Pruning Adversarially Robust Neural Networks without Adversarial Examples

Tong Jian[1,†], Zifeng Wang[1,†], Yanzhi Wang[2], Jennifer Dy[1], Stratis Ioannidis[1]
Department of Electrical and Computer Engineering
Northeastern University
[1]{jian, zifengwang, jdy, ioannidis}@ece.neu.edu
[2]yanz.wang@northeastern.edu

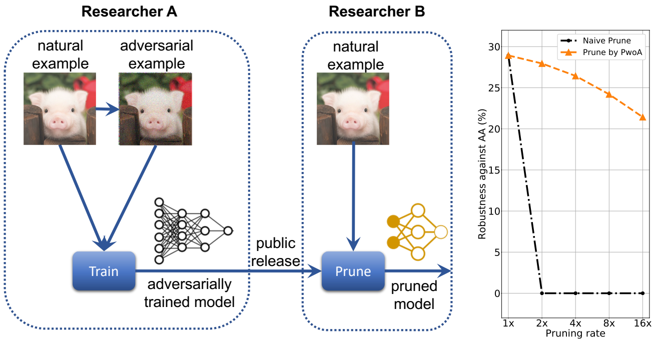
***Abstract*—Adversarial pruning compresses models while preserving robustness. Current methods require access to adversarial examples during pruning. This significantly hampers training efficiency. Moreover, as new adversarial attacks and training methods develop at a rapid rate, adversarial pruning methods need to be modified accordingly to keep up. In this work, we propose a novel framework to prune a previously trained robust neural network while maintaining adversarial robustness, *without* further generating adversarial examples. We leverage concurrent self-distillation and pruning to preserve knowledge in the original model as well as regularizing the pruned model via the Hilbert-Schmidt Information Bottleneck. We comprehensively evaluate our proposed framework and show its superior performance in terms of both adversarial robustness and efficiency when pruning architectures trained on the MNIST, CIFAR-10, and CIFAR-100 datasets against five state-of-the-art attacks.**

***Index Terms*—**

I. Introduction

The vulnerability of deep neural networks (DNNs) to adversarial attacks has been the subject of extensive research recently [1]–[3]. Such attacks are intentionally crafted to mislead DNNs towards incorrect predictions, e.g., by adding delicately but visually imperceptible perturbations to original, natural examples [4]. Adversarial robustness, i.e., the ability of a trained model to maintain its predictive power despite such attacks, is an important property for many safety-critical applications [5]–[7]. The most common and effective way to attain adversarial robustness is via *adversarial training* [8]–[10], i.e., training a model over adversarially generated examples. Adversarial training has shown reliable robustness performance against improved attack techniques such as projected gradient descent (PGD) [1], the Carlini & Wagner attack (CW) [2] and AutoAttack (AA) [3]. Nevertheless, adversarial training is computationally expensive [1], [11], usually 3×–30× [12] longer than natural training, precisely due to the additional cost of generating adversarial examples.

As noted by Madry et al. [1], achieving adversarial robustness requires a significantly wider and larger architecture than that for natural accuracy. The large network capacity required by adversarial training may limit its deployment on resource-constrained hardware or real-time applications. Weight pruning is a prominent compression technique to reduce model size without notable accuracy degradation [13]–[16]. While researchers have extensively explored weight pruning, only a few recent works have studied it jointly with adversarial robustness. Ye et al. [17], Gui et al. [18], and Sehwag et al. [19] apply active defense techniques with pruning in their research. However, these works require access to adversarial examples during pruning. Pruning is itself a laborious process, as effective pruning techniques simultaneously finetune an existing, pre-trained network; incorporating adversarial examples to this process significantly hampers training efficiency. Moreover, adversarial pruning techniques tailored to specific adversarial training methods need to be continually revised as new methods develop apace.



(a) Motivation of our PwoA framework

(b) Naïve Prune vs. PwoA

Fig. 1: (a) A DNN publicly released by researcher A, trained adversarially at a large computational expense, is pruned by Researcher B and made executable on a resource-constrained device. Using PwoA, pruning by B is efficient, requiring only access to natural examples. (b) Taking a pre-trained WRN34-10 pruned on CIFAR-100 as an example, pruning an adversarially robust model in a naïve fashion, without generating any adversarial examples, completely obliterates robustness against AutoAttack [3] even under a 2× pruning ratio. In contrast, our proposed PwoA framework efficiently preserves robustness for a broad range of pruning ratios, without any access to adversarially generated examples. To achieve similar robustness, SOTA adversarial pruning methods require 4×–7× more training time (see Figure 2 in Section V-C).

In this paper, we study how take a dense, adversarially robust DNN, that has already been trained over adversarial examples, and prune it *without any additional adversarial training*. As a motivating example illustrated in Figure 1(a), a DNN publicly released by researchers or a company, trained

[†] Both authors contributed equally to this work.

adversarially at a large computational expense, could be subsequently pruned by other researchers to be made executable on a resource-constrained device, like an FPGA. Using our method, the latter could be done efficiently, without access to the computational resources required for adversarial pruning.

Restricting pruning to access only natural examples poses a significant challenge. As shown in Figure 1(b), naïvely pruning a model without adversarial examples can be catastrophic, obliterating all robustness against AutoAttack. In contrast, our PwoA is notably robust under a broad range of pruning rates.

Overall, we make the following contributions:

1) We propose PwoA, an end-to-end framework for pruning a pre-trained adversarially robust model without generating adversarial examples, by (a) *preserving robustness* from the original model via self-distillation [20] and (b) *enhancing robustness* from natural examples via Hilbert-Schmidt independence criterion (HSIC) as a regularizer [21], [22]. Our code is publicly available. [†]
2) Our work is the *first to study how an adversarially pre-trained model can be efficiently pruned without access to adversarial examples.* This is an important, novel challenge: prior to our study, it was unclear whether this was even possible. Our approach is generic, and is neither tailored nor restricted to specific pre-trained robust models, architectures, or adversarial training methods.
3) We comprehensively evaluate PwoA on pre-trained adversarially robust models publicly released by other researchers. In particular, we prune five publicly available models that were pre-trained with state-of-the-art (SOTA) adversarial methods on the MNIST, CIFAR-10, and CIFAR-100 datasets. Compared to SOTA adversarial pruning methods, PwoA can prune a large fraction of weights while attaining comparable–or better–adversarial robustness, at a 4×–7× training speed up.

We omit related work and experimental details from this short paper; both can be found in the extended version [23].

II. Background

We use the following standard notation throughout the paper. In the standard k-ary classification setting, we are given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^{d_X}, y_i \in \{0,1\}^k$ are i.i.d. samples drawn from joint distribution P_{XY}. Given an L-layer neural network $h_{\boldsymbol{\theta}} : \mathbb{R}^{d_X} \to \mathbb{R}^k$ parameterized by weights $\boldsymbol{\theta} := \{\boldsymbol{\theta}_l\}_{l=1}^L \in \mathbb{R}^{d_{\boldsymbol{\theta}_l}}$, where $\boldsymbol{\theta}_l$ is the weight corresponding to the l-th layer, for $l = 1, \ldots, L$, we define the standard learning objective as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{XY}[\ell(h_{\boldsymbol{\theta}}(X), Y)] \approx \frac{1}{n}\sum_{i=1}^{n} \ell(h_{\boldsymbol{\theta}}(x_i), y_i), \tag{1}$$

where $\ell : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ is a loss function, e.g., cross-entropy.

A. Adversarial Robustness

We call a network *adversarially robust* if it maintains high prediction accuracy against a constrained adversary that perturbs input samples. Formally, prior to submitting an input sample $x \in \mathbb{R}^{d_X}$, an adversary may perturb x by an arbitrary $\delta \in \mathcal{B}_r$, where $\mathcal{B}_r \subseteq \mathbb{R}^{d_X}$ is the ℓ_∞-ball of radius r, i.e.,

$$\mathcal{B}_r = B(0, r) = \{\delta \in \mathbb{R}^{d_X} : \|\delta\|_\infty \leq r\}. \tag{2}$$

The *adversarial robustness* [1] of a model $h_{\boldsymbol{\theta}}$ is measured by the expected loss attained by such adversarial examples, i.e.,

$$\begin{aligned}\tilde{\mathcal{L}}(\boldsymbol{\theta}) &= \mathbb{E}_{XY}\left[\max_{\delta\in\mathcal{B}_r} \ell\left(h_{\boldsymbol{\theta}}(X+\delta), Y\right)\right] \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \max_{\delta\in\mathcal{B}_r} \ell(h_{\boldsymbol{\theta}}(x_i+\delta), y_i).\end{aligned} \tag{3}$$

An adversarially robust neural network $h_{\boldsymbol{\theta}}$ can be obtained via *adversarial training*, i.e., by minimizing the adversarial robustness loss in (3) empirically over the training set $\mathcal{D}$. In practice, this amounts to stochastic gradient descent (SGD) over adversarial examples $x_i+\delta$ (see, e.g., [1]). In each epoch, δ is generated on a per sample basis via an inner optimization over $\mathcal{B}_r$, e.g., via projected gradient descent (PGD).

Adversarial pruning preserves robustness while pruning. Current approaches combine adversarial training into their pruning objective. In particular, AdvPrune [17] directly minimizes adversarial loss $\tilde{\mathcal{L}}(\boldsymbol{\theta})$ constrained by sparsity requirements. HYDRA [19] also uses $\tilde{\mathcal{L}}(\boldsymbol{\theta})$ to jointly learn a sparsity mask along with $\boldsymbol{\theta}_l$. Both are combined with and tailored to specific adversarial training methods, and require considerable training time. This motivates us to propose our PwoA framework, described in Section IV.

B. Knowledge Distillation

In knowledge distillation [24], [25], a student model learns to mimic the output of a teacher. Consider a well-trained teacher model T, and a student model $h_{\boldsymbol{\theta}}$ that we wish to train to match the teacher's output. Let $\sigma : \mathbb{R}^k \to [0,1]^k$ be the softmax function, i.e., $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{j'} e^{z_{j'}}}$, $j = 1, \ldots, k$. Let

$$T^\tau(x) = \sigma\left(\frac{T(x)}{\tau}\right) \text{ and } h_{\boldsymbol{\theta}}^\tau(x) = \sigma\left(\frac{h_{\boldsymbol{\theta}}(x)}{\tau}\right) \tag{4}$$

be the softmax outputs of the two models weighed by temperature parameter $\tau > 0$ [24]. Then, the knowledge distillation penalty used to train $\boldsymbol{\theta}$ is:

$$\mathcal{L}_{\text{KD}}(\boldsymbol{\theta}) = (1-\lambda)\mathcal{L}(\boldsymbol{\theta}) + \lambda\tau^2\mathbb{E}_X[\text{KL}(h_{\boldsymbol{\theta}}^\tau(X), T^\tau(X))], \tag{5}$$

where $\mathcal{L}$ is the classification loss of the tempered student network $h_{\boldsymbol{\theta}}^\tau$ and KL is the Kullback–Leibler (KL) divergence. Intuitively, the knowledge distillation loss $\mathcal{L}_{\text{KD}}$ treats the output of the teacher as *soft labels* to train the student, so that the student exhibits some inherent properties of the teacher, such as adversarial robustness.

[†] `https://github.com/neu-spiral/PwoA/`

C. Hilbert-Schmidt Independence Criterion

The Hilbert-Schmidt Independence Criterion (HSIC) is a statistical dependency measure introduced by Gretton et al. [26]. HSIC is the Hilbert-Schmidt norm of the cross-covariance operator between the distributions in Reproducing Kernel Hilbert Space (RKHS). Similar to Mutual Information (MI), HSIC captures non-linear dependencies between random variables. HSIC is defined as:

$$\begin{aligned}\mathrm{HSIC}(X,Y) &= \mathbb{E}_{XYX'Y'}\left[k_X\left(X,X'\right)k_{Y'}\left(Y,Y'\right)\right]\\ &+\mathbb{E}_{XX'}\left[k_X\left(X,X'\right)\right]\mathbb{E}_{YY'}\left[k_Y\left(Y,Y'\right)\right]\\ &-2\mathbb{E}_{XY}\left[\mathbb{E}_{X'}\left[k_X\left(X,X'\right)\right]\mathbb{E}_{Y'}\left[k_Y\left(Y,Y'\right)\right]\right],\end{aligned} \tag{6}$$

where X' and Y' are independent copies of X and Y respectively, and k_X and k_Y are kernel functions. In practice, we often approximate HSIC empirically. Given n i.i.d. samples $\{(x_i, y_i)\}_{i=1}^n$ drawn from P_{XY}, we estimate HSIC via:

$$\widehat{\mathrm{HSIC}}(X,Y) = (n-1)^{-2}\,\mathrm{tr}\left(K_X H K_Y H\right), \tag{7}$$

where K_X and K_Y are kernel matrices with entries $K_{X_{ij}} = k_X(x_i, x_j)$ and $K_{Y_{ij}} = k_Y(y_i, y_j)$, respectively, and $H = I - \frac{1}{n}\mathbb{1}\mathbb{1}^T$ is a centering matrix.

III. Problem Formulation

Given an adversarially robust model $h_{\boldsymbol{\theta}}$, we wish to efficiently prune non-important weights from this pre-trained model while preserving adversarial robustness of the final pruned model. We minimize the loss function subject to constraints specifying sparsity requirements. More specifically, the weight pruning problem can be formulated as:

$$\begin{aligned}&\underset{\boldsymbol{\theta}}{\text{Minimize:}} \quad \mathcal{L}(\boldsymbol{\theta}),\\ &\text{subject to} \quad \boldsymbol{\theta}_l \in S_l, \quad l = 1, \cdots, L,\end{aligned} \tag{8}$$

where $\mathcal{L}(\boldsymbol{\theta})$ is the loss function optimizing both the accuracy and the robustness, and $S_l \subseteq \mathbb{R}^{d_{\boldsymbol{\theta}_l}}$ is a weight sparsity constraint set applied to layer l, defined as

$$S_l = \{\boldsymbol{\theta}_l \mid \|\boldsymbol{\theta}_l\|_0 \le \alpha_l\}, \tag{9}$$

where $\|\cdot\|_0$ is the size of $\boldsymbol{\theta}_l$'s support (i.e., the number of non-zero elements), and $\alpha_l \in \mathbb{N}$ is a constant specified as sparsity degree parameter.

IV. Methodology

We now describe PwoA, our unified framework for pruning a robust network without additional adversarial training.

A. Robustness-Preserving Pruning

Given an adversarially pre-trained robust model, we aim to preserve its robustness while sparsifying it via weight pruning. In particular, we leverage soft labels generated by the robust model and directly incorporate them into our pruning objective with only access to natural examples. Formally, we denote the well pre-trained model by T and its sparse counterpart by $h_{\boldsymbol{\theta}}$. The optimization objective is defined as follows:

$$\begin{aligned}&\underset{\boldsymbol{\theta}}{\text{Min.:}} \quad \mathcal{L}_{\mathtt{D}}(\boldsymbol{\theta}) = \tau^2\mathbb{E}_X[\mathrm{KL}(h^{\tau}_{\boldsymbol{\theta}}(X), T^{\tau}(X))],\\ &\text{subj. to} \quad \boldsymbol{\theta}_l \in S_l, \quad l = 1, \cdots, L,\end{aligned} \tag{10}$$

where τ is the temperature hyperparameter. Intuitively, our distillation-based objective forces the sparse model $h_{\boldsymbol{\theta}}$ to mimic the soft label produced by the original pre-trained model T, while the constraint enforces that the learnt weights are subject to the desired sparsity. This way, we preserve adversarial robustness via distilling knowledge from soft labels efficiently, without regenerating adversarial examples. Departing from the original distillation loss in (5), we remove the classification loss where labels are used, as we observed that it did not contribute to adversarial robustness (see in extended version [23]). Solving optimization problem (10) is not straightforward; we describe how to deal with the combinatorial nature of the sparsity constraints in Section IV-C.

B. Enhancing Robustness from Natural Examples

In addition to preserving adversarial robustness from the pre-trained model, we can further enhance robustness directly from natural examples. Inspired by the recent work that uses information-bottleneck penalties, [21], [22], [27], [28], we incorporate HSIC as a Regularizer (HBaR) into our robust pruning framework. To the best of our knowledge, HBaR has only been demonstrated effective under usual adversarial learning scenarios; we are the first to extend it to the context of weight pruning. Formally, we denote by $Z_l \in \mathbb{R}^{d_{Z_l}}$, $l \in \{1, \ldots, L\}$ the output of the l-th layer of $h_{\boldsymbol{\theta}}$ under input X (i.e., the l-th latent representation). The HBaR learning penalty [21], [22] is defined as follows:

$$\mathcal{L}_{\mathtt{H}}(\boldsymbol{\theta}) = \lambda_x \sum_{l=1}^{L} \mathrm{HSIC}(X, Z_l) - \lambda_y \sum_{l=1}^{L} \mathrm{HSIC}(Y, Z_l), \tag{11}$$

where $\lambda_x, \lambda_y \in \mathbb{R}_+$ are balancing hyperparameters.

Intuitively, since HSIC measures dependence between two random variables, minimizing $\mathrm{HSIC}(X, Z_l)$ corresponds to removing redundant or noisy information from X. Hence, this term also naturally reduces the influence of adversarial attack, i.e. perturbation added on the input data. Meanwhile, maximizing $\mathrm{HSIC}(Y, Z_l)$ encourages this lack of sensitivity to the input to happen while retaining the discriminative nature of the classifier, captured by the dependence to useful information w.r.t. the output label Y.

PwoA combines HBaR with self-distillation during weight pruning. We formalize PwoA to solve the following problem:

$$\begin{aligned}&\underset{\boldsymbol{\theta}}{\text{Minimize:}} \quad \mathcal{L}_{\mathtt{PwoA}}(\boldsymbol{\theta}) = \lambda\mathcal{L}_{\mathtt{D}}(\boldsymbol{\theta}) + \mathcal{L}_{\mathtt{H}}(\boldsymbol{\theta}),\\ &\text{subject to} \quad \boldsymbol{\theta}_l \in S_l, \quad l = 1, \cdots, L.\end{aligned} \tag{12}$$

C. Solving PwoA via ADMM

Problem (12) has combinatorial constraints due to sparsity. Thus, it cannot be solved using stochastic gradient descent as in the standard CNN training. To deal with this, we follow the ADMM-based pruning strategy by Zhang et al. [13] and Ren et al. [14]. We describe the complete procedure detail in [23]. In short, ADMM is a primal-dual algorithm designed for constrained optimization problems with decoupled objectives (e.g., problem (12)). Through the definition of an augmented Lagrangian, the algorithm alternates between two

Algorithm 1 PwoA Framework

```
Input: input samples $\{(x_i, y_i)\}_{i=1}^n$, a pre-trained robust neural network $T$ with $L$ layers, mini-batch size $m$, sparsity parameter $\alpha$, learning rate $\beta$, proximal parameters $\{\rho_l\}_{l=1}^L$.
Output: parameter of classifier $\theta$
while $\theta$ has not converged do
    Sample a mini-batch of size $m$ from input samples.
    SGD step:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla(\mathcal{L}_{\texttt{PwoA}}(\boldsymbol{\theta}) + \sum_{l=1}^{L} \frac{\rho_l}{2} \|\boldsymbol{\theta}_l - \boldsymbol{\theta}'_l + \boldsymbol{u}_l\|_F^2)$.
    Projection step:
    $\boldsymbol{\theta}'_l \leftarrow \Pi_{S_l}\left(\boldsymbol{\theta}_l + \boldsymbol{u}_l\right)$, for $l = 1, \ldots, L$.
    Dual variable update step:
    $\boldsymbol{u} \leftarrow \boldsymbol{u} + \boldsymbol{\theta} - \boldsymbol{\theta}'$
end
```

TABLE I: Summary of the pre-trained models used for datasets.

Dataset	Architecture	Training Method	Natural	PGD^{20}	CW	AA
MNIST	LeNet	PGD [22]	98.66	96.44	95.10	91.57
CIFAR-10	ResNet-18	TRADES [22]	84.10	52.92	51.00	49.43
	WRN34-10	TRADES [9]	84.96	55.44	53.92	52.34
	WRN34-10	LBGAT [29]	88.24	54.89	54.47	52.61
CIFAR-100	WRN34-10	LBGAT [29]	60.66	34.69	30.78	28.93

primal steps that can be solved efficiently and separately. The first subproblem optimizes objective $\mathcal{L}_{\texttt{PwoA}}$ augmented with a proximal penalty; this is an unconstrained optimization solved by classic SGD. The second subproblem is solved by performing Euclidean projections $\Pi_{S_l}(\cdot)$ to the constraint sets S_l; even though the latter are not convex, these projections can be computed in polynomial time. The overall PwoA framework is summarized in Algorithm 1.

V. Experiments

A. Experimental Setting

We conduct our experiments on three benchmark datasets, MNIST, CIFAR-10, and CIFAR-100. To setup adversarially robust pre-trained models for pruning, we consider five adversarially trained models provided by open-source state-of-the-art work, including Wang et al. [22], Zhang et al. [9], and Cui et al. [29], summarized in Table I.

To understand the impact of each component of PwoA to robustness, we examine combinations of the following non-adversarial learning objectives for pruning: $\mathcal{L}_{\texttt{CE}}$, $\mathcal{L}_{\texttt{H}}$, and $\mathcal{L}_{\texttt{D}}$. All of these objectives are optimized based on natural examples. We also compare PwoA with three adversarially pruning methods: APD [30], AdvPrune [17] and HYDRA [19].

Performance Metrics and Attacks. For all methods, we evaluate the final pruned model via the following metrics. We first measure (a) *Natural* accuracy (i.e., test accuracy over natural examples). We then measure adversarial robustness via test accuracy under (b) *FGSM*, the fast gradient sign attack [31], (c) *PGD*^{m}, the PGD attack with m steps used for the internal PGD optimization [1], (d) CW (CW-loss within the PGD framework) attack [2], and (e) *AA*, AutoAttack [3], which is the strongest among all four attacks. All five metrics are reported in percent (%) accuracy. Following prior adversarial learning literature, we set step size to 0.01 and $r = 0.3$ for MNIST, and step size to $2/255$ and $r = 8/255$ for CIFAR-10 and CIFAR-100, optimizing over ℓ_∞-norm balls in all cases. All attacks happen during the test phase and have full access to model parameters. Since there is always a trade-off between natural accuracy and adversarial robustness, we report the best model when it achieves the lowest average loss among the two, as suggested by Ye et al. [17] and Zhang et al. [9]. We measure and report the overall training time over a Tesla V100 GPU with 32 GB memory and 5120 cores.

B. A Comprehensive Understanding of PwoA

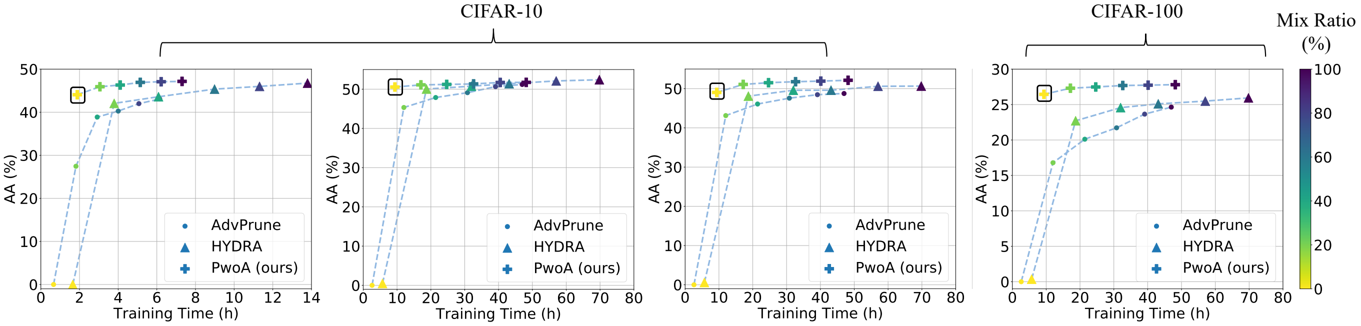
Ablation Study and PwoA Robustness. We first examine the synergy between PwoA terms in the objective in Eq. (12) and show how these terms preserve and even improve robustness while pruning. We studied multiple combinations of $\mathcal{L}_{\texttt{CE}}$, $\mathcal{L}_{\texttt{H}}$, and $\mathcal{L}_{\texttt{D}}$ in Table II. We report the natural test accuracy and adversarial robustness under various attacks of the pruned model under 3 pruning rates (4×, 8×, and 16×) on MNIST, CIFAR-10, and CIFAR-100. For each result reported, we explore hyperparameters λ , λ_x, and λ_y as described in [23] and report here the best performing values.

Overall, Table II suggests that our method PwoA (namely, $\mathcal{L}_{\texttt{D}} + \mathcal{L}_{\texttt{H}}$) prunes a large fraction of weights while attaining the best adversarial robustness for all three datasets. In contrast, a model pruned by $\mathcal{L}_{\texttt{CE}}$ alone (i.e., with no effort to maintain robustness) catastrophically fails under adversarial attacks on all the datasets. The reason is that when the dataset is more complicated and/or pruning rate is high, $\mathcal{L}_{\texttt{CE}}$ is forced to maintain natural accuracy during pruning, making it deviate from the adversarial robustness of the pre-trained model. In contrast, *concurrent self-distillation ($\mathcal{L}_{\texttt{D}}$) and pruning is imperative for preserving substantial robustness without generating adversarial examples during pruning*. We observe this for all three datasets, taking AA under 4× pruning rate for example, from 0.00% by $\mathcal{L}_{\texttt{CE}}$ to 89.28%, 48.26%, and 25.52% by $\mathcal{L}_{\texttt{D}}$ on MNIST, CIFAR-10 and CIFAR-100, respectively.

We also observe that *incorporating $\mathcal{L}_{\texttt{H}}$ while pruning is beneficial for maintaining high accuracy while improving adversarial robustness against various attacks*. By regularizing $\mathcal{L}_{\texttt{CE}}$ with $\mathcal{L}_{\texttt{H}}$, we observe a sharp adversarial robustness advantage on MNIST, taking AA for example from 0.00% by $\mathcal{L}_{\texttt{CE}}$ to 47.49%, 40.71%, and 13.04% by incorporating $\mathcal{L}_{\texttt{H}}$ under 4×, 8×, and 16× pruning rate, respectively; by regularizing $\mathcal{L}_{\texttt{D}}$ with $\mathcal{L}_{\texttt{H}}$, we again see that the regularization improves adversarial robustness on all the cases, especially w.r.t. the strongest attack (AA). We note that the robustness improvement of incorporating $\mathcal{L}_{\texttt{H}}$ with $\mathcal{L}_{\texttt{D}}$ is not caused by a trade-off between accuracy and robustness: in fact, $\mathcal{L}_{\texttt{D}} + \mathcal{L}_{\texttt{H}}$ *consistently improves both natural accuracy and robustness under all pruning rates on all datasets.* Motivated by the above observations, we further analyze how the two terms in HBaR defined in Eq. (11) affect natural accuracy and robustness; these can be found in the extended version [23].

TABLE II: **Prune LeNet (PGD), WRN34-10 (LBGAT), and WRN34-10 (LBGAT) on MNIST, CIFAR-10, and CIFAR-100, respectively.** For all the non-adversarial learning objectives, we report natural test accuracy (in %) and adversarial robustness (in %) on FGSM, PGD, CW, and AA attacked test examples under different pruning rates.

PR	$\mathcal{L}_{\texttt{CE}}$	$\mathcal{L}_{\texttt{D}}$	$\mathcal{L}_{\texttt{H}}$	MNIST LeNet (PGD)						CIFAR-10 WRN34-10 (LBGAT)						CIFAR-100 WRN34-10 (LBGAT)					
				Natural	FGSM	PGD^{10}	PGD^{20}	CW	AA	Natural	FGSM	PGD^{10}	PGD^{20}	CW	AA	Natural	FGSM	PGD^{10}	PGD^{20}	CW	AA
4×	✓			**99.18**	35.73	0.07	0.00	0.00	0.00	93.59	48.47	2.47	0.74	0.21	0.00	71.55	20.92	7.21	5.64	3.93	0.00
	✓		✓	98.54	91.86	89.78	78.32	79.16	47.49	**93.68**	46.52	8.45	1.69	0.25	0.00	**71.83**	23.45	7.57	5.95	4.07	0.00
		✓		98.67	95.42	97.08	95.61	95.19	89.28	88.69	62.72	52.86	50.96	50.29	48.26	60.91	36.21	32.69	31.87	27.74	25.52
		✓	✓	98.66	**95.89**	**97.35**	**96.16**	**96.15**	**90.00**	88.51	**63.44**	**53.54**	**51.51**	**50.89**	**49.03**	60.92	**36.70**	**33.08**	**32.59**	**28.40**	**26.44**
8×	✓			**99.18**	39.08	0.04	0.00	0.00	0.00	93.27	41.15	0.58	0.33	0.00	0.00	71.34	15.28	3.52	2.65	1.37	0.00
	✓		✓	98.63	88.70	88.89	70.67	71.42	40.71	**93.81**	40.08	2.95	1.04	0.28	0.00	**71.56**	17.32	3.73	2.65	1.60	0.00
		✓		98.66	**94.15**	96.94	95.98	94.74	86.48	88.40	61.93	50.76	48.13	48.07	44.87	61.10	35.27	30.46	29.65	25.52	23.34
		✓	✓	98.66	95.69	**97.13**	**95.61**	**95.60**	**87.37**	88.66	**62.64**	**51.41**	**48.98**	**48.81**	**46.09**	61.44	**35.61**	**31.19**	**30.45**	**26.32**	**24.20**
16×	✓			**98.96**	79.09	0.06	0.00	0.00	0.00	92.87	20.95	0.00	0.00	0.00	0.00	69.89	14.56	3.04	2.46	1.68	0.00
	✓		✓	98.70	81.24	83.70	50.82	54.31	13.04	**93.14**	29.88	0.84	0.11	0.04	0.00	**70.54**	16.88	3.56	2.72	1.62	0.00
		✓		98.33	94.51	95.89	93.15	93.14	76.00	88.30	60.77	48.80	46.32	45.76	42.01	62.34	34.65	28.48	27.19	23.30	20.11
		✓	✓	98.59	**95.03**	**96.34**	**94.43**	**94.48**	**77.21**	88.51	**61.52**	**49.68**	**47.19**	**47.01**	**43.33**	62.53	**35.15**	**29.05**	**27.88**	**24.08**	**21.43**



(a) ResNet-18 (TRADES) (b) WRN34-10 (TRADES) (c) WRN34-10 (LBGAT) (d) WRN34-10 (LBGAT)

Fig. 2: Robustness comparison with AdvPrune and HYDRA across different pre-trained models and datasets, under a varying *mix ratio*, i.e., fraction (in %) of natural examples replaced by adversarial examples during training. We plot AA robustness v.s. training time as we modify the mix ratio; boxes □ indicate PwoA with 0% mix ratio (no adversarial examples). We observe that, competitors are not robust without access to adversarial examples; to achieve PwoA's robustness at 0% mix ratio, AdvPrune and HYDRA require 4×–7× more training time. On CIFAR-100, they never meet the performance attained by PwoA. We also observe that PwoA improves by partial access to adversarial examples; overall, it attains a much more favorable trade-off between robustness and training efficiency than the two competitors. In fact, in all cases except (b), PwoA consistently outperforms competitors at 100% mix ratio, w.r.t. *both* robustness and training time.

C. *Comparison to Adversarial Pruning (AP) Methods*

Robustness with Partial Access to Adversarial Examples. We compare PwoA with two state-of-the art AP baselines, i.e., AdvPrune and HYDRA, in terms of adversarial robustness and training efficiency on the CIFAR-10 and CIFAR-100 datasets. Both AdvPrune and HYDRA require access to adversarial examples. To make a fair comparison, we generate adversarial examples progressively for all methods, including PwoA: in Figure 2, we change the *mix ratio*, i.e., the fraction of total natural examples replaced by adversarial examples generated by PGD^{10}. We plot AA robustness vs. training time, under a 4× pruning rate. We observe that, without access to adversarial examples (mix ratio 0%), both competing methods fail catastrofically, exhibiting no robustness whatsoever. Moreover, to achieve the same robustness as PwoA, they require between **4× and 7×** more training time; on CIFAR-100, they actually never meet the performance attained by PwoA. We also observe that PwoA improves by partial access to adversarial examples; overall, it attains a much more favorable trade-off between robustness and training efficiency than the two competitors. Interestingly, with the exception of the case shown in Figure 2(b) (WRN34-10 over CIFAR-10), PwoA consistently outperforms competitors at 100% mix ratio, w.r.t. *both* robustness and training time.

Impact of Pre-training Method. We also observe that HYDRA performs well when pruning models pre-trained with TRADES, but gets worse when dealing with model pre-trained with LBGAT. This is because HYDRA prunes the model using TRADES as adversarial loss, and is thus tailored to such pre-training. When models are pre-trained via LBGAT, this change of loss hampers performance. In contrast, PwoA can successfully prune an arbitrary pre-trained model, irrespective of the architecture or pre-training method.

Pruning Rate Impact. We further measure the performance of our PwoA and SOTA methods against all five attacks under 4×, 8×, and 16× pruning rate. We report these at 20% mix ratio, so that training times are roughly equal across methods, in Table III for CIFAR-100. *Overall, we can clearly see that PwoA consistently outperforms other SOTA methods*

TABLE III: **Prune WRN34-10 (LBGAT) on CIFAR-100**: Comparison of PwoA with SOTA methods w.r.t various attacks and training time (TT, in *h*) under different pruning rates at 20% mix ratio.

PR	Methods	Natural	FGSM	PGD^{10}	PGD^{20}	CW	AA	TT
4×	AdvPrune	**68.39**	40.77	24.71	22.42	21.45	14.95	12.14
	HYDRA	60.61	29.54	25.88	25.21	24.22	22.81	18.69
	PwoA (ours)	60.93	**36.92**	**33.62**	**33.30**	**29.10**	**27.31**	17.03
8×	AdvPrune	**68.33**	40.73	24.34	22.03	20.97	12.73	12.31
	HYDRA	61.04	29.90	25.55	25.04	24.11	22.36	18.73
	PwoA (ours)	61.58	**36.39**	**33.09**	**32.50**	**28.29**	**26.46**	17.05
16×	AdvPrune	**68.24**	38.98	23.20	20.50	19.13	8.40	12.08
	HYDRA	61.35	29.14	25.53	24.85	23.92	21.95	18.77
	PwoA (ours)	61.84	**35.78**	**32.24**	**31.34**	**27.31**	**25.28**	17.09

against all five attacks, under similar (or lower) training time. Specifically, PwoA maintains high robustness against AA with only 1.62% drop (under 4× PR) from the pre-trained model by LBGAT (see Table I), while the AA robustness achieved by HYDRA and AdvPrune drop by 6.12% and 13.98%, respectively. This again verifies that, when pruning a robust model pre-trained with different adversarial training methods, PwoA is more stable in preserving robustness. Improvements are also pronounced while increasing pruning rate: PwoA outperforms HYDRA against AA by 4.50%, 4.10%, and 3.33% under 4×, 8×, and 16× pruning rates, respectively. For completeness, we also report performance at 0% mix ratio on CIFAR-100 in [23]; in contrast to PwoA, competitors exhibit virtually negligible robustness in this case.

VI. Conclusions and Future Work

We proposed PwoA, a unified framework for pruning adversarially robust networks without adversarial examples. Our method leverages pre-trained adversarially robust models, preserves adversarial robustness via self-distillation and enhances it via the Hilbert-Schmidt independence criterion as a regularizer. Comprehensive experiments on MNIST, CIFAR-10, and CIFAR-100 datasets demonstrate that PwoA prunes a large fraction of weights while attaining comparable adversarial robustness with up to 7× training speed up. Future directions include extending PwoA framework to structured pruning and weight quantization. Another interesting future direction is to use distillation and novel penalties to prune a pre-trained robust model even without access to natural examples.

VII. Acknowledgements

The authors gratefully acknowledge support by the National Science Foundation under grants CCF-1937500 and CNS-2112471.

References

[1] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[2] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.

[3] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *ICML*, vol. 119, 2020, pp. 2206–2216.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[5] A. Chernikova, A. Oprea, C. Nita-Rotaru, and B. Kim, "Are self-driving cars secure? Evasion attacks against deep neural networks for steering angle prediction," in *IEEE Symposium on Security and Privacy Workshops*, 2019, pp. 132–137.

[6] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[7] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *CVPR Workshops*, 2019.

[8] Z. Yan, Y. Guo, and C. Zhang, "Deep defense: Training DNNs with improved adversarial robustness," in *NeurIPS*, 2018.

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

[10] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *ICLR*, 2019.

[11] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," *CVPR*, 2019.

[12] 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!" in *NeurIPS*, vol. 32, 2019.

[13] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *ECCV*, 2018, pp. 184–199.

[14] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "Admm-nn: An algorithm-hardware co-design framework of DNNs using alternating direction methods of multipliers," in *ASPLOS*, 2019.

[15] T. Jian, Y. Gong, Z. Zhan, R. Shi, N. Soltani, Z. Wang, J. Dy, K. R. Chowdhury, Y. Wang, and S. Ioannidis, "Radio frequency fingerprinting on the edge," *IEEE Transactions on Mobile Computing*, 2021.

[16] Z. Wang, Z. Zhan, Y. Gong, G. Yuan, W. Niu, T. Jian, B. Ren, S. Ioannidis, Y. Wang, and J. Dy, "SparCL: Sparse continual learning on the edge," *NeurIPS*, 2022.

[17] 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 *ICCV*, 2019.

[18] S. Gui, H. N. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," in *NeurIPS*, vol. 32, 2019.

[19] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "HYDRA: Pruning adversarially robust neural networks," in *NeurIPS*, vol. 33, 2020.

[20] M. Goldblum, L. Fowl, S. Feizi, and T. Goldstein, "Adversarially robust distillation," in *AAAI*, vol. 34, no. 04, 2020, pp. 3996–4003.

[21] W.-D. K. Ma, J. Lewis, and W. B. Kleijn, "The HSIC bottleneck: Deep learning without back-propagation," in *AAAI*, 2020, pp. 5085–5092.

[22] Z. Wang, T. Jian, A. Masoomi, S. Ioannidis, and J. Dy, "Revisiting Hilbert-Schmidt information bottleneck for adversarial robustness," in *NeurIPS*, 2021.

[23] T. Jian, Z. Wang, Y. Wang, J. Dy, and S. Ioannidis, "Pruning adversarially robust neural networks without adversarial examples (extended version)," *arXiv preprint arXiv:2210.04311*, 2022.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[25] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, pp. 1–31, 2021.

[26] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *International conference on algorithmic learning theory*, 2005, pp. 63–77.

[27] I. Fischer, "The conditional entropy bottleneck," *Entropy*, vol. 22, no. 9, p. 999, 2020.

[28] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *ICLR*, 2017.

[29] J. Cui, S. Liu, L. Wang, and J. Jia, "Learnable boundary guided adversarial training," in *ICCV*, 2021.

[30] J. Lee and S. Lee, "Robust CNN compression framework for security-sensitive embedded systems," *Applied Sciences*, vol. 11, no. 3, 2021.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.