IEEE Transactions on Information Forensics & Security

FReP: Efficient Training of Adversarially Robust Sparse Networks

Journal:	*Transactions on Information Forensics & Security*
Manuscript ID	T-IFS-18890-2024
Manuscript Type:	Regular Paper
Date Submitted by the Author:	28-Jun-2024
Complete List of Authors:	Li, Chenhao; Institute of Computing Technology Chinese Academy of Sciences, Li, Lin; Institute of Computing Technology Chinese Academy of Sciences Zhang, Zhibin; Chinese Academy of Sciences, Institute of Computing Technology Qiu, Qiang; Institute of Computing Technology Chinese Academy of Sciences Guo, Jiafeng; Institute of Computing Technology Chinese Academy of Sciences Cheng, Xueqi; Institute of Computing Technology, China Academy of Sciences,
Subject Category Please select at least one subject category that best reflects the scope of your manuscript:	COMMUNICATION AND INFORMATION THEORETIC SECURITY
EDICS:	CIT-ADV-CLA-Adversarial classification < CIT-COMMUNICATION AND INFORMATION THEORETIC SECURITY

Author's Responses to Custom Submission Questions

Is this manuscript a resubmission of, or related to, a previously rejected manuscript?	No
If "Yes", specify the publication venue and manuscript ID of the previous submission and upload a supporting document detailing how the resubmission has addressed the concerns raised during the previous review. If this does not apply, type N/A.	N/A
Is this manuscript an extended version of a conference publication?	Yes
If "Yes", provide the full citation of the conference submission or publication. If this does not apply, type N/A.	Li, Chenhao, et al. "Learning adversarially robust sparse networks via weight reparameterization." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. No. 7. 2023.
Is this manuscript related to any other papers of the authors that are either published, accepted for publication, or currently under review, and that are not included among the references cited in the manuscript?	No
If "Yes", please list these papers below. Except for permitted preprints, explain why these papers are not included among the references cited in the manuscript and how they are different from the manuscript. Include any unpublished papers as "Supporting Documents". If this does not apply, type N/A.	N/A
What is the contribution of this paper, within the scope of Transactions on Information Forensics & Security?	Deep neural network applications in fields such as biometrics and surveillance often require compression to be deployed on resource-constrained devices. This article focuses on the robustness of pruned neural networks to provide adversarial robustness guarantees for deployed compression deep neural networks. The paper will be of great interest to readers in the field of lightweight neural network security.
Why is the contribution significant (What impact will it have)?	Improving the robustness of pruned models is beneficial for the application of deep learning models in resource-constrained and safety-critical scenarios. This is an important issue as more and more edge devices are deploying neural network applications. However, current robust pruning methods not only damage the robustness of the model, but are also very time-consuming, making them difficult to apply. The proposed technology addresses both of these issues, making it an advantageous choice for both lightweight and robustness when deploying models.
What are the three papers in the published literature most closely related to this paper? Please provide full citation details, including DOI references where possible.	1.Li, Chenhao, et al. "Learning adversarially robust sparse networks via weight reparameterization." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. No. 7. 2023.

	https://doi.org/10.1609/aaai.v37i7.26027. 2. Sehwag, Vikash, et al. "Hydra: Pruning adversarially robust neural networks." Advances in Neural Information Processing Systems 33 (2020): 19655-19666. 3. Lee, Byung-Kwan, Junho Kim, and Yong Man Ro. "Masking adversarial damage: Finding adversarial saliency for robust and sparse network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. https://doi.org/10.1109/CVPR52688.2022.01470.
What is distinctive/new about the current paper relative to these previously published works?	Compared to our previous work, the current paper not only provides additional theoretical and experimental analysis, but also improves the algorithm to significantly reduce the training cost of robust sparse models. Compared with the two related works, the current article not only develops a novel robust pruning algorithm, but also significantly reduces the time cost of training and pruning. The proposed pruning algorithm achieves higher robustness at the same pruning rate and shortens the entire pruning process by 7 to 10 times.

FReP: Efficient Training of Adversarially Robust Sparse Networks

Chenhao Li, Lin Li, Zhibin Zhang, Qiang Qiu, Jiafeng Guo, Xueqi Cheng

***Abstract*—The robustness of pruned models against adversarial attacks has gradually attracted attention, wherein pruning is combined with adversarial training to obtain sparse and robust models. While the recent robust pruning methods show promising results with acceptable drops in robustness, their performance deteriorates significantly under high sparsity regimes. Concurrently, the high computational cost associated with multi-step adversarial training hinders the practical application of these approaches. In this work, we present the fast adversarial training approach with weight reparameterization (FReP) to learn adversarially robust sparse networks efficiently. Firstly, FReP reparameterizes the original weights into the product of two factors, implicitly introducing a sparsity constraint. This structural reformulation enables the training of sparse robust networks, thereby mitigating the performance degradation caused by pruning. Secondly, FReP introduces a temporal ensemble method to exploit the potential of single-step adversarial training, significantly reducing training costs without sacrificing robustness. During inferring, we restore the original weight structure to obtain compact and robust networks. Extensive experiments on diverse datasets demonstrate that FReP achieves high robustness under extreme sparsity while significantly reducing training time compared to current methods. For example, when pruning 99% of the parameters of the ResNet-18 model, FRep outperforms current best methods by 5.2% under AutoAttack on CIFAR-10. Moreover, FReP accelerates the optimization process by $7\times$ and $10\times$ compared to state-of-the-art MAD and HYDRA, respectively, making it an advantageous choice for robust pruning without incurring excessive costs.**

***Index Terms*—Adversarial defense, robustness, network pruning, sparse training.**

I. INTRODUCTION

Recent progress in edge computing [1] calls for the necessity for deep neural network compression. As a popular compression and acceleration technique, pruning has been widely adopted to reduce computational overhead by removing redundant parameters in models [2]–[4]. Although compressed models achieve impressive performance on clean samples, deep neural networks (DNNs) are vulnerable to adversarial attacks with carefully crafted perturbations [5]–[7]. In safety-critical and resource-constrained environments, both robustness and compactness are simultaneously necessary. While the robustness of DNNs and the accuracy of pruned DNNs have been extensively studied in isolation, limited research has focused on the robustness of pruned networks.

The authors are with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences,Beijing 100864, China. E-mail: {lichenhao19b, lilin2020, zhangzhibin, qiuqiang, guojiafeng, cxq}@ict.ac.cn.

Some works have tried to generate sparse and robust networks by directly combining network pruning and adversarial training techniques [8]–[11]. These works inherit heuristic pruning algorithms developed for natural training, which drops connections in neural networks with the lowest weight. However, as adversarially trained networks generally demand more non-zero parameters than naturally trained networks [8], [9], these pruning technologies perform poorly under adversarial scenarios when a relatively high ratio is set. The main difficulty arises from the traditional penalty-based paradigm's inherent contradiction between robustness and sparsity. For example, training with a strong sparse 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 significant pruning damage. This problem motivates the design of our previous work [12].

Concurrently, the high computational cost of adversarial training makes these robust pruning methods unsuitable for practical pruning scenarios. Effective adversarial training involved multiple iterations of a gradient-based optimizer on the network's inputs to generate strong adversarial perturbations [13]–[15]. This optimization process requires significantly more computing resources than conventional training. Current adversarial pruning methods use multi-step adversarial training to preserve the adversarial robustness when pruning, which naturally brings much computational overhead. In contrast, single-step adversarial training methods can generate perturbations with much fewer costs. However, these approaches are not adopted in robust pruning because they may collapse under stronger multi-step attacks. [16], [17].

In order to achieve high sparsity and high robustness at an acceptable cost, we propose FReP, a fast training method that learns adversarially robust sparse networks with reduced cost. Specifically, FReP builds adversarially robust sparse models through the following pipeline:

- **Architecture Perspective:** We overcome the contradiction between robustness and sparsity by reparameterizing network architecture. As illustrated in Fig. 1a, we construct the training-time structure by decomposing the original weights into two matrices/tensors. Since the magnitudes of these two factors are small, their product implicitly increases the sparsity of the network, thus achieving low pruning damage. On the other hand, the parameters are only learned by adversarial loss, which has enough ability to achieve high robustness. Notably, we optimize the parameters together with the robust objective, and thus, the efficiency is similar to the training of the original structure. The trained model is restored to

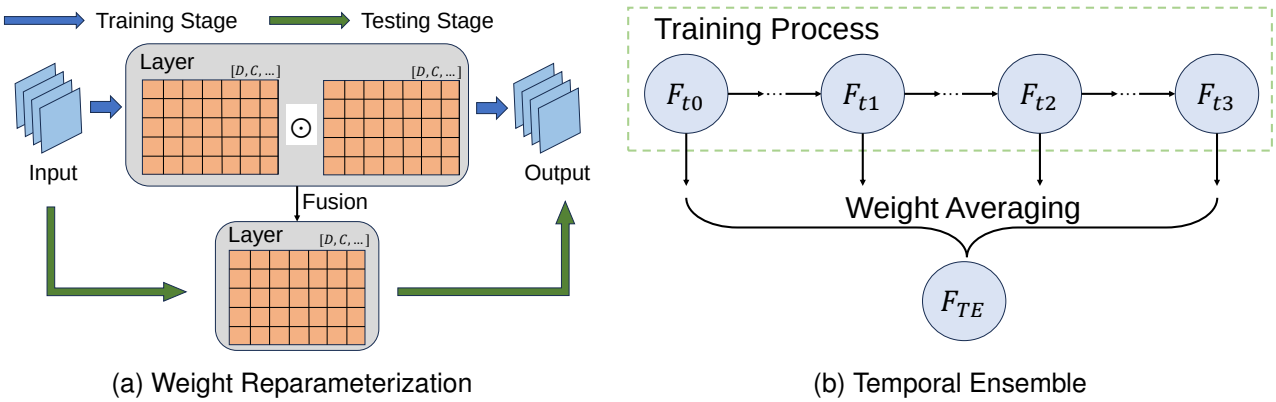


Fig. 1. The schematic diagram of FReP. (a) Network weights are reparameterized during training and restored during inference. (b) We integrate the weights of historical models obtained from the training process.

its original structure during inference, so model size and computational effort do not increase.

- **Optimization Perspective:** To avoid extreme computational costs while preserving the adversarial robustness, we propose the temporal ensemble method to exploit the potential of single-step adversarial training. As shown in Fig. 1b, we obtain individual models from the training history and integrate them via weight averaging. The intuition behind this approach is that ensemble have been found to boost adversarial robustness [18]–[20]. Meanwhile, temporal ensemble incurs little additional computational costs, so improved robustness can be gained almost free. We also adopt a smooth robust loss function to enhance the quality of aggregated individuals and further improve the robustness of the ensemble model.

Our empirical results demonstrate the effectiveness and efficiency of the proposed FReP. Compared with the state-of-the-art robust pruning methods, FReP achieves higher robustness for pruned models on various datasets and network structures. Furthermore, FReP accelerates the optimization process by $5\times \sim 13\times$ than current robust pruning methods, making it an advantageous choice for robust pruning without incurring high costs. In summary, our main contributions are as follows:

- A novel weight reparameterization technique for learning robust sparse models from our work [12]. We provide an extended theoretical analysis and empirical study of it effectiveness. It implicitly adds the sparsity without changing the robust training objective, achieving high compression ratios and robustness.
- We introduce a temporal ensemble method to extend our work [12], which learns robust sparse models more efficiently. By integrating historical models during training, it taps the potentials of models obtained from single-step adversarial training, achieving comparable robustness to multi-step training with significantly less training time.
- We conduct extensive experiments to validate the performance of FReP, of which the results demonstrate its superiority compared to baselines. Moreover, FReP significantly lowers the computational overhead of obtaining robust sparse models.

II. Related Work

A. Adversarial Robustness

The security issue of DNNs has received significant attention recently [21], [22]. By applying an imperceptible perturbation to clean data, the adversary can make the deep learning model predict incorrectly. The threat of adversarial attacks has motivated the development of various empirical defense mechanisms. One of the most effective defenses known to date is adversarial training [23], which utilizes adversarial examples to optimize the network with a min-max formulation. This method models a universal first-order attack through the inner maximization problem, while outer minimization represents training DNNs with adversarial examples:

$$\min_{\mathbf{W}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\widetilde{x}\in\mathcal{B}(x,\epsilon)} \mathcal{L}(\mathbf{W}, \widetilde{x}, y) \right], \tag{1}$$

where the adversarial example $\widetilde{x}$ is defined within the l_p-norm ball around the clean example x: $\mathcal{B}(x,\epsilon) := \{\widetilde{x} : ||\widetilde{x} - x||_p \leq \epsilon\}$, with a perturbation strength $\epsilon > 0$. In this paper, we follow the popular evaluation procedures to set $p = \infty$ [24], [25].

Adversarial training methods often leverage multiple forward and backward propagations to sufficiently approximate the inner maximization step, such as Projecting Gradient Descent (PGD) [23]. Following this, TRADES [13] quantified the trade-off between accuracy and robustness to achieve better robustness. MART [14] explicitly differentiated the distinctive influence of misclassified and correctly classified examples on the robustness during training. RST [26] proposed a robust self-training 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 larger network capacity to obtain strong adversarial robustness [27], limiting its use in resource-constrained scenarios.

B. Adversarial Robustness and Sparsity

The need to deploy DNNs in resource-constrained application systems has motivated the development of various network pruning approaches [28]. In parallel with studying network pruning in benign scenarios, achieving pruning in adversarial settings has also drawn attention. Several studies have investigated the relationship between the adversarial robustness and sparsity. Gou *et al.* [29] theoretically and experimentally demonstrated that adversarial robustness can be improved with an appropriate sparsity, whereas a very large sparsity tends to be harmful. However, they do not aim to prune robust models. Subsequently, several studies have attempted to compress robust networks using a magnitude-based pruning heuristic. Ye *et al.* [9] proposed a concurrent adversarial training and weight pruning scheme by solving a constrained optimization problem with an alternating direction method of multipliers. ATMC [10] constructed a similar constrained optimization formulation, where pruning and quantization were all integrated into the constraints. While ADMM-based pruning techniques are successful in natural training, they incur a significant performance drop when combined with adversarial training. Madaan *et al.* [30] proposed a Bayesian framework to prune highly vulnerable features for better robustness. RSR [8] applied the l_1 sparse-induced penalty in adversarial training to produce robust sparse models. Nevertheless, these early attempts fail to achieve competitive robustness at high pruning rates.

Some recent works constitute state-of-the-art methods for achieving robust networks at high sparsity. HYDRA [31] exploited the mask learning approach to search robust subnetworks in a pre-trained robust, dense network. Later, Özdenizci *et al.* [32] proposed a robust sparse training method that trained a robust network under strict sparse connectivity constraints. MAD [33] employed second-order information of adversarial loss to estimate adversarial saliency for model parameters and determine which parameters can be pruned without weakening adversarial robustness. CSTAR [34] gradually imposed the desired low-rankness on models to perform structured pruning of robust models. Despite achieving adversarial robustness on sparse models, these methods require significant computational resources. Adversarial training typically requires several times more computation than natural training, making robust pruning techniques based on it also time-consuming.

C. Efficient Adversarial Training

In order to reduce the computational overhead of adversarial training process, some fast training techniques have been proposed to efficiently train robust models by reducing redundant computation in multi-step training. Shafahi *et al.* [35] recycled the gradient information for model parameters to reduce computation. Zhang *et al.* [36] reused the gradients of all layers except the first layer to accelerate the training procedure. Although these methods reduce the generation overhead, the less robustness of models is also often highlighted.

Another line of work is devoted to improving the Fast Gradient Sign Method (FGSM) [37]. Models trained using FGSM were previously believed to be overwhelmingly susceptible to multi-step attacks due to the Gradient Masking effect [38]. Singel-step adversarial training methods are contingent on the local linearity assumption of the loss surface, which is often compromised during training, leading to invalid adversaries and false robustness [16], [39]. Wong *et al.* [39] found that FGSM could become effective with uniform random initialization to adversarial perturbations. Regularization methods have also been widely studied to improve the quality of the FGSM solution by smoothing the loss surface [40]–[43]. GradAlign maximized the cosine similarity of gradients of clean adversarial inputs to prevent catastrophic overfitting [41]. GAT minimized the squared l_2 distance between the softmax outputs of clean and adversarial images to find more suitable gradient directions [42]. NuAT replaced the l_2 norm with the Nuclear norm in GAT to utilize batch statistics for achieving function smoothing [43]. Although single-step defenses gain noticeable improvements, to our knowledge, no robust pruning method uses them to provide robustness.

III. METHOD

In this section, we propose FReP, which performs fast robust pruning under high sparsity regimes. We first show our observation that penalty-based pruning suffers from a contradiction between robustness and sparsity. Next, we discuss the details of the proposed weight reparameterization strategy, which implicitly introduces sparsity constraints without sacrificing robustness. Then, we provide a temporal ensemble method to effectively train a network with high adversarial robustness. Finally, we introduce a robust pruning algorithm based on FReP.

A. Preliminaries: Pruning Adversarailly Robust Networks

In natural training with only benign examples, many redundant parameters in the model can be pruned without weakening accuracy. However, adversarially trained models are less sparse than naturally trained models [8], [9], making it harder to prune an adversarially trained model. Fig. 2a shows the weight distribution of ResNet-18 models trained trained with natural training and adversarial training. It can be observed that the adversarially trained model is denser, and thus pruning causes more performance drops. We also report the degradation in performance at different pruning rates. As shown in Fig. 2b, robustness decreases faster than clean accuracy with increasing sparsity. Since adversarially trained networks generally demand more non-zero parameters, pruning algorithms developed for naturally trained models perform poorly on adversarially trained models. We further study the impact of the sparse-induced penalty on natural training and adversarial training. Fig. 2c experimentally demonstrates that a stronger penalty hurts more with robust training.

The above observations show that the traditional penalty-based pruning paradigm suffers from a contradiction between robustness and sparsity under adversarial settings. We further analyze this issue from a theoretical perspective. Let $\mathbf{W}$ be a set of trainable parameters, and $\mathcal{L}_{\text{perf}}$ be the performance-related loss function (e.g., cross-entropy). The traditional

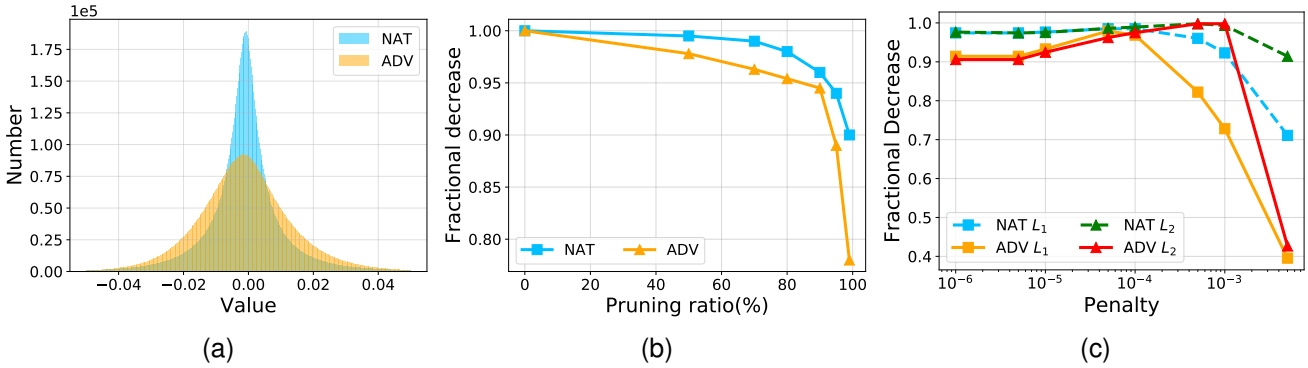


Fig. 2. Illustration of contradiction between robustness and sparsity. (a) The weight distribution of ResNet-18 models trained with natural training and adversarial training on the CIFAR-10 dataset. (b) Fraction decrease in accuracy for ResNet-18 trained with natural training and robustness for ResNet-18 trained with adversarial training. (c) Fraction decrease in performance for networks trained with natural training and adversarial training with different weight penalty factors.

paradigm adds a sparse-induced penalty term $\mathcal{L}_{\text{penalty}}$ by a pre-defined strength factor λ:

$$\mathcal{L}_{\text{total}}(\mathbf{W}, x, y) = \mathcal{L}_{\text{perf}}(\mathbf{W}, x, y) + \lambda \mathcal{L}_{\text{penalty}}(\mathbf{W}), \quad (2)$$

where the common forms of $\mathcal{L}_{\text{penalty}}(\mathbf{W})$ include l_1 and l_2-regularization. With $g(w)$ as the gradient, we take the derivative of with respect to a specific weight w:

$$g(w) = \frac{\partial \mathcal{L}_{\text{perf}}(\mathbf{W}, x, y)}{\partial w} + \lambda \frac{\partial \mathcal{L}_{\text{penalty}}(\mathbf{W})}{\partial w}, \quad (3)$$

Suppose w resides near the local optima and has a non-zero value. The first term in Eq. 3 is close to 0, but the second is not, so w is pushed towards 0. If w significantly impacts performance, the performance-related gradient will compete with the penalty gradient and intend to maintain the weight magnitude. Otherwise, w will keep moving towards 0 under the effect of the penalty gradient. The experimental results in Fig. 2 show that the competition between the two gradient terms in adversarial training is more intense, leading to the pruning dilemma. On the one hand, with a mild penalty, we cannot achieve high prunability, because most of the weights not small enough for low-damage pruning. On the other hand, a strong penalty deviates the parameters from the optima of the objective function and hinders training. To this end, we propose our weight reparameterization method to implicitly add the sparsity in the next subsection.

B. Weight Reparameterization Strategy

Instead of explicitly applying a sparse-induced penalty, we implicitly add a sparsity constraint through weight reparameterization. Specifically, we perform a simple mapping on trainable weights $\mathbf{W}$, which reparameterizes network parameters to the Hadamard product of two matrices/tensors. For each layer l, we use the following parameters during the training process:

$$\mathbf{W}^l = \mathbf{W}_1^l \odot \mathbf{W}_2^l, \quad (4)$$

where $\mathbf{W}_1^l$ and $\mathbf{W}_2^l$ have the same shape as the original parameters. Considering the empirical observation that weights are distributed over a small range, most products of the reparameterized weights will be very small. To verify this conjecture, we first analyze the impact of weight reparameterization on model training. From an optimization perspective, the update rule for $\mathbf{W}^l$ is:

$$\begin{aligned}&(\mathbf{W}_1^l - \eta \nabla_{\mathbf{W}^l}\mathcal{L} \odot \mathbf{W}_2^l) \odot (\mathbf{W}_2^l - \eta \nabla_{\mathbf{W}^l}\mathcal{L} \odot \mathbf{W}_1^l) \\ \approx &\mathbf{W}^l - \eta \nabla_{\mathbf{W}^l}\mathcal{L} \odot (\mathbf{W}_1^l \odot \mathbf{W}_1^l + \mathbf{W}_2^l \odot \mathbf{W}_2^l),\end{aligned} \quad (5)$$

where $\nabla_{\mathbf{W}^l}\mathcal{L}$ is the gradient of the loss function with respect to $\mathbf{W}^l$, and η is the learning rate. We ignore the term $\eta^2 \nabla_{\mathbf{W}^l}\mathcal{L} \odot \nabla_{\mathbf{W}^l}\mathcal{L} \odot \mathbf{W}^l$ as it is relatively small. Assuming that the absolute values of $\mathbf{W}_1^l$ and $\mathbf{W}_2^l$ are the same initially, we have:

$$|\mathbf{W}_1^l - \eta \nabla_{\mathbf{W}^l}\mathcal{L} \odot \mathbf{W}_2^l| = |\mathbf{W}_2^l - \eta \nabla_{\mathbf{W}^l}\mathcal{L} \odot \mathbf{W}_1^l| \quad (6)$$

The absolute values of the two weights remain equal after a single update. It can be easily proved that the absolute values of the two weights are always equal during training. Therefore, the update rule in Eq. 5 can be rewritten as:

$$\mathbf{W}^l - 2\eta |\mathbf{W}^l| \odot \nabla_{\mathbf{W}^l}\mathcal{L}, \quad (7)$$

The learning rate of each parameter is weighted by its magnitude. Small weights have a small learning rate, which limits their growth during training, while large weights have the opportunity to be updated to a small value. In addition, the fusion of the two weights is also sparser at initialization. It can be predicted that we will eventually train a sparse model. Therefore, the weight reparameterization method implicitly increases network sparsity.

We further relax the previous assumption and no longer require the absolute values of the two weights to be equal at initialization, allowing both weights to be randomly initialized. We plot the changes in weight magnitude during training for experimental verification. As shown in Fig. 3a, the magnitude of reparameterized weights decreases significantly and then maintains a small value. Since most weights are small enough to be safely removed after merging $\mathbf{W}_1$ and $\mathbf{W}_2$, our method can effectively alleviate the performance degradation caused by pruning. On the contrary, the original weights have an increasing stage and the final value is also much larger. This

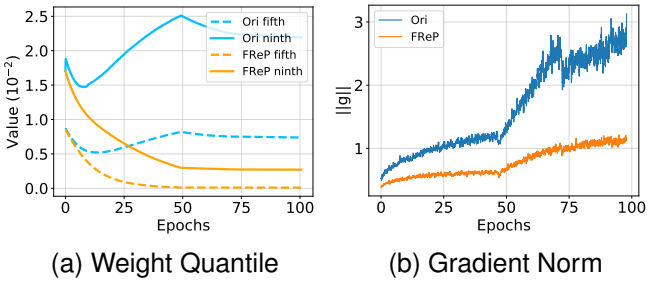


Fig. 3. Weight quantiles and gradient norms of reparameterized weights and original weights.

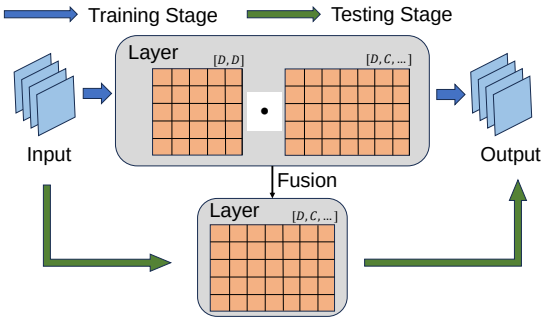


Fig. 4. Schematic diagram of weight reparameterization in FReP-CH.

result confirms that our proposed weight reparameterization method implicitly increases network sparsity.

Besides, a mild l_2-regularization may have positive effects on robustness. We also apply weight decay to both reparameterized weights:

$$\begin{aligned}&(\mathbf{W}_1^l - \lambda \mathbf{W}_1^l) \odot (\mathbf{W}_2^l - \lambda \mathbf{W}_2^l) \\ \approx &\mathbf{W}^l - 2\lambda \mathbf{W}^l,\end{aligned} \tag{8}$$

It is equivalent to adding weight decay to the original weights, which usually brings positive effects on performance and generalization.

One might worry that the additional sparsity constraint hinders model optimization. Considering that W_1 and W_2 are much less than 1, the gradients $\nabla_{\mathbf{W}}\mathcal{L} \odot \mathbf{W}_2$ and $\nabla_{\mathbf{W}}\mathcal{L} \odot \mathbf{W}_1$ may be too small to update the parameters. To check this risk, we plot the gradient norm of the original weights and one of the reparameterized weights. As shown in Fig. 3b, smaller gradients are expected for reparameterized weights, but still sufficient for network optimization. Moreover, our approach only optimizes the robust objective without adding any term, thus leading to little damage to robustness.

Weight Reparameterization Variant: We also design another form of weight reparameterization for better channel sparsity, named FReP-CH. Let D be the number of output channels. For each layer l, we add a $D \times D$ matrix and fuse the two weights using matrix multiplication:

$$\mathbf{W}^l = \mathbf{W}_1^l \cdot \mathbf{W}_2^l, \tag{9}$$

where $\mathbf{W}_1^l \in \mathbb{R}^{D \times D}$ and $\mathbf{W}_2^l$ has the same shape as the original parameters. FReP-CH does not lead to extremely sparse weight distribution but integrates the feature extraction capabilities of each channel. Note that the linear combination of convolution weights is equivalent to the linear combination of pre-activation output features. The matrix multiplication weight fusion operation aggregates the information of all channels. This property allows the remaining channels to contain sufficient information, thus FReP-CH can maintain the representational capacity of layers after pruning most output channels.

C. Efficient Robust Optimization with Temporal Ensemble

The process of training robust networks with reparameterized weights can be formalized as follows:

$$\min_{\mathbf{W}_1 \odot \mathbf{W}_2} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\widetilde{x} \in \mathcal{B}(x,\epsilon)} \mathcal{L}(\mathbf{W}_1 \odot \mathbf{W}_2, \widetilde{x}, y) \right], \tag{10}$$

where $\mathcal{L}$ is the objective function of model with parameters $\mathbf{W}_1 \odot \mathbf{W}_2$. The inner maximization is typically solved by projected gradient descent (PGD), which generates perturbations through a large number of iterations:

$$\widetilde{x}^{t+1} = \prod_{\mathcal{B}(x,\epsilon)} (\widetilde{x}^t + \alpha \cdot \text{sign}(\nabla_{\widetilde{x}^t} \mathcal{L}(\mathbf{W}_1 \odot \mathbf{W}_2, \widetilde{x}^t, y))), \tag{11}$$

where α is the step size, $\text{sign}(\cdot)$ returns the sign of the vector, Π is the projection operator on l_∞ norm-ball $B(x, \epsilon)$ around x with radius ϵ, and $\widetilde{x}^t$ denotes the adversarial example at the t-th PGD step. Although multi-step iteration is beneficial to find the strongest attack example, it incurs significant computational cost. To save the optimization cost, we generate adversarial examples with only a single propagation:

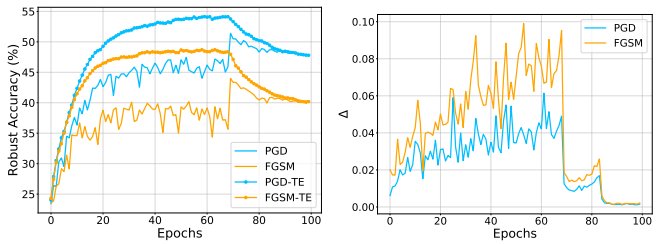
$$\widetilde{x} = \prod_{\mathcal{B}(x,\epsilon)} (x + \epsilon \cdot sign(\nabla_x \mathcal{L}(\mathbf{W}_1 \odot \mathbf{W}_2, x, y))), \tag{12}$$

Nevertheless, simply solving the inner maximization with single-step optimization cannot achieve competitive robustness compared to multi-step methods. We intend to improve robustness through model ensemble, which has attracted attention for its strong defense against adversarial examples [18], [19], [44]. The conventional ensemble method of training multiple diverse models for joint predictions fails to meet our goal of reducing training costs. Instead of training multiple models, we utilize weight averaging to aggregate historical models on the trajectory of a single training. One additional bonus of weight averaging is to provide a flatter loss landscape, which indicates better robustness [15], [40].

Considering that models obtained at a relatively late training stage are generally more robust, giving historical models the same ensemble weight may not be a good strategy. Therefore, we implement a temporal ensemble by an exponential moving average of the model parameters with a decay rate τ:

$$\mathbf{W}_{TE} = \tau \cdot \mathbf{W}_{TE} + (1 - \tau) \cdot \mathbf{W}, \tag{13}$$

During the initial training period, the models are not adequately learned, which may harm the ensemble model. Therefore, we accelerate the decay of the early models, denoted as



(a) (b)

Fig. 5. The deterioration of ensemble model in adversarial training. We plot robust accuracy as well as the difference between ensemble models and SGD-optimized models.

$\tau = min(\tau_0, \frac{i}{i+c})$, where c is a constant factor controlling the length of warm-up periods.

Simply integrating weight averaging into the adversarial training working flow suffers from performance degradation in the last stage of training. Due to the robust overfitting [45] in the adversarial training of individuals, the performance of the ensemble model also degrades. As shown in Fig. 5a, regardless of whether it is applied to single-step or multi-step adversarial training, the performance of the ensemble model degenerates to the ordinary model at the end of optimization. We wonder why the ensemble model assembled from overfitting individuals performs poorly while inferior models can still be aggregated to produce a superior one in the middle of the training stages.

From a model ensemble perspective, we guess that the deterioration of the ensemble model is due to homogenization at later stages. As the learning rate decreases in the late stage of training, model weights change little per iteration, and the historical models become homogeneous. Therefore, the ensemble model loses the superiority of the ensemble. To provide empirical evidence of the homogenization of models at the last periods, we visualize the difference between the ensemble model and SGD-optimized model along the training process, denoted as:

$$\Delta = \frac{1}{||\mathcal{D}||} \sum ||f_{\mathbf{W}_{TE}}(x, y) - f_{\mathbf{W}}(x, y)||_2^2, \qquad (14)$$

where $f_{\mathbf{W}}(x, y)$ is the model outputs. As shown in Fig. 5b, Δ increases in the early stage until the learning rate decreases (at the 70th and 85th epoch). Then, the ensemble model degenerates to the SGD-optimized model, which indicates that the integrated models are homogeneous. Based on this observation, a simple but effective strategy is to reduce integration frequency when the learning rate decreases. We update the ensemble model every 100 iterations at the beginning and divide the update frequency by 10 as the learning rate decreases. As aggregated models accumulate enough updates and become different, the ensemble model can gain benefits again.

While weight averaging improves the robustness of single-step adversarial training, it still has a large performance gap with multi-step methods. The gap is mainly due to the low quality of the integrated models. The validity of the solution in Eq. 12 relies on the local linearity assumption, and the following condition should be satisfied:

$$\mathcal{L}(\mathbf{W}, \widetilde{x}, y) - \mathcal{L}(\mathbf{W}, x, y) \approx (\nabla_x \mathcal{L})^T (\widetilde{x} - x), \qquad (15)$$

However, as training progresses, the loss surface becomes highly curved and non-linear. This phenomenon is known as gradient masking, which causes adversarial perturbations to deviate from the solution of inner maximization and the resulting models are not robust to multi-step attacks.

In order to enhance the quality of the integrated individuals, we use the self-consistent robust error (SRE) proposed by Pang *et al.* [46], which substitutes KL divergence with squared error in TRADES [13]:

$$\mathcal{L}_{SRE} = ||f_{\mathbf{W}}(x) - y||_2^2 + \lambda ||f_{\mathbf{W}}(\widetilde{x}) - f_{\mathbf{W}}(x)||_2^2 \qquad (16)$$

The first term corresponds to the mean squared error loss on clean images, and the second term corresponds to the squared l_2 distance between the softmax output of clean images and their corresponding adversaries. Their experiments have verified the effectiveness of this loss function in multi-step adversarial training. Moreover, the squared l_2 distance has been shown to serve as a relaxation term to produce a smoother loss surface [10]. Therefore, we apply it to single-step defense for better robustness.

D. FRePP-Based Robust Pruning

We apply the above temporal ensemble approach to reparameterized networks to train adversarially robust sparse networks with low computational costs. For the purpose of reducing the time consumption of the pruning process, we focus on the combination of run-time pruning approaches with our proposed FReP, as they achieve good pruning results without additional fine-tuning. We use a simple heuristic pruning algorithm which removes weights of the least magnitude. Despite the simplicity, it achieves highly competitive results when combined with our approach.

Our FReP-based robust pruning adopts a global pruning strategy. For unstructured pruning, we select top-k weights with the highest absolute values and set others to zero. For structured pruning, we measure the importance of output channels using the l_2 norm. Taking the convolutional layer as an example, we define the metric as follows:

$$||\mathbf{W}_{i:}||_2 = \sqrt{\sum_{c=1}^{C} \sum_{p=1}^{K} \sum_{q=1}^{K} \mathbf{W}_{i,c,p,q}^2}, \quad \forall 1 \leq i \leq D, \qquad (17)$$

where C and D are the number of input and output channels, respectively, and K is the kernel size. The metric for linear layers can be computed similarly. We select channels with the highest top-k values and zero out others.

The training and pruning procedure based on FReP is outlined in Algorithm 1. First, we add a uniformly random noise with a radius ϵ to the clean image. Next, we generate perturbation by maximizing self-consistent robust error using single-step optimization. We equally optimize the reparameterized weights by Eq. 16 and update the ensemble model every f_e iterations. After several training iterations, we perform

Algorithm 1 Robust Pruning Based on FReR

Input: Dataset $\mathcal{D}$, neural network $f_{\mathbf{W}}$ with reparameterized weights $\mathbf{W}_1$ and $\mathbf{W}_2$, perturbation strength ϵ, learning rate η, pruning ratio $p\%$, total iterations T, ensemble frequency f_e, pruning timing t_p.
Output: a pruned robust network

Init: Initialize the ensemble model $f_{\mathbf{W_{TE}}}$ with $f_{\mathbf{W}}$
for $t = 1$ **to** T **do**
  Sample a mini-batch $(x, y) \subset \mathcal{D}$
  $\widetilde{x} = x + Uniform(-\epsilon, \epsilon)$
  Generate adversarial samples:
  $\widetilde{x} = \Pi_\epsilon(\widetilde{x} + \epsilon \cdot sign(\nabla_{\widetilde{x}} \mathcal{L}_{SRE}(\mathbf{W}, \widetilde{x}, y)))$
  Compute robust loss:
  $\mathcal{L}_{SRE} = ||f_{\mathbf{W}}(x) - y||_2^2 + \lambda ||f_{\mathbf{W}}(\widetilde{x}) - f_{\mathbf{W}}(x)||_2^2$
  update $\mathbf{W}_1$ and $\mathbf{W}_2$:
  $\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \eta \nabla_{\mathbf{W}_1} \mathcal{L}_{SRE}(\mathbf{W}, \widetilde{x}, y)$,
  $\mathbf{W}_2 \leftarrow \mathbf{W}_2 - \eta \nabla_{\mathbf{W}_2} \mathcal{L}_{SRE}(\mathbf{W}, \widetilde{x}, y)$
  if $t \ \% \ f_e == 0$ **then**
    $\mathbf{W}_{TE} = \tau \cdot \mathbf{W}_{TE} + (1 - \tau) \cdot \mathbf{W}$
  end if
  if $t == t_p$ **then**
    Prune $p\%$ parameters in model $f_{\mathbf{W}}$ and synchronize to ensemble model $f_{\mathbf{W_{TE}}}$.
  end if
end for
Merge parameters and output the pruned network $f_{\mathbf{W}_{TE}}$

network pruning and synchronize the pruning scheme to the ensemble model. Then, we only update the selected weights or channels during the remainder of the training schedule. Finally, the reparameterized weights are fused to restore the original structure.

IV. Experiments

In this section, we present the empirical results to demonstrate the effectiveness and efficiency of the proposed method. Firstly, we specify the detailed experimental setups in Sec. IV-A. We then provide the evaluation of FReP on different datasets and network architectures in Sec. IV-B. Afterwards, we analyzed the effect of weight reparameterization and temporal ensemble in Sec. IV-C and Sec. IV-D. Lastly, we provide a rich collection of ablation studies in Sec. IV-E.

A. Experimental Settings

1) Datasets: We conduct exhaustive experiments on CIFAR-10 and SVHN, which equally have 32×32 dimensional images with 10 classes. For a more comprehensive evaluation, we also conduct evaluations on CIFAR-100 and Tiny-ImageNet in subsequent experiments.

2) Baselines: We compare against the following adversarial robustness and compression baselines:

- LWM [11] directly prunes connections with the lowest weight magnitude in the adversarially robust network. We implemented a version of during-training pruning instead of the original 3-step pruning, which achieves better robustness.
- ADMM [9] employs an alternating direction method of multipliers pruning framework to solve a constrained robust optimization problem.
- HYDRA [31] performs mask learning for a robust network with the desired pruning ratio. It optimizes a robust empirical risk minimization problem and exploits a learning approach to discover efficient sub-networks.
- MAD [33] employs second-order information of adversarial loss to estimate adversarial saliency for model parameters. It optimizes real-number masks in robust pretrained networks.

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

- *White-box Attacks:* We follow the conventional settings 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: FGSM [37], 50-step PGD [23], 50-step momentum method (MIM) [47], and the AutoAttack (AA) [48], which is an ensemble of four attacks.
- *Black-box Attack:* The model architecture and parameters are unknown to the attacker in real scenarios. Hence we also 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 [49].

Implementation Details. All models are optimized using SGD with a momentum of 0.9 and weight decay of 2×10^{-4}. The learning rate is initialized to 0.1 and is divided by 10 at the 70-th and 85th epochs. All models are trained for 100 epochs with a batch size of 128, and are pruned at the 30th epoch. Network weights ($\mathbf{W}_1$ and $\mathbf{W}_2$) are randomly initialized with no additional restrictions.

B. Performance Evaluation and Analysis

1) Comparisons with Current State-of-the-art Methods: To validate the effectiveness of FReP, we compare adversarial robustness with strong baselines for robust pruning. We particularly report comparisons for VGG-16, ResNet-18 and WideResNet-28-10 models on CIFAR-10 and SVHN datasets as indicated in the papers [31], [33]. As shown in Tab. I, our method achieves consistently better robustness across three models and two datasets, demonstrating the stability of our method. Besides, several observations can be drawn: (1) Compared with the best baseline, FReP achieves gains up to 1.5 and 0.6 percentage points in robust accuracy under AutoAttack for CIFAR-10 and SVHN each on average of three networks. The result indicates that our method has stronger comprehensive defense capabilities. (2) Compared with the performance on clean samples, our method obtains more consistent robust improvement, indicating that our method is more conducive to robustness under high sparsity. (3) For WideResNet-28-10 models, our method has smaller improvements over the baselines. This result may be because complex models have more redundancy and the effect of implicitly introducing sparsity is reduced. Overall, FReP shows less performance

TABLE I
COMPARISON OF ADVERSARIAL ROBUSTNESS AT A 90% (SPARSITY) PRUNING RATIO ON CIFAR-10 AND SVHN DATASETS WITH VGG-16, RESNET-18, AND WIDERESNET-28-10. WE ALSO REPORT THE PERFORMANCE OF UNPRUNED MODELS TRAINED WITH STANDARD ADVERSARIAL TRAINING (AT).

Model	Method	CIFAR-10						SVHN					
		Clean	FGSM	PGD	MIM	Square	AA	Clean	FGSM	PGD	MIM	Square	AA
VGG-16	AT (0%)	79.5	56.7	49.5	49.7	51.4	44.6	92.5	66.1	54.4	54.2	51.0	47.6
	LWM	78.0	50.5	45.8	45.9	47.2	39.4	88.4	62.2	50.6	50.5	45.7	43.8
	ADMM	79.3	52.3	45.9	45.8	47.6	39.7	**91.8**	62.4	50.8	50.7	44.2	43.3
	HYDRA	76.6	50.8	46.7	46.8	48.5	40.7	87.9	63.8	52.3	52.2	47.2	45.1
	MAD	78.9	50.4	47.4	47.0	47.2	40.4	88.2	62.8	51.1	51.0	46.5	44.8
	FReP (**ours**)	**80.5**	**56.7**	**48.9**	**48.8**	**51.6**	**43.2**	89.8	**64.5**	**53.1**	**53.0**	**48.3**	**46.0**
ResNet-18	AT (0%)	81.5	57.4	51.1	51.3	54.2	46.9	93.2	73.5	55.4	55.6	53.9	49.0
	LWM	78.0	56.0	48.2	48.5	50.2	43.5	90.4	65.8	52.9	53.2	49.7	46.2
	ADMM	**81.4**	55.8	47.2	47.2	49.2	42.8	**92.7**	65.2	52.2	52.5	48.5	45.5
	HYDRA	80.9	56.1	48.6	48.5	50.9	43.6	90.9	66.8	53.4	53.6	50.6	46.7
	MAD	80.1	56.4	47.9	48.0	49.4	43.0	90.2	65.4	52.7	52.9	49.8	46.1
	FReP (**ours**)	80.4	**57.3**	**49.7**	**49.8**	**52.2**	**44.7**	91.3	**68.1**	**54.0**	**54.7**	**52.6**	**47.3**
WRN-28-10	AT (0%)	87.1	61.7	54.2	54.4	55.9	51.3	93.7	75.6	57.6	57.9	55.6	52.7
	LWM	85.6	60.3	52.0	52.0	53.9	47.8	91.1	67.9	55.0	55.1	52.8	48.4
	ADMM	**87.4**	59.8	51.6	51.7	53.4	47.4	**93.5**	70.6	54.8	55.0	52.5	47.8
	HYDRA	85.9	60.6	53.2	53.1	54.3	48.1	92.6	69.0	56.4	56.6	54.4	49.2
	MAD	84.6	60.9	52.5	52.3	52.8	47.6	92.0	71.9	55.6	55.8	53.1	48.6
	FReP (**ours**)	85.0	**61.1**	**53.5**	**53.4**	**55.3**	**48.9**	93.1	**72.3**	**56.8**	**57.1**	**55.0**	**49.7**

TABLE II
COMPARING ADVERSARIAL ROBUSTNESS AT 99% SPARSITY ON CIFAR-10 ACROSS DIFFERENT ARCHITECTURES.

Method	VGG-16			ResNet-18			ResNet-34			ResNet-50			WRN-28-10		
	Clean	PGD	AA	Clean	PGD	AA	Clean	PGD	AA	Clean	PGD	AA	Clean	PGD	AA
LWM	61.8	36.4	31.5	71.2	40.5	34.9	73.4	44.0	38.4	74.2	44.8	39.6	75.3	45.9	40.2
ADMM	55.2	34.1	28.9	58.7	36.1	30.4	68.8	41.5	35.5	69.1	42.2	36.0	70.6	42.8	37.0
HYDRA	59.9	37.9	32.5	69.0	41.6	35.9	71.8	44.4	39.0	73.9	45.3	40.1	75.6	46.2	40.7
MAD	61.5	41.0	35.0	70.1	41.5	35.4	71.5	43.8	38.2	72.8	44.3	38.4	73.2	45.0	39.8
FReP (**ours**)	**78.3**	**46.5**	**40.3**	**78.9**	**47.1**	**41.1**	**81.2**	**50.8**	**44.7**	**81.6**	**51.0**	**44.8**	**82.5**	**51.8**	**45.5**

degradation of the robustness compared to the performance of the full-parameter model.

To validate the scalability of our approach to different architectures, we conduct experiments on more network structures under high compression. Tab. II shows the robustness at 99% sparsity along with five network architectures. Our approach achieves superior results across all of them, outperforming the four compared methods on all architectures. Compared with the results at 90% sparsity, the advantages of our method are more evident at 99% sparsity. FReP has improved robustness by 5.1% on average of five networks under AutoAttack compared to the best baseline. The sparsity implicitly added by our method is noticeable when pruning 99% of the parameters, significantly reducing pruning damage to robustness. Meanwhile, our approach only optimizes the robust objective, thus maintaining high robustness.

2) Black-box Robustness: We evaluate block-box robustness under the query-based attack [49], using l_∞-norm perturbation limited with radius $\epsilon = 8/255$. Fig. 6 represents robustness results, where the query access is limited to 2,000. Again, the proposed approach achieves better robustness than other baselines at 90% and 99% sparsity. Compared with the results at 90% sparsity, our method achieves a larger gap with baselines at 99% sparsity. This result also demonstrates the superiority of our method under high sparsity.

TABLE III
ROBUSTNESS OF STRUCTURED PRUNING AT A 70% PRUNING RATIO

Method	CIFAR-10		CIFAR-100	
	VGG-16	ResNet-18	VGG-16	ResNet-18
LWM	34.5	36.8	15.2	18.5
ADMM	35.6	38.0	16.1	18.9
CSTAR	44.4	45.3	20.4	21.6
FReP-CH	**45.1**	**45.9**	**21.2**	**22.9**

3) Structured Pruning with FReP-CH: Structured pruning methods align well with hardware architectures, thereby enabling wider practical application. We aim to investigate the ability of our approach to generalize to structured pruning. Tab. III displays the results of filter pruning on CIFAR-10 and CIFAR-100. As HYDRA and MAD do not release code for structured pruning versions, the baseline method does not include them in this experiment. Our approach achieves superior results for structured pruning, consistently outperforming the baselines with different networks and datasets. When pruning 70% of filters with our weight reparameterization technique, the robust accuracy remains competitive. Our method shows significant performance improvement over LWM and ADMM,

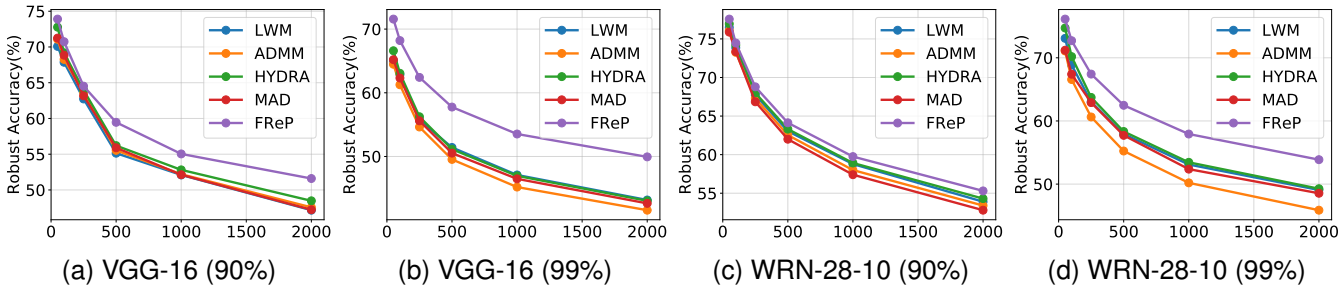


Fig. 6. Black-box Square Attack evaluations with different queries. We display robust accuracy with VGG-16 models and WideResNet-28-10 models on CIFAR-10. Sub-labels indicate the architecture (sparsity) for each evaluation.

TABLE IV
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. ORI IS PRUNING ON ORIGINAL WEIGHTS.

Dataset	Settings	AT	unstructured pruning			structured pruning		
		0%	70%	90%	99%	50%	60%	70%
CIFAR-10	Ori	81.5/51.1	80.2/49.1	79.2/48.6	71.9/41.0	78.0/47.4	74.6/43.7	69.7/37.1
	FReP-CH		**82.0**/49.4	**81.4**/49.0	75.6/43.7	**80.6/49.8**	**79.9/49.1**	**79.2/48.6**
	FReP		81.1/**50.2**	80.4/**49.7**	**78.9/47.1**	79.2/47.8	78.6/47.4	75.8/45.9
CIFAR-100	Ori	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
	FReP-CH		51.6/26.2	50.7/25.0	44.1/21.9	**52.4/25.1**	**51.5/23.6**	**48.4/22.9**
	FReP		**52.5/27.1**	**51.8/26.3**	**49.8/24.0**	51.6/24.7	48.2/23.0	41.3/19.8
Tiny-ImageNet	Ori	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
	FReP-CH		45.8/20.2	44.4/19.0	37.6/13.1	**46.2/20.1**	**44.8/18.9**	**38.0/14.8**
	FReP		**46.4/20.8**	**45.8/19.7**	**39.7/15.6**	44.8/19.3	43.4/17.3	27.9/12.2

demonstrating the effectiveness of weight reparameterization. Even compared with the state-of-the-art pruning strategies CSTAR, our method achieves 0.8% and 1.3% improvement for two models on CIFAR-100. Note that pruning the filter also needs to remove the corresponding input channels of the next layer, so the actual compression ratio is much larger than the settings.

4) Comparison of Different Weight Structures: We further investigate the impact of different reparameterization methods on pruning performance. Specifically, we evaluate the performance of FReP and its variant FReP-CH on three datasets. We also perform the same pruning algorithm on the original weight structure as a baseline approach. As shown in Tab. IV, we observe that FReP achieves higher robustness for unstructured pruning while FReP-CH performs better for structured pruning. Models trained with FReP are extremely sparse and suffer less damage from irregular pruning, thus performing much better. FReP-CH mixes information from all output channels, thus the remaining filters can extract abundant information for robustness.

5) Computational Overhead and Storage Consumption: FReP 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 $\mathbf{W} \in \mathbb{R}^{D\times C\times K^2}$, and suppose the output feature map is $\mathbf{A} \in \mathbb{R}^{D\times H\times W}$. The computation of a single convolutional layer is $H \times W \times D \times C \times K^2$. FReP only adds $D \times C \times K^2$ MACs, which are much smaller than the original computational overhead. Considering the batch processing, 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.

TABLE V
PRUNING COST COMPARISON WITH THE PRIOR PRUNING METHODS. WE PRUNING RESNET-18 ON CIFAR-10 WITH A BATCH SIZE OF 128.

	#Epochs	Time (hr)	Memory (GB)
Natural Training	100	0.6	3.60
LWM	100	5.8	3.63
ADMM	300	16.0	3.65
HYDRA	220	11.7	3.83
MAD	240	8.5	3.83
FReP-CH	100	1.5	3.71
FReP	100	1.2	3.83

To illustrate the time cost of our algorithm, we measure the time to obtain a pruned ResNet-18 model on CIFAR-10. As shown in Tab. V, Our method takes less time to obtain a robust sparse model. Specifically, our method takes about 1.21 hours on an NVIDIA Tesla V100, significantly faster than the baselines. The improvement is two-fold. Firstly, most robust pruning methods necessitate three stages: robust pretraining, mask training, and model fine-tuning. However, our method implements pruning during training, compressing the three stages together and only need 100 epochs. Secondly, we replace the commonly used 10-step PGD adversarial training

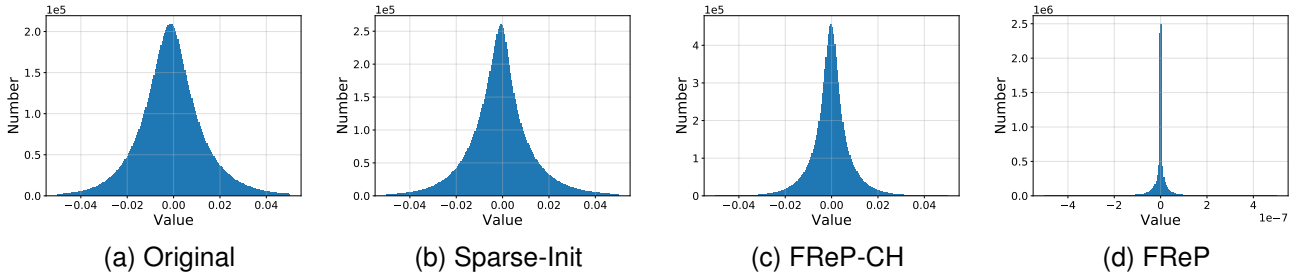


Fig. 7. Weight distribution of the ResNet-18 model trained with (a) Original weights, (b) Original weights with sparse initialization (c) FReP reparameterized weights, (d) FReP-CH reparameterized weights.

with an improved single-step method, which significantly reduced the training time. The above improvements make our robust pruning method have an acceptable time cost.

C. The Effect of Weight Reparameterization

This section investigates the impact of weight reparameterization on model sparsity. We conduct extensive experiments comparing the original weights and our reparameterized weights.

1) Performance of Robust Pruning: We first compare the performance of robust pruning in two settings, where we train networks using standard adversarial training:

- **Pruning after pre-training (PT)**: We pre-train networks from scratch for 100 epochs and prune them without any fine-tuning process. This setting can reflect the damage of pruning to the model robustness.
- **Pruning with training**: We perform pruning at the 30th epoch in the total 100 epochs training process.

Fig. 8 shows that our weight-reparameterized networks achieve stable robustness at high pruning ratios, while the performance drops significantly with original weights. Notably, even without fine-tuning, our method maintains robustness when pruning 99% of connections of a pre-trained model, indicating that the network is sparse enough to suffer little pruning damage. This result 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.

2) Fusing Weights at Initialization: The weight distribution at initialization will be sparser than regular initialization if we fuse W_1 and W_2 before training, which may reduce pruning damage. To exclude the effect of sparse initialization, we compare our method with the strategy of fusing weights at initialization (Sparse-Init). As shown in Tab. VI, Sparse-Init achieves robustness similar to Regular-Init at different pruning ratios. The results highlight the importance of the reparameterized structure for maintaining sparsity. FReP adds sparsity to network weights and maintains it during training, resulting in less robustness degradation at high pruning ratios.

3) Visualization of Weight Distribution: To intuitively analyze the sparseness effect of weight reparameterization, we plot the weight distribution of models with different weight structures in Fig. 7. FReP leads to a significantly sparser model, while fusing weights before training (Sparse-Init) fails to introduce sparsity. Although sparsely initialized, the weights grow larger during training, eventually resulting in a weight distribution similar to regular initialization. In contrast, FReP produces an extremely sparse weight distribution and thus suffers little from pruning damage. The weight distribution of models also demonstrates the importance of FReP for maintaining sparse weights during training.

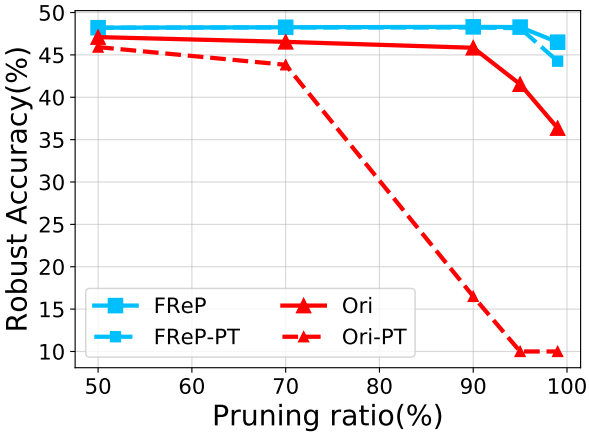


Fig. 8. Comparison of robust accuracy with pruning on original weights and reparameterized weights. We display results with VGG-16 on CIFAR-10 under different pruning ratios.

TABLE VI
ROBUST ACCURACY(%) WITH RESNET-18 ON CIFAR-10.

	50%	70%	90%	95%	99%
Regular-Init	49.5	49.1	48.6	45.8	41.0
Sparse-Init	49.6	49.2	48.6	45.9	41.1
FReP-CH	50.1	49.4	49.0	47.6	43.7
FReP	50.8	**50.2**	**49.7**	**49.2**	**47.1**

D. The Effect of Temporal Ensemble and SRE loss

This section investigates the effectiveness of temporal ensemble and self-consistent robust error for improved single-step adversarial training.

TABLE VII
COMPARISON OF OUR ALGORITHM WITH DIFFERENT MULTI-STEP ADVERSARIAL TRAINING METHODS. WE PRUNE 90% OF THE PARAMETERS OF RESNET-18 MODELS AND ALL MODELS USE OUR REPARAMETERIZED STRUCTURE.

Method	CIFAR10				CIFAR100				Tiny-ImageNet			
	Clean	FGSM	PGD	AA	Clean	FGSM	PGD	AA	Clean	FGSM	PGD	AA
FReP+FGSM	82.7	56.0	40.0	34.9	54.3	27.4	18.5	14.1	49.7	21.4	14.8	9.6
FReP+PGD	79.8	56.8	48.5	44.1	51.8	31.8	25.8	21.1	46.1	23.2	19.2	13.9
FReP+TRADES	78.8	56.5	49.3	44.6	51.3	31.4	26.7	21.9	44.5	23.5	20.6	14.5
FReP+MART	79.2	57.6	49.4	45.1	51.0	32.0	26.7	21.8	45.2	23.8	20.4	14.6
FReP	80.4	57.3	49.7	44.7	51.8	**32.2**	26.3	21.5	45.8	24.3	19.7	14.3

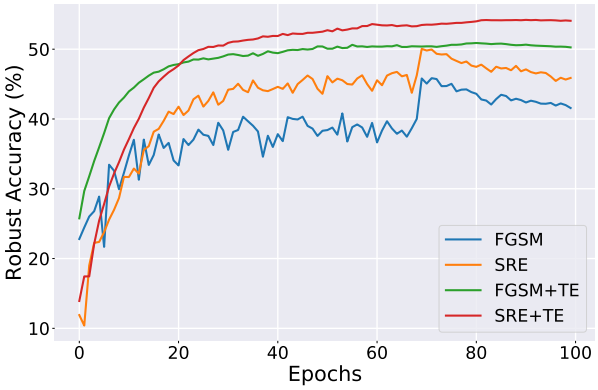


Fig. 9. Robustness of models trained with different combinations.

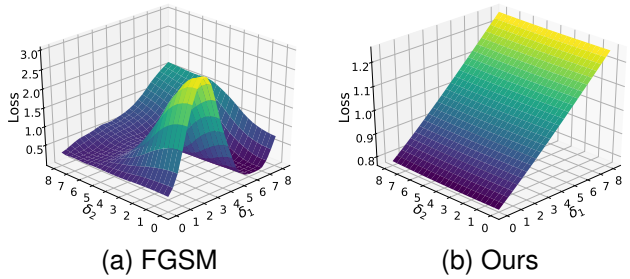


Fig. 10. Loss landscapes of models trained by FGSM and our method. z axis denotes the loss value $z = \mathcal{L}(x + \delta_1 r_1 + \delta_2 r_2)$, where $r_1 = \text{sign}(\nabla_x \mathcal{L}(x))$ is the sign of the gradient direction and r_2 is an orthogonal direction.

1) Comparison with Multi-step Adversarial Training: Tab. VII shows the pruning results when replacing our temporal ensemble with other adversarial training methods. The proposed approach significantly outperforms the performance of the basic single-step adversarial training FGSM on all three datasets. Since our method simultaneously smooths the weights and logits in adversarial training, the inner maximization problem can be better approximated. Moreover, our method achieves comparable robust accuracy to multi-step defenses such as TRADES and MART, demonstrating that our method accelerates training without excessively sacrificing robustness.

2) Effectiveness of Each Component: To study the effectiveness of each component, we trained models with different combinations on the CIFAR-10 dataset. As shown in Fig. 9, both temporal ensemble and SRE substantially contribute to enhancing the robustness of single-step adversarial training. In particular, we found that SRE has a constant robustness improvement throughout the training process but still suffers from robust overfitting. Temporal ensemble improves robustness more obviously, and can help mitigate the overfitting with appropriate update frequency. When these two components are combined, robustness is further improved.

3) Visualizing Loss Surface: We further visualize the loss surface of models trained using our method and FGSM. As shown in Fig. 10, FGSM produces gradient masking with a convoluted loss landscape. Taking a step of size $\epsilon = 8/255$ in the gradient direction does not reach the maximum, preventing the generation of strong adversaries. On the contrary, our method leads to a smooth loss landscape in the input space, and attack samples that approximately maximize the loss can be found in a single step, thus achieving much higher robustness.

E. Ablation Studies

In this section, we discuss the optimal time for pruning during training. Then, we study the effect of update frequency on temporal ensemble. Finally, we investigate the impact of initialization.

1) Number of Training Epochs Before Pruning: Previous studies suggested that important connections have been highlighted at the early stages of training [50], [51]. Does this conclusion also hold in adversarial training, even though we have reparameterized the weights? To answer this question, we prune networks at different training epochs ranging from 10 to 60. The results are presented in Fig. 11a, where the pruning ratios are 90% and 99% for ResNet50 on the CIFAR-10 dataset. For both 90% and 99% pruning ratios, pruning at the 10th epoch hurts performance, indicating important connections are not all highlighted at this point. However, we have identified most of the important connections since we only observe slight robust accuracy drops when pruning 90% of connections. In addition, although the robust accuracy improves as the pruning timing is delayed, we see gains in performance saturate after the 30th epoch. Since the total training time is fixed, pruning too late may even lead to insufficient time for subsequent training. Overall, our results show that important connections are also identified early in adversarial training.

2) Update Frequency of the Temporal Ensemble Model: In Sec. III-C, we proposed an adjustment strategy that updates

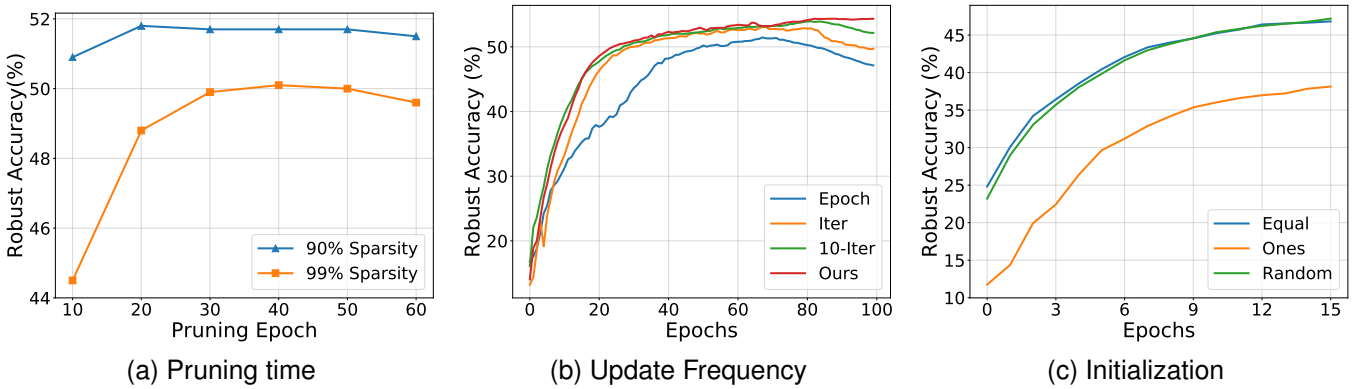


Fig. 11. The comprehensive ablation experiments on CIFAR-10. (a) Robust accuracy of compressed networks with varying number of training epochs before the pruning operation. (b)Robust accuracy of ensemble models at different update frequencies during training. (c) Robust accuracy of first 15 training epochs with three initialization methods.

TABLE VIII
CLEAN/ROBUST ACCURACY (%) OF MODELS TRAINED WITH DIFFERENT INITIALIZATION TECHNIQUES. WE PRUNE RESNET-18 MODELS ON THE CIFAR-10 DATASET.

Initialization	0%	90%	99%
Equal	81.9/51.4	80.5/49.5	78.9/47.2
Ones	77.6/42.3	76.4/41.0	74.1/38.4
Random	82.0/51.4	80.4/49.7	78.9/47.1

the ensemble model every 100 iterations at the beginning, and then divide the update frequency by 10 as the learning rate decreases. We select three schemes to compare with: updates the temporal ensemble model each iteration, every ten iterations, and each epoch. As shown in Fig. 11b, Frequently updating the ensemble model suffers from degradation in the late training stages, while infrequent updates do not fully enjoy the benefits of the temporal ensemble. The 10-iteration-based scheme and our adjustment strategy are nearly the same in their prime, except that our strategy avoids degradation. Our strategy fully exploits the advantages of model temporal integration without late deterioration.

3) Impact of Initialization Techniques: The explanation of the implicit sparsity effect in Sec. III-B assumes that the absolute values of the reparameterized weights are initialized to be equal. To study the impact of initialization, we compared three different initialization methods, including:

- Equal: $\mathbf{W}_1$ is randomly initialized and $\mathbf{W}_2 = |\mathbf{W}_1|$
- Ones: $\mathbf{W}_1$ is randomly initialized and $\mathbf{W}_2$ is an all-ones matrix.
- Random: Both $\mathbf{W}_1$ and $\mathbf{W}_2$ are randomly initialized.

Fig. 11c shows the robust accuracy in the early stages of training. The training dynamics of the two initialization methods, Equal and Random, are similar, which indicates that we do not need to place special restrictions on the initialization of the reparameterized weights. However, initializing one of the matrices to all ones provides a poor starting point for optimization. We also report the pruning results for different initializations in Tab. VIII. Unsurprisingly, methods Equal and Random achieve similarly good performance while method Ones performs poorly.

V. CONCLUSION

This paper proposes FReP, an Efficient and effective approach that can simultaneously impose high sparsity and high adversarial robustness on the DNN models. It consists of two essential components, *i.e.*, the weight reparameterization strategy and the temporal ensemble method. Specifically, weight reparameterization decomposes the original weights into two tensors during training to implicitly add the sparsity constraint. Thus, the compressed DNNs achieves high compression performance and strong adversarial robustness at the same time. Temporal ensemble taps the potentials of single-step adversarial training by integrating the weights of historical models, thereby significantly reducing the time to complete robust pruning without sacrificing robustness. Extensive experimental results demonstrate the effectiveness of our approach. Our method outperforms the state-of-the-art baselines across various datasets and network architectures, while demonstrating adaptability to both unstructured and structured pruning. Notably, FReP accelerates the optimization process by $7\times$ and $10\times$ compared to state-of-the-art MAD and HYDRA, respectively, making it an advantageous choice for robust pruning.

REFERENCES

[1] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing ai to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, 2022.
[2] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–12.
[3] N. Lee, T. Ajanthan, and P. Torr, "Snip: single-shot network pruning based on connection sensitivity," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–12.
[4] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4857–4867.
[5] C. Chen, X. Zhao, and M. C. Stamm, "Generative adversarial attacks against deep-learning-based camera model identification," *IEEE Trans. Inf. Forensics Security*, 2019.
[6] H. Ma, K. Xu, X. Jiang, Z. Zhao, and T. Sun, "Transferable black-box attack against face recognition with spatial mutable adversarial patch," *IEEE Trans. Inf. Forensics Security*, 2023.

[7] J. Dong, Y. Wang, J. Lai, and X. Xie, "Restricted black-box adversarial attack against deepfake face swapping," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2596–2608, 2023.

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

[9] S. Ye, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, and X. Lin, "Adversarial robustness vs. model compression, or both?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 111–120.

[10] S. Gui, H. N. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1285–1296.

[11] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Towards compact and robust deep neural networks," *arXiv preprint arXiv:1906.06110*, 2019.

[12] C. Li, Q. Qiu, Z. Zhang, J. Guo, and X. Cheng, "Learning adversarially robust sparse networks via weight reparameterization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 8527–8535.

[13] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7472–7482.

[14] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–13.

[15] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 2958–2969.

[16] J. Uesato, B. O'donoghue, P. Kohli, and A. Oord, "Adversarial risk and the dangers of evaluating against weak attacks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5025–5034.

[17] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.

[18] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.

[19] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4970–4979.

[20] J. Liu, C. P. Lau, H. Souri, S. Feizi, and R. Chellappa, "Mutual adversarial training: Learning together is better than going alone," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2364–2377, 2022.

[21] S. Han, C. Lin, C. Shen, Q. Wang, and X. Guan, "Interpreting adversarial examples in deep learning: A review," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–38, 2023.

[22] H. Liang, E. He, Y. Zhao, Z. Jia, and H. Li, "Adversarial attack and defense: A survey," *Electronics*, vol. 11, no. 8, p. 1283, 2022.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–10.

[24] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021.

[25] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Trans. Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021.

[26] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. Liang, "Unlabeled data improves adversarial robustness," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 11 190–11 201.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[28] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[29] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse dnns with improved adversarial robustness," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 240–249.

[30] D. Madaan, J. Shin, and S. J. Hwang, "Adversarial neural pruning with latent vulnerability suppression," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6575–6585.

[31] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "HYDRA: pruning adversarially robust neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2020, pp. 19 655–19 666.

[32] O. Özdenizci and R. Legenstein, "Training adversarially robust sparse networks via bayesian connectivity sampling," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8314–8324.

[33] B.-K. Lee, J. Kim, and Y. M. Ro, "Masking adversarial damage: Finding adversarial saliency for robust and sparse network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 15 126–15 136.

[34] H. Phan, M. Yin, Y. Sui, B. Yuan, and S. Zonouz, "Cstar: Towards compact and structured deep neural networks with adversarial robustness," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 2, 2023, pp. 2065–2073.

[35] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 1–13, 2019.

[36] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 1–12, 2019.

[37] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–11.

[38] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM CCS*, 2017, pp. 506–519.

[39] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[40] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, "Robustness via curvature regularization, and vice versa," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 9078–9086.

[41] M. Andriushchenko and N. Flammarion, "Understanding and improving fast adversarial training," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 16 048–16 059, 2020.

[42] G. Sriramanan, S. Addepalli, A. Baburaj, and V. B. R., "Guided adversarial attack for evaluating and enhancing adversarial defenses," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20 297–20 308.

[43] ——, "Towards efficient and effective adversarial training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 11 821–11 833.

[44] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 369–385.

[45] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8093–8104.

[46] T. Pang, M. Lin, X. Yang, J. Zhu, and S. Yan, "Robustness and accuracy could be reconcilable by (proper) definition," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 17 258–17 277.

[47] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 9185–9193.

[48] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.

[49] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 484–501.

[50] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks," in *International Conference on Learning Representations*, 2018, pp. 1–12.

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