Toward Adversary-aware Non-iterative Model Pruning through Dynamic Network Rewiring of DNNs

SOUVIK KUNDU, YAO FU, BILL YE, PETER A. BEEREL, and MASSOUD PEDRAM,
University of Southern California, USA

We present a dynamic network rewiring (DNR) method to generate pruned deep neural network (DNN) models that both are robust against adversarially generated images and maintain high accuracy on clean images. In particular, the disclosed DNR training method is based on a unified constrained optimization formulation using a novel hybrid loss function that merges sparse learning with robust adversarial training. This training strategy dynamically adjusts inter-layer connectivity based on per-layer normalized momentum computed from the hybrid loss function. To further improve the robustness of the pruned models, we propose DNR++, an extension of the DNR method where we introduce the idea of sparse parametric Gaussian noise tensor that is added to the weight tensors to yield robust regularization. In contrast to existing robust pruning frameworks that require multiple training iterations, the proposed DNR and DNR++ achieve an overall target pruning ratio with only a single training iteration and can be tuned to support both irregular and structured channel pruning. To demonstrate the efficacy of the proposed method under the no-increased-training-time “free” adversarial training scenario, we finally present FDNR++, a simple yet effective training modification that can yield robust yet compressed models requiring training time comparable to that of an unpruned non-adversarial training. To evaluate the merits of our disclosed training methods, experiments were performed with two widely accepted models, namely VGG16 and ResNet18, on CIFAR-10 and CIFAR-100 as well as with VGG16 on Tiny-ImageNet. Compared to the baseline uncompressed models, our methods provide over 20× compression on all the datasets without any significant drop of either clean or adversarial classification performance. Moreover, extensive experiments show that our methods consistently find compressed models with better clean and adversarial image classification performance than what is achievable through state-of-the-art alternatives. We provide insightful observations to help make various model, parameter density, and prune-type selection choices and have open-sourced our saved models and test codes to ensure reproducibility of our results.

CCS Concepts: • **Hardware** → **Robustness**; **Communication hardware, interfaces and storage**;

Additional Key Words and Phrases: Pruning, adversarial robustness, model pruning

ACM Reference format:

Souvik Kundu, Yao Fu, Bill Ye, Peter A. Beerel, and Massoud Pedram. 2022. Toward Adversary-aware Non-iterative Model Pruning through Dynamic Network Rewiring of DNNs. *ACM Trans. Embedd. Comput. Syst.* 21, 5, Article 52 (December 2022), 24 pages.

<https://doi.org/10.1145/3510833>

This work was supported in part by NSF grant number 1763747 and DARPA grant number HR00112190120.

Authors’ address: S. Kundu, Y. Fu, B. Ye, P. A. Beerel, and M. Pedram, University of Southern California, Los Angeles, California, 90089; emails: {souvikku, yaof, biao, pabeerel, pedram}@usc.edu.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2022 Copyright held by the owner/author(s).

1539-9087/2022/12-ART52 \$15.00

<https://doi.org/10.1145/3510833>

1 INTRODUCTION

Despite being the critical component of various computer vision tasks [5, 10, 21, 24, 36, 45, 50], **deep neural networks (DNNs)** suffer from significant safety concerns due to their vulnerability to *adversarial images* [4, 11, 20, 34, 47]. These maliciously generated images are perceptually similar to the clean ones yet can fool the models to make wrong classifications. Given the adverse consequences of misclassification of such images, efficient training of a robust classifier that can correctly label the majority of adversarial images has been an active area of research [11, 19, 41, 51].

Meanwhile, to reduce the large computation and storage costs of DNN models, particularly for IoT devices, reduced-size models [9, 28, 29] and model *pruning* techniques [6, 7, 17, 30] have gained significant attention. In particular, various heuristic pruning techniques have proven to be extremely effective in reducing the parameter count of apparently over-parameterized models by up to ~90% [6, 7, 30]. Moreover, ensuring the pruned models have structure can yield significant performance improvements on a broad range of compute platforms [18, 38]. However, as Figure 1(a) illustrates, adversarial training [40] generally demands more non-zero parameters than needed for only clean images [40]. Thus, a naively pruned model performing well on clean images becomes vulnerable to adversarial images. Moreover, compared to normally trained models, it is more difficult to aggressively prune adversarially trained models [54] without sacrificing classification performance. However, with the growing deployment of DNNs in safety-critical edge applications on resource-constrained devices such as drones, compressed yet robust models are often necessary. Unfortunately, despite a plethora of work on model pruning performed on clean data, there have been only a few studies that analyze pruned models under adversarial attacks. In particular, prior work [14, 54] proposed to design a pruned yet robust model through a unified constrained optimization formulation based on the **alternating direction method of multipliers (ADMM)** in which dynamic L_2 regularization is the key to outperforming other pruning techniques [46]. In addition, [48] proposed to provide provable robustness of pruned models. However, all these techniques require the added overhead of pre-training a model. Moreover, ADMM requires the model pruner to specify layer-wise pruning ratios. This either demands human expertise to provide suitable pruning ratios or requires the use of cumbersome evolutionary search methods to evaluate the model sensitivity for a specific global pruning ratio.¹ Other schemes like Lasso [43] cannot guarantee a specified pruning ratio and requires a subsequent re-training phase to increase the accuracy after the assignment of non-significant weights to zero, resulting in increased training time.

Our contributions: To address the above-mentioned concerns, this article presents **dynamic network rewiring (DNR)**. In particular, we propose a unified training framework to find a pruned model with increased robustness that does not require the knowledge of individual per-layer pruning ratios and present our initial results in [25]. DNR uses a unique hybrid loss function for robust pruning having three major components: a clean image classification loss, a dynamic L_2 -regularizer term inspired by a relaxed version of ADMM [8], and an adversarial image classification loss. Using this hybrid loss, we then propose a non-iterative robust *sparse-learning* scheme. We take inspiration from [6] and use normalized momentum as a metric to dynamically arrange per-layer pruning ratios and help maintain the target pruning ratio every epoch, eliminating the need for subsequent fine-tuning. To further improve the robustness of models via robust regularization [19, 32], this article also proposes DNR++, a sparse-learning strategy where the weights are purposefully regularized via a form of trainable sparse noise tensors. In particular, models generated through DNR++ can yield improved performance of up to ~9% on adversarial images with similar or improved clean image performance as that generated through DNR.

¹For a model with target parameter density d , its global pruning ratio is defined as $(1-d) \times 100\%$. For example, in a model with $d = 0.05$, the pruning ratio is 95%.

Fig. 1. (a) Weight distribution of the 14th convolution layer of a ResNet18 model for different training schemes: normal, adversarial [40], and noisy adversarial [19]. (b) An adversarially generated image ($\hat{x}$) obtained through a FGSM attack, which is predicted to be the number 5 instead of 8 (x).

Table 1. Comparison between Various Robust Model Pruning Schemes

Method	Baseline Robustness	Ultra-high Compression	Non-iterative Training	Improved Robustness via Noisy Weights	Single-shot Training
ADMM [14, 46]	✓	✓	×	×	×
RSR [43]	✓	×	×	✓	×
DNR (this article)	✓	✓	✓	×	×
DNR++ (this article)	✓	✓	✓	✓	×
FDNR++ (this article)	✓	✓	✓	✓	✓

Training a model with strong gradient-based attack-generated images (e.g., a PGD attack) generally increases the training time and cost by nearly $k\times$, where k is the PGD attack iteration constant. This, in particular, is challenging for resource-constrained devices even for incremental training. To reduce this large training cost, we leverage the idea of “free” adversarial training [27, 49] and propose a fast one-shot training variant of DNR++, namely FDNR++, which allows sparse weight updates during each iteration of perturbed image generation. In particular, this technique helps FDNR++ produce pruned yet robust models with training times comparable to that of unpruned models training with no robustness. To provide inference speedup with no custom compiler or hardware design, we extend the proposed approaches to support structured pruning, namely *channel pruning*. To the best of our knowledge, we are the first to propose a non-iterative robust training framework that supports both irregular and channel pruning.

Table 1 summarizes the general benefits of our proposed methods compared to the SOTA. We quantify these benefits analyzing the performance of both irregular and channel-pruned robust models on an extensive set of experiments on CIFAR-10 [23] and CIFAR-100 [23] using both ResNet18 [16] and VGG16 [50], and on Tiny-ImageNet [15] using VGG16. Models yielded through our method consistently outperform the **state of the art (SOTA)** [43, 54] with no significant drop in clean image classification performance compared to the respective unpruned baselines. We further empirically demonstrate the superiority of this scheme when used to target model pruning on clean-only image classification tasks compared to SOTA non-iterative pruning mechanisms [6, 7, 17, 33].

Lastly, based on our comprehensive investigations and observations on adversarial robustness, we demonstrate a simple yet extremely effective method to push the boundary of model robustness using **random pruned models for inference (RPMI)**. Specifically, RPMI randomly selects one of multiple stored pruned models for inference.²

²Here, our assumption is the model selector mechanism to be largely hidden from the attacker despite the attacker’s knowledge of the various models’ weights.

The remainder of this article is structured as follows. In Section 2 we present necessary background. Section 3 describes our proposed training methods and enhancements, and Section 4 presents our detailed experimental evaluations and analysis. We conclude in Section 5 and detail the potential broader impact of this work in Section 6.

2 BACKGROUND WORK

2.1 Preliminaries on Adversarial Attacks

Recently, various gradient-based adversarial attacks have been proposed to generate perturbed or adversarial images. These images, despite having barely visible perturbations from real images, manage to fool a trained DNN. One of the most common gradient-based attacks is the **fast gradient sign method (FGSM)** [11]. Given a vectorized input $\mathbf{x}$ of the real image and corresponding output label $\mathbf{t}$, FGSM perturbs each element x in $\mathbf{x}$ along the sign of the associated element of the gradient of the inference loss w.r.t. $\mathbf{x}$ as shown in Equation (1) and illustrated in Figure 1(b). **Projected gradient descent (PGD)** [40], another well-known attack, is a multi-step variant of FGSM, where $\hat{\mathbf{x}}^{k=1} = \mathbf{x}$ and the iterative update of the perturbed image $\hat{\mathbf{x}}$ in the k^{th} step is given in Equation (2).

$$\hat{\mathbf{x}} = \mathbf{x} + \epsilon \times \text{sgn}(\nabla_{\mathbf{x}} J(g(\mathbf{x}; \boldsymbol{\theta}), \mathbf{t})) \quad (1)$$

$$\hat{\mathbf{x}}^k = \text{Proj}_{P_\epsilon(\mathbf{x})}(\hat{\mathbf{x}}^{k-1} + \alpha \times \text{sgn}(\nabla_{\mathbf{x}} J(g(\hat{\mathbf{x}}^{k-1}; \boldsymbol{\theta}), \mathbf{t}))) \quad (2)$$

$g(\mathbf{x}, \boldsymbol{\theta})$ generates the output of the DNN, parameterized by $\boldsymbol{\theta}$. Here, Proj projects the updated adversarial sample onto the projection space $P_\epsilon(\mathbf{x})$ defined as the ϵ - L_∞ neighbourhood of the benign sample $\mathbf{x}$.³ ϵ is a scalar that sets the limit to the perturbation and α is the attack step size. We note that PGD is one of the strongest L_∞ adversarial example generation algorithms [40] and use it as the attack for our adversarial image generation during training of the proposed framework. We note that there have been efforts in an orthogonal research direction to look at various modifications [12, 13, 22, 56] of the original adversarial training strategy [40] including mixing of augmentation techniques [44] in further improving model robustness. Our proposed training strategy can be easily adopted to these techniques to yield even further improved performances.

Note that for the above attacks we consider two different scenarios: (1) a **white-box (WB)** attack, in which the attacker has complete access to the model parameters, and (2) a **black-box (BB)** attack, in which the attacker has no knowledge of the model parameters and thus involves weaker perturbations.

2.2 Model Pruning

ADMM is a powerful optimization method used to solve problems with non-convex, combinatorial constraints [3]. It decomposes the original optimization problem into two sub-problems and solves the sub-problems iteratively until convergence. Pruning **convolutional neural networks (CNNs)** can be modeled as an optimization problem where the cardinality of each layer's weight tensor is bounded by its pre-specified pruning ratio. In the ADMM framework, such constraints are transformed to ones represented with indicator functions, such as $\mathbb{I}_\theta(\mathbf{z}) = 0$ for $|\mathbf{z}| \leq n$ and $+\infty$ otherwise. Here, $\mathbf{z}$ denotes the duplicate variable [3] and n represents the target number of non-zero weights determined by pre-specified pruning ratios.

³It is noteworthy that the generated $\hat{\mathbf{x}}$ are clipped to a valid range, which for our experiments is $[0, 1]$.

Next, the original optimization problem is reformulated as

$$\mathcal{L}_\rho(\boldsymbol{\theta}, \mathbf{z}, \lambda) = J(g(\mathbf{x}; \boldsymbol{\theta}), \mathbf{t}) + \mathbb{I}_\theta(\mathbf{z}) + \langle \lambda, \boldsymbol{\theta} - \mathbf{z} \rangle + \frac{\rho}{2} \|\boldsymbol{\theta} - \mathbf{z}\|_2^2, \quad (3)$$

where λ is the Lagrangian multiplier and ρ is the penalization factor when parameters $\boldsymbol{\theta}$ and $\mathbf{z}$ differ. Equation (3) is broken into two sub-problems that solve $\boldsymbol{\theta}$ and $\mathbf{z}$ iteratively until convergence [46]. The first sub-problem uses **stochastic gradient descent (SGD)** to update $\boldsymbol{\theta}$, while the second sub-problem applies projection to find the assignment of $\mathbf{z}$ that is closest to $\boldsymbol{\theta}$ yet satisfies the cardinality constraint, effectively pruning weights with small magnitudes.

Not only can ADMM prune a model's weight tensors but also it has a dynamic regularizer. Such adaptive regularization is one of the main reasons behind the success of its use in pruning. However, ADMM-based pruning has several drawbacks. First, ADMM requires prior knowledge of the per-layer pruning ratios. Second, ADMM does not guarantee the pruning ratio will be met, and therefore, an additional round of hard pruning is required after ADMM completes. Third, not all problems solved with ADMM are guaranteed to converge. Fourth, to improve the convergence, ρ needs to be progressively increased across several rounds of training, which increases training time [3].

Sparse learning [6] addresses the shortcomings of ADMM by leveraging exponentially smoothed gradients (momentum) to prune weights. It redistributes pruned weights across layers according to their mean momentum contribution. The weights that will be removed and transferred to other layers are chosen according to their magnitudes, while the weights that are brought back (reactivated) are selected based on their momentum values. On the other hand, a major shortcoming of sparse learning compared to ADMM is that it does not benefit from a dynamic regularizer and thus often yields lower levels of accuracy. Furthermore, prior sparse-learning schemes only support irregular forms of pruning, limiting speedup on many compute platforms. Finally, sparse learning, to the best of our knowledge, has not previously been extended to robust model pruning.

3 DYNAMIC NETWORK REWIRING

To tackle the shortcomings of ADMM and sparse learning we introduce DNR, a network rewiring method that optimizes the model parameters based on a hybrid loss function and enables non-iterative training to provide high-accuracy compressed yet robust DNN models. We then detail our improved method DNR++, which adds regularizing sparse noise tensors to the trainable weight tensors. Then, to improve the robust training time, we present a simple yet effective extension FDNR++ that achieves robust compression with training times comparable to that of training normal unpruned models with only clean images. Finally, we describe our extension method to support structured pruning.

3.1 Dynamic L_2 Regularizer

For a DNN parameterized by $\boldsymbol{\theta}$ with L layers, we let $\boldsymbol{\theta}_l$ represent the weight tensor of layer l . In our sparse-learning approach, these weight tensors are element-wise multiplied ($\odot$) by corresponding binary mask tensors ($\mathbf{m}_l$) to retain only a fraction of non-zero weights, thereby meeting a target pruning ratio. We update each layer mask in every epoch similar to [6]. The number of non-zeros is updated based on the layer's normalized momentum, and the specific non-zero entries are set to favor large magnitude weights. We incorporate an ADMM dynamic L_2 regularizer [46] into this framework by introducing duplicate variable $\mathbf{z}$ for the non-zero weights, which is in turn updated at the start of every epoch. Unlike [46], we only penalize differences between the masked weights ($\boldsymbol{\theta}_l \odot \mathbf{m}_l$) of a layer l and their corresponding duplicate variable $\mathbf{z}_l$. Because the total cardinality constraint of the masked parameters is satisfied, i.e., $\sum_{l=1}^L \text{card}(\boldsymbol{\theta}_l \odot \mathbf{m}_l) \leq n$, the indicator penalty

Fig. 2. (a) Training loss vs. epochs and (b) pruning sensitivity per layer for VGG16 on CIFAR-10.

factor is redundant and the loss function may be simplified as

$$\mathcal{L}_\rho(\theta, \mathbf{z}, \mathbf{m}) = J(g(\mathbf{x}; \theta, \mathbf{m}), \mathbf{t}) + \frac{\rho}{2} \sum_{l=1}^L \|\theta_l \odot \mathbf{m}_l - \mathbf{z}_l\|_2^2, \quad (4)$$

where ρ is the dynamic L_2 penalizing factor. This simplification is particularly important because the indicator function used in Equation (3) is non-differentiable and its removal in Equation (4) enables the loss function to be minimized without decomposition into two sub-problems.⁴ Moreover, SGD with this loss function converges similarly to the SGD with $J(g(\mathbf{x}; \theta, \mathbf{m}), \mathbf{t})$ and more reliably than ADMM. Intuitively, the key role of the dynamic regularizer in this simplified loss function is to encourage the DNN to not change values of the weights that have large magnitude unless the corresponding loss is large, similar to what the dynamic regularizer does in ADMM-based pruning.

3.2 Proposed Hybrid Loss Function for DNR

For a given input image $\mathbf{x}$, adversarial training can be viewed as a min-max optimization problem that finds the model parameters θ that minimize the loss associated with the corresponding adversarial sample $\hat{\mathbf{x}}$, as shown below:

$$\arg \min_{\theta} \left\{ \arg \max_{\hat{\mathbf{x}} \in \mathcal{P}_\epsilon(\mathbf{x})} J(g(\hat{\mathbf{x}}; \theta), \mathbf{t}) \right\}. \quad (5)$$

In our framework, we use SGD for loss minimization and PGD to generate adversarial images. More specifically, to boost classification robustness on a perturbed image we propose using a hybrid loss function that combines the proposed simplified loss function in Equation (4) with adversarial image loss, i.e.,

$$J_{tot} = \beta \mathcal{L}_\rho(\theta, \mathbf{z}, \mathbf{m}) + (1 - \beta) J(g(\hat{\mathbf{x}}; \theta, \mathbf{m}), \mathbf{t}), \quad (6)$$

where β provides a tunable trade-off between the two loss components.

OBSERVATION 1. *A DNN only having a fraction of weights active throughout the training can be trained with the proposed hybrid loss to finally converge similar to that of the unpruned model (mask $\mathbf{m} = 1$) to provide a robust yet compressed model. This is exemplified in Figure 2(a), which shows similar convergence trends for both pruned and unpruned models, simultaneously achieving both the target pruning ratio and robustness while avoiding the need for multiple training iterations.*

⁴Note this simplified loss function also drops the term $\langle \lambda, \theta - \mathbf{z} \rangle$ because $\mathbf{z}$ is updated with θ at the beginning of each epoch, forcing the Lagrangian multiplier λ and its contribution to the loss function to be always 0.

Fig. 3. Simplified diagram of pruned weight generation policy in (a) DNR and (b) DNR++. In particular, for DNR++, we evaluate a *pruned noise* tensor for a corresponding *pruned weight* tensor and add them to finally generate the *pruned noisy weight* tensor.

3.3 DNR++: Dynamic Network Rewiring with Noisy Weights

The addition of noise to the weight tensors to provide robust regularization has been proven effective in improving model performance against various gradient-based adversarial attacks [32, 37]. In this work, we adopt a recently proposed trainable **parametric noise injection (PNI)** scheme [19]. However, to ensure target model pruning with the addition of noise, a naive adaptation of the scheme is not possible. We now detail our proposed additive **sparse PNI (SPNI)**-based parameter update strategy in DNR++, exemplified in Figure 3. Let the weight tensor of a convolutional layer l be denoted as $\theta_l \in \mathbb{R}^{M \times N \times h \times w}$, where h and w are the height and width of the convolutional kernel, and M and N represent the number of filters and channels per filter, respectively. Let us now assume a noise tensor v_l having the same dimension as that of the weights. The SPNI weights are generated via following equation:

$$\tilde{\theta}_l^i = f_{SPNI}(\theta_l^i) = \theta_l^i + m_l^i \cdot \eta_l^i \cdot v_l^i; v_l^i \sim N(0, (\sigma_l^i)^2), \quad (7)$$

where θ_l^i and v_l^i are elements of the noise-free weight and noise tensors, respectively, with m_l^i as the binary mask value for the scalar weight of layer l at the i^{th} epoch. Note that in our network rewiring-based sparse-learning scheme, the m_l^i value may change from epoch to epoch based on the significance of a layer weight. η_l^i is a trainable noise scaling factor that controls the strength of the additive noise and its simplistic gradient-based update is given by

$$\eta_l^{i+1} = \eta_l^i - \varrho \cdot \frac{\partial \mathcal{L}}{\partial \eta_l^i}, \quad (8)$$

where

$$\frac{\partial \mathcal{L}}{\partial \eta_l^i} = \sum \frac{\partial \mathcal{L}}{\partial f_{SPNI}(\theta_l^i)} \frac{\partial f_{SPNI}(\theta_l^i)}{\partial \eta_l^i} \quad (9)$$

and ϱ is the learning rate. We initialize the $\eta_l = 0.25$ and use no weight decay for this learnable parameter, following the recommendation of [19].

3.4 FDNR++: A Single-shot Training Extension of DNR++

Excessive training time and compute costs are well-known issues of adversarial training. This is primarily due to the fact that internally the training performs two tasks in different back-propagation instances: (1) trainable weights update via gradient-based back-propagation and (2) iterative perturbation update via back-propagating gradient of loss with respect to the pixel values. Thus, despite our proposed DNR and DNR++ being non-iterative, we cannot call these methods “single shot” as multiple iterative back-propagation is necessary to create strong perturbed images via

the PGD attack. Thus, to achieve the goal of a fast single-shot robust model pruning method we incorporate the idea of merging these two tasks [49]. In particular, during the perturbation generation of an image batch, we allow the weight gradients to update the weights with an associated mask value of 1. The details of our proposed fast DNR++ (FDNR++) are presented in Algorithm 2. Interestingly, in FDNR++ we do not have a separate adversarial loss component because we consider the perturbed variants of an image batch as a separate sub-batch and update the weights based on loss computed from each sub-batch (Algorithm 2). The number of sub-batches for a batch is a hyperparameter and is determined by the value *numSteps* (Algorithm 2). The sub-batch perturbation grows stronger with increasing *numSteps*. Note, the primary goal of FDNR++ is to show the feasibility of a “truly” single-shot robust compression framework, which has been largely missing in the literature. DNR++ can be easily adapted to other recently proposed techniques [52] to reduce adversarial training complexity as well.

3.5 Support for Channel Pruning

We first convert the 4D weight tensor θ_l to a 2D weight matrix, with M and $N \times h \times w$ being the number of rows and columns, respectively. We then partition this matrix into N sub-matrices of M rows and $h \times w$ columns, one for each channel. To compute the importance of a channel c , we find the **Frobenius norm (F-norm)** of a corresponding sub-matrix, thus effectively computing $O_l^c = \|\theta_l^{c, :, :, :}\|_F^2$. Based on the fraction of non-zero weights that need to be rewired during an epoch i , denoted by the pruning rate p_i , we compute the number of channels that must be pruned from each layer, $c_l^{p_i}$, and prune the $c_l^{p_i}$ channels with the lowest F-norms. We then compute each layer’s importance based on the normalized momentum contributed by its non-zero channels. These importance measures are used to determine the number of zero-F-norm channels $r_l^i \geq 0$ that should be re-grown for each layer l . More precisely, we re-grow the r_l^i zero-F-norm channels with the highest Frobenius norms of their momentum. Note that the iterative prune and regrow strategy of DNR enables the model’s sparse weight tensors to evolve during training.

We note that this approach can easily be extended to enable various other forms of structured pruning. Moreover, despite supporting pruning of both convolution and linear layers, this article focuses on reducing the computational complexity of a DNN. We thus experiment with pruning only convolutional layers because they dominate the computational complexity [26]. Figure 4 shows an illustrative example of key differentiating factors between irregular and structured pruning at the convolutional layer level. The detailed pseudo-code of the proposed training framework is shown in Algorithm 1.

OBSERVATION 2. *DNN layers’ sensitivity toward clean and perturbed images are not necessarily equal, and thus determining layer pruning ratios for robust models is particularly challenging.*

As exemplified in Figure 2(b), for $x = 95\%$ there is a significant difference in the sensitivity of the layers for clean and perturbed image classification. DNR, on the contrary, automatically finds per-layer pruning ratios (overlaid as pruning sensitivity as in [7]) that serves well for both types of image classification targeting a global pruning of 95%.

4 EXPERIMENTS

In this section, we first describe the experimental setup we used to evaluate the effectiveness of the proposed robust training methods. We evaluate the results under both white-box and black-box attack scenarios, comparing our method against other SOTA robust pruning techniques based on ADMM [54] and L_1 lasso [43]. Based on the model performances on adversarially generated images, we further provide a detailed analysis and present a RPMI-based robustness improvement strategy for safety-critical applications with limited compute budget. We then present a thorough

Fig. 4. Illustrative example of (a) irregular and (b) structured channel pruning for a CONV layer l parameterized by θ_l . Here each element θ_l^{mn} of the tensor θ_l is a tensor of $h \times w$ scalar weights.

evaluation of the improved robustness under the lens of gradient obfuscation analysis. We also evaluate the merit of DNR as a clean-image pruning scheme and show that it consistently outperforms contemporary non-iterative model pruning techniques [6, 7, 17, 33]. We finally present an ablation study to empirically evaluate the importance of the dynamic regularizer in the DNR's loss function. We used Pytorch [42] to write the models and trained/tested on an NVIDIA RTX 2080 Ti GPU.⁵

4.1 Experimental Setup

4.1.1 Models and Datasets. We selected three widely used datasets, CIFAR-10 [23] CIFAR-100 [23], and Tiny-ImageNet [15], and picked two well-known CNN models, VGG16 [50] and ResNet18 [16], to evaluate the performances of the disclosed training methods. Both CIFAR-10 and CIFAR-100 datasets have 50K training samples and 10K test samples with an input image size of $32 \times 32 \times 3$. Training and test data size for Tiny-ImageNet are 100k and 10k, respectively, where each image size is of $64 \times 64 \times 3$. For all the datasets we used standard data augmentations (horizontal flip and random crop with reflective padding) to train the models with a batch size of 128.

4.1.2 Adversarial Attack, DNR, DNR++, and FDNR++ Training Settings. Unless stated otherwise, for both PGD-based perturbation and perturbation for FDNR++, we set the attack bound ϵ to $8/255$. For PGD-based attack the attack step size α and the number of attack iterations k were kept to 0.01 and 7, respectively, the same values as in [19]. For FGSM, we chose the same ϵ value as above.

We performed DNR and DNR++ based training for 240/240/120 epochs for CIFAR-10/CIFAR-100/Tiny-ImageNet, with a starting learning rate of 0.1, momentum value of 0.9, and weight decay value of $5e^{-4}$. For CIFAR-10 and CIFAR-100, the **learning rate (LR)** was reduced by a factor of 0.2 after 80, 120, and 160 epochs. For Tiny-ImageNet we reduced the LR value after 40, 60, and 80 epochs. In addition, we hand-tuned ρ to 10^{-4} and set the pruning rate $p = 0.5$. We linearly decreased the pruning rate every epoch by $\frac{p}{total\ epochs}$. Finally, to balance between the clean and adversarial loss, we set β to 0.5 in the DNR and DNR++ training experiments. Lastly, note that we performed warm-up sparse learning [6] for the first five epochs with only the clean image loss function before using the hybrid loss function with dynamic regularization (see Equation (6)) for robust compression for the remaining epochs.

For the scaled sparse noisy tensor-based regularization technique in DNR++, at channel number c of layer l , we define a trainable noise scaling factor η_{lc} . Thus, the noise scaling tensor for a convolutional layer l with N channel is given by $\eta_l = [\eta_{l_0}, \eta_{l_1}, \dots, \eta_{l_N}]$ and we follow the recommendation of [19] to initialize each element to 0.25.

⁵Our models and test codes are available at [this](#) hyperlink.

ALGORITHM 1: DNR Training

Data: weight θ_l , momentum μ_l , binary mask M_l , $l = 0..k$

Data: density d , $i = 0..numEpochs$, pruning rate $p = p_{i=0}$ pT: **irregular** or **channel**

```

1 for  $l \leftarrow 0$  to  $L - 1$  do
2    $\theta_l \leftarrow \text{init}(\theta_l)$  &
    $M_l \leftarrow \text{createMaskForWeight}(\theta_l, d)$ 
3    $\text{applyMaskToWeights}(\theta_l, M_l)$ 
4    $z_l \leftarrow \theta_l \odot M_l$ 
5 end
6 for  $i \leftarrow 0$  to  $numEpochs$  do
7   for  $j \leftarrow 0$  to  $numBatches$  do
8      $\mathcal{L} = \text{computeCleanLoss}(x) +$ 
       $\text{updateDyamicRegularizr}(\theta, z)$ 
9      $\mathcal{L}_{adv} = \text{computePerturbedLoss}(\hat{x})$ 
10     $\mathcal{L}_{tot} = \text{updateRobustLoss}(\mathcal{L}, \mathcal{L}_{adv})$ 
11     $\frac{\partial \mathcal{L}_{tot}}{\partial \theta} = \text{computeGradients}(\theta, \text{batch})$ 
12     $\text{updateMomentumAndWt}(\frac{\partial \mathcal{L}_{tot}}{\partial \theta}, \mu)$ 
13    for  $l \leftarrow 0$  to  $L - 1$  do
14       $\text{applyMaskToWeights}(\theta_l, M_l)$ 
15    end
16  end
17   $tM \leftarrow \text{getTotalMomentum}(\mu)$ 
18   $pT \leftarrow \text{getTotalPrunedWeights}(\theta, p_i)$ 
19   $p_i \leftarrow \text{linearDecay}(p_i)$ 
20  for  $l \leftarrow 0$  to  $L - 1$  do
21     $\mu_l \leftarrow$ 
       $\text{getMomentumContribution}(\theta_l, M_l, tM, pT)$ 
22     $\text{Prune}(\theta_l, M_l, p_i, pT)$ 
23     $\text{Regrow}(\theta_l, M_l, \mu_l \cdot tM, pT)$ 
24     $\text{applyMaskToWeights}(\theta_l, M_l)$ 
25     $z_l \leftarrow \theta_l \odot M_l$ 
26  end
27 end

```

ALGORITHM 2: FDNR++

Data: weight θ_l , momentum μ_l , binary mask M_l , $l = 0..k$, $numSteps$, ϵ .

Data: density d , $i = 0..numEpochs$, pruning rate $p = p_{i=0}$ pT: **irregular** or **channel**

```

1 for  $l \leftarrow 0$  to  $L - 1$  do
2    $\theta_l \leftarrow \text{init}(\theta_l)$  &
    $M_l \leftarrow \text{createMaskForWeight}(\theta_l, d)$ 
3    $\text{applyMaskToWeights}(\theta_l, M_l)$ 
4    $z_l \leftarrow \theta_l \odot M_l$ 
5 end
6  $\kappa \leftarrow 0$ 
7 for  $i \leftarrow 0$  to  $numEpochs/numSteps$  do
8   for  $j \leftarrow 0$  to  $numBatches$  do
9     for  $k \leftarrow 0$  to  $numSteps$  do
10       $\mathcal{L} = \text{computeLoss}(x + \kappa) +$ 
         $\text{updateDyamicRegularizr}(\theta, z)$ 
11       $\frac{\partial \mathcal{L}}{\partial \theta} = \text{computeGradients}(\theta, \text{batch})$ 
12       $\eta \leftarrow \text{updateNoise}(\theta, M)$ 
13       $g_a \leftarrow \text{computeGradients}(x + \kappa, \text{batch})$ 
14       $\text{updateMomentumAndNoisyWt}(\frac{\partial \mathcal{L}}{\partial \theta}, \eta, \mu)$ 
15
16      for  $l \leftarrow 0$  to  $L - 1$  do
17         $\text{applyMaskToWeights}(\theta_l, M_l)$ 
18      end
19       $\kappa \leftarrow \kappa + \epsilon \cdot \text{sign}(g_a)$ 
20       $\kappa \leftarrow \text{clip}(\kappa, -\epsilon, \epsilon)$ 
21    end
22  end
23   $tM \leftarrow \text{getTotalMomentum}(\mu)$ 
24   $pT \leftarrow \text{getTotalPrunedWeights}(\theta, p_i)$ 
25   $p_i \leftarrow \text{linearDecay}(p_i)$ 
26  for  $l \leftarrow 0$  to  $L - 1$  do
27     $\mu_l \leftarrow$ 
       $\text{getMomentumContribution}(\theta_l, M_l, tM, pT)$ 
28     $\text{Prune}(\theta_l, M_l, p_i, pT)$ 
29     $\text{Regrow}(\theta_l, M_l, \mu_l \cdot tM, pT)$ 
30     $\text{applyMaskToWeights}(\theta_l, M_l)$ 
31     $z_l \leftarrow \theta_l \odot M_l$ 
32  end
33 end

```

For FDNR++, we set the *numSteps* value to 4, meaning during every epoch each image batch is repeated four times to generate perturbed image variants to the model. Hence, to not incur any additional training time cost, we reduce the total *numEpochs* by a factor of 4. Thus, the effective epoch count for CIFAR-10/CIFAR-100/Tiny-ImageNet becomes 60/60/30.

4.2 Results for White-box Attack

Definition (Compression Ratio). We define the model compression ratio as $1/d$, where d represents the target global parameter density of the model. Hence, compression ratio⁻¹ is same as the parameter density of the pruned model.

Fig. 5. DNR: Model compression vs. accuracy (on both clean and adversarially generated images) for irregular and channel pruning evaluated with VGG16 on CIFAR-10 ((a) and (b), respectively) and ResNet18 on CIFAR-100 ((c) and (d), respectively). (c) and (f) show the percentage of effective non-zero channels present for channel and irregular pruned models at different values of target parameter density, for VGG16 and ResNet18, respectively.

As shown in Figures 5(a), 5(b), 5(d), and 5(e), for various white-box attack-generated images, DNR can effectively find a robust model with high compression ratio and negligible compromise in accuracy. In particular, for irregular pruning, our method can compress up to $\sim 20\times$ with a negligible drop in accuracy on clean as well as PGD- and FGSM-based perturbed images, compared to the baseline non-pruned models, tested with VGG16 on CIFAR-10 and ResNet18 on CIFAR-100.⁶

Figures 6(a), 6(b), 6(d), and 6(e) show the performance of DNR++ trained VGG16 and ResNet18 on CIFAR-10 and CIFAR-100, respectively, under various pruning schemes. Interestingly, the robustness improvement achieved via DNR++ is valid even at ultra high model compression of $100\times$. For example, at $d = 0.05$ (irregular), the improved robustness against PGD white-box attack for VGG16 on CIFAR-10 is 52.48% with DNR++, which is 9.23% more than that achievable through DNR. Similarly, Figures 7(a), 7(b), 7(d), and 7(e) show the accuracy vs. compression ratio⁻¹ plots for models trained through FDNR++. FDNR++ consistently outperforms DNR by providing improved robustness against FGSM and PGD white-box attack-generated images. From these results it is clear that our proposed class of robust training methods based on the proposed hybrid loss and rewiring method can indeed find a robust DNN model even at ultra-high compression ratios.

OBSERVATION 3. *As the target compression ratio increases, channel pruning degrades adversarial robustness more significantly than irregular pruning.*

As we can consistently see in Figures 5, 6, and 7(b)–7(e), the achievable model compression with negligible accuracy loss for structured (channel) pruned models is $\sim 10\times$ lower than that achievable through irregular pruning. This trend is similar to that of the model’s performance on clean image. However, as we can also see in Figures 5, 6, 7(c), and 7(f), the percentage of

⁶ A similar trend is observed for VGG16 on CIFAR-100 and ResNet18 on CIFAR-10. These are not included in the article for the sake of brevity.

Fig. 6. DNR++: Model compression vs. accuracy (on both clean and adversarially generated images) for irregular and channel pruning evaluated with VGG16 on CIFAR-10 ((a) and (b), respectively) and ResNet18 on CIFAR-100 ((c) and (d), respectively). (e) and (f) show the percentage of effective non-zero channels present for channel and irregular pruned models at different values of target parameter density, for VGG16 and ResNet18, respectively.

Fig. 7. FDNR++: Model compression vs. accuracy (on both clean and adversarially generated images) for irregular and channel pruning evaluated with VGG16 on CIFAR-10 ((a) and (b), respectively) and ResNet18 on CIFAR-100 ((c) and (d), respectively). (e) and (f) show the percentage of effective non-zero channels present for channel and irregular pruned models at different values of target parameter density, for VGG16 and ResNet18, respectively.

Table 2. Results on VGG16 to Classify Tiny-ImageNet

Method	Pruning Type	Compression Ratio	% Channel Present	Accuracy (%)		
				Clean	FGSM	PGD
DNR	Unpruned-baseline	1×	100	50.91	18.19	13.87
	Irregular	11.26×	98.76	52.49	18.36	14.43
	Channel	1.45×	74	51.09	17.92	13.54
DNR++	Unpruned-baseline	1×	100	51.12	22.44	18.02
	Irregular	11.25×	98.77	52.92	22.23	18.05
	Channel	2.01×	60	50.68	20.51	16.69

channels present in the channel-pruned models can be up to $\sim 10\times$ lower than the irregular pruned counterparts, implying the potential for a similarly large speedup in inference time on a large range of compute platforms [6, 39].

Results on Tiny-ImageNet: As shown in Table 2, DNR can compress the model up to $11.26\times$ without any compromise in performance for both clean and perturbed image classification. For DNR++, the robustness improves by up to 3.87% in absolute scale, with similar clean image classification performance, under similar compression. For channel-pruned models, DNR++ can improve adversarial classification performance by 3.15% over that achievable through DNR with a reduced effective channel presence of only $\sim 60\%$. A similar trend is observed for FDNR++ but not included in the article for the sake of brevity.

4.3 Results for Black-box Attack

Table 3 shows the performance details of the models generated via DNR, DNR++, and FDNR++ under various black-box adversarial attack scenarios. In particular, similar to the trend of the white-box attack scenario, DNR++ provides the best-performing black-box robust models. Here, to generate black-box attacks we used attacker models of the same architecture as but of different target density than the victim model.⁷

4.4 DNR vs. DNR++ vs. FDNR++

This subsection provides an empirical performance comparison of DNR, DNR++, and FDNR++. Figures 8(a) and 8(b) show the accuracy comparison among the proposed methods on clean and PGD-generated perturbed images, respectively. In particular, DNR++ performs better than both DNR and FDNR++ on the strongly perturbed images, with similar performance on clean images. FDNR++, on the other side, performs superior to DNR on perturbed images, with negligible performance drops on clean images. Finally, Figure 8(c) demonstrates the superiority of FDNR++ in terms of improved training speed of $\sim 6\times$ compared to the other two methods. These results indeed show FDNR++ to be a low-cost training alternative of DNR++. Note, however, that FDNR++ generally yields models that are less robust than obtained through DNR++. For example, on PGD-generated adversarial CIFAR-10 test images, FDNR++ generates a pruned VGG16 with classification performance of 61.09%, which is significantly lower than the 65.85% performance of the DNR++ generated model.

4.5 Comparison with SOTA

We now compare the performance of DNR, DNR++, and FDNR++ with ADMM- [54], L_1 lasso-, and RSR-based [43] robust pruning. For ADMM-based robust pruning, we followed a three-stage

⁷Here, the attacker model is the model that generates adversarial images and the victim model is the one to which these adversarial images are sent to fool or degrade classification performance.

Table 3. Results for Black-box Attack-generated Images on VGG16 and ResNet18

Method	Model Name	Baseline	Pruning Type	Compression	Accuracy (%)		
		Clean Acc. (%)		Ratio	Clean	FGSM	PGD
Dataset: CIFAR-10							
DNR	VGG16	86.35	Irregular	20.85×	86.74	64.75	62.39
DNR++		86.32		20.92×	86.8	72.88	70.14
FDNR++		83.10		20.8×	84.32	65.92	61.43
DNR	VGG16	86.35	Channel	20×	85.83	65.28	62.8
DNR++		86.32		20×	85.77	68.66	65.85
FDNR++		83.10		20×	84.0	64.74	61.09
DNR	ResNet18	87.9	Irregular	21.57×	87.32	66.34	64.32
DNR++		88.01		21.1×	87.97	69.76	66.71
FDNR++		86.91		21.1×	85.53	65.6	61.51
DNR	ResNet18	87.9	Channel	20×	87.49	67.4	64.85
DNR++		88.01		20×	87.74	69.7	66.95
FDNR++		86.91		20×	86.52	65.9	62.11

Here, the baseline clean acc. % corresponds to the clean image classification accuracy of an unpruned robust model of same architecture type.

Fig. 8. Comparison of performance between DNR, DNR++, and FDNR++ on (a) clean and (b) adversarial image generated via PGD attack evaluated on CIFAR-10 with VGG16. (c) shows the normalized training time required to yield robust compressed models for VGG16 (on CIFAR-10) and ResNet18 (on CIFAR-100).

compression technique, namely *pre-training*, *ADMM-based pruning*, and *masked retraining*, performing pruning for 30 epochs with $\rho_{admm} = 10^{-3}$ as described in [54]. L_1 lasso-based pruning adds a L_1 regularizer to its loss function to penalize the weight magnitudes, where the regularizer coefficient determines the penalty factor. RSR improved the performance of L_1 lasso by proposing a robust regularizing noise. Table 4 shows that our proposed DNR++ method outperforms both ADMM- and L_1 lasso-based approaches by a considerable margin, retaining advantages of both worlds. In particular, compared to ADMM, with the VGG16 (ResNet18) model on CIFAR-10, DNR++ provides up to 11.86% (7.49%) increased classification accuracy on perturbed images with 1.24× (1.44×) higher compression. Compared to L_1 lasso, we achieve 10.38× (3.15×) higher compression and up to 10.47% (7.26%) and 3.56% (2.05%) increased accuracy on perturbed and clean images, respectively, for VGG16 (ResNet18) on CIFAR-10 classification. Finally, compared to RSR, DNR++ provides up to 5.91% higher performance on perturbed images with up to $\sim 10\times$ better compression.

OBSERVATION 4. *Naively tuned per-layer pruning ratio degrades both robustness and clean-image classification performance of a model.*

For this, we evaluated robust compression using naive ADMM, i.e., using naively tuned per-layer pruning ratio (all but the first layer $\sim x\%$ for a $x\%$ total pruning ratio). As shown in Table 4,

Table 4. Comparison of Our Dynamic Rewiring-based Methods with ADMM-, L_1 Lasso-, and RSR-Based Robust Pruning Schemes on CIFAR-10

Model	Method	No Pre-trained Model	Per-layer Pruning Ratio Knowledge Not Needed	Target Pruning Met	Pruning Type	Compression Ratio	Accuracy (%)		
							Clean	FGSM	PGD
VGG16	ADMM [54]	×	×	✓	Irregular	16.78×	86.34	49.52	40.62
	ADMM naive	×	✓	✓		19.74×	83.87	42.46	32.87
	L_1 Lasso [43]	✓	✓	×		2.01×	83.24	50.32	42.01
	RSR [43]	✓	✓	×		2.17×	83.02	54.16	47.7
	DNR	✓	✓	✓		20.85×	86.74	52.92	43.21
	DNR++	✓	✓	✓		20.92×	86.8	60.07	52.48
	FDNR++	✓	✓	✓		20.8×	84.32	57.7	51.7
ResNet18	ADMM [54]	×	×	✓	Irregular	14.6×	87.15	54.65	46.57
	ADMM naive	×	✓	✓		19.74×	86.10	50.49	42.24
	L_1 Lasso [43]	✓	✓	×		6.84×	85.92	55.20	46.80
	RSR [43]	✓	✓	×		6.83×	86.79	60.35	53.03
	DNR	✓	✓	✓		21.57×	87.32	55.13	47.35
	DNR++	✓	✓	✓		21.1×	87.97	61.14	54.06
	FDNR++	✓	✓	✓		21.1×	85.53	56.93	50.58

Romanized numbers in the table are results of our experiments, and italicized values are directly taken from the respective original papers.

Fig. 9. FDNR++: Model performance under attacks trained with different seeds having same model architecture as the victim models. (a) and (b) show performance on FGSM- and PGD-generated images, respectively, when both attacker and victim have $d = 0.1$, and (c) and (d) show the respective performance when $d = 0.05$. Note the x and y directions signify the seeds of the victim model and attacker model, respectively. We used ResNet18 on CIFAR-100 to conduct these experiments.

this clearly degrades the performance, implying layer-wise parameter density tuning is necessary for ADMM to perform well.

4.6 Random Pruned Model Selection for Inference (RPMI)

OBSERVATION 5. Consider the set of possible models used to create adversarial images. These models should, in general, have the same architecture and similar clean-image accuracy as the victim model. However, they could have varying parameter densities and be generated using different random seeds. Interestingly, the strength of adversarial images on the victim model that are generated from models of the same density but different seeds is similar to images generated from models with varying densities.

Figures 9(a) to 9(d) show the performance of a model on perturbed images generated from a different model of the same architecture with the same density trained with different seeds. The diagonal elements from the right represent a white-box attack because the seed is the same for both the attacker and the victim models and thus shows lower classification accuracy on perturbed images.

Fig. 10. FDNR++: Model performance under attacks from attacker models of various target density for (a) FGSM- and (b) PGD-generated adversarial images. Similar trend is observed for DNR and DNR++ as well. We used a fixed seed value to train ResNet18 on CIFAR-100 to conduct these experiments.

Fig. 11. FDNR++: Weight histogram plot (of non-zero weights only) of the 4th CONV layer of ResNet18 for CIFAR-100 for model trained with different seeds (s_0 and s_1) to meet various target densities.

The performance of the victim model improves significantly when the target is trained with a different seed. As observed above, we see similar performance improvements when the target densities are changed, comparing the non-right-diagonal values of Figure 9 to that of Figure 10. Interestingly, as depicted in Figure 11, the distributions of the weights of two models of the same d trained with different seeds are largely similar, despite one being able to largely act as a black box to another.

Based on this observation, we propose a simple enhancement of our method to significantly improve model robustness. First, we choose a set of adversarially trained candidate models to have the same target density to ensure optimal performance of the underlying hardware, in case it was optimized for a specific target density. In particular, we propose to create a library of adversarially trained models compressed to a density similar to the victim but trained with different seeds, and randomly choose models from that library to perform inference on specific input image batches.

Intuitively, this RPMI, detailed in Algorithm 3, makes the attacker's task difficult in two respects: (1) it must collect model information of all candidate models, and (2) it must predict the sequence of models to determine which model to be used for inference for a specific image batch. This is particularly important because, even with the knowledge of all candidate models, the attacker may still fail to create a white-box attack because it fails to guess the specific sequence of random models. To evaluate the efficacy of the proposed RPMI method, we took two candidate models g_{θ_0} and g_{θ_1} in the model pool each with target density $d = 0.1$ but trained with two different seeds s_0 and s_1 , respectively. We assume the attacker has knowledge of one of the models g_{θ_1} . Figure 12 shows that, as we increase the probability of selecting g_{θ_0} (which acts as a black box to the attacker) in the random model sampling of RPMI, the accuracy on adversarially perturbed images improves. The baseline plots are the performance of the victim model under a white-box attack.

ALGORITHM 3: RPMI Algorithm

Data: Pruned model library $lib_p = [g_{\theta_1}, g_{\theta_2}, \dots, g_{\theta_i}]$, adversarial dataset D_{adv} generated from g_{θ_m} , where $m \in [1, \dots, i]$

- 1 **for** $\hat{x} \in D_{adv}$ **do**
- 2 Randomly choose a pruned model g_{θ_j} from lib_p
- 3 Evaluate $\hat{y} = g_{\theta_j}(\hat{x})$
- 4 **end**
- 5 **Return** $\hat{y}$

Fig. 12. Performance of RPMI evaluated with ResNet18 on CIFAR-100. The density of the candidate models is $d = 0.1$.

4.7 Estimate of Parameter Storage and Computation Cost of the Pruned Models

We now discuss the parameter storage and compute cost (FLOPs) of the pruned models. For an unpruned model, let $\mathcal{P}_l$ be the number of parameters present in layer l . With homogeneous FP-32 representation, the total storage requirement (MB) of the model weights can be given by [31]

$$M_{fp32} = 4 * \left(\sum_{l=0}^{L-1} \frac{\mathcal{P}_l}{2^{20}} \right). \quad (10)$$

For a pruned model, the parameter storage cost can be estimated as

$$M_{fp32}^P = 4 * \left(\sum_{l=0}^{L-1} \frac{\mathcal{P}_l}{2^{20}} \right) * d, \quad (11)$$

where d represents the parameter density of the model. Here, the estimated storage cost saving is directly proportional to the saving in non-zero weight count. Note, however, that this estimate ignores the cost of indexing the irregularly pruned models, which depends on the specifics of the compression technique applied.

The computation cost (FLOPs) of an unpruned convolutional layer l with parameter tensor $\theta_l \in \mathbb{R}^{M \times N \times h \times w}$ that creates an output feature map of size $H_o \times W_o \times M$ can be given by [26]

$$F_l^{conv} = H_o * W_o * h * w * M * N. \quad (12)$$

For a linear layer l with node fanin f_{in} and node fanout f_{out} , the FLOPs can be given by

$$F_l^{lin} = f_{in} * f_{out}. \quad (13)$$

From the above equation, we can compute the FLOPs for a pruned L -layer model as

$$F^{total} = \sum_{l=0}^{L-1} F_l * d_l, \quad (14)$$

where d_l is the parameter density of layer l . Note that here, we assume that the computation cost for zero-valued weights is saved through clock gating. Figure 13 compares per-layer FLOPs between unpruned and pruned ResNet18 models.

4.8 Gradient Obfuscation Analysis

We conducted the experiments listed in Table 5 to verify whether the improved robustness of the generated models comes from any approximation of the true gradient based on a single sample.

Fig. 13. Per-layer FLOP comparison of an unpruned ($d = 1.0$) and an irregularly pruned ResNet18 model ($d = 0.05$) used to classify CIFAR-10. Here, C_n and S_m represent convolutional layer n and shortcut layer m , respectively.

Table 5. Checklist Set of Tests for Characteristic Behaviors Caused by Obfuscated and Masked Gradients [2]

Checks to Identify Gradient Obfuscation	Fail	Pass
(i) One-step attack performs worse compared to iterative attacks		✓
(ii) Black-box attacks perform worse compared to white-box attacks		✓
(iii) Increasing perturbation bound can increase attack strength		✓
(iv) Unbounded attacks can reach ~100% success		✓
(v) Adversarial example can't be found through random sampling		✓

In particular, the five experiments listed in the table were proposed in [2] and can identify potential gradient obfuscation. Note that here we say an attack performs worse than another when the classification performance on the attack-generated images is higher as opposed to the images generated through other attacks.

For all the models on all datasets, the one-step FGSM performs poorly compared to its iterative counterpart PGD as exemplified in Figures 5, 6, and 7. This certifies the success of Test (i). Table 3 shows that black-box attack-generated images yield weaker attack than the white-box counterpart, certifying the success of Test (ii). To conduct Tests (iii) and (iv) we analyzed the adversarial classification performance of ResNet18 on CIFAR-10 by increasing the attack bound ϵ . Figures 14(a) and 14(c) present the performances of the DNR-generated ResNet as a function of the PGD attack bound ϵ . In particular, we can see that for both irregular and channel-pruned models, the accuracy degrades as the ϵ increases. Test (v) can fail if and only if gradient-based attacks cannot provide adversarial examples for the model to misclassify. It is clear from our extensive experiments that FGSM and PGD, both variants of gradient-based attacks, can largely fool the models despite our training.

We also evaluated the model performance with increased attack strength by increasing the number of iterations K of PGD (Figures 14(b) and 14(d)) and found that the model's robustness decreases with increasing K . Thus, based on these evaluations, we conclude that even if the models include obfuscated gradients, they are not primary sources of the robustness for the model generated by our training methods.

4.9 Ablation Study

To understand the performance of the proposed hybrid loss function with a dynamic L_2 regularizer, we performed ablation with both VGG16 and ResNet18 generated by DNR on CIFAR-10 for a target parameter density of 5% and 50% using irregular and channel pruning, respectively. As shown in

Fig. 14. On CIFAR-10, the perturbed data accuracy of ResNet18 under PGD attack versus increasing (a), (c) attack iteration, and (b), (d) attack bound ϵ for irregular (5% density) and channel-pruned (50% density) models, respectively.

Table 6. Comparison of DNR with and without the Dynamic Regularizer for CIFAR-10 Classification

Model	Method: DNR	Accuracy (%) with Irregular Pruning			Accuracy (%) with Channel Pruning		
		Clean	FGSM	PGD	Clean	FGSM	PGD
VGG16	Without dynamic L_2	87.01	50.09	40.62	86.28	49.49	41.25
	With dynamic L_2	86.74	52.92	43.21	85.83	51.03	42.36
ResNet18	Without dynamic L_2	87.45	53.52	45.33	87.97	53.10	45.91
	With dynamic L_2	87.32	55.13	47.35	87.49	56.09	48.33

Table 6, using the dynamic regularizer improves the adversarial classification accuracy by up to 2.83% for VGG16 and $\sim 3\%$ for ResNet18 with similar clean-image classification performance.

4.10 Pruning to Classify Clean-only Images

To evaluate the merit of DNR as a clean-image-only pruning scheme (DNR-C), we performed training using DNR with the same loss function minus the adversarial loss term (by setting $\beta = 1.0$ in Equation (6)) to reach a target pruning ratio. Table 7 shows that our approach consistently outperforms other state-of-the-art non-iterative pruning approaches based on momentum information [6, 7], reinforcement-learning-driven auto-compression (AMC) [17], and connection sensitivity [33]. We have further compared the performance of DNR-C with existing state-of-the-art approaches [1, 35, 53, 55]⁸ that are iterative and require pre-trained unpruned model for model compression. The δ value in the seventh column of Table 7 represents the error difference from corresponding non-pruned baseline models as mentioned in the respective original manuscripts. We also present the performance on CIFAR-100 for VGG16 and on ResNet18 and Tiny-ImageNet for VGG16.⁹ In particular, we achieve up to $34.57\times$ ($12.61\times$) compression on the CIFAR-10 dataset with irregular (channel) pruning, maintaining accuracy similar to the baseline. On CIFAR-100, compression of up to $22.45\times$ ($5.57\times$) yields no significant accuracy drop (less than 2.7% in top-1 accuracy) with irregular (channel) pruning. Moreover, our evaluation shows that a possible practical speedup of up to $6.06\times$ for CIFAR-10 and $2.41\times$ for CIFAR-100 can be achieved through channel pruning using DNR-C. For Tiny-ImageNet, DNR-C can provide compression and speedup of up to $11.55\times$ and $1.53\times$, respectively, with negligible accuracy drop.

⁸For NISP [55] we report accuracy with ResNet56.

⁹To have an “apples to apples” comparison we provide results on the ResNet50 model for classification on CIFAR-10. All other simulations are done on only the ResNet18 variant of ResNet.

Table 7. Comparison with State-of-the-art Non-iterative Pruning Schemes on CIFAR-10 and Comparison of Deviation from Baseline on CIFAR-100 and Tiny-ImageNet

Dataset	Model	Method	Pruning Type	Compression Ratio	Error (%) Top-1	δ from Baseline	Speedup
CIFAR-10	VGG16	SNIP [33]	Irregular	32.33×	8.00	-0.26	-
		Sparse learning [6]		32.33×	7.00	-0.5	-
		DNR-C		34.57×	6.50	-0.09	1.29×
		Filter pruning [35]	Structured	2.78×	6.60	+0.15	1.52×
		AutoPrune [53]		4.72×	8.50	-0.90	4.35×
		SuperSlash [1]		16.67×	8.70	-2.10	3.49×
		DNR-C		12.61×	8.00	-1.5	6.06×
	ResNet50	GSM [7]	Irregular	10×	6.20	-0.25	-
		AMC [17]		2.5×	6.45	+0.02	-
		DNR-C		20×	4.8	-0.07	1.75×
	ResNet18	NISP [55]	Structured	1.76×	6.99	-0.03	1.77×
CIFAR-100	VGG16	DNR-C	Irregular	20×	27.14	-1.04	1.07×
			Structured	2.76×	28.78	-2.68	2.06×
	ResNet18	DNR-C	Irregular	22.45×	24.9	-1.17	1.13×
			Structured	5.57×	25.28	-1.55	2.41×
Tiny ImageNet	VGG16	DNR-C	Irregular	11.55×	40.96	+0.36	1.01×
			Structured	1.74×	42.61	-1.28	1.53×

Romanized numbers in the table are results of our experiments, and italicized values are directly taken from the respective original papers.

5 CONCLUSIONS AND FUTURE WORK

This article addresses the largely unexplored problem of achieving ultra-high compression of DNN models while maintaining their robustness through a non-iterative training approach. In particular, the proposed network rewiring-based DNR training method leverages a sparse-learning strategy with a hybrid loss function that has a dynamic regularizer to achieve better tradeoffs between accuracy, model size, and robustness. We then proposed a robustness improvement strategy via sparse parametric noise regularization and extended the idea to “free” adversarial training by presenting DNR++ and FDNR++, respectively. In particular, DNR++ can provide an increased accuracy on perturbed images by up to $\sim 10.5\%$ with improved compression with respect to the existing SOTA methods. Furthermore, our extension to support channel pruning shows that compressed models produced by DNR can have a practical inference speedup of up to $\sim 10\times$.

More generally, exploring a deeper biological foundation and interpretability of model compression are important directions of future research.

6 BROADER IMPACT

The demand for model robustness to perform well under various adversarial attacks has grown rapidly over the past few years, primarily due to deployment of AI-enabled hardware for safety-critical applications, including autonomous vehicles and drones. As most of these devices are limited by computation and/or storage budgets, deployment of large over-parameterized models is not an option in the majority of the cases. In this context, the retention of model robustness for a compute-efficient compressed model is of the utmost importance. We believe our proposed training method for compressed yet robust DNNs will be a key factor in helping enable limited-resource AI-driven hardware where safety is a concern.

A APPENDIX

A.1 DNR and Per-layer Pruning Ratios

As the architectures of modern DNNs deepen to hundreds of layers, manually setting per-layer pruning ratios to reach a desired global compression ratio becomes increasingly impractical. In particular, we assert that layers may not be equally *sensitive* toward clean and perturbed images and thus determining layer pruning ratios for robust models is particularly challenging.

To quantify per-layer sensitivity, we follow a similar procedure as proposed in [7]. In particular, we estimate the sensitivity of a layer by measuring the percentage reduction in classification accuracy on both clean and adversarial images caused by pruning a layer by a fixed percentage x , keeping all other layers unpruned. The work in [7] argued that the closer this percentage is to 1, the more sensitive the layer and thus less layer-pruning ratio is recommended. The results for the 13 layers of VGG16 with a target pruning ratio of 95% are plotted in Figure 15. The plot shows that there is indeed a wide variance in the estimated sensitivity for both clean and adversarial images across layers.

We then compared these estimates to the percentage of non-zero weights present after DNR targeting the same 95% global pruning ratio, the results of which are also plotted in Figure 15. Interestingly, the plots show that the estimated layer sensitivity exhibits the same general trend as the percentage of non-zero weights obtained by DNR (i.e., earlier layers are more sensitive than later layers and thus need more % non-zero weights). Although empirical in nature, this result suggests that DNR is effective in evaluating the sensitivity and assigning weights to each layer without any human intervention.

Figure 16 shows a few more examples of layer sensitivity found by DNR for both irregular and channel pruning. Interestingly, we discovered that for both types of pruning, the earlier layers are more sensitive than the later layers.

Fig. 15. Layer-wise sensitivity estimated by both layer-wise pruning and DNR on VGG16 for the CIFAR-10 dataset, with a target pruning (irregular) ratio of 95%. Here, both the unpruned and DNR-found models trained with both clean and PGD-generated samples.

Fig. 16. Layer-wise sensitivity scores estimated through DNR on VGG16 for CIFAR-10 and CIFAR-100 dataset with (a) irregular pruning to a target global pruning ratio of 95% and (b) channel pruning to a target global pruning ratio of 50%.

A.2 Details of Models Used

Table 8 shows the details of the models we used for our experimental evaluation.

Table 8. Models Used for Our Experiments

Dataset	CIFAR-10, CIFAR-100	
Model	VGG16	ResNet18
# Parameters	14.7 M	11.16 M
Architecture	$[C64/3 \times 3] \times 2$, MaxPool/ $2 \times 2 - 2$ $[C128/3 \times 3] \times 2$, MaxPool/ $2 \times 2 - 2$ $[C256/3 \times 3] \times 3$, MaxPool/ $2 \times 2 - 2$ $[C512/3 \times 3] \times 3$, MaxPool/ $2 \times 2 - 2$ $[C512/3 \times 3] \times 3$, MaxPool/ $2 \times 2 - 2$ 1FC	$C64/3 \times 3$ $[C64/3 \times 3] \times 4$ $[C128/3 \times 3] \times 4$ $[C256/3 \times 3] \times 4$ $[C512/3 \times 3] \times 4$ AvgPool, 1FC

We omitted batch-norm and activation layers for brevity. We specify convolution (C) layers with number of filters and kernel size. MaxPool layers are specified with kernel and stride size. The repetition number of a layer is specified outside the corresponding bracket.

REFERENCES

- [1] Hazoor Ahmad, Tabasher Arif, Muhammad Abdullah Hanif, Rehan Hafiz, and Muhammad Shafique. 2020. Super-Slash: A unified design space exploration and model compression methodology for design of deep learning accelerators with reduced off-chip memory access volume. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 4191–4204.
- [2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* (2018).
- [3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122.
- [4] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*. IEEE, 39–57.
- [5] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. 2015. DeepDriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*. 2722–2730.
- [6] Tim Dettmers and Luke Zettlemoyer. 2019. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840* (2019).
- [7] Xiaohan Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, Ji Liu, et al. 2019. Global sparse momentum SGD for pruning very deep neural networks. In *Advances in Neural Information Processing Systems*. 6379–6391.
- [8] Thu Dinh and Jack Xin. 2018. Convergence of a relaxed variable splitting method for learning sparse neural networks via l_1, l_0 , and transformed- l_1 penalties. *arXiv preprint arXiv:1812.05719*.

- [9] Arash Fayyazi, Souvik Kundu, Shahin Nazarian, Peter A. Beerel, and Massoud Pedram. 2019. CSrram: Area-efficient low-power ex-situ training framework for memristive neuromorphic circuits based on clustered sparsity. In *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI'19)*. IEEE, 465–470.
- [10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 580–587.
- [11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- [12] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. 2020. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*.
- [13] Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. 2019. An alternative surrogate loss for PGD-based adversarial testing. *arXiv preprint arXiv:1910.09338*.
- [14] Shupeng Gui, Haotao N. Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. 2019. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*. 1283–1294.
- [15] Lucas Hansen. 2015. Tiny ImageNet challenge submission. *CS 231N*.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
- [17] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. AMC: AutoML for model compression and acceleration on mobile devices. In *The European Conference on Computer Vision (ECCV'18)*.
- [18] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 1389–1397.
- [19] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. 2019. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 588–597.
- [20] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2021. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*. 15262–15271.
- [21] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.
- [22] Lang Huang, Chao Zhang, and Hongyang Zhang. 2020. Self-adaptive training: Beyond empirical risk minimization. *Advances in Neural Information Processing Systems* 33 (2020).
- [23] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. *Technical Report, Citeseer* (2009).
- [24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.
- [25] Souvik Kundu, Mahdi Nazemi, Peter A. Beerel, and Massoud Pedram. 2021. DNR: A tunable robust pruning framework through dynamic network rewiring of DNNs. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 344–350.
- [26] Souvik Kundu, Mahdi Nazemi, Massoud Pedram, Keith M. Chugg, and Peter A. Beerel. 2020. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Computers*.
- [27] Souvik Kundu, Massoud Pedram, and Peter A. Beerel. 2021. Hire-SNN: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5209–5218.
- [28] Souvik Kundu, Saurav Prakash, Haleh Akrami, Peter A. Beerel, and Keith M. Chugg. 2019. pSConv: A pre-defined sparse kernel based convolution for deep CNNs. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton'19)*. IEEE, 100–107.
- [29] Souvik Kundu, Qirui Sun, Yao Fu, Massoud Pedram, and Peter A. Beerel. 2021. Analyzing the confidentiality of undistillable teachers in knowledge distillation. In *Advances in Neural Information Processing Systems (NeurIPS'21)*, 34.
- [30] Souvik Kundu and Sairam Sundaresan. 2021. AttentionLite: Towards efficient self-attention models for vision. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'21)*. 2225–2229.
- [31] Souvik Kundu, Shikai Wang, Qirui Sun, Peter A. Beerel, and Massoud Pedram. 2022. BMPQ: Bit-gradient sensitivity-driven mixed-precision quantization of DNNs from scratch. In *DATE*.
- [32] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP'19)*. IEEE, 656–672.
- [33] Namhoon Lee, Thalaisyngam Ajanthan, and Philip H. S. Torr. 2018. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

- [34] Dongze Li, Wei Wang, Hongxing Fan, and Jing Dong. 2021. Exploring adversarial fake images on face manifold. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*. 5789–5798.
- [35] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.
- [36] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I. Sánchez. 2017. A survey on deep learning in medical image analysis. *Medical Image Analysis* 42 (2017), 60–88.
- [37] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 369–385.
- [38] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.
- [39] Xiaolong Ma, Sheng Lin, Shaokai Ye, Zhezhi He, Linfeng Zhang, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, et al. 2021. Non-structured DNN weight pruning—is it beneficial in any platform? *IEEE Transactions on Neural Networks and Learning Systems*.
- [40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.
- [41] Dongyu Meng and Hao Chen. 2017. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 135–147.
- [42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.
- [43] Adnan Siraj Rakin, Zhezhi He, Li Yang, Yanzhi Wang, Liqiang Wang, and Deliang Fan. 2019. Robust sparse regularization: Simultaneously optimizing neural network robustness and compactness. *arXiv preprint arXiv:1905.13074*.
- [44] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. 2021. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*.
- [45] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7263–7271.
- [46] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang. 2019. ADMM-NN: An algorithm-hardware co-design framework of DNNs using alternating direction methods of multipliers. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 925–938.
- [47] Athena Sayles, Ashish Hooda, Mohit Gupta, Rahul Chatterjee, and Earlene Fernandes. 2021. Invisible perturbations: Physical adversarial examples exploiting the rolling shutter effect. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*. 14666–14675.
- [48] Vikash Sehwal, Shiqi Wang, Prateek Mittal, and Suman Jana. 2020. Hydra: Pruning adversarially robust neural networks. *arXiv preprint arXiv:2002.10509*.
- [49] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *arXiv preprint arXiv:1904.12843*.
- [50] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [51] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.
- [52] Eric Wong, Leslie Rice, and J. Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*.
- [53] Xia Xiao and Zigeng Wang. 2019. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems (NeurIPS'19)*, 32.
- [54] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. 2019. Adversarial robustness vs. model compression, or both. In *The IEEE International Conference on Computer Vision (ICCV'19)*, Vol. 2.
- [55] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. 2018. NISP: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9194–9203.
- [56] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*. PMLR, 7472–7482.

Received 6 August 2021; revised 30 November 2021; accepted 7 January 2022