One Less Reason for Filter-Pruning: Gaining Free Adversarial Robustness with Structured Grouped Kernel Pruning

Shaochen (Henry) Zhong[1], **Zaichuan You**[*2], **Jiamu Zhang**[*2], **Sebastian Zhao**[*3], **Zachary LeClaire**[2], **Zirui Liu**[1], **Daochen Zha**[1], **Vipin Chaudhary**[2], **Shuai Xu**[2], and **Xia Hu**[1]

[1]Department of Computer Science, Rice University
{shaochen.zhong, zirui.liu, daochen.zha, xia.hu}@rice.edu
[2]Department of Computer and Data Sciences, Case Western Reserve University
{zxy456, jxz1217, zjl16, vipin, sxx214}@case.edu
[3]Electrical Engineering and Computer Sciences, UC Berkeley
sebbyzhao@berkeley.edu

Abstract

Densely structured pruning methods utilizing simple pruning heuristics are capable of delivering immediate compression and acceleration benefits with acceptable benign performances. However, empirical findings indicate such naively pruned networks are extremely fragile under simple adversarial attacks. Naturally, we would be interested in knowing if such phenomenon also hold true to carefully designed modern structured pruning methods. If so, then to what extent the severity? And what kind of remedies are available? Unfortunately, both the questions and the solution remain largely unaddressed: no prior art is able to provide a thorough investigation on the adversarial performance of modern structured pruning methods (spoiler: it is not good), yet the few works that attempt to provide mitigation often done so at various extra costs with only to-be-desired performance. In this work, we answer both questions by fairly and comprehensively investigate the adversarial performance of 10+ popular structured pruning methods. Solution-wise, we take advantage of *Grouped Kernel Pruning (GKP)*'s recent success in pushing densely structured pruning freedom to a more fine-grained level. By mixing up kernel smoothness — a classic robustness-related kernel-level metric — into a modified GKP procedure, we hereby present an one-shot-post-train-data-free GKP method capable of advancing SOTA performance on both benign and adversarial scale, while requiring no extra (in fact, often less) cost than a standard pruning procedure.

1 Introduction

Convolutional neural networks (CNNs) have demonstrated solid performance on tasks centered around computer vision. However, with modern CNNs growing in both widths and depths, the issue of over-parameterization has drawn increasing attention due to such networks often requiring large computational resources and memory capacity. To mitigate the burden, network pruning — the study of removing redundant parameters from original networks without significant performance loss — has become a popular approach for its simplicity and directness.

* Equal contribution. Order determined alphabetically by last name.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Despite the popularity of the pruning field in general, **few prior arts have been available to provide *improved adversarial robustness* under the constraint of *(densely) structured pruning***; even though empirical findings show vanilla structured pruning methods implemented with naive pruning strategies often experience huge performance drop on such adversarial tasks [Wang et al., 2018, Sehwag et al., 2020, Vemparala et al., 2021]. More concerning, no prior art has made an effort to provide a comprehensive investigation on whether the same phenomenon also exists under carefully designed modern structure pruning methods, where such methods are often capable of delivering excellent benign accuracy retention after pruning (sometimes, even improvements).

Below we provide a walk-through of why are such a constraint (densely structured) and property (being adversarially robust) considered preferable and important, to how we developed our solution by leveraging the power of increased structural pruning freedom (grouped kernel pruning) with kernel-level metrics (kernel smoothness). In the later sections of this paper, we replicate and test out around 13 popular densely structured pruning methods and variants against various white box (evasion) adversarial attacks, where our proposed method showcases clear dominance.

1.1 Structured v.s. Unstructured Pruning: Accuracy-Efficiency Trade-off

Most of the existing CNN pruning methods can be roughly categorized into *structured* and *unstructured* pruning. Note we said *roughly* because there is no universally agreed delineation between structured and unstructured pruning methods. The general consensus is that methods considered more unstructured often enjoy a higher degree of freedom on where to apply their pruning strategies (e.g., weight-level pruning) and therefore resulting in better accuracy retention.

In contrast, structured pruning methods often prune weights in a grouped manner following some kind of architecturally-defined constraints (e.g., filter-level pruning). Compared to their unstructured counterparts, structured pruning methods are more hardware-friendly and easier to obtain acceleration on commodity harddware, though at the cost of worse accuracy retention. This is due to an unstructurally pruned network is often left with pruned parameters randomly distributed in the weight matrix, leading to poor data reuse and locality [Yang et al., 2018]. It barely has wall-clock time speed up without supports like custom-indexing, special operation design, sparse operation libraries, or even dedicated hardware setups [Yang et al., 2018, Han et al., 2016].

Among all structured pruning methods, one popular line of research is to produce pruned networks that are entirely dense, a.k.a. ***densely structured, where the pruned weights are stored in the normal dense tensor format***. Such format of a pruned network is considered to be most library/hardware-friendly and, therefore, most deployable in a practical context. With such significant benefits, densely structured pruning methods consist of the absolute majority of structured pruning methods.

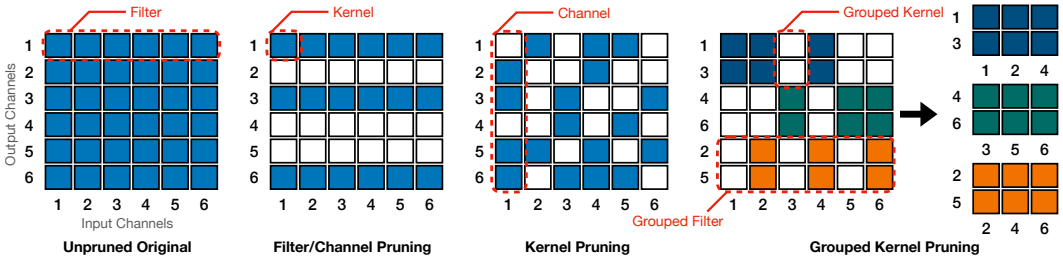

Figure 1: Visualization of different pruning granularities.

Densely structured pruning methods come with different pruning granularities, where a significant portion of prior arts prune at a filter or channel level. These two types of pruning are historically considered to be the limit of densely structured pruning, as showcased in Figure 1: if we go down one more level, we will have kernel pruning, yet its pruned network is not dense. This is until recently, authors from Zhong et al. [2022] utilized kernel pruning with grouped convolution, where they prune at a grouped kernel level to have an entirely dense pruned network. To the best of our knowledge, grouped kernel pruning carries the highest degree of pruning freedom among all densely structured pruning methods.

1.2 Pruned Models Are Fragile Under Adversarial Attacks — But Why Do We Care?

Empirical findings like Wang et al. [2018] suggest that although pruned neural networks may have acceptable benign accuracy, they are often more vulnerable to adversarial attacks. Adversarial robustness is recognized as a long-standing metric to evaluate model quality, as a model with undesired adversarial robustness can be easily exploited to produce wrong and potentially harmful output, resulting in fairness and accountability issues.

We would argue such robustness properties are especially valued under the context of (structured) pruning, where pruned models are often deployed to resource-constraint devices with less central oversight available and required to be executed in a more real-time manner. Imagine if an OCR model for real-time check redeeming can be maliciously exploited to read the number 1 as 9; the result will surely be unpleasant for many parties involved.

To alleviate such a problem (though not under a pruning context), prior arts like Wang et al. [2020] demonstrate the adversarial robustness of a convolutional network is largely correlated to its sensitiveness to high-frequency components (HFC), where such sensitiveness can be mitigated with some simple kernel-level operations like kernel smoothness.

1.3 Solution: Grouped Kernel Pruning with Adversarial-Robustness-Boosting Kernel Metrics

With the recent *Grouped Kernel Pruning (GKP)* framework pushed the pruning freedom of densely structured pruning to a (close) kernel-level [Zhong et al., 2022], we explore the unique possibility of mixing up adversarial-robustness-boosting kernel metrics — such as kernel smoothness — into the procedure of GKP. We present you Smoothly Robust Grouped Kernel Pruning (SR-GKP), a densely structured pruning method that works in a simple post-train one-shot manner, but often capable of delivering competitive benign performance and much stronger adversarial performance against SOTA filter and channel pruning methods with much more sophisticated procedures required. Solution-wise, our main claims and contributions are:

- **Free improvement on adversarial robustness** Our method has no extra (in fact, often less) cost compared to a standard pruning method, making the gained adversarial robustness entirely free.
- **One-shot & post-train: the simplest procedure with most compatibility.** The procedure of our method is both one-shot and post-train, which means it is compatible with any trained CNNs (as it does not interfere with the training pipeline), yet straightforward to execute.
- **Raise attention to the important but overlooked field of adversarially robust structured pruning.** Our method is among the few structured pruning methods capable of delivering pruned networks with improved adversarial performance — a field presents with severe problems, but receives little recognition nor solutions.

On the investigation side, we are the first to comprehensively reveal:

- **Drastic adversarial performance difference under a similar benign report**. We found that while different carefully designed modern densely structured pruning methods may showcase similar benign performance, some are done so at the cost of adversarial robustness.
- **One less reason for filter/ pruning: further endorsing GKP**. Filter and channel pruning methods have dominated the field of densely structured pruning for years; our work — together with Zhong et al. [2022] and Park et al. [2023] — showcased that when done right, grouped kernel pruning-based methods are superior under both benign and adversarial tasks, making it a promising direction for future densely structured pruning exploration.

For added bonuses, we are the first ones to reproduce and comprehensively report the benign and adversarial performances of multiple structured pruning methods under a fair setting. We believe the lack of such fair and comprehensive reports (on both benign and adversarial tasks) is mainly due to the lack of user-friendly tools. Thus, alongside our method implementation and checkpoint files, we also provide the pruning community **a lightweight open-sourced tool capable of a plug-and-play style of testing different victim models with various adversarial attacks while supporting all procedures a modern pruning method may require**, pending acceptance of this manuscript.

2 Related Work and Discussion

Due to page limitation, we will discuss related work regarding white box evasion adversarial attacks, adversarially robust structured pruning, and grouped kernel pruning. Other related topics, such as the compression/acceleration implications of structured and unstructured pruning methods, and input component frequency with kernel smoothness, will be introduced in Appendix B.

Adversarial Attacks. Neural networks are known to be vulnerable to adversarial attacks, i.e., a small perturbation applied to the inputs can mislead models to make wrong prediction Szegedy et al. [2014], Goodfellow et al. [2014].

In practice, adversarial attacks are often categorized as white-box and black-box evasion attacks, the difference being white-box attacks have access to the entirety of the model, including input features, architectures, and model parameters, while black-box attacks' access is often constrained (e.g.,,, only input-output pair). Thus, white-box attacks are almost always more effective and efficient than black-box; in fact, many classic black-box attacks are constructed in a way to approximate the information which is directly accessible by white-box attacks (e.g., gradient) [Chen et al., 2020]. Given one significant use case of the pruned model is edge-device deployments, where the model is more likely to be accessed; also, to have a straightforward workflow with harder challenges posing, we opt for white-box attacks for the scope of this paper.

Structured Pruning for Adversarial Robustness. Structured pruning methods, which arguably carry the most practical significance, has been heavily studied throughout the years [Molchanov et al., 2017, Yu et al., 2018, He et al., 2019, Wang et al., 2019a,b, Lin et al., 2019, He et al., 2018, Li et al., 2021, Zhong et al., 2022]. Despite their popularity, few of them focus on adversarial robustness. To the best of our knowledge, there are only three prior arts presented structured pruning methods while claiming improved performance on adversarial robustness metrics [Vemparala et al., 2021, Ye et al., 2019, Sehwag et al., 2020]. Unfortunately, Vemparala et al. [2021] does not have a public repository for code, Ye et al. [2019] does not have any experiment on standard `BasicBlock` ResNets for comparative investigation despite their popularity [Blalock et al., 2020], and Sehwag et al. [2020] is mostly proposed as an unstructured method with only one structurally pruned ablation study conducted on VGG-16, yet the structured pruning implementation is not published.

The lack of traffic, infrastructure, or baseline in this area has undoubtedly created deterrents to all interested scholars. To fill the gap, **we provide the community an open-sourced toolkit capable of testing various victim models against different adversarial attacks under a pruning context, and it comes with 10+ popular structured pruning methods already integrated for comparative evaluation.**

Grouped Kernel Pruning. Our work relies on the GKP framework — particularly inspired by the recent work Zhong et al. [2022]. The core of GKP is grouped kernel pruning and reconstruction to a grouped convolution format, where this combination has been explored a few times under the context of structure pruning.

Specifically, Zhong et al. [2022] proposed their take on grouped kernel pruning with a three-stage procedure: filter clustering, generate then decide which grouped kernel pruning strategy to employ, then reconstruct to grouped convolution format via permutation. This particular framework solved the previous drawback of requiring a complex procedure, yet a rich set of experiment results were showcased to demonstrate its performance advantages against many SOTA structured pruning methods. Our proposed method is largely enabled by the extra pruning freedom that GKP provides. Outside of Zhong et al. [2022], concurrent work like Zhang et al. [2022] and follow-up work like Park et al. [2023] have showcased the excellent benign performance of different GKP implementations.

3 Proposed Method

3.1 Motivation: Pruning May Amplify Overfitting to High-Frequency Components

Wang et al. [2020] suggests CNNs are prone to overfitting high-frequency components (HFC) of inputs — a type of feature that is not robust yet can be easily replicated with adversarial perturbations.

Table 1: Unpruned and channel-wise pruned ResNet-56 v. HFC/LFC-reconstructed CIFAR-10 test set. θ represents the cutoff threshold (for $0 \leq \theta \leq 1$). With a higher θ, the HFC-reconstructed images will exclusively include more high-frequency information; vice-versa for a lower θ.

INPUT	UNPRUNED BASELINE	CC PRUNED	NPPM PRUNED	L1NORM-B PRUNED
FULL ($\theta = 0.0$)	93.24	**94.04**	93.55	92.62
HFC ($\theta = 0.3$)	77.05	**80.22**	78.08	79.83
HFC ($\theta = 0.5$)	50.77	**57.49**	55.47	56.06
HFC ($\theta = 0.7$)	22.79	25.78	21.92	**27.15**

Though Wang's finding is towards an unpruned model, such phenomenon can be found, and in fact, even amplified, under a structural pruning setting.

A classic demonstration of such phenomena can be seen in Figure 2. Despite the `frog`-labeled figure reconstructed with only low-frequency components showing visible resemblance to its original benign format, a ResNet-56 model pruned by L1Norm filter pruning cannot classify it correctly. However, such pruned models can somehow correctly classify the same input reconstructed with only HFCs, even if it already lost all semantics of a frog to a human audience. This indicates a model pruned by methods without having adversarial robustness in consideration is more likely to overfit to HFC.

We emphasize that the above example is not a cherry-picked one. As shown in Table 1, by reconstructing the entire test set of CIFAR-10 with solely their high-frequency components, we find that a structurally pruned model is even more prone to fitting HFCs than its unpruned counterpart. This is potential because HFCs are easily learnable features under a benign setting, so pruned models want to "make most use" of their remaining weights given the reduced network capacity, and therefore become even more overfitted to HFCs by treating them as short-cut features.

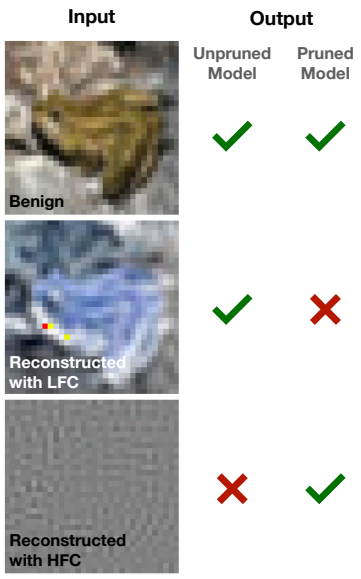

Figure 2: A `frog` figure from CIFAR-10 test set in its original, LFC, and HFC-reconstructed formats. The output indicates the correctness of classification results when test through of an unpruned and an L1Norm pruned ResNet-56.

Kernel smoothness as an indicator for learning from HFC. Fortunately, Wang et al. [2020] suggests ***kernel smoothness*** is highly correlated to the learning of HFC. Specifically, Wang et al. [2020] founds that a CNN with "smoother" kernels — where neighbor weights within a 2D kernel has less of a value difference — will reduce the overfitting of HFCs, thus making the model more robust against adversarial perturbations. For the ease of the following conversation, we define the kernel smoothness of a convolution kernel k for $k \in \mathbb{R}^{H \times W}$ to be:

$$\text{smoothness}(k) = \sum_{i=1}^{h \times w} \sum_{\substack{j \in \text{values} \\ \text{border with } k_i}} \left| k_j^2 - k_i^2 \right|, \tag{1}$$

where H and W are the kernel dimensions; in most CNNs, it is 3×3.

The finding of kernel smoothness and adversarial robustness presents a unique opportunity under the context of the GKP framework. Since prior to GKP, densely structured pruning is almost always done at a filter/channel level, where kernel-level metrics/operation have a little bearing when relaxed. However, GKP prunes at a (close) kernel level, where a kernel-level metric may still retain its power.

3.2 Mixing Smoothness into Grouped Kernel Pruning Procedure

It is natural to want to encapsulate some kernel-level operations/metrics — in this case, kernel smoothness — into the procedure of GKP. However, the challenge comes with how we can do it in an

efficient and effective manner. Particularly, how can we archive improved adversarial performance without sacrificing benign tasks?

For the ease of illustration, let $\boldsymbol{W}^\ell \in \mathbb{R}^{C^\ell_{\text{out}} \times C^\ell_{\text{in}} \times H^\ell \times W^\ell}$ be the weight of ℓ-th convolutional layer, which consist of C^ℓ_{out} filters, with each filter consists of C^ℓ_{in} number of $H^\ell \times W^\ell$ 2D kernels. According to Zhong et al. [2022], a standard GKP procedure has two potential stages:

Stage 1: **Filter grouping stage**: where the C^ℓ_{out} filters are clustered into n equal-sized filter groups $\{\boldsymbol{FG}^\ell_i, \boldsymbol{FG}^\ell_j, \ldots, \boldsymbol{FG}^\ell_n\}$, with each filter group $\boldsymbol{FG}^\ell \in \mathbb{R}^{C^\ell_{\text{out}}/n \times C^\ell_{\text{in}} \times H^\ell \times W^\ell}$.

Stage 2: **Pruning strategies obtaining/pruning stage**: where the pruning method generates a set of "candidate" grouped kernel pruning strategies to be evaluated and select from; where a grouped kernel is defined as a $\boldsymbol{GK} \in \boldsymbol{FG}^\ell$ with $\boldsymbol{GK}$ having a shape of $C^\ell_{\text{out}}/n \times 1 \times H^\ell \times W^\ell$. Finally, we evaluate all collected candidate strategies and decide which to pursue.

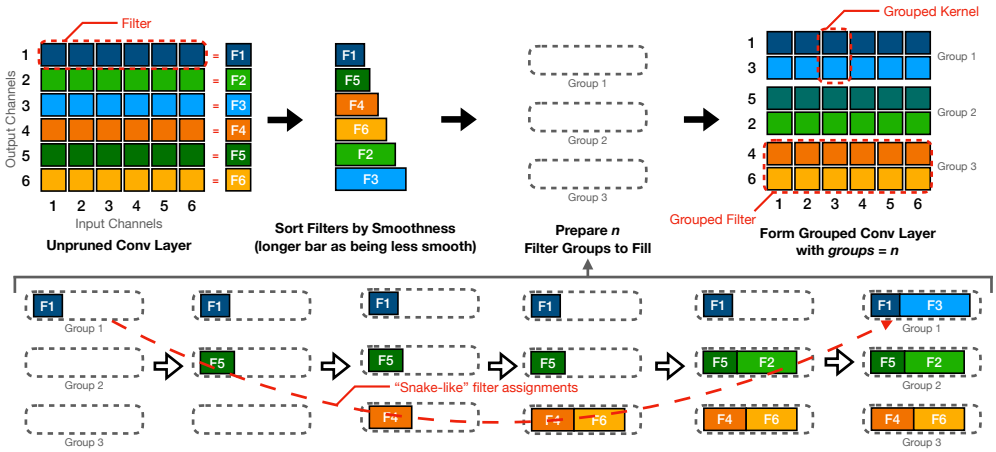

Figure 3: Smoothness Snaking For Filter Grouping (Stage 1)

However, how to apply kernel smoothness-related criteria to such stages is a non-trivial question. Our empirical results from some proof-of-concept experiments suggest some naive applications either will not work at all or will only work by significantly sacrificing the performance on benign tasks: e.g., if we simply replace Stage 2 above by pruning the grouped kernels with greater $\sum_{k \in \boldsymbol{GK}} \text{smooth}(k)$, we had experienced a huge drop on benign accuracy Appendix C. Therefore, we must drive our attention to discover some more sophisticated way of mixing up such criteria into the above stages.

3.2.1 Smoothness Snaking: Is Filter Clustering the Only Right Answer for GKP?

Per TMI-GKP [Zhong et al., 2022], filters within the same convolutional layer are grouped into several equal-sized groups by a *filter clustering schema*, which consist of some different combinations of dimensionality reduction and clustering techniques. The motivation of grouping by clustering is natural as it establishes a preferable search space for pruning algorithms, where the power of pruned components can likely be retrained via similar unpruned components within the same group. However, later procedures of TMI-GKP (e.g., Stage 2 and 3 per Section 3.2) then seek to maintain a diverse representation of kept components within each filter group, **which beg the question: is filter clustering the only right answer for GKP? If finding a diverse set of unpruned grouped kernels is the goal, why not have filter groups with filter diversity to start with?** And if the answer is "No." How can we utilize this opportunity to mix up with kernel smoothness criteria?

Grouped-based pruning will always seek out some kind of balance between groups to avoid skewed distribution [Zhong et al., 2022]. Following such principle, we want our filter groups to have balanced smoothness — so that we do not end up having any filter group that is "over" or "under-smoothed" to start with for pruning. However, this is equivalent to the partition problem, which is known to be NP-hard. Given the filter grouping stage is often robust to adjustments[1], we proposed to sort filters according to their smoothness, then assign them iteratively in an S-shaped "snaking" manner across a

[1] In Zhong et al. [2022], various filter grouping strategies — including random assignment — were proposed, yet many of them tend to perform reasonably well.

predefined number of filter groups, as shown in Figure 3, namely ***Smoothness Snaking***. Empirical results suggest although smoothness snaking may not as optimal as the dynamic clustering scheme in Zhong et al. [2022] in terms of benign performance, it is able to provide better adversarial robustness under adversarial attacks and is faster to execute due to the absence of dimensionality reduction & clustering procedures (Appendix C). We consider this to be a successful mix-up.

3.2.2 Smooth Beam Greedy GKP Search

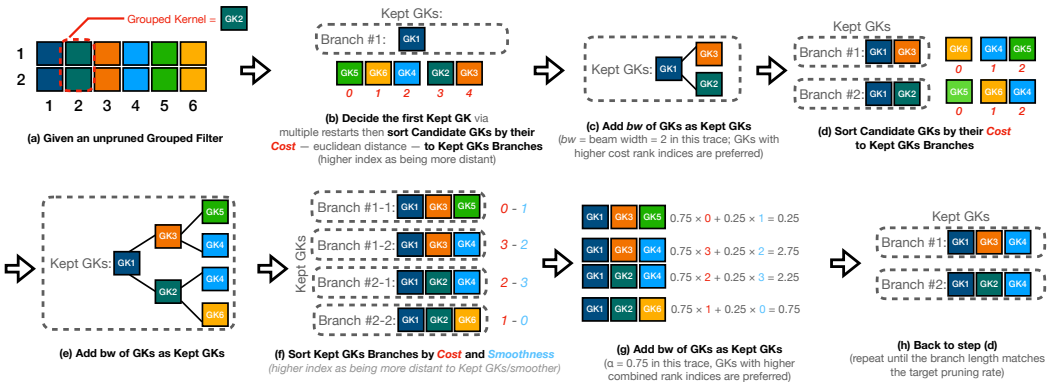

Figure 4: Smoothness-aware Beam Greedy Search for Grouped Kernel Pruning (Stage 2).

Known that the pruning strategies obtaining/decision stage (Stage 2 per Section 3.2) of GKP is sensitive to temper, where a direct application of smoothness criteria won't work Appendix C. This indicates the distance-based *cost* formula (Equation 5) proposed in Zhong et al. [2022] carries a significant influence on the performance of a pruned network, which is non-surprising given the vast popularity of distance-based pruning arts. Since we can't directly replace this *cost* formula, we propose to widen the capacity of pruning strategies obtaining stage with smoothness in mind while keeping the decision stage akin to the *cost* formula. By doing this, the chances of more "smoothness-aware" pruning strategies being employed are increased.

Again, to keep our approach simple, we implemented a custom beam search element to consider more grouped kernels during each advancement (Figure 4). Per each iteration, all gathered candidate grouped kernel pruning strategies will be evaluated against a mixture of smoothness and cost criteria, where only a $Beam_{\text{width}}$ amount of grouped kernels will be kept for further advancement (until the desired pruning ratio is reached). The scoring formula to determine which subset of strategies may be kept in the beam is defined as:

$$\underset{\text{mix-up}}{\text{Score}}(\boldsymbol{GK}_{\text{branch}}, \alpha) = \alpha \cdot \phi\big(\text{Cost}(\boldsymbol{GK}_{\text{branch}})\big) + (1-\alpha) \cdot \phi\big(\text{Smoothness}(\boldsymbol{GK}_{\text{branch}})\big), \quad (2)$$

where $\phi\big(\texttt{Criterion}(\boldsymbol{GK}_{\text{branch}})\big)$ represents the rank of such $\boldsymbol{GK}_{\text{branch}}$ when all collected $\boldsymbol{GK}_{\text{branch}}$ candidates are sorted according to the given `Criterion` in a descending order. So $\phi\big(\text{Smoothness}(\boldsymbol{GK}_{\text{branch}})\big) = 0$ would suggest this particular $\boldsymbol{GK}_{\text{branch}}$ has a greater smoothness value (a.k.a. "less smooth") then all other $\boldsymbol{GK}_{\text{branch}}$ candidates in considerations. α is a tunable parameter that adjusts the importance of one metric over the other. The $\text{Smoothness}(\boldsymbol{GK}_{\text{branch}})$ equation is simply the sum of all kernels within all $\boldsymbol{GK}_{\text{kept}}$ over Equation 1; yet the $\text{Cost}(\boldsymbol{GK}_{\text{branch}})$ equation is a modified version of (Equation 5) in Zhong et al. [2022], where we removed some hyper-parameters for simplicity and to reduce tuning workload. Please refer to Figure 4 and Appendix C and for more details.

4 Experiments and Results

4.1 Experiment Setups

We evaluate the efficacy of our method on ResNet-32/56/110 with the `BasicBlock` implementation, ResNet-50/101 with the `BottleNeck` implementation, and VGG-16 [He et al., 2016, Simonyan and

Zisserman, 2015]. For datasets, we choose CIFAR-10 [Krizhevsky, 2009], Tiny-ImageNet [Wu et al., 2017], and ImageNet-1k [Deng et al., 2009] for a wide range of coverage. For all compared methods and under most model-dataset combinations, we tried our best to replicate them with a ≈ 300 epochs (except for ImageNet, where we only employ 100 epochs) of fine-tuning/retraining budget while maintaining all other settings either identical or proportional to their original publications.

4.2 Compared Methods and Evaluating Criteria

We evaluate our proposed method against up to 13 popular densely structured pruning methods and variants shown in Appendix D.3. We produce pruned-and-finetuned (or retrained) models upon identical unpruned baseline models with similar post-prune MACs and Params. Then, we compare their inference accuracy for benign inputs as well as adversarially-perturbed inputs powered by *FGSM* and *PGD*-perturbed in various perturbation budgets and intensities.

One reporting mechanism that is probably unique to our paper — in comparison to standard pruning art under the benign space — is for some of our methods, results of multiple epochs are reported, each showcasing a method's peak performance against different accuracy metrics. This is because for benign tasks, following He et al. [2019], only the epoch with the best benign accuracy needs to be reported. But under an adversarial context, if a pruning method is capable of producing more than one fully pruned model during the fine-tuning/retraining stage, often time the best performer per each evaluation metric does not overlap. We believe it is responsible for reporting them all as there are no dominate evaluation metrics in our experiments; yet this is an important advantage for methods that can generate multiple usable pruned models (e.g., one-shot pruning) over methods that can only generate a few or just one fully pruned model (e.g., layer-wise iterative pruning).

We also investigate each pruning method against a checklist of questions, including pruning granularity, procedure, when is the first fully pruned epoch, zero-masked or hard-pruning, and many other important questions. This, along with the epoch budget constraint and baseline control, should provide our audience with a more leveled playing field for informed methods comparison.

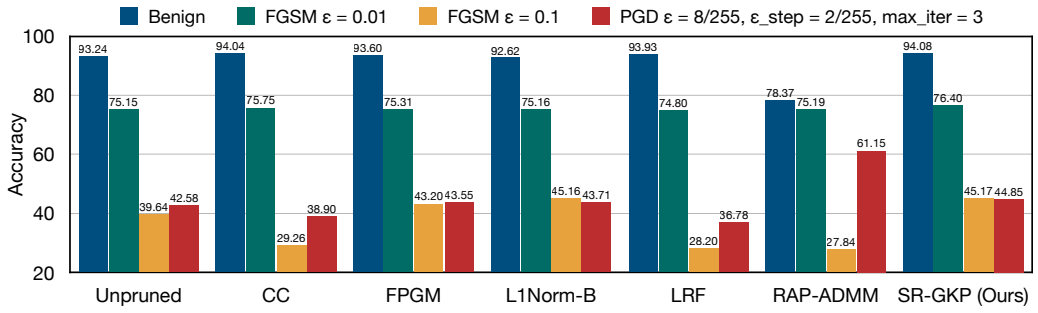

Figure 5: Visualization of ResNet-56 on CIFAR-10 — note this plot is done in a "superscore" manner for a concise presentation; the four bars of a method may not belong to the same model checkpoint.

4.3 Results and Analysis

Table 2 and Table 3 showcased the performance of various modern SOTA methods as well as our proposed methods on the two most popular model-dataset combinations [Blalock et al., 2020]. For ResNet-56 on CIFAR-10, our method outperformed every other method on all evaluating criteria, except for PGD; as RAP-ADMM outperformed SR-GKP significantly with 61.16% v.s. 44.85%. However, it is worth noting that RAP-ADMM does adversarial training on PGD-perturbed data, so it is not surprising that it performs well against the seen type of attack. Unfortunately, it seems like the adversarially trained RAP-ADMM cannot generalize its defense to other adversarial attacks, even though they are similar in nature. Also, RAP-ADMM has the worst benign performance across all showcased methods, yet its training time is significantly longer due to the need to perturb its training data on-the-fly constantly.

Table 2 (and similar experiments showcased in) may also answer one of our research questions: are carefully designed modern structured pruning methods also fragile under adversarial attacks? The answer is an unfortunate "Yes," as not only do recent structured pruning methods show serious performance drops under adversarial attacks, such drops are often more severe than their predecessors

Table 2: ResNet-56 on CIFAR-10. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	93.24	75.15	39.64	42.58	126.561	0.853
CC [Li et al., 2021]	Best Benign	94.04	74.78	29.25	37.85	69.837	0.616
	Best (a)	93.73	75.75	29.26	38.86		
	Best (b)	93.70	75.56	29.89	38.90		
FPGM [He et al., 2019]	Best Benign	93.60	75.31	43.20	43.55	71.661	0.482
	Best (a)	93.37	76.28	44.96	44.66		
	Best (b)						
HRank [Lin et al., 2020]	Best Benign	92.27	72.32	19.11	32.51	79.237	0.584
	Best (b)						
	Best (a)	92.07	72.59	18.94	32.21		
L1Norm-B [Li et al., 2016]	Best Benign	92.62	72.97	41.30	41.79	72.115	0.586
	Best (a)	91.94	75.16	42.49	43.71		
	Best (b)	91.70	74.40	45.16	43.41		
LRF [Joo et al., 2021]	Best Benign	93.93	73.47	25.59	34.86	71.009	0.490
	Best (a)	93.68	74.80	27.39	36.78		
	Best (b)	93.63	74.06	28.20	35.98		
NPPM [Gao et al., 2021]	Best Benign	93.55	74.82	29.07	37.12	70.843	0.601
	Best (a)	93.35	75.50	30.29	38.09		
	Best (b)	93.43	75.27	31.18	38.13		
RAP-ADMM[Ye et al., 2019]	-	78.37	75.19	27.84	**61.15**	71.661	0.482
SFP[He et al., 2018]	-	93.15	75.63	43.83	44.10	71.462	0.481
TMI-GKP [Zhong et al., 2022]	Best Benign	93.95	75.18	42.18	43.46	71.855	0.482
	Best (a)	93.37	75.88	42.55	43.74		
	Best (b)	93.66	75.74	44.09	44.51		
SR-GKP (Ours)	Best Benign	**94.08**	75.89	42.60	43.85	71.855	0.482
	Best (a)	93.83	**76.40**	**45.17**	**44.85**		
	Best (b)						

Table 3: ResNet-50 on ImageNet-1k. Note this table includes two baselines: "self-trained" and "`torchvision`". This is because TMI-GKP requires training epoch snapshots, which is not supplied with the `torchvision` pretrained ResNet-50.

Method	Baseline	Benign	FGSM$_{\varepsilon=0.001}$	FGSM$_{\varepsilon=0.01}$	FGSM$_{\varepsilon=0.1}$	PGD$^{\varepsilon=4/255,\ \varepsilon_{\text{step}}=1/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	Self-trained	75.70	67.57	25.82	16.16	6.66	4122.828	25.557
TMI-GKP [Zhong et al., 2022]	Self-trained	75.02	67.62	25.86	15.82	7.26	2725.954	17.069
SR-GKP		75.29	68.02	26.45	15.83	8.08	2725.954	17.069
Unpruned	`torchvision`	76.13	70.18	28.70	13.93	9.27	4122.828	25.557
FPGM [He et al., 2019]	`torchvision`	75.04	**68.50**	25.84	13.06	7.43	2641.670	18.310
SFP [He et al., 2018]		58.50	55.72	25.82	9.01	**11.00**	2635.129	17.302
SR-GKP (Ours)		**75.34**	68.04	**26.65**	**15.94**	7.85	2759.672	17.803

— which often rely on much naive design. Figure 5 provides a vivid illustration with LRF — a 2021 method — showing the weakest adversarially robustness across the plotted methods.

Upon careful comparison, we noticed that SFP and FPGM — two pre-2020 methods — tend to be the best filter pruning methods under the double scrutiny of benign and adversarial tasks. However, this only holds true to smaller scale experiments, as SFP is significantly outperformed by SR-GKP on ResNet-50 on ImageNet (58.50% v.s. 74.34%). Though FPGM tends to perform better on the same task, it is still behind SR-GKP in a general sense.

Last, although TMI-GKP is optimized for benign tasks, and its procedure makes no consideration for adversarial tasks, it naturally comes with strong adversarial robustness. We believe this has a lot to do with the increased pruning freedom enabled by the GKP granularity, which further illustrates the potential of GKP-based methods outside its achievements in benign space. Due to page limitation, we hereby only showcase an abbreviated version of the experiments. **We strongly encourage our readers to check out our full experiments at Appendix D.3 with a lot more comprehensive coverage on many more dataset-model combinations.**

5 Conclusion

Our work studies the area of adversarially robust structured pruning: a topic presents with severe problems, but lacks proper recognition or explorations. On the investigation side, we reveal that — just like their naive predecessors — carefully-designed modern structured pruning methods are also fragile under adversarial attacks, yet different pruning methods may yield drastically different adversarial performance, while hiding behind similar benign reports. On the solution side, we propose SR-GKP: a simple one-shot GKP method showcases competitive benign performance with a

significant advantage under adversarial attacks against comparable SOTA methods, while requiring no extra cost from a pruning procedure perspective.

We believe the overlooked nature of this field is mainly two-fold: the lack of pruning freedom to utilize findings of other fields, and the lack of tools for doing adversarial evaluations under a pruning context. We present our take and contribution to both issues by showcasing the capability of GKP-based methods and provide our community an open-sourced tool for future studies.

6 Acknowledgment

This research was supported, in part, by NSF Awards OAC-2117439, OAC-2112606, IIS-2224843, and IIS-1900990. This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.

References

D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

J. Chen, M. I. Jordan, and M. J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 1277–1294. IEEE, 2020. doi: 10.1109/SP40000.2020.00045. URL `https://doi.org/10.1109/SP40000.2020.00045`.

T. Chen, L. Liang, D. Tianyu, Z. Zhu, and I. Zharkov. Otov2: Automatic, generic, user-friendly. In *International Conference on Learning Representations*, 2023.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

S. Gao, F. Huang, W. Cai, and H. Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9270–9280, June 2021.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv 1412.6572*, 12 2014.

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. *ACM SIGARCH Computer Architecture News*, 44(3): 243–254, 2016.

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, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90.

Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2234–2240. ijcai.org, 2018. doi: 10.24963/ijcai.2018/309.

Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4340–4349. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00447.

D. Joo, E. Yi, S. Baek, and J. Kim. Linearly replaceable filters for deep network channel pruning. In *AAAI Conference on Artificial Intelligence*, 2021.

A. Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009.

F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*, 2021.

H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rJqFGTslg`.

Y. Li, S. Gu, K. Zhang, L. Van Gool, and R. Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *Proceedings of the European Conference on Computer Vision*, 2020.

Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6438–6447, 2021.

M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao. Hrank: Filter pruning using high-rank feature map. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1526–1535. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00160.

S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

J.-H. Park, Y. Kim, J. Kim, J.-Y. Choi, and S. Lee. Dynamic structure pruning for compressing cnns. *ArXiv*, abs/2303.09736, 2023.

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

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6199`.

M. R. Vemparala, N. Fasfous, A. Frickenstein, S. Sarkar, Q. Zhao, S. Kuhn, L. Frickenstein, A. Singh, C. Unger, N. S. Nagaraja, C. Wressnegger, and W. Stechele. Adversarial robust model compression using in-train pruning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 66–75, 2021.

H. Wang, X. Wu, Z. Huang, and E. P. Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8681–8691, 2020. doi: 10.1109/CVPR42600.2020.00871.

L. Wang, G. W. Ding, R. Huang, Y. Cao, and Y. C. Lui. Adversarial robustness of pruned neural networks. In *Submission to International Conference on Learning Representations (Workshop)*, 2018. URL `https://openreview.net/forum?id=SJGrAisIz`.

W. Wang, C. Fu, J. Guo, D. Cai, and X. He. COP: customized deep model compression via regularized correlation-based filter-level pruning. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3785–3791. ijcai.org, 2019a. doi: 10.24963/ijcai.2019/525.

W. Wang, S. Zhao, M. Chen, J. Hu, D. Cai, and H. Liu. DBP: discrimination based block-level pruning for deep model acceleration. *CoRR*, abs/1912.10178, 2019b.

J. Wu, Q. Zhang, and G. Xu. Tiny imagenet challenge. *Technical Report*, 2017.

C. Yang, A. Buluç, and J. D. Owens. Design principles for sparse matrix multiplication on the gpu. In *European Conference on Parallel Processing*, pages 672–687. Springer, 2018.

S. Ye, X. Lin, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, and Y. Wang. Adversarial robustness vs. model compression, or both? pages 111–120, 10 2019. doi: 10.1109/ICCV.2019.00020.

R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis. NISP: pruning networks using neuron importance score propagation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9194–9203. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00958.

G. Zhang, S. Xu, J. Li, and A. J. X. Guo. Group-based network pruning via nonlinear relationship between convolution filters. *Applied Intelligence*, Jan. 2022. doi: 10.1007/s10489-021-02907-0.

S. Zhong, G. Zhang, N. Huang, and S. Xu. Revisit kernel pruning with lottery regulated grouped convolutions. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=LdEhiMG9WLO`.

A Few Corrections to the Main Paper. We have found some typos and editing errors in our main paper after our submission, and we would like to utilize the appendix to highlight a couple of them to resolve any confusion they may have caused:

- On Page 6, Section 3.2.1, Line 230. It should be "However, later procedures of TMI-GKP (e.g., Stage 2 ~~and 3~~ per Section 3.2)...", as there is no Stage 3 in Section 3.2.
- On Page 7, the caption of Table 2 is incomplete. It should read "ResNet-56 on CIFAR-10. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats the showcased MACs/Params reduction that perform best against criterion (a)."
- On Page 9, Section 4.3, Line 314, the Figure anchor is not rendered correctly. It should refer to Figure 5.
- On Page 9, Table 3. The four accuracy reports of SR-GKP with the "self-trained" baseline should be **embolden**, as it is strictly better than TMI-GKP across all evaluating criteria.

Hope those edits help, we apologize the inconvenience they may have caused.

A Limitation and Broader Impact

Though our work mainly focuses on the benign and adversarial performances of densely structured pruning methods, we expect grouped kernel pruning to provide impressive performance under other reasonable evaluating metrics — as it simply comes with a higher degree of pruning freedom than typical filter/channel pruning. We encourage our colleagues to explore more variants of GKP for different compression tasks.

Note though our investigation revealed serious performance issues under adversarial tasks, they are still limited to benign and artificially perturbed input. We will leave a more comprehensive investigation against other robustness metrics for future work.

B Extended Related Work and Discussion

B.1 Structured v.s. Unstructured Pruning

Pruning methods can be roughly divided into structured pruning and unstructured pruning according to the pruning granularity. Specifically, unstructured pruning often means we prune each weight independently. In contrast, structured pruning bundles weights into groups, then prunes the whole group instead of the individual weight (e.g., block-wise Lagunas et al. [2021], channel-wise He et al. [2017], group-wise pruning Zhong et al. [2022]).

Unstructured pruning can maintain the model performance better with the same number of parameters. However, unstructured pruned models yield marginal wall-clock time efficiency, or even slower than the unpruned model at the low sparsity regime. This is because unstructured pruned matrices need to be stored in sparse matrix format, as the zeros are randomly distributed in these matrices Yang et al. [2018]. Operations executed on sparse matrices (e.g., sparse matrix multiplication, sparse embedding table look-up) are notoriously inefficient on commodity hardware, e.g., GPUs and CPUs, due to the limited data reuse and random memory access Yang et al. [2018], Han et al. [2016].

In contrast, although structured pruning has less flexibility compared to the unstructured one, it is much more hardware/library-friendly since structurally pruned matrices can still be stored in the dense matrix format. Thus, operations executed on structured pruned matrices are the same to those in the unpruned model, which are highly optimized. Consequently, the compression provided by structured pruning can often translate into the real wall-clock time speedup upon proper implementations.

B.2 Learning of High-Frequency Components and its Adversarial Implications

As we consult adversarial-robustness-boosting kernel metrics and operations, we heavily rely on the findings from Wang et al. [2020], a work that discusses how learning components with different frequencies may affect the adversarial robustness of a CNN, and how may some kernel-level metrics like kernel smoothness influence such type of learning. We will elaborate more on this in Section **??** above.

C Additional Details on Proposed Method

C.1 Naive Mix-Up Attempts of Kernel Smoothness

As per Section 3.2, there are generally two stages in a GKP procedure: *Filter Grouping* and *Grouped Kernel Pruning*. The following experiments shall attempt to mix-up kernel smoothness criteria into each of such stages, and we can therefore find out which stage is "friendly" to such mix-up operations and how such operations should look like.

C.1.1 During GKP Filter Grouping (Stage 1)

In TMI-GKP [Zhong et al., 2022], the filter grouping stage is driven by the *tickets magnitude increase* score, known as *TMI-driven Clustering*. We utilize it as a baseline to investigate whether our smoothness-aware filter grouping operation — *Smoothness Snaking* (Section 3.2.1 and Figure 3) — can maintain the baseline performance and provide improvements. Note, we denote TMI-GKP's grouped kernel pruning scheme (GK Pruning) as *Greedy* in the following Table 4.

Table 4: Comparison of different *Filter Grouping* methods in GKP. All compared models are pruned to identical MACs/Params for fairness.

Model	Filter Grouping	GK Pruning	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$
ResNet-32	Unpruned		-	92.80	71.93	31.35
	TMI-driven Clustering	Greedy	Best Benign	**92.99**	69.15	19.59
			Best (a)	92.03	70.61	15.85
			Best (b)	92.77	69.90	30.51
	Smoothness Snaking	Greedy	Best Benign Best (a)	92.77	**71.47**	**30.64**
			Best (b)	92.65	**70.82**	**30.65**
ResNet-56	Unpruned		-	93.24	75.15	39.64
	TMI-driven Clustering	Greedy	Best Benign	**93.95**	75.18	42.18
			Best (a)	93.37	75.88	42.55
			Best (b)	93.66	75.74	44.09
	Smoothness Snaking	Greedy	Best Benign	93.62	75.68	41.21
			Best (a)	93.40	**76.05**	43.03
			Best (b)	93.44	75.69	**44.62**

It can be observed that though Smoothness Snaking may yield a slightly lower benign performance than its TMI-driven baseline, it may improve the adversarial robustness when used with the same grouped kernel pruning procedure. We would also note Smoothness Snaking is significantly faster (up to 1,600x) than TMI-driven clustering due to the absence of dimensionality reduction and clustering procedure (see Table 7 for details).

C.1.2 During GKP Pruning Strategies Obtaining/Pruning (Stage 2)

Following the section above, here we investigate the performance of different *Grouped Kernel Pruning* methods. Our proposed method is denoted as *Smooth Beam Greedy* (see Figure 4 for details), TMI-GKP's grouped kernel pruning method is denoted as *Greedy* as above, and *Lease Smooth* represent a vanilla adaptation of smoothness-driven pruning, where the grouped kernels that are least smooth are pruned.

Table 5: Comparison of different *Grouped Kernel Pruning* methods in GKP. All compared models are pruned to identical MACs/Params for fairness.

Model	Filter Grouping	GK Pruning	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$
ResNet-32	Unpruned		-	92.80	71.93	31.35
	Smoothness Snaking	Greedy	Best Benign Best (a)	92.77	71.47	30.64
			Best (b)	92.65	70.82	30.65
	Smoothness Snaking	Least Smooth	Best Benign	90.09	63.91	17.22
			Best (a)	88.90	70.41	15.33
			Best (b)	48.37	47.80	28.55
	Smoothness Snaking	Smooth Beam Greedy	Best Benign	**93.02**	70.80	30.04
			Best (a)	91.94	**71.48**	15.49
			Best (b)	92.73	**70.82**	**31.11**
ResNet-56	Unpruned		-	93.24	75.15	39.64
	Smoothness Snaking	Greedy	Best Benign	93.62	75.68	41.21
			Best (a)	93.40	76.05	43.03
			Best (b)	93.44	75.69	44.62
	Smoothness Snaking	Least Greedy	Best Benign	90.97	73.32	24.95
			Best (a)	90.01	74.63	14.10
			Best (b)	90.59	73.60	26.40
	Smoothness Snaking	Smooth Beam Greedy	Best Benign	**94.08**	75.89	42.60
			Best (a) Best (b)	**93.83**	**76.40**	**45.17**

From Table 5, we may tell that *Smooth Beam Greedy* may significantly improve the adversarial robustness of the pruned networks, yet, it also provides remedies to the decrease in benign performance due to the smoothness snaking operation. It may also be worth noting that the *Least Smooth* operation is extremely detrimental to almost all tracking metrics, suggesting a vanilla mix-up is not appropriate.

C.1.3 Bonus Investigation: Is Smoothness-aware Filter Pruning Possible?

One main purpose of our paper is to endorse the potential of GKP under adversarial tasks, after [Zhong et al., 2022, Zhang et al., 2022, Park et al., 2023] showcased the power of GKP in a benign context, thus "one less reason for filter pruning." But to make such a claim proper, we will need to investigate whether it is possible to do the same smoothness-aware mix-up with a filter pruning procedure.

Here in Table 6, we use SFP [He et al., 2018] as the baseline, which is considered one of the strongest filter pruning methods on CIFAR-10 [Krizhevsky, 2009]. Then, we try to apply the same ranked-based kernel-smoothness mix-up algorithm as in Equation 2 to find out if such a strong filter pruning baseline can withstand the same mix-up under a filter level.

Table 6: Filter pruning method SFP [He et al., 2018] applied with the same mix-up algorithm in Equation 2. Note "CSB" represents "cost smoothness balancer", which is also α in Equation 2 — so a higher CSB means more biased towards the distance-based *Cost* metrics. All compared models are pruned to identical MACs/Params for fairness.

Model	Method	Benign	$\text{FGSM}_{\varepsilon=0.01}$	$\text{FGSM}_{\varepsilon=0.1}$	$\text{PGD}^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max\_iter}=3}$
ResNet-32	SFP	91.94	69.25	30.34	30.18
	CSB = 0.5	83.49	40.86	21.12	24.28
	CSB = 0.75	91.03	65.94	22.92	25.40
ResNet-56	SFP	93.15	75.63	43.83	44.10
	CSB = 0.5	81.83	60.46	19.59	28.45
	CSB = 0.75	92.26	72.11	37.42	36.64
	CSB = 0.9	93.09	74.78	40.64	41.41

It can be observed that SFP under such mix-up is completely unusable, which suggests the same mix-up strategy, though effective on GKP, is not transferable to filter pruning.

C.2 Filter Grouping Speed Comparison: TMI-Driven Clustering v.s. Smoothness Snaking

Table 7: Wall-clock runtime comparsion between SR-GKP (Ours) and TMI-GKP [Zhong et al., 2022].

Method	ResNet-32 Group	ResNet-32 Total	ResNet-56 Group	ResNet-56 Total	ResNet-110 Group	ResNet-110 Total
TMI-GKP	47m 6s	1h 20m 10s	2h 32m 12s	2h 36m 22s	4h 53m 33s	5h 30m 18s
SR-GKP	3s	11m 36s	5s	20m 43s	11s	1h 10m 16s

In Table 7, we showcased the significant runtime advantage of SR-GKP to TMI-GKP due to the absence of clustering and dimensionality reduction procedure.

C.3 SR-GKP Procedure

C.3.1 Simplifying the Cost Formula from TMI-GKP

For better readability, we hereby follow the notation of TMI-GKP [Zhong et al., 2022] (Equation 4), where we assume V^* represents a set of kept grouped kernels provided by a pruning strategy, where $\boldsymbol{g}^\ell$ the convolutional layer in question. The quality V^*, under our design, is deemed by:

$$\underset{\text{grouped kernel pruning}}{\text{Score}}(V^*, \boldsymbol{g}^\ell) = \sum_{s_u, s_v \in \binom{V^*}{2}} w(s_u, s_v) - \beta\big(\sum_{s_{i=1} \in V^*} w(p_i, s_i)\big). \tag{3}$$

Where $w(s_u, s_v)$ represents the Euclidean distance between grouped kernels s_u and s_v, yet s_i represents the kept grouped kernel that has the least $w(p_i, s_i)$ to a pruned grouped kernel p_i. Thus,

the former term of this equation calculates the inner distance sum of grouped kernels within strategy V^*, yet the latter term represents the outer distance between a pruned grouped kernel and its closest kept grouped kernel. Intuitively, we would like the former term to be large, as we would prefer our ideal V^* to have great diversity. Following the same idea, we would like to have the latter term small, as we want the kept kernels to cover the representation power of a certain pruned kernel. By using a $-\beta$ to connect two term, we have $V^*_{\text{best}} = \arg\max_{V^*}(\text{Score}(V^*, \boldsymbol{g}^\ell))$ for all V^*s obtained in Stage 2 (Figure 4).

Though similar, we differ from TMI-GKP [Zhong et al., 2022] (Equation 4) in two spots: first, we set $\beta = \binom{V^*}{2}/p_{\text{num}}$, where p_{num} represent the number of pruned grouped kernels in layer $\boldsymbol{g}^\ell$, balancing the two terms automatically. Secondly, for the latter term of Equation 3, we only match one pruned kernel p_i with one kept kernel s_i, instead of several of them. We made these modifications for the main purpose of removing hyperparameters. We kept the grouped kernel strategy selection stage similar to TMI-GKP, but only expanded its search scope using the Smooth Beam Greedy search (Figure 4), because prior investigation suggests this stage is sensitive to smoothness-aware mix-up operations.

D Extended Experiments and Results

D.1 Preliminary

D.1.1 Details of Experiment Setups

For all experiments on VGG-16 and ResNet-32/56/110, we aim to provide all pruning methods an around 300 epochs fine-tuning/retraining budget. Experiments conducted on ResNet-50/101 are budgeted with 100 epochs. We allow comparing methods to utilize their own training schedule and vanilla SGD optimizer setup.

SR-GKP utilizes an initial `lr` of 0.01, with a step size of 100 or 30, depending on if it is on the 300 epoch or 100 epoch schedule. `BasicBlock` ResNets with VGG use a weight decay of `5e-4`, yet `BottleNeck` ResNets use a weight decay of `1e-4`. SR-GKP strictly employs a batch size of `64` for all CIFAR-10 experiments, `128` for Tiny-ImageNet experiments, and `256` for ImageNet-1k experiments.

D.1.2 Details of Evaluation Criteria

We manually ensure all pruned models have a similar MACs/Params reduction from the identical baseline models. Then, following Li et al. [2017], we report all model checkpoints — yield during the pruning procedure — that may reach the target MACs/Params reduction. For performance (accuracy) evaluation, we test each model against FGSM [Goodfellow et al., 2014] and PGD [Madry et al., 2018] under various settings and intensities.

D.1.3 Details of Compared Methods

We provide a method overview in Table 8 to provide our readers with a more comprehensive understanding of such methods.

Table 8: Details of Compared Methods. Note "C/F/GK" under "Type" represents Channel/Filter/Grouped Kernel Pruning. Whether a method requires "Special Setup" is determined by whether it follows the most vanilla train-prune-fine-tune procedure. "Fully Pruned Epoch" reflects if given a 300 fine-tune/retrain budget, what would be the first epoch that meets the target MACs/Params reduction? "Zero-Masked?" reflects whether the pruning method can easily yield a compressed model without masking (a.k.a. hard pruning).

Method	Venue	Type	Procedure	Special Setup?	Zero-Masked?	Fully Pruned Epoch
CC [Li et al., 2021]	CVPR	C	One-shot	Y(requires data)	N	1
DHP [Li et al., 2020]	ECCV	F	Iterative (from scratch)	Y (hypernet)	Y	100
FPGM [He et al., 2019]	CVPR	F	Iterative	N	Y	1
GAL [Lin et al., 2019]	CVPR	F	Iterative	Y (GAN)	Y	close to 300
HRank [Lin et al., 2020]	CVPR	F	Iterative	N	Y	325 or 327
L1Norm [Li et al., 2016]	ICLR	F	One-shot	Y (dynamic pruning rate)	N	1
LRF [Joo et al., 2021]	AAAI	C	One-shot	Y (requires data, adding 1x1, dark knowledge)	N	1
NPPM [Gao et al., 2021]	CVPR	C	One-shot	Y (hypernet)	N	1
RAP-ADMM [Ye et al., 2019]	ICCV	F	Iterative	Y (adv. training)	Y	151
SFP [He et al., 2018]	IJCAI	F	Iterative	Y (soft pruning)	Y	300
TMI-GKP [Zhong et al., 2022]	ICLR	GK	One-shot	N	N	1
SR-GKP (Ours)	-	GK	One-shot	N	N	1

D.2 Additional Experiments

D.2.1 ResNet-32/56/110 on CIFAR-10 (in addition to Table 2)

Please refer to Table 9, Table 10, and Table 11 for details.

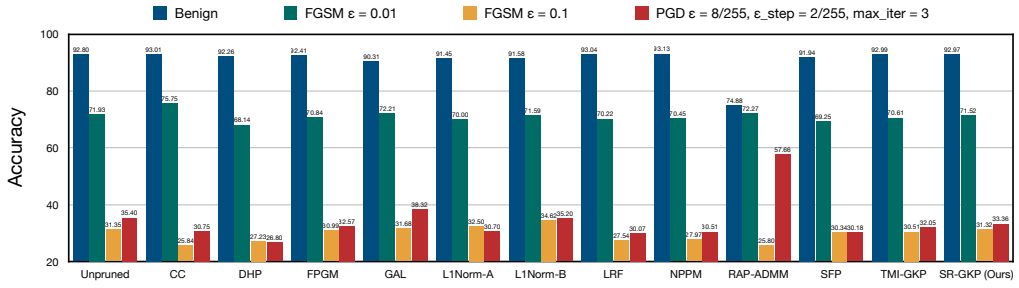

Figure 6: "Superscore" visualization of ResNet-32 on CIFAR-10.

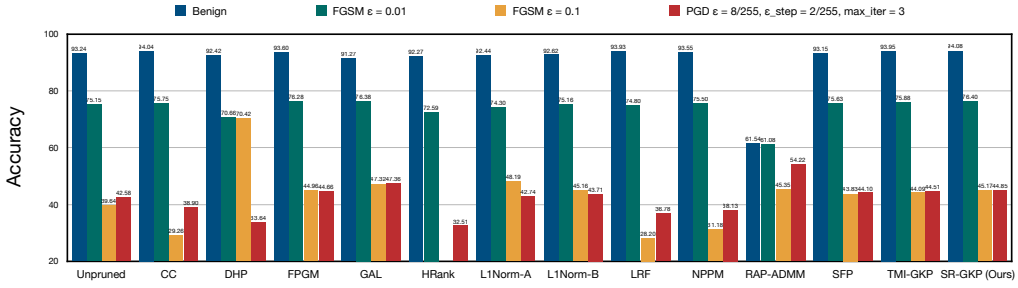

Figure 7: "Superscore" visualization of ResNet-56 on CIFAR-10.

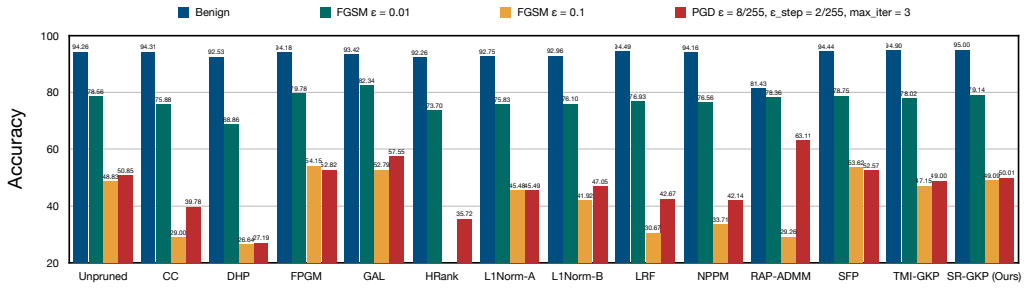

Figure 8: "Superscore" visualization of ResNet-110 on CIFAR-10.

D.2.2 VGG-16 on CIFAR-10

Please refer to Table 12 for details.

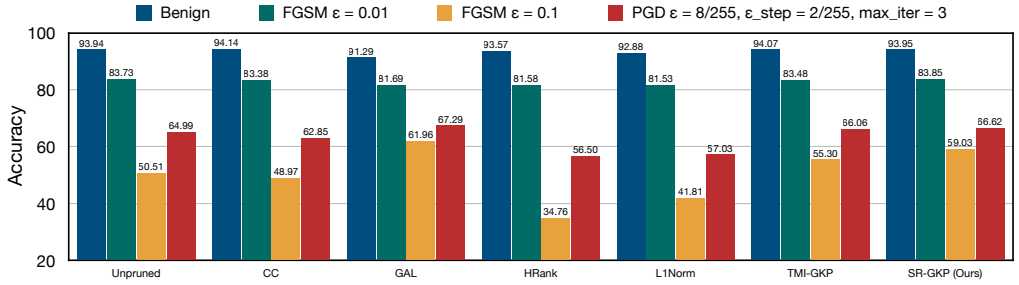

Figure 9: "Superscore" visualization of VGG-16 on CIFAR-10.

D.2.3 ResNet-56/101 on Tiny-ImageNet

Please refer to Table 13.

D.3 Ablation Studies

Please refer to Table 14, Table 15, and Table 16 for ablation studies on hyperparameter α in Equation 2. We denote it as CSB as it is in essence a "cost-smoothness balancer." It can be observed that a relatively high CSB — meaning giving more bias to the distance-based cost metrics — may yield better performance.

Table 9: Full experiments of ResNet-32 on CIFAR-10. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats the showcased MACs/Params reduction that perform best against criterion (a).

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	92.80	71.93	31.35	35.40	69.479	0.464
CC [Li et al., 2021]	Best Benign	93.01	70.06	25.26	30.40	39.261	0.312
	Best (a)	92.69	70.92	25.84	30.75		
	Best (b)	92.82	70.50	26.79	31.20		
DHP [Li et al., 2020]	Best Benign	92.26	67.25	21.35	26.12	40.091	0.283
	Best (a)	91.88	68.14	26.64	26.80		
	Best (b)	91.80	67.62	27.23	26.65		
FPGM [He et al., 2019]	Best Benign	92.41	69.75	29.61	**32.32**	39.352	0.262
	Best (a)	92.06	70.84	29.74	**32.57**		
	Best (b)	92.23	69.96	30.99	**32.43**		
GAL [Lin et al., 2019]	-	90.31	72.21	31.68	38.32	38.80	0.233
L1Norm-A [Li et al., 2016]	Best Benign	91.45	68.04	24.03	28.71	39.861	0.252
	Best (a)	91.03	70.00	25.92	30.70		
	Best (b)	90.38	67.61	**32.50**	29.67		
L1Norm-B [Li et al., 2016]	Best Benign	91.58	67.04	25.24	28.91	39.630	0.313
	Best (a)	91.03	**71.59**	29.11	35.20		
	Best (b)	90.59	69.67	**34.62**	34.02		
LRF [Joo et al., 2021]	Best Benign	**93.04**	69.38	25.82	30.07	38.791	0.260
	Best (a)	92.63	70.22	25.13	29.74		
	Best (b)	92.79	69.21	27.54	28.64		
NPPM [Gao et al., 2021]	Best Benign	**93.13**	69.88	26.03	29.75	39.605	0.326
	Best (a)	92.88	70.45	27.27	30.41		
	Best (b)	92.89	70.08	27.97	30.51		
RAP-ADMM [Ye et al., 2019]	-	74.88	72.27	25.80	**57.66**	39.370	0.271
SFP [He et al., 2018]	-	91.94	69.25	30.34	30.18	40.375	0.266
TMI-GKP [Zhong et al., 2022]	Best Benign	92.99	69.15	19.59	28.09	39.545	0.263
	Best (a)	92.03	70.61	15.85	29.44		
	Best (b)	92.77	69.90	30.51	**32.05**		
SR-GKP (Ours)	Best Benign	**92.97**	70.57	29.31	32.01	39.545	0.263
	Best (a)	92.88	**71.52**	30.39	**33.36**		
	Best (b)	92.86	70.79	**31.32**	**32.89**		

Table 10: Full experiments of ResNet-56 on CIFAR-10. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats the showcased MACs/Params reduction that perform best against criterion (a).

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	93.24	75.15	39.64	42.58	126.561	0.853
CC [Li et al., 2021]	Best Benign	94.04	74.78	29.25	37.85	69.837	0.616
	Best (a)	93.73	75.75	29.26	38.86		
	Best (b)	93.70	75.56	29.89	38.90		
DHP [Li et al., 2020]	Best Benign	92.42	69.50	29.93	32.56	73.289	0.480
	Best (a)	92.34	70.66	30.17	33.64		
	Best (b)	92.07	31.54	70.42	32.83		
FPGM [He et al., 2019]	Best Benign	93.60	75.31	43.20	43.55	71.661	0.482
	Best (a)	93.37	76.28	44.96	44.66		
	Best (b)						
GAL [Lin et al., 2019]	-	91.27	76.38	**47.32**	**47.36**	98.24	0.700
HRank [Lin et al., 2020]	Best Benign	92.27	72.32	19.11	32.51	79.237	0.584
	Best (b)						
	Best (a)	92.07	72.59	18.94	32.21		
L1Norm-A [Li et al., 2016]	Best Benign	92.44	73.00	41.00	40.76	67.995	0.487
	Best (a)	91.65	74.30	43.59	42.74		
	Best (b)	91.34	71.94	48.19	42.10		
L1Norm-B [Li et al., 2016]	Best Benign	92.62	72.97	41.30	41.79	72.115	0.586
	Best (a)	91.94	75.16	42.49	43.71		
	Best (b)	91.70	74.40	45.16	43.41		
LRF [Joo et al., 2021]	Best Benign	93.93	73.47	25.59	34.86	71.009	0.490
	Best (a)	93.68	74.80	27.39	36.78		
	Best (b)	93.63	74.06	28.20	35.98		
NPPM [Gao et al., 2021]	Best Benign	93.55	74.82	29.07	37.12	70.843	0.601
	Best (a)	93.35	75.50	30.29	38.09		
	Best (b)	93.43	75.27	31.18	38.13		
RAP-ADMM [Ye et al., 2019]	-	78.37	75.19	27.84	**61.15**	71.661	0.482
SFP [He et al., 2018]	-	93.15	75.63	43.83	44.10	71.462	0.481
TMI-GKP [Zhong et al., 2022]	Best Benign	93.95	75.18	42.18	43.46	71.855	0.482
	Best (a)	93.37	75.88	42.55	43.74		
	Best (b)	93.66	75.74	44.09	44.51		
SR-GKP (Ours)	Best Benign	**94.08**	75.89	42.60	43.85	71.855	0.482
	Best (a)	93.83	**76.40**	**45.17**	**44.85**		
	Best (b)						

Table 11: Full experiments of ResNet-110 on CIFAR-10. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats the showcased MACs/Params reduction that perform best against criterion (a).

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	94.26	78.56	48.83	50.85	254.995	1.728
CC [Li et al., 2021]	Best Benign	94.31	75.12	27.31	38.24	144.414	1.046
	Best (a)	94.17	75.88	28.14	39.78		
	Best (b)	94.23	75.52	29.00	39.64		
DHP [Li et al., 2020]	Best Benign	92.53	67.94	25.08	26.83	101.350	0.612
	Best (a)	92.21	68.86	25,51	27.19		
	Best (b)	92.25	67.82	26.64	26.67		
FPGM [He et al., 2019]	Best Benign	94.18	79.32	**52.16**	**52.46**	114.357	0.976
	Best (a)	94.02	**79.78**	**53.33**	**52.73**		
	Best (b)	94.10	79.70	**54.15**	**52.82**		
GAL [Lin et al., 2019]	-	93.42	82.34	52.79	**57.55**	180.677	1.186
HRank [Lin et al., 2020]	Best Benign	92.96	73.70	16.87	35.72	158.992	1.060
L1Norm-A [Li et al., 2016]	Best Benign	92.75	74.84	38.03	41.54	143.454	0.958
	Best (a)	91.93	75.83	39.71	44.04		
	Best (b)	92.21	75.26	45.48	45.49		
L1Norm-B [Li et al., 2016]	Best Benign	92.96	75.26	41.19	44.86	144.909	1.094
	Best (a)	92.40	76.10	41.36	47.05		
	Best (b)	91.71	73.58	41.92	42.69		
LRF [Joo et al., 2021]	Best Benign	94.49	76.60	29.50	42.15	144.405	0.997
	Best (a)	94.22	76.93	29.27	42.59		
	Best (b)	94.38	76.71	30.67	42.67		
NPPM [Gao et al., 2021]	Best Benign	94.16	76.16	32.84	41.54	146.722	1.120
	Best (a)	94.01	76.56	33.71	42.14		
	Best (b)						
RAP-ADMM [Ye et al., 2019]	-	81.43	78.36	29.26	**63.11**	144.357	0.976
SFP [He et al., 2018]	-	94.44	78.75	**53.62**	**52.57**	144.274	0.976
TMI-GKP [Zhong et al., 2022]	Best Benign	**94.90**	77.50	47.15	49.00	144.551	0.976
	Best (a)	**94.63**	78.02	45.95	48.81		
	Best (b)	**94.90**	77.50	47.15	49.00		
SR-GKP (Ours)	Best Benign	**95.00**	78.01	46.53	48.86	144.551	0.976
	Best (a)	94.69	**79.14**	47.49	**50.01**		
	Best (b)	94.60	78.92	**49.09**	**49.83**		

Table 12: Full experiments of VGG-16 on CIFAR-10. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats the showcased MACs/Params reduction that perform best against criterion (a).

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	93.94	83.73	50.51	64.99	313.433	14.728
CC [Li et al., 2021]	Best Benign	94.14	83.09	47.83	62.81	178.107	-
	Best (a)	93.88	83.38	47.70	62.74		
	Best (b)	93.97	83.14	48.97	62.85		
GAL [Lin et al., 2019]	-	91.29	81.69	61.96	67.29	203.224	7.732
HRank [Lin et al., 2020]	Best Benign	93.57	81.45	31.80	56.28	212.264	8.700
	Best (a)	93.53	81.58	34.00	56.50		
	Best (b)	93.54	81.52	34.76	56.34		
L1Norm [Li et al., 2016]	Best Benign	92.88	80.85	37.21	56.43	179.561	9.135
	Best (a)	92.41	81.53	31.10	56.17		
	Best (b)	92.06	80.48	41.81	57.03		
TMI-GKP [Zhong et al., 2022]	Best Benign	**94.07**	83.48	55.30	66.06	178.184	8.293
SR-GKP (Ours)	Best Benign	93.95	**83.49**	**56.64**	65.57	178.184	8.293
	Best (a)	93.77	**83.85**	**57.48**	**66.51**		
	Best (b)	93.86	**83.61**	**59.03**	**66.62**		

Table 13: Full experiments of ResNet-56/101 on Tiny-Imagenet. All pruning are performed on the same baseline model. "Best (a)" represents the performance of a model checkpoint that meats the showcased MACs/Params reduction that perform best against criterion (a).

Model	Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.001}$	(b) FGSM$_{\varepsilon=0.01}$	(c)FGSM$_{\varepsilon=0.1}$	(d) PGD$^{\varepsilon=4/255,\ \varepsilon_{\text{step}}=1/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
ResNet-56	Unpruned	-	55.59	53.55	28.29	8.00	15.80	506.254	0.865
	SR-GKP (Ours)	Best Benign	**54.83**	**54.14**	**29.22**	**7.71**	**17.08**	318.690	0.547
	TMI-GKP [Zhong et al., 2022]	Best Benign	51.48	50.07	27.23	7.62	15.42	318.690	0.547
ResNet-101	Unpruned	-	65.51	65.12	48.10	10.13	37.66	10081.092	42.902
	SR-GKP (Ours)	Best Benign	**67.21**	66.52	46.98	8.95	**37.34**	5721.113	24.226
		Best (a)	65.69	**64.91**	37.95	10.95	25.94		
		Best (b)	65.61	64.71	**38.27**	10.85	25.97		
		Best (c)	65.69	64.91	37.95	**10.96**	25.95		
ResNet-101	TMI-GKP [Zhong et al., 2022]	Best Benign	64.69	64.03	42.57	8.40	32.52	5721.113	24.226
		Best (a)	63.53	62.23	33.47	10.46	20.61	5721.113	24.226
		Best (b)	63.60	61.94	33.71	10.71	20.47	5721.113	24.226
		Best (c)	63.50	61.93	33.67	10.77	20.63	5721.113	24.226

Table 14: Ablation study of cost-smoothness balancer "CSB" (α in Equation 2) on ResNet-32 with CIFAR-10.

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	MACs (M)	Params (M)
Unpruned	-	92.80	71.93	31.35	69.479	0.464
0.1 CSB	Best Benign	92.79	70.39	28.09	39.545	0.263
	Best (a)	92.59	71.50	29.03		
	Best (b)	92.47	70.94	29.69		
0.25 CSB	Best Benign	93.01	70.45	27.91	39.545	0.263
	Best (a)	92.70	71.32	30.46		
	Best (b)					
0.5 CSB	Best Benign	93.07	68.74	28.07	39.545	0.263
	Best (a)	92.54	70.63	21.42		
	Best (b)	92.78	70.29	30.47		
0.75 CSB	Best Benign	92.93	70.55	30.02	39.545	0.263
	Best (a)	92.70	71.09	30.89		
	Best (b)					
0.9 CSB	Best Benign	92.97	70.57	29.31	39.545	0.263
	Best (a)	92.88	71.52	30.39		
	Best (b)	92.86	70.79	31.32		

Table 15: Ablation study of cost-smoothness balancer "CSB" (α in Equation 2) on ResNet-56 with CIFAR-10.

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	MACs (M)	Params (M)
Unpruned	-	93.24	75.15	39.64	126.561	0.853
0.1 CSB	Best Benign	93.72	75.17	39.90	71.855	0.482
	Best (a)	93.38	75.79	42.53		
	Best (b)					
0.25 CSB	Best Benign	93.83	75.68	40.40	71.855	0.482
	Best (a)	93.70	76.49	42.35		
	Best (b)	93.57	75.72	43.92		
0.5 CSB	Best Benign	93.76	75.61	41.20	71.855	0.482
	Best (a)	93.41	76.69	42.58		
	Best (b)	93.45	75.97	42.69		
0.75/0.9 CSB	Best Benign	94.08	75.89	42.60	71.855	0.482
	Best (a)	93.83	76.40	45.17		
	Best (b)					

Table 16: Ablation study of cost-smoothness balancer "CSB" (α in Equation 2) on ResNet-56 with CIFAR-110.

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	MACs (M)	Params (M)
Unpruned	-	94.26	78.56	48.83	254.995	1.728
0.1 CSB	Best Benign	94.60	78.44	44.20	144.551	0.976
	Best (a)	94.47	79.62	46.03		
	Best (b)	94.35	79.41	47.5		
0.25 CSB	Best Benign	94.50	77.86	47.07	144.551	0.976
	Best (a)	94.35	78.65	46.04		
	Best (b)	94.40	78.52	48.10		
0.5 CSB	Best Benign	95.00	78.01	46.53	144.551	0.976
	Best (a)	94.69	79.14	47.49		
	Best (b)	94.60	78.92	49.09		
0.75 CSB	Best Benign	94.49	77.97	45.93	144.551	0.976
	Best (a)	94.26	78.91	47.84		
	Best (b)	94.32	78.44	48.89		
0.9 CSB	Best Benign	94.54	78.21	47.16	144.551	0.976
	Best (a)	94.29	78.81	46.71		
	Best (b)	94.31	78.25	48.55		

Table 17: ResNet-32 on CIFAR-10 with pr = 62.5

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	92.80	71.93	31.35	35.40	69.479	0.464
CC [Li et al., 2021]	Best Benign	92.39	70.71	15.45	30.87	26.904	0.210
	Best (a)	91.83	69.99	28.84	30.91		
	Best (b)	92.01	67.97	29.07	29.29		
DHP [Li et al., 2020]	Best Benign	91.73	66.87	28.00	26.71	-	-
	Best (a)	91.36	67.71	26.83	26.89		
	Best (b)	91.31	66.84	28.75	26.39		
FPGM [He et al., 2019]	Best Benign	91.32	65.41	20.91	24.97	-	-
	Best (a)	90.47	67.77	15.95	26.61		
	Best (b)	91.04	56.51	24.47	25.68		
L1Norm-A [Li et al., 2016]	Best Benign	89.96	66.06	20.44	27.29	26.511	0.163
	Best (a)	89.52	67.65	18.07	27.93		
	Best (b)	89.23	66.75	23.21	27.88		
L1Norm-B [Li et al., 2016]	Best Benign	90.01	64.89	19.39	24.52	26.157	0.146
	Best (a)	89.78	67.14	17.75	26.91		
	Best (b)	89.42	66.66	21.63	27.55		
LRF [Joo et al., 2021]	Best Benign	92.79	68.97	22.02	27.95	29.915	0.196
	Best (a)	92.46	70.56	20.91	28.90		
	Best (b)	92.43	69.50	25.40	29.22		
NPPM [Gao et al., 2021]	Best Benign	91.92	66.83	22.23	25.63	26.998	0.198
	Best (a)	91.79	67.56	22.39	25.91		
	Best (b)	91.67	67.16	23.97	25.60		
OTOv2 (from scratch) [Chen et al., 2023]	-	90.97	66.36	17.28	27.15	-	-
OTOv2 (post train) [Chen et al., 2023]	-	92.14	70.63	28.01	31.96	-	-
SFP [He et al., 2018]	-	90.28	66.71	20.35	25.47	-	-
SR-GKP (Ours)	Best Benign	**92.21**	66.38	21.91	25.98	26.717	0.176
	Best (a)	91.52	**69.33**	14.83	27.94		
	Best (b)	92.04	66.37	**23.55**	25.80		

Table 18: ResNet-56 on CIFAR-10 with pr = 62.5

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	93.24	75.15	39.64	42.58	126.561	0.853
CC [Li et al., 2021]	Best Benign	93.57	73.63	25.30	35.40	48.692	0.421
	Best (a)	93.33	74.29	24.96	35.54		
	Best (b)	93.37	62.56	26.12	35.25		
DHP [Li et al., 2020]	Best Benign	91.66	70.66	29.75	31.40	-	-
	Best (a)	91.36	71.22	26.36	31.05		
	Best (b)	91.48	70.41	30.15	31.27		
FPGM [He et al., 2019]	Best Benign	92.64	71.80	35.17	35.17	-	-
	Best (a)	92.31	72.58	35.98	35.94		
	Best (b)	92.62	71.86	35.99	35.61		
HRank [Lin et al., 2020]	-	90.63	69.49	17.14	29.51	-	-
L1Norm-A [Li et al., 2016]	Best Benign	91.79	68.95	24.68	34.90	47.562	0.355
	Best (a)	91.12	71.76	37.01	36.97		
	Best (b)	91.47	70.10	39.83	37.03		
L1Norm-B [Li et al., 2016]	Best Benign	91.56	69.56	32.80	33.61	47.794	0.322
	Best (a)	90.66	71.24	33.22	35.09		
	Best (b)	91.07	69.25	26.19	33.66		
NPPM [Gao et al., 2021]	Best Benign	93.07	73.23	29.66	35.24	52.550	0.446
	Best (a)	92.84	74.21	28.27	35.38		
	Best (b)	93.02	72.91	30.39	34.03		
OTOv2 (from scratch) [Chen et al., 2023]	-	91.41	67.83	22.02	29.78	78.361	0.488
OTOv2 (post train) [Chen et al., 2023]	-	93.19	75.11	41.46	42.63	65.072	0.555
SFP [He et al., 2018]	-	92.24	72.21	33.65	35.39	-	-
SR-GKP (Ours)	Best Benign	**92.93**	70.94	21.15	32.01	48.409	0.323
	Best (a)	92.69	**73.54**	35.42	38.39		
	Best (b)	92.69	73.38	**36.53**	38.95		

Table 19: ResNet-110 on CIFAR-10 with pr = 62.5

Method	Criterion	Benign	(a) FGSM$_{\varepsilon=0.01}$	(b) FGSM$_{\varepsilon=0.1}$	(c) PGD$^{\varepsilon=8/255,\ \varepsilon_{\text{step}}=2/255}_{\text{max_iter}=3}$	MACs (M)	Params (M)
Unpruned	-	94.26	78.55	48.84	50.85	254.995	1.728
CC [Li et al., 2021]	Best Benign	94.29	73.77	24.50	36.32	98.582	0.727
	Best (a)	94.05	74.23	24.87	36.71		
	Best (b)	94.03	73.70	26.03	36.44		
DHP [Li et al., 2020]	Best Benign	92.73	71.39	23.19	35.51	-	-
	Best (a)	92.35	72.41	23.70	36.22		
	Best (b)	92.50	71.27	25.13	34.86		
FPGM [He et al., 2019]	Best Benign	94.11	76.11	47.62	47.54	-	-
	Best (a)	94.00	76.52	48.39	47.75		
	Best (b)	93.93	76.41	49.33	47.62		
HRank [Lin et al., 2020]	-	91.94	70.13	15.04	30.19	-	-
L1Norm-A [Li et al., 2016]	Best Benign	92.50	73.06	40.19	41.77	97.952	0.622
	Best (a) & (b)	91.51	74.99	42.62	43.45		
L1Norm-B [Li et al., 2016]	Best Benign	94.04	74.81	41.82	41.28	101.256	0.484
	Best (a)	93.79	75.55	42.36	41.99		
	Best (b)	93.86	74.96	43.43	41.99		
LRF [Joo et al., 2021]	Best Benign	94.10	75.47	20.66	39.87	94.479	0.638
	Best (a)	93.88	76.96	33.44	42.52		
	Best (b)	93.98	76.21	34.55	41.77		
NPPM [Gao et al., 2021]	Best Benign	93.93	74.71	31.37	38.81	99.915	0.746
	Best (a)	93.76	75.32	31.12	39.26		
	Best (b)	93.79	75.02	32.62	39.26		
OTOv2 (from scratch) [Chen et al., 2023]	-	91.58	71.43	22.02	29.78	78.361	0.488
OTOv2 (post train) [Chen et al., 2023]	-	93.19	75.11	41.46	42.63	65.072	0.555
SFP [He et al., 2018]	-	92.98	76.08	52.15	47.26	-	-
SR-GKP (Ours)	Best Benign	**94.31**	76.31	43.88	45.44	97.217	0.654
	Best (a)	94.17	**76.52**	43.98	46.01		
	Best (b)	94.27	76.17	**44.56**	45.74		

Table 20: Methods ranked against each other on each model with pruning rate ≈ 62.5.

Method	ResNet-32 Mean Rank	ResNet-56 Mean Rank	ResNet-110 Mean Rank	All Models Mean Rank
CC	**#1**	#3.5	#5.75	#3.42
DHP	#3.75	#7.5	#7.75	#6.33
FPGM	#4.25	#3.75	**#2**	#3.33
L1Norm-A	#5.5	#3.75	#5.75	#5.00
L1Norm-B	#6.25	#6.25	#4.25	#5.58
NPPM	#5	#4	#5.5	#4.83
SFP	#7.5	#5	#3	#5.17
SR-GKP (Ours)	#2.75 (2nd-best)	**#2.25**	**#2**	**#2.33**