.8	38.3/20.6
Tiny-ImageNet	Baseline	**45.6/21.4**	44.1/20.8	42.5/19.4	28.0/12.6	42.4/19.7	38.9/17.5	25.1/12.0
	Channel-Rep	45.0/20.9	44.2/20.6	43.4/19.9	33.4/14.8	**44.6/20.4**	**43.3/19.6**	**35.2/15.6**
	Twin-Rep	45.5/21.3	**44.9/21.0**	**44.5/20.8**	**38.5/17.6**	44.3/20.1	42.5/18.9	27.2/13.1

Table 3: Comparisons with two variants on CIFAR-10/100 and Tiny-ImageNet with ResNet-18. We report clean/robust accuracy for both unstructured pruning and structured pruning. Baseline is pruning on original weights.

Understanding the proposed weight reparameterization. A key driver behind the success of Twin-Rep is that most weights are very close to zero after merging $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. Therefore, it can effectively alleviate the performance degradation caused by the pruning operation. We wondered why the two factors do not get larger with training, eventually resulting in a weight distribution similar to the original structure. In Twin-Rep, the gradients are $\nabla_{\boldsymbol{\theta}}\mathcal{L} \odot \boldsymbol{W}_2$ for $\boldsymbol{W}_1$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L} \odot \boldsymbol{W}_1$ for $\boldsymbol{W}_2$. Considering that W_1 and W_2 are much less than 1, the gradients are also smaller than original structure. The effect of weight decay is more significant, thus weights in Twin-Rep are not larger than weights in the original structure. Figure 4 shows that the fifth and ninth deciles of the absolute value of W_1 first decrease and then stabilize, but deciles of original weights has an increasing process. This observation is consistent with our claim that weight decay is more pronounced. And the smaller gradients for Twin-Rep also support our analysis.

Why does weight reparameterization work well. Our approach does not change the optimization objective. It adds the desired property in the optimization process and does not disturb the optimal solution of the adversarial training objective. Twin-Rep maintains extremely sparse network weights during training that most element-wise products of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are small enough to be safely removed. Therefore, it achieves comparable robustness with 99% weights removed, much higher than the performance of the original structure. One concern may be whether the gradient will be too small to hinder optimization. Figure 4 shows that smaller gradients for reparameterized weights are expected, but still sufficient for network optimization. Moreover, Channel-Rep aggregates the information of all channels, thus achieving significant improvement compared to the original structure, when pruning a large number of channels.

Computational overhead and storage consumption. Twin-Rep brings almost no additional Computational overhead. The added element-wise product is insignificant compared with the original calculations. Take a convolution layer as an example, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^{D\times C\times K\times K}$. Suppose the output feature map is $\boldsymbol{A} \in \mathbb{R}^{D\times H\times W}$. The computation of a single convolutional layer is $H\times W\times D\times C\times K\times K$. However, Twin-Rep only adds $D\times C\times K\times K$ calculations. Considering the batch processing during training, the computational cost added by our method is negligible. Moreover, since intermediate features are the main memory consumption during training, our method does not bring much memory overhead. For instance, Twin-Rep consumes 1.22 times more memory than the original structure when the batch size is 128. More detailed results can be found in Appendix C.

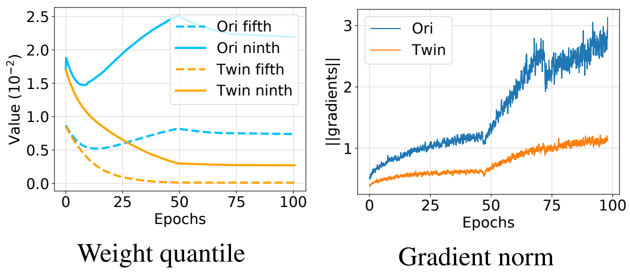

Figure 4: Weight quantiles and gradient norms of reparameterized weights $\boldsymbol{W}_1$ and original weights.

Conclusion

We propose a weight reparameterization approach to overcome the limitations of the traditional heuristic pruning strategy applied to adversarial training. We construct the training-time structure by decomposing the original weights into two tensors, which are merged by element-wise multiplication during inference. Our approach implicitly adds the sparsity constraint while increasing trainable parameters for better robustness under high sparsity. Experiments demonstrate that the proposed Twin-Rep pruning method out-performs recent state-of-the-art baselines across various datasets and network architectures. Besides, we also design a variant for better channel pruning robustness. An open research question is to combine weight reparameterization with advanced pruning methods to further improve the performance of pruned networks.

Acknowledgments

This work was sponsored by Beijing Nova Program under Grant No.Z211100002121141, and supported by the Innovation Funding of the Institute of Computing Technology (ICT) of Chinese Academy of Sciences under Grant No.E161040.

References

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2020. Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In *European Conference on Computer Vision*, 484–501.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, 274–283. PMLR.

Bellec, G.; Kappel, D.; Maass, W.; and Legenstein, R. 2018. Deep Rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*.

Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L. D.; Monfort, M.; Muller, U.; Zhang, J.; et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Brendel, W.; Rauber, J.; Kümmerer, M.; Ustyuzhaninov, I.; and Bethge, M. 2019. Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems*, 12841–12851.

Carmon, Y.; Raghunathan, A.; Schmidt, L.; Duchi, J. C.; and Liang, P. 2019. Unlabeled Data Improves Adversarial Robustness. In *Advances in Neural Information Processing Systems*, 11190–11201.

Chen, T.; Zhang, Z. A.; Wang, P.; Balachandra, S.; Ma, H.; Wang, Z.; and Wang, Z. 2022. Sparsity Winning Twice: Better Robust Generalization from More Efficient Training. *ArXiv*, abs/2202.09844.

Collins, M. D.; and Kohli, P. 2014. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, 2206–2216. PMLR.

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4690–4699.

Ding, X.; Hao, T.; Tan, J.; Liu, J.; Han, J.; Guo, Y.; and Ding, G. 2021. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4510–4520.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations, ICLR*.

Gui, S.; Wang, H. N.; Yang, H.; Yu, C.; Wang, Z.; and Liu, J. 2019. Model compression with adversarial robustness: A unified optimization framework. *Advances in Neural Information Processing Systems*, 32: 1285–1296.

Guo, Y.; Yao, A.; and Chen, Y. 2016. Dynamic Network Surgery for Efficient DNNs. In *Advances in Neural Information Processing Systems*, 1379–1387.

Guo, Y.; Zhang, C.; Zhang, C.; and Chen, Y. 2018. Sparse DNNs with Improved Adversarial Robustness. In *Advances in Neural Information Processing Systems*, 240–249.

Han, S.; Pool, J.; Tran, J.; and Dally, W. J. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems*, 1135–1143.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*.

Lee, N.; Ajanthan, T.; and Torr, P. H. S. 2019. Snip: single-Shot Network Pruning based on Connection sensitivity. In *International Conference on Learning Representations*.

Lin, J.; Rao, Y.; Lu, J.; and Zhou, J. 2017. Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2178–2188.

Louizos, C.; Welling, M.; and Kingma, D. P. 2018. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*.

Madaan, D.; Shin, J.; and Hwang, S. J. 2020. Adversarial neural pruning with latent vulnerability suppression. In *International Conference on Machine Learning*, 6575–6585. PMLR.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations, ICLR*.

Nakkiran, P. 2019. Adversarial robustness may be at odds with simplicity. *arXiv preprint arXiv:1901.00532*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Özdenizci, O.; and Legenstein, R. 2021. Training adversarially robust sparse networks via Bayesian connectivity sampling. In *International Conference on Machine Learning*, 8314–8324. PMLR.

Rakin, A. S.; He, Z.; Yang, L.; Wang, Y.; Wang, L.; and Fan, D. 2019. Robust sparse regularization: Simultaneously optimizing neural network robustness and compactness. *arXiv preprint arXiv:1905.13074*.

Ramanujan, V.; Wortsman, M.; Kembhavi, A.; Farhadi, A.; and Rastegari, M. 2020. What's Hidden in a Randomly Weighted Neural Network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11893–11902.

Sehwag, V.; Wang, S.; Mittal, P.; and Jana, S. 2020. HYDRA: Pruning Adversarially Robust Neural Networks. In *Advances in Neural Information Processing Systems*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Wang, L.; Ding, G. W.; Huang, R.; Cao, Y.; and Lui, Y. C. 2018. Adversarial robustness of pruned neural networks. *International Conference on Learning Representations*.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2019. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.

Ye, S.; Xu, K.; Liu, S.; Cheng, H.; Lambrechts, J.-H.; Zhang, H.; Zhou, A.; Ma, K.; Wang, Y.; and Lin, X. 2019. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 111–120.

You, H.; Li, C.; Xu, P.; Fu, Y.; Wang, Y.; Chen, X.; Baraniuk, R. G.; Wang, Z.; and Lin, Y. 2019. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 7472–7482. PMLR.