Adversarial Robust Model Compression using In-Train Pruning

Manoj-Rohit Vemparala[1], Nael Fasfous[2], Alexander Frickenstein[1], Sreetama Sarkar[1], Qi Zhao[3], Sabine Kuhn[1], Lukas Frickenstein[1], Anmol Singh[1] Christian Unger[1], Naveen-Shankar Nagaraja[1], Christian Wressnegger[3], Walter Stechele[2]

[1]*BMW Autonomous Driving*, [2]*Technical University of Munich*, [3]*Karlsruhe Institute of Technology*

Abstract

Efficiently deploying learning-based systems on embedded hardware is challenging for various reasons, two of which are considered in this paper: The model's size and its robustness against attacks. Both need to be addressed even-handedly. We combine adversarial training and model pruning in a joint formulation of the fundamental learning objective during training. Unlike existing post-train *pruning approaches, our method does not use heuristics and eliminates the need for a pre-trained model. This allows for a classifier which is robust against attacks and enables better compression of the model, reducing its computational effort. In comparison to prior work, our approach yields* $6.21\,\text{pp}$ *higher accuracy for an* $85\,\%$ *reduction in parameters for ResNet20 on the CIFAR-10 dataset.*

1. Introduction

While convolutional neural networks (CNNs) have been proven effective in various computer vision applications, such as image classification [1], semantic segmentation [2], and object detection [3], their deployment on resource-limited hardware such as in-vehicle systems remains challenging. In real-world applications, the high memory requirements and energy consumption of neural networks can be a limiting factor. Recent works on CNN optimization are categorized into methods for parameter pruning [4–6], quantization [7, 8], and knowledge distillation [9]. These methods contribute to significant improvements in reducing the computational complexity of neural networks. Moreover, neural networks are vulnerable to attacks, questioning their suitability for safety-critical applications such as autonomous driving. Adversarial examples, for instance, are small perturbations to the input that appear insignificant or even imperceptible to the human eye, but cause a different/incorrect prediction by the classifier [10]. As a remedy, the research community has invested significant efforts to learn more robust models, for instance using adversarial training [11, 12], where such adversarial examples are actively incorporated in the learning process. However, only few studies analyze the impact of adversarial robustness *and* network compression [13–16]

Techniques for pruning neural networks aim to remove redundant structural parameters like channels, kernels or individual weight elements of neural networks in order to decrease the memory requirements and accelerate the computation of the network at hand, while maintaining the network's accuracy. Most pruning methods [*e.g.* 4–6] follow a three step approach: First, a model is learned to solve a task at hand. Second, this very model is pruned according to a separate objective function. Third, the model is fine-tuned to maintain the overall accuracy.

Pruning often relies on magnitude-based heuristics that require incorporating iterative fine-tuning during the pruning search to maintain the model's effectiveness [6, 17]. This, however, significantly increases the computation effort (the GPU hours) for the pruning process. To improve upon this, recent research proposes reinforcement learning (RL) agents to automate the process of finding the optimal model pruning strategies [5, 18]. While, these learning-based compression techniques outperform pure heuristic-based approaches both in efficiency and compression ratio, they often do not yield an optimal solution.

In this paper, we propose to incorporate the pruning process, that is, learning an appropriate pruning mask, in the underlying optimization function of the training. We thereby break through the barrier between training and pruning, and circumvent the need for magnitude-based heuristics. In an extensive evaluation, we demonstrate that our method yields $80\,\%$ reduction of multiply and accumulate (MAC) operations in a ResNet56 network with minimal degradation in accuracy. Our joint formulation of the learning and pruning objectives allows us to additionally incorporate recent advantages from adversarial training [12] and

increase the robustness of the pruned network. We provide a trade-off between task-specific accuracy, adversarial accuracy and compression rate. With this, we achieve higher adversarial accuracy than RL-based approaches and also outperform the state of the art robust pruning methods.
We summarize our contributions as follows:

1. **Higher Compression.** Our method identifies redundant weights by minimizing a hardware-aware auxiliary loss when updating the network's connections. We obtain $80\,\%$ reduction of operations with only 2.22 pp (percent point) degradation in accuracy for a ResNet56-based channel-pruned configuration on CIFAR-10.

2. **Sustained Robustness.** We show the effectiveness of our in-train pruning scheme under attack. By augmenting the trainable pruning masks with adversarial training, our method produces 5.91 pp and 8.65 pp higher natural accuracy with similar adversarial robustness compared to post-training RL pruning at a $70\,\%$ reduction of operations for ResNet20 and ResNet56.

3. **Improved Accuracy.** We compare our approach to state-of-the-art robust pruning methods. We achieve a 6.21 pp higher natural accuracy than Robust-ADMM [13] while maintaining a similar level of adversarial robustness for $85\,\%$ channel pruning.

The remainder of the paper is structured as follows: In Section 2, we discuss related work on post-training and in-train pruning, as well as adversarial robustness. Section 3 introduces our approach on *in-train* adversarial robust model compression, which is then extensively evaluated in Section 4. Section 5 concludes the paper.

2. Related Work

Post-Train Pruning Han et al. [6] determined the importance of individual elements in the weight matrix based on their magnitude, demonstrating the redundancies in deep neural networks. Pruning individual weights, referred to as irregular pruning, leads to inefficient memory accesses, making it impractical for general-purpose computing platforms. Regularity in pruning becomes an important criterion towards accelerator-aware optimization [17]. He et al. [4] prune redundant channels by applying LASSO regression and solving for least square minimization of the output error of the remaining feature maps. However, the above pruning methods are based on heuristics, which cannot be guaranteed to generalize well for different tasks and objectives. Recently, auto-machine-learning (Auto-ML) based approaches are leveraged in the research of model pruning techniques [5, 18]. Huang et al. [18] proposed a reinforcement learning based filter pruning framework to achieve layer-wise filter pruning. An RL-agent prunes a single layer at a time before fine-tuning and then moving on to the next layer. This leads to longer GPU hours and provides no guarantee of a global optimal pruning strategy for the entire model. In AMC [5], a deep deterministic policy gradient (DDPG) based RL-agent is utilized in regular filter pruning. The RL-agent provides the environment with a continuous action that can be defined as the compression ratio of each layer. Based on the magnitude obtained from the L2-norm heuristic and the sparsity ratio of each layer given by the RL-agent, the redundant channels are pruned. In this work, we eliminate the need for heuristics and search-time for pruning and determine the pruning strategy based on gradient updates *during* the training process.

In-Train Pruning Integrating the pruning process into the training phase to jointly optimize the weights and prune connections is referred to as in-train pruning. The autoencoder-based low-rank filter-sharing technique (ALF) proposed by Frickenstein et al. [19] utilizes sparse autoencoders that extract the most salient features of convolutional layers, discarding filters in an unsupervised manner. ALF is limited to filter pruning and does not support other pruning regularities. ALF also adds an additional expansion layer which prohibits the extraction of inter-layer filter pruning benefits. Zhang et al. [20] present a systematic weight pruning framework for neural networks, where pruning is formulated as a constrained non-convex optimization problem and solved using alternating direction method of multipliers (ADMM) [21] during the training process. The authors subsequently extend their work in StructADMM [22] to structured sparsity and provide analysis on row pruning, column pruning and filter pruning. Although the task-specific and pruning objectives are solved simultaneously, the authors use predefined sparsity ratios from other pruning works to ensure the convergence. Sparse learning or training sparse networks from scratch [23–25] can also be considered as an in-train pruning technique, which has achieved extremely high pruning rates with negligible accuracy degradation. This method does not require a pre-trained dense model and the network topology is updated during training through pruning and regrowing connections. Parameters are pruned based on magnitude and grown back at random [23] or based on gradient [25] or momentum [24] information. However, these methods often require predefined layer-wise sparsities and are mostly effective in reducing model size through weight pruning, rather than focusing on the hardware advantages through structured pruning.

Model Compression and Adversarial Robustness Galloway et al. [26] evaluated and interpreted the adversarial robustness of binary neural networks (BNNs). They highlight the most commonly mentioned benefits of BNNs,

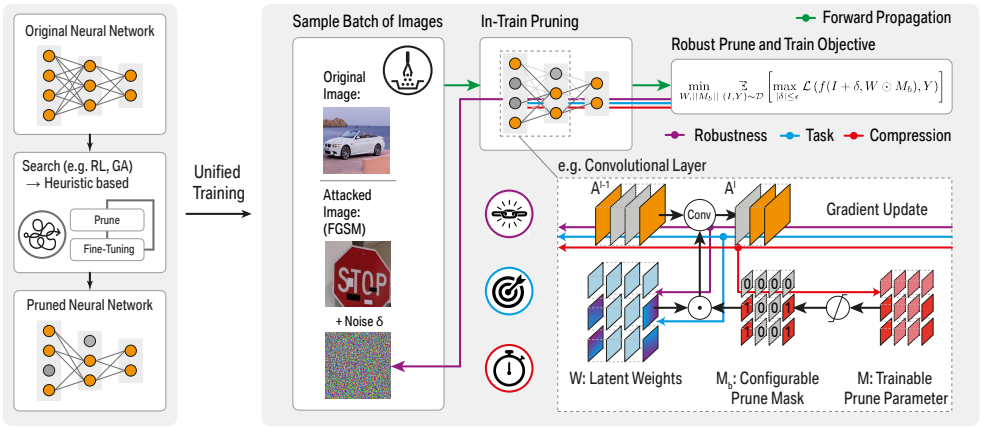

Figure 1: Depiction of existing post-train pruning approaches (left) in comparison to our newly proposed method (right).

i.e. the reduced memory consumption and the faster inference. They also point out the improvement in robustness against adversarial attacks for BNNs compared to full-precision models. The inherently discontinuous and approximated gradients of BNNs gives them an advantage over full-precision networks for adversarial attacks. In this work, we focus on model pruning as the main compression technique, orthogonal to quantization and binarization approaches. Inspired by the work of [20], Ye et al. [13] incorporated adversarial robustness into the ADMM objective. One of the main findings of the work for improving robustness of a compact model is to concurrently prune and adversarially train an over-parameterized network. A similar approach followed by Gui et al [16] constructed framework to realize a unified constrained robust-aware optimization on DNN models. The objective function of ADMM is reconstructed to improve adversarial robustness using three main compression strategies: pruning, factorization and quantization. However, a pre-trained model with adversarial robustness is required before the compression. Differently, our approach minimizes the associated loss term to obtain sparsity by updating the differentiable prune masks without relying on a magnitude-based heuristic. Sehwag et al. [14] recently proposed a new method to make pruning techniques aware of robust training objectives. After pre-training an over-parameterized model, the pruning is based on the importance score computed from the loss. After determining the pruned subgraph, fine-tuning is performed. As noted in their results, the lowest magnitude based pruning is not suitable for robustness-aware pruning technique. Our approach does not require a pre-trained model as we prune and train the compact robust model simultaneously.

3. Adversarial Robust Model Compression

In this paper, we target two objectives: 1) Compressing a model to reduce the computational effort of a neural network, and 2) increasing the robustness against an adversary manipulating input data. Both can be effectively achieved by formulating a joint optimization problem as shown in Fig 1.

We adopt the concept of adversarial training [11] but incorporate pruning edges in the network using a binary mask $M_b \in \{0, 1\}$ derived from a trainable continuous mask M, that is, weights W are canceled out if the corresponding dimension of the mask is 0 and left unchanged if it is set to 1: $W \odot M_b$ (*c.f.* Sections 3.1 and 3.2). Attacks against a neural networks are described as finding a minimal perturbation δ of an image I (forming the the adversarial example $I_{\text{adv}} = I + \delta$) that changes the outcome of a given model represented by the prediction function $f(\cdot)$ [27]. For adversarial training, we make use of this generation principle, while maximizing the loss $\mathcal{L}$ for a given perturbation budget ϵ:

$$\min_{W,M} \mathop{\mathbb{E}}_{(I,Y)\sim\mathcal{D}} \left[\max_{|\delta|\leq\epsilon} \mathcal{L}\left(f(I+\delta, W \odot M_b), Y\right) \right]. \quad (1)$$

The outer minimization problem in Eq. (1) involves a set of randomly sampled images from dataset $\mathcal{D}$, where the expected loss $\mathbb{E}$ on the random samples is minimized through an adversarial training scheme, such as Fast Adversarial Training (FastAT) [12].

Exposing a model to adversarial images I_{adv} results in the adversarial accuracy Acc_{adv}, representing the measure of adversarial robustness of the underlying model. Our in-train pruning approach aims to achieve a balanced trade-off between natural accuracy Acc_{nat} (calculated from the ground-truth labels Y for the corresponding images I), adversarial robustness Acc_{adv}, and model complexity $\text{sum}(M_b)$ during the training process, rather than introducing separate (post-training) phases for pruning and fine-tuning.

In principle, one may use different methods for generating adversarial examples for training, such as Fast Gra-

dient Sign Method (FGSM) [27], Projected Gradient Descent (PGD) [11] and Carlini-Wagner (C&W) [28]. Wong et al. [12], however, show that using FGSM in combination with random initialization is particularly effective. With this, the cost of training, measured in GPU hours, with one iteration of FGSM is significantly lower than with other variants like PGD-based adversarial training [11]. We integrate the in-train update operations of the pruning mask M_b in the FastAT procedure as shown in Algorithm 1.

Algorithm 1: Joint selection of pruning masks and adversarial training.

Require: Training samples $\mathcal{D}$, perturbation strength ϵ, step size α

1 Initialize θ, $M_b \leftarrow 1$
2 **for** $Epoch = 1, \dots N_{epochs}$ **do**
3 **for** *Batch* $B \subset \mathcal{D}$ **do**
4 Initialize perturbation $\delta \leftarrow random\_uniform(-\epsilon, +\epsilon)$
5 Sample a batch of K examples $\{(I^{(1)}, Y^{(1)}), \cdots, (I^{(K)}, Y^{(K)})\}$ from data distribution.
6 Use FGSM attack to generate perturbations on batch K to update δ
7 $\delta \leftarrow \delta + \alpha \cdot sign(\nabla_\delta \mathcal{L}(f(I+\delta, W \odot M_b), Y)$
8 $\delta \leftarrow \max(\min(\delta, \epsilon), -\epsilon)$
9 $I_{\text{adv}} \leftarrow I + \delta$
10 Update weights W and pruning masks M using SGD for adversarial images:
11 $W \leftarrow W - \eta \cdot \nabla_W \mathcal{L}(f(I_{\text{adv}}, W \odot M_b), Y)$
12 $M \leftarrow M - \eta \cdot \nabla_M \mathcal{L}(f(I_{\text{adv}}, W \odot M_b), Y)$
13 **end**
14 **if** $E_{\textit{Prune, Start}} \leq Epoch \leq E_{\textit{Prune, End}}$ **then**
15 **if** $Epoch \bmod Prune\_step = 0$ **then**
16 $M \leftarrow M - \eta \cdot \nabla_M \mathcal{L}_{\text{Prune}}(M, \psi_{\text{base}})$
17 **end**
18 $M_b \leftarrow \text{round}(0.5 \cdot \tanh(\text{M}) + 0.5)$
19 **end**
20 **end**

During each training step, we generate a uniform random initialization for the adversarial perturbation as shown in line 4, followed by performing a step into the ascent gradient direction (line 7) is scaled by the step size α. We update the weights and pruning masks of the neural network jointly in lines 11 and 12 for clean and adversarial images with learning rate η. During these update steps the importance scores for masks M_b get accumulated. Lines 15 and 16 zero out prune masks based on a hardware loss $\mathcal{L}_{\text{Prune}}$ (*c.f.* Section 3.2). As shown in line 14, we start and freeze the optimization of prune masks at the epoch corresponding to $E_{\text{Prune, Start}}$ and $E_{\text{Prune, End}}$ respectively. As part of our experiments in Section 4, we discuss the training behavior for different pruning constraints with and without the adversarial setting in more detail.

3.1. Pruning

We aim for obtaining a pruning strategy directly when optimizing the network's weights W during the training process and thus save the effort of additional post-train pruning. We use binary pruning masks M_b to derive pruned weights as $\tilde{W}$=$W \odot M_b$. At each layer $l \in \{1, ..., N\}$ of an N-layer CNN, we append a binary pruning mask M_b^l to the network's weights W^l. All but the last layer have an input shape $L^{l-1} \in \mathbb{R}^{A_{in} \times B_{in} \times C_{in}}$, where A_{in}, B_{in}, and C_{in} indicate the spatial height, width, and input channels, respectively. L^0 represents the input image I and L^N the classification output of the CNN. The weights $W \in \mathbb{R}^{K_h \times K_w \times C_{in} \times C_o}$ are the trainable parameters of the individual layers, where K_h, K_w, and C_o refer to the kernel's dimensions, and the number of output channels/filters, respectively.

The binary masks for irregular weight pruning are structured as $M_b^l = \{0,1\}^{K_h \times K_w \times C_{in} \times C_o}$, kernel pruning requires masks as $M_b^l = \{0,1\}^{1 \times 1 \times C_{in} \times C_o}$ and channel pruning requires masks $M_b^l = \{0,1\}^{1 \times 1 \times C_{in} \times 1}$. The size of the binary mask increases as the pruning tends to become more irregular leading to higher compression rates. However, irregular and kernel pruning demands dedicated hardware implementation [29] for load balancing and additional memory for mask indices, resulting in sub-optimal benefits on general-purpose platforms. The masked weights are obtained using the the Hadamard product $\odot$ along the pruning dimension as $\tilde{W}^l$ as shown in Eq. (2).

Our training scheme intends to influence M_b using cross-entropy and hardware (HW) objectives (*c.f.* Section 3.2), by updating the continuous-valued, and thus trainable, mask M with the same shape as M_b. We use $\tanh$, scale, and shift operations to normalize the value range of masks M_{norm} to $[0, 1]$ as shown in Eq. (2). We apply the round operation and restrict the mask values to the binary set $\{0, 1\}$.

$$\begin{aligned} \tilde{W}^l &= W^l \odot M_b, \ \tilde{W}^l \in \mathbb{R}^{K_h \times K_w \times C_{in} \times C_o} \\ M_b &= \text{round}(M_{\text{norm}}) \\ M_{\text{norm}} &= 0.5 \cdot \tanh(\text{M}) + 0.5 \end{aligned} \tag{2}$$

Any discrete parameter with a limited range set would introduce zero gradients. We use Straight-Through Estimators (STE) similar to [30] to overcome the vanishing gradient effect and obtain updates for continuous masks M, later discretized to M_b for applying pruning decisions on the weights.

3.2. Loss Formulation

We define the loss function that allows us to account for hardware-specific compression objectives. The inference complexity of the CNN depends on the number of non-zero values in the binary pruning masks $\text{sum}(M_b^l)$ at every layer l. We represent the shape of layer l after the masks are applied as l_{shape} and hardware inference complexity as a function of l_{shape} and M_b^l given as $\psi_l(l_{\text{shape}}, M_b^l)$. Increasing the number of zeros in the prune masks leads to a lower number of computations and parameters. However, this impacts accuracy Acc_{nat} for extreme compression rates.

The latent weights W are optimized to improve the task-specific accuracy and adversarial robustness with respect to the sum of the cross-entropy loss $\mathcal{L}_{ce}$ and regularization loss $\mathcal{L}_{reg}$. The trainable prune masks M are also considered in the regularization loss to avoid too many binary masks M_b elements biased at the early stages due to exploding magnitude. We provide more details about the regularization $\mathcal{L}_{reg}$ in supplementary material S.1. We optimize the trainable masks M based on an auxiliary loss term $\mathcal{L}_{HW}$, which captures hardware HW benefits. It is important to select pruning masks which not only produce HW benefits but also allow smooth minimization of cross-entropy loss during the training process. Therefore, we formulate prune loss $\mathcal{L}^i_{\text{prune}}$ at step i in Eq. (3), which is an accumulation of $\mathcal{L}_{ce}$ and $\mathcal{L}_{HW}$. The HW loss $\mathcal{L}_{HW}$ is the difference between the relative complexity of neural networks at iteration i and a target constraint ψ^*. We accumulate the complexity of all the N layers to obtain the complexity of neural network.

$$\begin{aligned} \mathcal{L}^i_{\text{Prune}} &= \mathcal{L}^i_{ce} + b \times \mathcal{L}^i_{HW} \\ \mathcal{L}^i_{HW} &= max(\frac{\sum_{l=1}^{N}(\psi_l^i)}{\sum_{l=1}^{N}(\psi_l^0)} - \psi^*, 0) \end{aligned} \tag{3}$$

We use the scaling factor b to control the convergence speed for the prune masks M during the training process. For extreme constraints such as 70% HW reductions, we use higher b=50. The complexity of the neural network can be represented using the number of parameters or MAC operations. In Eq. (4), we represent the complexity by also incorporating the binary prune masks M_b. We first calculate the compression ratio μ_l for every layer l based on the number of non zeros present in the weight matrix. For this purpose, we introduce M^l_{base} having the same dimensionality as M^l_{b}, consisting of all ones, representing the unpruned model. We observe that the number of zeros in the binary prune masks directly affect the complexity of layer l, which can be represented using either parameters $\psi_l(\text{params})$ or operations $\psi_l(\text{ops})$.

$$\begin{aligned} \mu_l &= ||M_b^l||/||M^l_{\text{base}}|| \\ \psi_l(\text{params}) &= K_w^l \times K_h^l \times C_{in}^l \times C_o^l \times \mu^l \\ \psi_l(\text{ops}) &= A_o^l \times B_o^l \times K_w^l \times K_h^l \times C_{in}^l \times C_o^l \times \mu^l \end{aligned} \tag{4}$$

Eq. (4) can be extended to pruning regularities such as channel/filter pruning, where inter-layer HW benefits must be taken into consideration. For channel pruning, we capture the inter-layer benefits by incorporating μ_l and μ_{l+1}, thereby reducing C_{in}^l and C_o^l respectively. We use an optimizer similar to that of adversarial training, such as Momentum/ADAM, to update the prune masks. As shown in Eq. (5) and Eq. (6), we approximate the gradients gm_{ce} and gm_{HW} derived from $\mathcal{L}_{ce}$ and $\mathcal{L}_{HW}$ to update the continuous prune mask M, incorporating STE as mentioned in Section 3.1.

$$\begin{aligned} gm^l_{ce} = \frac{\partial \mathcal{L}_{ce}}{\partial M^l} &= \frac{\partial \mathcal{L}_{ce}}{\partial \tilde{W}} \cdot \frac{\partial \tilde{W}}{\partial M_b^l} \cdot \frac{\partial M_b^l}{\partial M^l_{\text{norm}}} \cdot \frac{\partial M^l_{\text{norm}}}{\partial M^l} \\ &\overset{\text{STE!}}{=} \frac{\partial \mathcal{L}_{ce}}{\partial \tilde{W}} \cdot \frac{\partial \tilde{W}}{\partial M_b^l} \cdot \frac{\partial M^l_{\text{norm}}}{\partial M^l} \end{aligned} \tag{5}$$

As shown in Eq. (6), the gradients updating prune masks due to $\mathcal{L}_{HW}$ scales depending on the baseline complexity ψ^l_{base} of the layer l. We derive ψ^l_{base} by setting μ_l = 1. We discuss the influence of various hyper-parameters on the pruning efficiency in supplementary material S.1.

$$\begin{aligned} gm^l_{HW} = \frac{\partial \mathcal{L}_{HW}}{\partial M^l} &= \frac{\partial \psi^l}{\partial M^l} = \frac{\partial \psi^l}{\partial M_b^l} \cdot \frac{\partial M_b^l}{\partial M^l_{\text{norm}}} \cdot \frac{\partial M^l_{\text{norm}}}{\partial M^l} \\ &\overset{\text{STE!}}{=} \frac{\partial \psi^l}{\partial M_b^l} \cdot \frac{\partial M^l_{\text{norm}}}{\partial M^l} = \frac{\psi^l_{\text{base}}}{||M^l_{\text{base}}||} \cdot \frac{\partial M^l_{\text{norm}}}{\partial M^l} \end{aligned} \tag{6}$$

4. Experiments

We evaluate the proposed in-train pruning technique on CIFAR-10 [31] and ImageNet [32] datasets. For CIFAR-10, we use 50K train and 10K validation images to train and evaluate our method respectively. The images have a resolution of 32 × 32 pixels. ImageNet consists of ∼ 1.28M train and 50K validation images with a resolution of 256 × 256 pixels. We use ResNet20 and ResNet56 as baseline models for the CIFAR-10 dataset, and ResNet18 as a baseline model for the ImageNet dataset. If not otherwise mentioned, all hyper-parameters specifying the task-related training were adopted from ResNet's base implementation [1]. For defensive training against adversarial attacks, we use FastAT [12].

This section is organized as follows. In Section 4.1, we analyze the effectiveness of incorporating trainable masks during standard training without any adversarial training. In Section 4.2, we demonstrate the effectiveness of in-train pruning on robust models by comparing the approach against an RL-search based state-of-the-art pruning scheme [5]. Finally, we compare our method with state-of-the-art robust pruning techniques in Section 4.3.

4.1. In-Train Pruning

We investigate the effectiveness of in-train channel pruning in Table 1 based on different constraints on the operation (Ops) reduction metric. As shown in column 3 of Table 1, we set the target reduction factor for operations ψ^* from Eq. (3) to $\{1.0, 0.4, 0.3, 0.2\}$ for ResNet20 and ResNet56 on the CIFAR-10 dataset, $\{1.0, 0.7, 0.5\}$ for ResNet18 on the ImageNet dataset. We observe -2.91 pp and -0.53 pp (percent point) of accuracy degradation for operation constraint $\psi^* = 0.4$ in ResNet20 and ResNet56 respectively. We also report the corresponding parameter reduction in column 5.

Table 1: In-train pruning for various operation constraints. We use ResNet20 and ResNet56 on CIFAR-10 dataset and ResNet18 on ImageNet dataset.

Model/ Dataset	Acc [%]	Ops Reduction Target	Ops Reduction Actual	Param Reduction
ResNet20 CIFAR-10	**92.47**	1.0	-	1.0
	89.56	0.4	0.38	0.68
	88.67	0.3	0.31	0.58
	88.17	**0.2**	**0.17**	**0.30**
ResNet56 CIFAR-10	**93.56**	1.0	-	1.0
	93.03	0.4	0.35	0.55
	92.38	0.3	0.28	0.50
	91.57	**0.2**	**0.18**	**0.37**
ResNet18 ImageNet	**68.53**	1.0	-	1.0
	67.22	0.7	0.69	0.88
	65.06	**0.5**	**0.45**	**0.78**

For an extreme target constraint $\psi^* = 0.2$, we observe an accuracy degradation of -4.3 and -1.99 pp for ResNet20 and ResNet56 respectively. The training behaviour which incorporates joint optimization of trainable weights and prune masks is analyzed in Fig. 2. We plot the Top1 accuracy and HW loss $\mathcal{L}_{HW}$, detailed in Eq. (3), across the training steps. The noisy behaviour in accuracy improvement can be seen across the iterations, indicating the joint optimization of the compression task (prune masks) and the learning task (weights). The operation constraints of $\psi^* = 0.3$ and 0.2 converge slightly slower compared to the operation constraint of $\psi^* = 0.4$. In Table 1, we also investigate the consistency of these trends on more challenging datasets such as ImageNet. We observe a minor degradation of -1.31 and -3.47 pp for operation constraints of 0.7 and 0.5 on the ResNet18 model trained on the ImageNet dataset. We also extend the method for the task of object detection to highlight its scalability in supplementary material S.3.

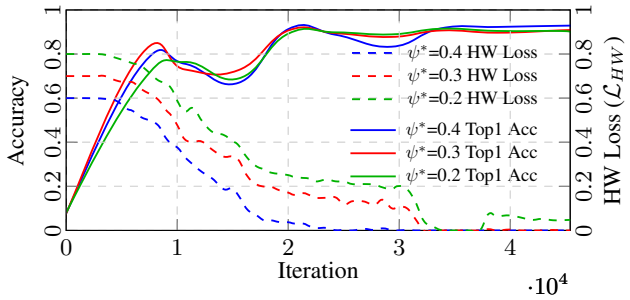

Figure 2: Comparison of in-train pruning behaviour across several training iterations for different operation constraints ψ^*=0.4, 0.3, 0.2.

Table 2 evaluates the proposed in-train pruning scheme for different pruning regularities. We observe that irregular weight pruning produces lower accuracy degradation (-1.16 pp, -0.38 pp) compared to structured channel pruning (-4.30 pp, -2.22 pp) for ResNet20 and ResNet56. Although weight pruning shows lower accuracy degradation for extreme target reductions, it is challenging to obtain inference benefits from such regularities on general-purpose structured execution hardware, *e.g.* GPUs.

Table 2: Exploring different pruning regularities for operation reduction factor ψ^*=0.2.

Model	Prune Regularity	Acc [%]	Ops Reduction Target	Ops Reduction Actual
ResNet20	baseline	**92.47**	1.0	-
	weight	91.31	0.2	0.16
	kernel	89.78	0.2	0.19
	channel	**88.17**	0.2	0.17
ResNet56	baseline	93.56	1.0	-
	weight	93.18	0.2	0.19
	kernel	92.25	0.2	0.21
	channel	**91.34**	0.2	0.21

4.2. Robust Pruning

In this section, we demonstrate our proposed in-train pruning method's ability to achieve compressed models which balance the trade-off between natural accuracy and adversarial robustness.

Baseline Training As a baseline for adversarial training, we implement FastAT [12] (see Section 3). For FastAT on the CIFAR-10 dataset, we use random FGSM with strength $\epsilon = 8/255$, step size $\alpha = 10/255$ to generate adversarial perturbations during the training process. We train for 300 epochs and set the initial learning rate to 0.1 and scale it down by a factor of 10 every 80 epochs. For evaluating robustness of the pruned models, the PGD attack is performed with $\epsilon = 8/255$ and $\alpha = 2/255$ for 20 iterations.

AMC-based Robust Pruning For the purpose of comparison with post-train pruning approach, we implement the state-of-the-art reinforcement learning-based pruning

scheme AMC [5] to find pruning configurations generating a trade-off between natural accuracy and adversarial accuracy. We constrain the number of operations to the target specified in Table 3 and Table 4 and allow the RL-agent to search for 500 episodes to obtain the pruning strategy. We adversarially fine-tune the resulting network with a cyclic learning rate of 0.1 for 30 epochs.

Improved Robustness with In-Train Pruning We augment our pruning approach with FastAT [12]-based adversarial training and start zeroing the prune masks at $E_{\text{Prune, Start}} = 20$ and freeze the masks at the $E_{\text{Prune, End}} = 240$. We use an initial learning rate of 0.1 and decrease it by a factor of 10 at the 80^{th} and 160^{th} epoch. We use the same attack strength as baseline training.

In Table 3, we make a comparison between the RL-based post-train pruning approach and the proposed in-train pruning method. Across all experiments, we observe an improvement in natural accuracy, while maintaining similar adversarial robustness. For a target reduction ψ^*=0.3 on ResNet20, we obtain an improvement of 5.91 pp in natural accuracy. For ResNet56 and ψ^*=0.3, we obtain an improvement of 8.65 pp in natural accuracy and with similar adversarial robustness. Similar improvements are achieved for the ImageNet dataset, see supplementary material S.2.

Table 3: Comparison between post-train RL-based robust pruning and the proposed in-train robust pruning for various operation constraints.

Model	Operations Reduction	FastAT + RL Prune Acc [%]	FastAT + RL Prune PGD-Acc [%]	FastAT + In-train Prune Acc [%]	FastAT + In-train Prune PGD-Acc [%]
ResNet20	1.0	81.52	40.65	81.52	40.65
	0.70	78.89	40.39	80.63	39.27
	0.50	77.11	39.65	80.32	40.14
	0.30	66.97	33.89	72.88	34.33
ResNet56	1.0	84.03	38.45	84.03	38.45
	0.70	82.78	42.47	84.52	36.91
	0.50	81.88	41.78	84.56	36.78
	0.30	74.75	36.95	83.40	36.89

In Fig. 3, we plot the training behaviour to compare the in-train pruning approach with (red) and without (blue) adversarial robustness, for $\psi^* = 0.3$. Compared to Fig. 2, we sample more data points to clearly perform the comparison. We observe noisy improvement in natural accuracy behaviour for the in-train robust pruning (red). The sudden fluctuation in accuracy at 15K and 30K iterations indicates the change in training behaviour due to the step learning rate policy. During these iterations, we observe large changes in the HW loss, indicating a phase of exploration in the binary prune masks M_b ($0 \leftrightarrow 1$). The changes in the pruning masks result in noisy accuracy improvement but eventually stabilize within 5K training iterations. We freeze the changes in pruning masks at the 45K iteration as the pruning constraint ψ^* is satisfied ($\mathcal{L}_{HW} = 0$).

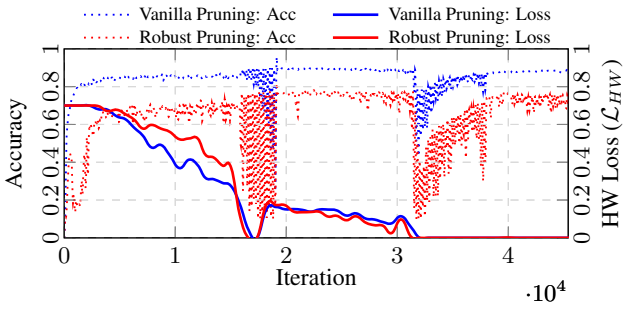

Figure 3: Comparison of the proposed in-train pruning scheme for operation constraint $\psi^* = 0.3$ with (red) and without (blue) the consideration of adversarial robustness.

We also verify the robustness of our in-train pruning scheme with stronger adversarial attacks such as Carlini-Wagner (C&W) [28] as shown in Fig. 4. C&W is an iterative attack guided by an optimizer such as Adam, which produces strong adversarial examples by simultaneously minimizing perturbation distance and manipulating the predictions based on a target class. Different loss functions can be applied in C&W attacks. In our experiments, the most efficient l_2-bounded loss is used for the evaluation. We run the attack for 100 iterations and set the C&W constant c=100, which is responsible for controlling the trade-off between the attacked image similarity and the success rate of the target class. We do not perform a binary search on the c value as suggested in the paper, since our focus is not on minimizing adversarial distance. Instead, we use a high value of c to ensure that the models are subjected to the strongest attack for reasonable image perturbations. In Fig. 4, we observe that the vanilla model trained without adversarial perturbations breaks very early at the 10^{th} iteration. However, robust models withstand the attack for more iterations ($\geq$ 20) with adversarial accuracy at least greater than 20%. Our pruned models obtain even higher adversarial accuracy than the unpruned RobustAT network after 30 iterations. We also observe higher adversarial accuracy starting from the 20^{th} iteration for our in-train pruned model with target constraint $\psi^* = 0.3$. This indicates the generalization capability of the in-train pruning approach as the compression rate increases.

In Table 4, we analyze the proposed in-train pruning approach with different pruning regularities in the context of adversarial robustness. Additionally, we compare our results with the post-train RL-based pruning scheme. The RL-agent proposed in the original work of AMC [5] is only suited for channel-wise pruning. We adapted the RL-agent to also perform pruning for different regularities. We ob-

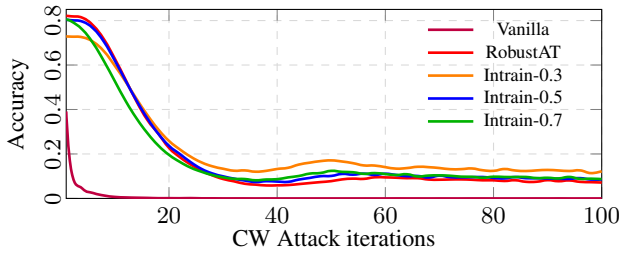

Figure 4: Comparison of adversarial robustness for different CNN models using C&W attack for ResNet20.

serve that irregular weight pruning gives the best trade-off between natural and adversarial accuracy. These observations also align with the robust pruning works in literature [13, 14]. The effectiveness of the in-train pruning scheme compared to the RL-based pruning scheme becomes more evident as the pruning becomes more structured (weight-wise $\rightarrow$ channel-wise). Compared to RL-based weight pruning, we observe a slight degradation in natural accuracy (-0.65 pp) for the in-train pruning scheme on ResNet20. However, the proposed method produces 5.91 pp, 0.44 pp better natural and adversarial accuracy respectively for channel pruning. The same trend also applies to ResNet56. As channel pruning is more advantageous on general-purpose accelerators such as GPUs, this strengthens the motivation for the proposed in-train pruning scheme.

Table 4: Robust pruning for various pruning regularities with target operation constraint $\psi^* = 0.3$.

Model	Pruning Regularity	FastAT + RL Prune Acc [%]	FastAT + RL Prune PGD-Acc [%]	FastAT + In-train Prune Acc [%]	FastAT + In-train Prune PGD-Acc [%]
ResNet20	1.0	81.52	40.65	81.52	40.65
	weight	79.08	40.35	78.43	38.59
	kernel	75.79	38.63	77.92	38.64
	channel	66.97	33.89	72.88	34.33
ResNet56	1.0	84.03	38.45	84.03	38.45
	weight	83.94	41.04	83.21	38.64
	kernel	81.68	40.82	83.31	38.68
	channel	74.75	36.95	83.40	36.89

4.3. Comparison with State of the Art

We compare the proposed in-train pruning approach to the robust pruning works in literature. In Table 5, we report the results of RobustADMM [13], Hydra [14] and ATMC [16]. RobustADMM, Hydra and ATMC use different baseline models, PGD evaluation parameters and adversarial training schemes. RobustADMM considers an over-parameterized ResNet as a baseline model and prunes it for various parameter constraints. We report their channel pruning results which achieve a model size of 0.04×10^6 (mentioned as $w = 1$ in [13]) and 0.17×10^6 (mentioned as $w = 2$ in [13]). Differently, our approach uses the smaller ResNet20 as a baseline model and achieves 6.21 pp and 6.31 pp better natural accuracy while maintaining similar adversarial robustness for model sizes with 0.04×10^6 and 0.16×10^6, respectively. The results from our approach dominate in terms of natural as well as adversarial accuracy for the same pruning constraints due to dynamic sparsity ratios across layers and heuristic-free pruning.

Compared to the work in Hydra [14], we achieve a significant improvement for channel pruning configurations. Different from RobustADMM, Hydra performs a PGD attack for 50 iterations to measure adversarial robustness. Compared to a 50% constrained channel-pruned VGG-16 model, we achieve 69.08% model reduction and 29.64 pp improvement in natural accuracy, while maintaining a similar level of adversarial robustness. The work in ATMC [16] evaluates robustness of compressed ResNet-34 with the PGD attack for 7 iterations. For the comparison, we use the weight pruned configuration of ATMC-32bit model with same attack hyper-parameters and obtain 6.63%, 14.43% higher robustness for $\epsilon = 4/255, 8/255$ respectively with similar model size. Different from [13, 14, 16], our pruning method does not require a pre-trained model.

Table 5: Comparing the in-train pruning scheme with SoTA on CIFAR-10 dataset.

Work	Baseline Model	Pre-trained Model	Pruning Regularity	PGD attack. $\epsilon, \alpha, iter$	Model Size $\times 10^6$	Acc [%]	Adv. Acc [%]
FastAT Wong et al. [12]	ResNet20	✗	no prune	8, 2, 10	0.27	82.06	40.97
				8, 2, 50	0.27	82.06	40.52
RobustADMM Ye et al. [13]	ResNet18	✓	channel	8, 2, 10	0.04	64.52	38.01
			channel	8, 2, 10	0.17	73.36	43.17
In-train Prune (Ours)	ResNet20	✗	channel	8, 2, 10	0.04	**70.73**	**39.31**
			channel	8, 2, 10	0.16	**79.67**	**43.22**
ATMC Gui et al. [16]	ResNet34	✓	weight	4, 0.7, 7	0.11	84.00	62.00
			weight	8, 1.4 , 7	0.11	84.00	39.00
In-train Prune (Ours)	ResNet56	✗	weight	4, 0.7, 7	0.13	82.68	**68.63**
			weight	8, 1.4, 7	0.13	82.68	**53.43**
Hydra Sehwag et al. [14]	VGG16	✓	weight	8, 2, 50	0.76	78.90	48.70
			weight	8, 2, 50	0.15	73.20	41.70
			channel	8, 2, 50	7.65	52.90	38.00
In-train Prune (Ours)	VGG16	✗	channel	8, 2, 50	**5.51**	**82.54**	**38.36**
			channel	8, 2, 50	**0.76**	73.40	30.20

5. Conclusion

In this work, we propose an *in-train* pruning technique, which eliminates the need for *pre-trained models*, *pruning heuristics* and computationally expensive *model exploration time*, traditionally required in literature. We highlight the effectiveness of our approach by comparing it against RL-based robust pruning method, producing improvements in natural accuracy and maintaining similar robustness at higher compression ratios. We also compare our approach with state-of-the-art robust pruning schemes. Our approach yields higher compression rates on channel pruning regularity, producing significant improvements in accuracy and adversarial robustness.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision (ECCV)*, 2018.

[3] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.

[4] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *International Conference on Computer Vision (ICCV)*, Oct 2017.

[5] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *European Conference on Computer Vision (ECCV)*, 2018.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[7] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: parameterized clipping activation for quantized neural networks," *CoRR*, vol. abs/1805.06085, 2018.

[8] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *NIPS*, 2017.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[12] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations (ICLR)*, 2020.

[13] 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 *International Conference on Computer Vision (ICCV)*, October 2019.

[14] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Hydra: Pruning adversarially robust neural networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[15] M. R. Vemparala, A. Frickenstein, N. Fasfous, L. Frickenstein, Q. Zhao, S. Kuhn, D. Ehrhardt, Y. Wu, C. Unger, N. S. Nagaraja, and W. Stechele, "BreakingBED – Breaking Binary and Efficient Deep Neural Networks by Adversarial Attacks," in *Intelligent Systems Conference (IntelliSys)*, September 2021.

[16] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[17] A. Frickenstein, M. R. Vemparala, C. Unger, F. Ayar, and W. Stechele, "DSC: Dense-sparse convolution for vectorized inference of convolutional neural networks," *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[18] Q. Huang, K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 709–718, 2018.

[19] A. Frickenstein, M.-R. Vemparala, N. Fasfous, L. Hauenschild, N.-S. Nagaraja, C. Unger, and W. Stechele, "ALF: Autoencoder-based low-rank filter-sharing for efficient convolutional neural networks," *Design Automation Conference (DAC)*, 2020.

[20] 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 *European Conference on Computer Vision (ECCV)*, 2018.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trendsin Machine Learning*, 2011.

[22] T. Zhang, K. Zhang, S. Ye, J. Li, J. Tang, W. Wen, X. Lin, M. Fardad, and Y. Wang, "ADAM-ADMM: A unified, systematic framework of structured weight pruning for dnns," *CoRR*, vol. abs/1807.11091, 2018.

[23] D. Mocanu, E. Mocanu, P. Stone, P. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Communications*, vol. 9, 2018.

[24] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," *ArXiv*, vol. abs/1907.04840, 2019.

[25] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *Proceedings of Machine Learning and Systems*, 2020.

[26] A. Galloway, G. W. Taylor, and M. Moussa, "Attacking Binarized Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[27] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[28] N. Carlini and D. A. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017.

[29] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *International Symposium on Computer Architecture (ISCA)*, 2016.

[30] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

[31] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (canadian institute for advanced research)."

[32] J. Deng, W. Dong, R. Socher, *et al.*, "ImageNet: A Large-Scale Hierarchical Image Database," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.