Flexible Group Count Enables Hassle-Free Structured Pruning

Jiamu Zhang[*♣♦] Shaochen Zhong[*♣] Andrew Ye[♡♦] Zirui Liu[∞] Sebastian Zhao[◇]
Kaixiong Zhou[†] Li Li[♠] Soo-Hyun Choi[♠] Rui Chen[♠] Xia Hu[♣] Shuai Xu[♦]
Vipin Chaudhary[♦]
[♣]Rice University, USA, [♦]Case Western Reserve University, USA, [♠]Samsung Electronics America, USA
[♡]Stanford University, USA, [∞]University of Minnesota–Twin Cities, USA, [◇]UC Berkeley, USA, [†]North Carolina State University, USA

Abstract

Densely structured pruning methods — which generate pruned models in a fully dense format, allowing immediate compression benefits without additional demands — are evolving due to their practical significance. Traditional techniques in this domain mainly revolve around coarser granularities, such as filter pruning, thereby limiting performance due to restricted pruning freedom. Recent advancements in Grouped Kernel Pruning (GKP) have enabled the utilization of finer granularities while maintaining a densely structured format. We observe that existing GKP methods often introduce dynamic operations to different aspects of their procedures at the cost of adding complications and/or imposing limitations (e.g., requiring an expensive mixture of clustering schemes), or contain dynamic pruning rates and sizes among groups that result in a reliance on custom architecture support for its pruned models. In this work, we argue that the best practice to introduce these dynamic operations to GKP is to make `Conv2d(groups)` *(a.k.a. group count) flexible under an integral optimization, leveraging its ideal alignment with the infrastructure support of Grouped Convolution. Pursuing such a direction, we present a one-shot, post-train, data-agnostic GKP method that is more performant, adaptive, and efficient than its predecessors while simultaneously being user-friendly, with little-to-no hyper-parameter tuning or handcrafting of criteria required.*

1. Introduction

Despite having a proven track record revolving around computer vision tasks, modern convolutional neural networks (CNNs) face deployment challenges for growing model capacities. To address this issue of over-parameterization, *network pruning* — a field studying how to insightfully remove components from the original model without significant degradation to its properties and performance — has undergone constant development for being an intuitive way of potentially reducing the computation and memory footprint required to practically utilize a model [2].

In this work, we advance the progress on *Grouped Kernel Pruning (GKP)* [60], a recently developed structured pruning granularity with many deployment-friendly properties, by investigating a common design choice among existing GKP methods: **dynamic operations**, which denotes the act of applying different operations to the same task (e.g., clustering CNN filters with various combinations of dimensionality reduction and clustering techniques, as in TMI-GKP [60]). We find that current GKP designs tend to include such operations in a sub-optimal manner, resulting in various complications and limitations. As a solution, we propose that **the best approach to implementing dynamic operations to GKP is to make `Conv2d(groups)` (a.k.a. group count) flexible** under an integral optimization, leveraging its ideal alignment with the existing and future infrastructure support of *Grouped Convolution* [26]. Our empirical evaluation shows that by making these group counts flexible, we can afford to "lean down" on the rest of the typical GKP procedures, and therefore obtain a new one-shot, post-train, data-agnostic[1] GKP method that is more performant, adaptive, and efficient than its predecessors while simultaneously being user-friendly with little-to-no hyper-parameter tuning or handcrafted criteria required. We concisely summarize our contribution as enabling *"hassle-free structured pruning,"* as suggested in the title.

* Equal contribution. The work corresponds to Jiamu Zhang `<mz81@rice.edu>`. Jiamu Zhang and Andrew Ye conducted the majority of their contribution while studying at the Case Western Reserve University.

[1] As of the pruning operation is not influenced, nor is making any assumption, of the task-consisting data.

2. Background

We purposely provide a rather extensive background section given that our work develops upon specific observations made on existing adaptations of Grouped Kernel Pruning (GKP) [60]. GKP is a recently proposed and recognized structured pruning granularity with limited exposure, which means it is likely — or at least partially — foreign to many readers. **Though unconventional, we believe the extensive background supplied below would ensure a self-contained reading experience** without sending our readers to jump through multiple GKP literature with nonunified notations and visualizations. For additional information, we refer readers to Zhong et al. [60], He and Xiao [17], and Appendix 8 for more information regarding different structured pruning granularities.

2.1. Trading performance for deployability: the practical advantage of structured pruning

Within the realm of network pruning, two general categories of techniques have been delineated, which are commonly referred to as *unstructured* and *structured* pruning [2, 17, 38]. While it can be faithfully concluded that these two categories have very different focuses and approaches, there is unfortunately no universally agreed distinction between what pruning methods constitute structured pruning and what do not.

Nonetheless, the general understanding follows the notion of a performance-deployability trade-off: an unstructured pruning method typically tends to enjoy a higher degree of pruning freedom — and thus better performance — but it is done so at the cost of leaving the pruned network sparse without a reduction in size, and consequently requires special libraries or hardware support to realize their compression/acceleration benefits [53] (e.g., weight pruning [27]). Conversely, a structured pruning method often removes model components in groups that follow the architecture design of the original network, resulting in a smaller network. In particular, the majority of structured pruning methods (e.g., filter pruning [28, 63]) are capable of delivering pruned models that are reduced in dimension yet entirely dense (a.k.a. *densely structured*) and therefore provide immediate compression benefits without additional overhead.

2.2. Exploring structured pruning with finer granularities: grouped kernel pruning (GKP)

To narrow the performance gap between unstructured and structured pruning methods, many structured pruning works explore finer pruning granularities, which are often regarded as *intra-channel pruning* methods due to the two most prevalent structured pruning approaches, channel pruning and filter pruning, which derive their pruning operations from the in and out channels of the original CNN model.

However, one major issue with current intra-channel methods is that their pruned models are no longer dense and therefore lose the benefits of being densely structured, such as improving network efficiency without additional environment or hardware support [53]. We highlight this in Figure 1: it can be seen that if the naive approach of seeking finer granularities than filter/channel pruning naturally results in kernel pruning, which is intrinsically sparse. This is also the case for all *intra-kernel* pruning methods (e.g., stride pruning [1], N:M sparsity [62]), in which kernel-level sparsity is introduced. These methods might be "structured" by definition, as they indeed remove model components in groups, but they often cannot provide efficiency benefits without external support due to the levels of sparsity introduced to pruned models.

In order to achieve both an increased degree of pruning freedom and a dense post-pruned structure, a special variant of intra-channel pruning granularity – *Grouped Kernel Pruning (GKP)* [60] – has been proposed[2], in which a finer pruning granularity than filter/channel pruning was achieved without introducing sparsity by leveraging grouped convolutions [24]. We illustrate this process in Figure 2. To the best of our knowledge, GKP provides the highest degree of pruning freedom under the context of remaining densely structured, and thus attracts the interests of the pruning community [17, 42, 58, 60].

2.3. A common recipe for GKP-based methods: dynamic operations

Although GKP is still a fairly under-developed pruning granularity given its recency, we have observed a consistent pattern among recent successful works in this direction (e.g., TMI-GKP [60] and DSP [42]). Both methods introduce *dynamic operations* to different stages of its procedure and achieve significant performance improvements than methods with only deterministic operations.

As shown in Figure 3: TMI-GKP opts to include dynamic choices of clustering schemes in each of its convolutional layers. Similarly, in Figure 4, DSP makes its filter grouping and group kernel pruning stages dynamic in the sense that they may enjoy different group sizes and different in-group pruning rates for components within the same layer. While both methods deliver impressive performance, we notice that their adoption of dynamic operations results in various complications and limitations. For instance, several clustering schemes trialed in TMI-GKP can be very expensive to run. Yet, many of the produced clustering results are eventually discarded according to their ticket magnitude increase (TMI) scores. On the other hand, DSP prunes grouped kernels in different sizes, where the resul-

[2]For the sake of rigor, this granularity was in fact revisited and refined by Zhong et al. [60] at ICLR'22 and coined as *grouped kernel pruning*. The granularity itself is, of course, naturally emerged in group convolution [24] and was first proposed under a pruning context by Yu et al. [55], though unfortunately, it did gain much traction. More about this in Appendix 8.

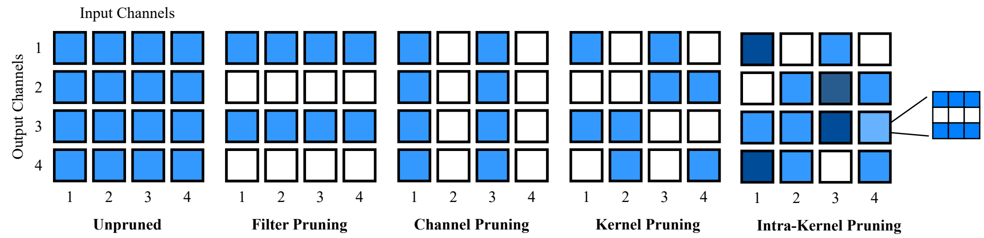

Figure 1. Different Structured Pruning Granularities

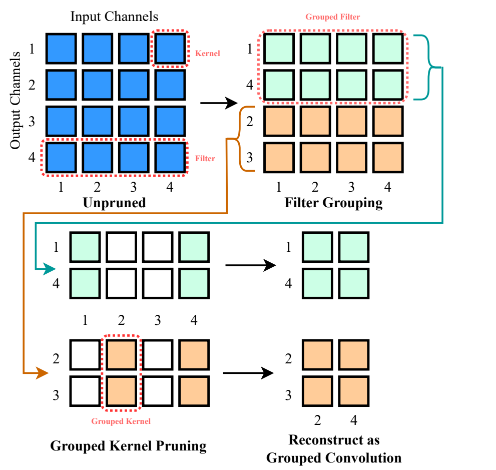

Figure 2. General Procedure of Grouped Kernel Pruning

tant pruned network is often irregularly shaped (i.e., having different dimensions of tensors within the same layer) and therefore relies on custom model definitions and convolutional operators to perform training and inference — more on this in Section 3.1.

To mitigate the complications and limitations caused by dynamic operations in existing GKP methods, we propose a new method that includes these operations in `Conv2d(groups)` (a.k.a. "group count" or "number of groups" in the grouped convolution). This means we allow each convolutional layer to take a flexible number of `groups` when grouping filters. **We argue this is the best area to integrate dynamic operations into a GKP procedure**, as this setup is directly supported by the well-adopted grouped convolution operator in modern ML frameworks and is, therefore, able to make use of existing and future infrastructure updates and support for grouped convolutions. Empirical evaluations also support the effectiveness of our approach.

Moreover, **after employing a flexible group count, we can simultaneously reduce the complexity and dependency of the rest of the GKP procedure and drastically improve the efficiency and usability of our method.** As an example, we utilize only one simple clustering operation rather than selecting one of the many TMI-score-dependent clustering schemes in Zhong et al. [60], thus removing dependencies on training snapshots or checkpoints of the original unpruned model. This is a meaningful trait, given the prevalent utilization of pretrained models in practical pruning. We name our method LeanFlex-GKP, emphasizing that it is a GKP method that is more "leaned down" than others by utilizing flexible group counts as its primary mechanism.

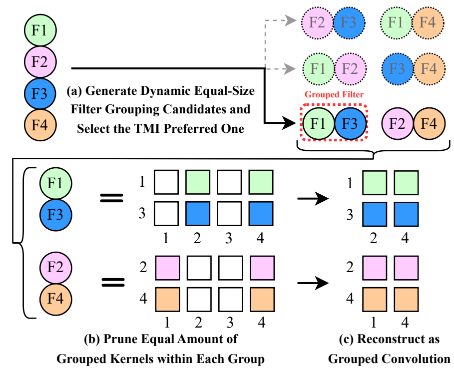

Figure 3. Procedure of TMI-GKP

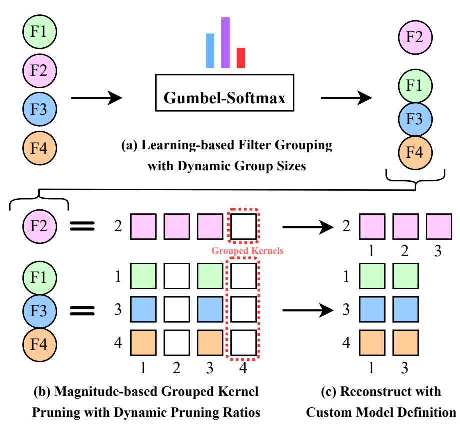

Figure 4. Procedure of Dynamic Structure Pruning (DSP)

We summarize the traits of our proposed method and the contributions of our work as follows:

- **Advancing the progress of GKP by identifying and solving a common pain point: dynamic operations**. We recognize the significance of dynamic operations to GKP, as well as the challenges of integrating them into current procedures. By utilizing flexible group counts as a medium, we tactfully introduce such operations to our procedure while avoiding the complications and limitations typically found in other GKP methods. Extensive empirical evaluation supports the effectiveness of our method.

- **Providing an efficient, hassle-free experience.** By reducing the complexity of various stages in the typical GKP procedure, our method provides a significant advantage in terms of efficiency and adaptability over others. LeanFlex-GKP is a post-train, one-shot, data-agnostic procedure with little-to-no hyper-parameter tuning or setting handcrafting required, making it one of the most usable structured pruning methods available.
- **Guiding future developments of GKP**. Aside from our proposed method itself, our work also contains the most comprehensive empirical evaluation and ablation studies on GKP to date. Given that GKP is an underdeveloped pruning granularity with many attractive properties, we believe our investigation provides valuable insights and guidance to future scholars working to adopt GKP and its variants.

3. Motivation

3.1. Flexible group counts as the dynamic operation in GKP

As mentioned in Section 2.3, the involvement of dynamic operations plays a significant role to the GKP procedure. Yet, current methods tend to adopt dynamic operations at the cost of adding significant complications or limitations. Take, for instance, TMI-GKP [60] and Dynamic Structure Pruning (DSP) [42]: TMI-GKP trials different *clustering schemes*[3] at its filter grouping stage per each convolutional layer of the unpruned model, forming a dynamic choice of clustering schemes across the depth of the pruned model. DSP, on the other hand, allows for dynamic group sizes and in-group pruning ratios in formed filter groups and thus enjoys a higher degree of pruning freedom than TMI-GKP.

While both methods demonstrate performance advantages over GKP methods with purely deterministic operations (e.g., KPGP by Zhang et al. [58]), the addition of such dynamic operations also comes with its own respective costs.

In the context of TMI-GKP, certain *clustering schemes*, which consist of combining a dimensionality reduction technique with a clustering algorithm like k-PCA + k-Means, may incur significant computational costs. For example, k-PCA — one of the candidate dimensionality reduction techniques utilized in TMI-GKP — requires an eigen decomposition of a convolutional layer's weight tensor, which is an expensive procedure requiring a complexity more than $\mathcal{O}(n^3)$ for a $n \times n$ matrix [41]. Yet, all produced clustering results except one are discarded if they result in a lower ticket magnitude increase (TMI) score: a weight-shift related metric inspired by the series of works on the *lottery ticket hypothesis* [9]. This makes the use of TMI-GKP challenging should the width of the target network become large, as outlined in Table 7 (TMI-GKP is unable to prune the WideResNet model within a reasonable time constraint).

In DSP, dynamic behavior is present in both the filter grouping and grouped kernel pruning stages, where the learned filter groups are allowed to be in different sizes. Each filter group can opt to remove a different amount of grouped kernels, resulting in a pruning granularity that is finer than typical equal-group-equal-pruning-ratio GKP methods [55, 58, 60]. However, due to the pruned network having different tensor shapes within the same layer, it can no longer be reconstructed into a grouped convolution format and instead relies on custom-defined model definitions and operators, ultimately diminishing its practical adaptability.

In this work, we integrate dynamic operations on `Conv2D(groups)` (also commonly known as "group count" or "number of groups" under a grouped convolution context). To achieve this, we group convolutions with different `groups` settings across model layers. We place emphasis on the fact that this setup is supported by the grouped convolution operator, and is therefore able to take advantage of existing and future infrastructure updates and support systems. Our setup improves upon and differs from the two most successful current methods in GKP: TMI-GKP and DSP. In the former, a hard-coded `groups=8` is applied for all models and layers without consideration of subsequent pruning schematics. We reveal that such an approach to be sub-optimal in our ablation studies in Section 10. Furthermore, our setup also differs from DSP, as the end group results still remain equally-sized and contain an identical pruning ratio among groups, thus allowing for quick and efficient implemention without custom support.

3.2. Leaning out for an efficient GKP procedure

Given the effectiveness of utilizing flexible group counts, we can afford to reduce the complexity of previous GKP procedures. Instead of trialing different cluster schemes or employing learn-based regularization procedures, we simply utilize a k-Means^{++} inspired clustering procedure to determine grouping, which drastically decreases the complexity and dependency requirements of filter grouping (Section 4.2).

During the grouped kernel pruning stage, methods like TMI-GKP formalize the procedure as a graph search problem solved with a multiple-restart greedy procedure, showcasing a significant performance advantage over vanilla magnitudes or distance-based alternatives [58]. However, we decide instead to use a tactfully designed distance and magnitude-based heuristic to achieve similar, if not bet-

[3] Where each *clustering scheme* consists of different combinations of various dimensionality reductions and clustering techniques.

ter, accuracy retention rates to the unpruned models (Section 4.3). Our replacement of this procedure significantly reduces the runtime of our pruning procedure (as clocked in Table 7) and improves its general usability.

3.3. Towards a hassle-free experience

Although the post-prune performance and efficiency of pruning procedures are certainly reasonable criteria when evaluating a method under a practical context, **usability across a broader scenario and being user-friendly are another vital set of factors to consider**. In fact, some of the most widely adopted pruning methods do not necessarily offer the best performance or the fastest runtime, but are often extremely user-friendly as they can be run and deployed with minimal adjustments. Two examples of such work are OTOv2 [3] and DepGraph [8], which are architecture-agnostic methods capable of pruning any model, with OTOv2 capable of pruning from scratch.

Our method, LeanFlex-GKP, being a GKP method limited to CNNs, is not at the same level of generalization as OTOv2 or DepGraph. Still, we strive to maximize its usability under such constraints by making it a post-train, one-shot, data-agnostic pruning method with standard fine-tuning procedures. As long as one has access to the weights of the CNN model and fine-tuning data, they may utilize our pruning method to prune their model and fine-tune via standard Stochastic Gradient Descent without further interference. In comparison, previous GKP methods like TMI-GKP require access to the training snapshots/checkpoints of the original unpruned model, and iterative GKP methods like DSP require regularization learning and pruning operations during the fine-tuning/retraining procedure.

On the note of user-friendliness, our method has little-to-no hyperparameters or handcrafted settings, reducing the requirement of human and resource efforts for trial-and-error testing different settings. Furthermore, **the user of our method can reliably predict the pruned model size and computation requirement by simply multiplying the pruning rate by the original unpruned model**, making the our procedure standardized and predictable. Surprisingly, this is a useful property lacking in many modern pruning methods, such as Lin et al. [32, 33], Park et al. [42] and Chen et al. [3], where the user will typically need to trial-and-error various hyperparameter combinations to achieve a certain accuracy in pruning reduction. The importance of being able to reliably prune model to a specific size cannot be overstated in a practical context, as the alternative will require massive computation or manual effort to search for the suitable hyperparameter setting. In some cases, such an endeavor might even be impossible.

4. Proposed method

Our proposed method, LeanFlex-GKP, consists of a four-stage procedure:

1. **Filter grouping**: where we group filters within a certain convolution layer into n equal-sized filter groups according to their distance towards k-Means^{++} determined centers (Figure 5).
2. **Group kernel pruning:** where we prune a certain amount of grouped kernels out of all filter groups within the same layer. The pruning is determined by each grouped kernel's L_2 norm and distance to their geometric median (Figure 6).
3. **Post-prune group count evaluation:** where we evaluate all grouping and pruning strategies obtained under different group count settings and then select the one where the preserved group kernels have the maximum inter-group distance and the minimum intra-group distance (Figure 7).
4. **Grouped convolution reconstruction:** where we convert the pruned model to a grouped convolution format, just like we showcased in the standard GKP procedure (Figure 2).

In general, we aim to develop lightweight and dependency-free measures to at each stage of the GKP process. We walk our readers through the technicalities of our method, as well as demonstrate that a SOTA-capable GKP method with many novel and favorable properties results by discerningly combining basic tools and leveraging the power of flexible group counts.

4.1. Preliminaries

Suppose there is a convolutional neural network model $\mathbf{W}$ with L convolutional layers, then the layer with index l is denoted as $\mathbf{W}^l$. A layer can be viewed as a 4D tensor $\mathbf{W}^l \in \mathbb{R}^{C_{\text{out}}^l \times C_{\text{in}}^l \times H^l \times W^l}$, in which C_{in}^l is the number input channels on layer l (number of kernels in a filter), C_{out}^l is the number output channels on layer l (number of filters in a layer), and $H^l \times W^l$ is the kernel size. The task to perform a grouped convolution reconstruction upon $\mathbf{W}^l$, as illustrated in Figure 2, can be described as converting $\mathbf{W}^l$ to a $\mathbf{G}^l \in \mathbb{R}^{n \times C_{\text{in}}^l \times m \times H^l \times W^l}$, where n stands for the group count setting of this conversion, and $m = C_{\text{out}}^l / n$ representing the group size.

4.2. KPP-aware filter grouping

The general goal of filter grouping is to cluster filters that are similar to each other within the same group, so that when such filters are partially removed during the pruning process, leftover components can cover the representation power of their removed counterparts. In previous works like TMI-GKP [60] and DSP [42], this procedure is rather resource-intensive, with TMI-GKP trialing expensive clustering schemes under the guidance of its TMI score, and DSP employing a learning-based procedure.

In order to streamline the grouping process and to mitigate complexity, we devise a cost-effective filter cluster-

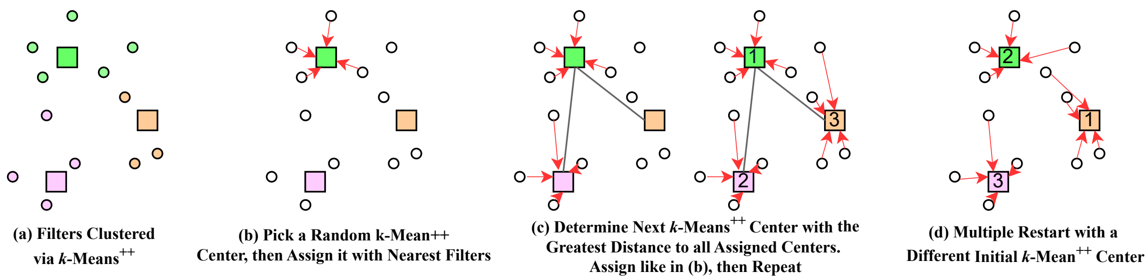

Figure 5. Visualization of the LeanFlex-GKP KPP-Aware Filter Grouping Procedure. We first cluster filters (the circles) via k-Means^{++} (KPP) into n groups with no constraint on having an equal group size to determine clustering centers (the squares), as in (a). Then, our operation can be viewed as a cycle between assigning m nearest filters into a KPP center to form a filter group, then finding the next KPP center to do subsequent filter assignments, as in (b) → (c); until n filter groups are formed (the first KPP center is picked at random). Last, we conduct a multiple restart and repeat the (b) ↔ (c) center-finding-filter-assignments, as showcased in (d). After all multiple restarts, we are left with n candidate filter grouping strategies, and select the strategy that has filters with the least intra-group distance to their respective KPP centers (having less summed length on red arrows).

ing algorithm based on the clustering centers obtained by k-Means^{++} (KPP). In contrast to a direct utilization of KPP cluster assignments, our approach exclusively leverages clustering centers, and is reinforced by two greedy strategies. Our procedure is illustrated in Figure 5. We denote n to be the group count and $m = C^l_{\text{o tut}}/n$ to be the group size (number of filters within each filter group). In this particular visualization, we have $n = 3$ and $m = 4$. We demonstrate the efficiency and performance advantage of our method with wall-clock results in Table 7 and accuracy results in Table 2, support our claims made in Section 3.2 and Section 3.1.

4.3. L_2 & geometric median-based GKP

Previous methods like TMI-GKP frame the problem of grouped kernel selection as a graph search problem, and utilized a greedy procedure with multiple restarts. While this procedure is generally efficient, it is still time and resource-consuming given a layer with a large amount of `in_channels`. Thus, inspired by the toolsets proposed in FPGM [20], we devise a simple combination utilizing the L_2 norm and Geometric Median-based distance to form a lightning-fast pruning procedure, as illustrated in Figure 6. We demonstrate the efficacy of our method with Table 7 (as mentioned in Section 3.2).

4.4. Post-prune group count evaluation as integral optimization

One primary motivation for our work is that our method makes use of flexible group counts under a GKP procedure. However, it is intrinsically challenging to evaluate clustering quality under different group counts (e.g., previously suggested metrics like a Silhouette score [60] have little bearing in a network pruning context). Thus, we opt to employ an additional Geometric Median-based evaluation similar to that in Section 4.3. We illustrate this process in Figure 7 and provide **a walk-through of the complete LeanFlex-GKP procedure in pseudocode as Algorithm 1**. Given that each group count evaluation is con-

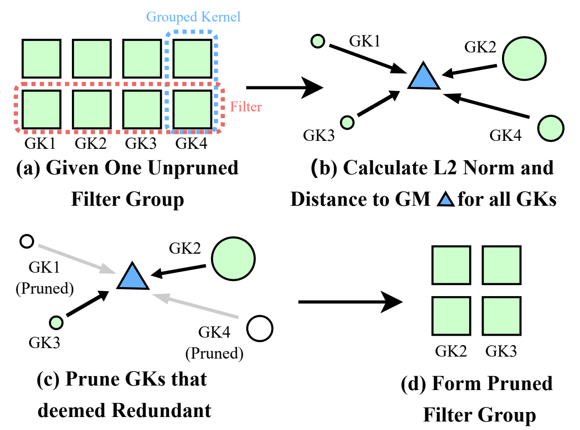

Figure 6. Visualization of LeanFlex-GKP L_2 & Geometric Median-based Grouped Kernel Pruning Procedure. Given an unpruned filter group as in (a), we first calculate the Geometric Median (GM) of its Grouped Kernels (GKs), as well as each GK's distance to the GM and their L_2 norm. These distances and the L_2 norm are visualized in (b) as the length of black arrows and the area of green circles, respectively. The GKs with large L_2 norms and small distances to their GMs are preserved and eventually reconstructed to the grouped convolution format, as shown (c) to (d).

ducted on a pruned convolutional layer (after being grouped with different `Conv2d(groups)`), our method makes integral connections between the (originally independent) filter grouping and grouped kernel pruning stage. Ablation studies in Table 4 confirm the advantage of this integral optimization design over other alternative setups.

5. Experiments

Experiment Coverage We extensively evaluate the effectiveness of our method against 25 other densely structured pruning methods (Table 9) on architectures including BasicBlock (20/32/56/110) and BottleNeck ResNets (50/101) [16], VGG11/13/16 [46], DenseNet40 [22], MobileNetV2 [45], and WideResNet [56]. The datasets used include CIFAR10/100 [25], Tiny-ImageNet [52], and ImageNet-1k

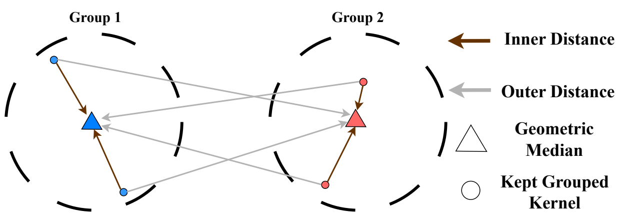

Figure 7. Visualization of LeanFlex-GKP Group Count Evaluation. We first compute the GM among retained grouped kernels and then calculate the inner and outer distance among them. After a normalization w.r.t. the group count, the one with the highest average (Outer Distance − Inner Distance) is chosen; please refer to Appendix 9.2 for details.

[4]. Please refer to Section 11 for full details on experiment settings.

We provide an abbreviated version of our experiments in Table 1. For each model-dataset combination listed in Table 1, we select the five most competitive structural pruning methods based on their performance in final accuracy and accuracy gain, and compare them with our method. **We refer our readers to Table 13 Table 21 in Section 11 for full experiment results, and Table 10 for accuracy gap between competitive methods and our method**, where we compare against 25 different structured pruning methods illustrated in Table 9 and evaluate our methods under 21 different settings specified in Table 5. We also provide a series of ablation studies in Section 10 to facilitate an anatomical understanding of our proposed method. Additionally, we apply our method to the UNet of SDXL-Base-1.0 for the image generation task (see Table 12).

Report Digestion For all experiment results reported like Table 1, **DA** represents if the method is data-agnostic (pruning can be done without access to data), **IP** indicates if a method is considered an iterative pruning method (utilizing a train-prune cycle), and **RB** reports recovery budget (in terms of epochs). All other reported criteria are in terms of %. **BA** and **Pruned** respectively report the unpruned (baseline) accuracy and the pruned accuracy. Methods marked with * are drawn from their original or (third-party) replicated publication; the rest are replicated by us to ensure a fair comparison (often with an identical baseline). Generally speaking, a method that is **DA** ✓, **IP** ✗, and demands a smaller **RB** is likely to be more user-friendly. ↓ **MACs** and ↓ **Params** represent the drop of MACs/Params after pruning (percentage of total components pruned).

In the most ideal setup, every pruning method should be evaluated against an identical unpruned baseline with identical MACs/Params drop. But practically, this is often impossible due to various technical or practical challenges (e.g., inaccessible baselines, lack of support of certain pruning ratio [32], different pruning granularity and targets [60], potential addition of architecture tweak [11]), and readers are expected to compare the ΔAcc readings when methods are pruning away similar amounts of MACs/Params upon baselines with similar accuracies. We authors understand the importance of evaluation alignment, where we have the majority of our reported experiments replicated under a fair pipeline to ensure aligned **BA** and **RB**, as well as a comparable (or overpruned to our disadvantage) ↓ **MACs/Params**. **To the best of our knowledge, few, if not none of the CNN structured pruning works outside ours have paid efforts in enforcing this alignment despite its importance.**

Result Discussion We believe it is fair to conclude that our proposed method showcases SOTA-competitive (if not beyond) performance across comprehensive combinations of models and datasets. Out of all 21 reported results of LeanFlex-GKP, 18 of them showcased accuracy improvements after pruning (yet, no other compared method is able to provide positive ΔAcc under the three exception setups), suggesting our pruning method actually helps on the generalization of the model given a reasonable setup. We also note the compute (MACs) and memory (Params) **reduction of our pruned models are almost always within 1% of their assigned pruning rates (e.g., see Table 19 and Table 21), which is a useful characteristic not found in many compared methods**[4]. This supports one of the hassle-free claims we made in Section 3.3. Additionally, we would like to mention the combinations of BasicBlock ResNets with CIFAR10 — though being some of the most commonly evaluated combinations [2] — are potentially getting saturated, as methods with significant performance gaps on more difficult model-dataset combinations tend to show little difference upon BasicBlock ResNets and CIFAR10. Further, it is worth noting that **our method exhibits significant efficiency advantage compared to methods with comparable accuracy performance**, like TMI-GKP [60] and NPPM [11], and this advantage becomes particularly pronounced as the size of the model is enlarged (Table 7, Table 10). Given we purposely showcase the most competitive methods in Table 1 of the main text, sometimes the accuracy gap can be less than ideal. But we note that this is the by-product of faithful and comprehensive reporting. A closer inspection reveals no single method is able to keep up with our LeanFlex-GKP across all featured tasks and settings as indicated in Table 10; let alone the many hassle-free features — which are often more of a deal breaker under practical scenarios.

6. Conclusion

Our work serves as a more performant, efficient, and user-friendly advancement to the *grouped kernel pruning* granularity and can be of particular interest to both scholars of the pruning community and end users with practical needs.

[4]This is evidenced by the many not-perfectly-aligned results in Table 19 and Table 21, where we tried to make all methods without the * mark — meaning we replicated such runs under our controlled pipeline — aligned with the pruning rate in caption, but failed to do so in multiple scenarios.

Table 1. **ABBREVIATED** Experiment Results. Results in **bold red** indicate being the second best among comparisons. Please refer to Section 5 for header definitions. We note that comparative methods showcased here are among the strongest methods we featured, and we feature rather comprehensively (see Table 9).

Method	DA	IP	RB	BA	Pruned	ΔAcc	↓ MACs	↓ Params
VGG16 on CIFAR10 MACs ≈ 313.4M Params ≈ 14.7M								
CC [30]	✗	✗	300	93.94	94.14	↑ 0.20	43.18	-
HRank [32]	✗	✓	300	93.94	93.57	↓ 0.37	32.28	40.82
L1Norm [28]	✓	✗	300	93.94	92.88	↓ 1.06	42.71	37.85
KPGP* [57]	✓	✗	300	94.27	94.17	↓ 0.13	43.15	43.59
TMI-GKP [60]	✓	✗	300	93.94	94.07	↑ 0.10	43.15	43.59
LeanFlex-GKP (ours)	✓	✗	300	93.94	**94.15**	↑ **0.21**	43.15	43.59
ResNet32 on CIFAR10 MACs ≈ 69.5M Params ≈ 0.46M								
CC [30]	✗	✗	300	92.80	92.39	↓ 0.41	61.29	54.35
NPPM [11]	✗	✗	300	92.80	91.92	↓ 0.88	61.15	56.52
L1Norm-B [28]	✓	✗	300	92.80	90.01	↓ 2.79	62.36	67.39
SFP [18]	✗	✓	300	92.80	90.28	↓ 2.52	59.74	60.65
FPGM [20]	✗	✓	300	92.80	91.32	↓ 1.48	58.28	59.57
LeanFlex-GKP (ours)	✓	✗	300	92.80	**92.40**	↓ **0.40**	61.56	61.74
ResNet110 on CIFAR10 MACs ≈ 255.0M Params ≈ 1.73M								
ChipNet* [48]	✗	✓	300	93.98	93.78	↓ 0.20	62.41	-
CC [30]	✗	✗	300	94.26	94.29	↑ 0.03	61.34	58.38
FPGM [20]	✗	✓	300	94.26	94.11	↓ 0.15	58.35	60.17
LRF [23]	✗	✗	300	94.26	94.10	↓ 0.16	62.94	63.12
L1Norm-B [28]	✓	✗	300	94.26	94.04	↓ 0.22	60.29	72.25
LeanFlex-GKP (ours)	✓	✗	300	94.26	**94.35**	↑ **0.09**	64.22	62.19
ResNet56 on CIFAR100 MACs ≈ 69.5M Params ≈ 0.46M								
TMI-GKP [60]	✓	✗	300	70.85	71.11	↑ 0.26	43.22	43.19
CC [30]	✗	✗	300	71.53	71.43	↓ 0.10	43.52	28.52
SFP [18]	✗	✓	300	71.53	69.80	↓ 1.73	44.29	44.82
NPPM [11]	✗	✗	300	71.53	71.57	↑ 0.04	33.54	13.04
FPGM [20]	✗	✓	300	71.53	69.48	↓ 2.05	43.38	43.19
LeanFlex-GKP (ours)	✓	✗	300	71.53	**72.11**	↑ **0.58**	43.22	43.18
ResNet110 on CIFAR100 MACs ≈ 255.001M Params ≈ 1.734M								
TMI-GKP [60]	✓	✗	300	72.99	72.79	↓ 0.20	43.31	43.37
NPPM [11]	✗	✗	300	73.20	72.38	↓ 0.82	42.77	18.69
L1Norm-A [28]	✓	✗	300	73.20	69.85	↓ 3.35	43.74	44.41
CC [30]	✗	✗	300	73.20	73.21	↑ 0.01	43.43	19.78
LRF [23]	✗	✗	300	73.20	73.58	↑ 0.38	43.38	42.16
LeanFlex-GKP (ours)	✓	✗	300	73.20	**73.63**	↑ **0.43**	43.31	43.36
ResNet56 on Tiny-ImageNet MACs ≈ 506.254M Params ≈ 0.865M								
TMI-GKP [60]	✓	✗	300	56.13	55.52	↓ 0.61	37.05	36.76
L1Norm-A [28]	✓	✗	300	56.13	55.41	↓ 0.72	35.51	32.14
L1Norm-B [28]	✓	✗	300	56.13	55.21	↓ 0.92	36.43	41.04
HRank [32]	✗	✓	300	56.13	54.16	↓ 1.97	37.39	30.98
LRF [23]	✗	✗	300	56.13	**55.95**	↓ **0.18**	35.90	34.68
LeanFlex-GKP (ours)	✓	✗	300	56.13	**55.67**	↓ **0.46**	37.05	36.76
ResNet50 on ImageNet-1K MACs ≈ 4122.828M Params ≈ 25.557M								
TMI-GKP* [60]	✓	✗	100	76.15	75.53	↓ 0.62	33.21	33.74
ThiNet* [37]	✗	✓	100	72.88	72.04	↓ 0.84	36.7	-
OTOv2* post-train) [3]	✗	✓	120	76.13	75.38	↓ 0.75	37.70	26.58
FPGM* [20]	✗	✓	100	76.13	75.04	↓ 1.09	35.93	28.36
KPGP* [57]	✓	✗		76.15	75.50	↓ 0.65	33.70	33.20
LeanFlex-GKP (ours)	✓	✗	100	76.13	**75.62**	↓ **0.51**	33.06	30.34

Acknowledgments

This research was partially supported by NSF Awards ITE-2429680, IIS-2310260, OAC-2112606, and OAC-2117439. Furthermore, this work was supported by the US Department of Transportation (USDOT) Tier-1 University Transportation Center (UTC) Transportation Cybersecurity Center for Advanced Research and Education (CYBER-CARE) grant #69A3552348332.

Further, this work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University (CWRU). We give special thanks to the CWRU HPC team for their prompt and professional help and maintenance. The views and conclusions in this paper are those of the authors and do not represent the views of any funding or supporting agencies.

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017. 2, 1

[2] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2: 129–146, 2020. 1, 2, 7

[3] Tianyi Chen, Luming Liang, DING Tianyu, Zhihui Zhu, and Ilya Zharkov. Otov2: Automatic, generic, user-friendly. In *International Conference on Learning Representations*, 2023. 5, 8, 1, 9, 11, 15

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 7, 8

[5] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search, 2019. 1

[6] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5840–5848, 2017. 1

[7] Sara Elkerdawy, Mostafa Elhoushi, Abhineet Singh, Hong Zhang, and Nilanjan Ray. One-shot layer-wise accuracy approximation for layer pruning. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2940–2944. IEEE, 2020. 9, 11

[8] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023. 5, 1, 9, 15

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 4

[10] Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1899–1908, 2020. 1, 9, 12

[11] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9270–9280, 2021. 7, 8, 9, 10, 11, 13, 14, 15

[12] Jinyang Guo, Wanli Ouyang, and Dong Xu. Multi-dimensional pruning: A unified framework for model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1508–1517, 2020. 1, 9, 12

[13] Qingbei Guo, X. Wu, J. Kittler, and Zhi quan Feng. Self-grouping convolutional neural networks. *Neural networks : the official journal of the International Neural Network Society*, 132:491–505, 2020. 2

[14] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1539–1547, 2020. 1

[15] Yi Guo, Huan Yuan, Jianchao Tan, Zhangyang Wang, Sen Yang, and Ji Liu. Gdp: Stabilized neural network pruning via gates with differentiable polarization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5239–5250, 2021. 1, 9, 12

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6, 8

[17] Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey, 2023. 2, 1

[18] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks, 2018. 8, 1, 9, 10, 11, 13, 14, 15

[19] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 9, 13

[20] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration, 2019. 6, 8, 1, 9, 11, 13, 14, 15

[21] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 10

[22] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 8

[23] Donggyu Joo, Eojindl Yi, Sunghyun Baek, and Junmo Kim. Linearly replaceable filters for deep network channel pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8021–8029, 2021. 8, 1, 9, 10, 11, 13, 14, 15

[24] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. 2

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 8

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. 1, 2

[27] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1989. 2

[28] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017. 2, 8, 1, 9, 11, 12, 13, 14, 15

[29] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks, 2020. 9, 11, 13, 14, 15

[30] Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression, 2021. 8, 9, 10, 11, 12, 13, 14, 15

[31] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020.

[32] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map, 2020. 5, 7, 8, 9, 11, 12, 13, 14, 15

[33] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5, 1, 9, 15

[34] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning, 2019. 12

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 10

[36] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 1

[37] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 8, 1, 9, 11

[38] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J. Dally. Exploring the granularity of sparsity in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 2

[39] Tanay Narshana, Chaitanya Murti, and Chiranjib Bhattacharyya. DFPC: Data flow driven pruning of coupled channels without data. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[40] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde

de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. 10

[41] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *STOC*, pages 507–516. ACM, 1999. 4

[42] Jun-Hyung Park, Yeachan Kim, Junho Kim, Joon-Young Choi, and SangKeun Lee. Dynamic structure pruning for compressing cnns. *ArXiv*, abs/2303.09736, 2023. 2, 4, 5

[43] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 10

[44] Sourjya Roy, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Anand Raghunathan. Pruning filters while training for efficiently optimizing deep learning networks. *CoRR*, abs/2003.02800, 2020. 1

[45] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018. 6, 8

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 6, 8

[47] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33:10936–10947, 2020. 1, 9, 12

[48] Rishabh Tiwari, Udbhav Bamba, Arnav Chavan, and Deepak K Gupta. Chipnet: Budget-aware pruning with heaviside continuous approximations. *arXiv preprint arXiv:2102.07156*, 2021. 8, 9, 12, 13

[49] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis, 2019.

[50] Wenxiao Wang, Minghao Chen, Shuai Zhao, Long Chen, Jinming Hu, Haifeng Liu, Deng Cai, Xiaofei He, and Wei Liu. Accelerate cnns from three dimensions: A comprehensive pruning framework. In *International Conference on Machine Learning*, pages 10717–10726. PMLR, 2021.

[51] Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 12273–12280. AAAI Press, 2020. 1

[52] Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imagenet challenge. *Technical report*, 2017. 6, 8

[53] Carl Yang, Aydın Buluç, and John D Owens. Design principles for sparse matrix multiplication on the gpu. In *European Conference on Parallel Processing*, pages 672–687. Springer, 2018. 2

[54] Yinan Yang, Yu Wang, Ying Ji, Heng Qi, and Jien Kato. One-shot network pruning at initialization with discriminative image patches. *arXiv preprint arXiv:2209.05683*, 2022. 1, 9, 11

[55] Niange Yu, Shi Qiu, Xiaolin Hu, and Jianmin Li. Accelerating convolutional neural networks by group-wise 2d-filter pruning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2502–2509, 2017. 2, 4

[56] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017. 6, 8

[57] Guanqun Zhang, Shuai Xu, Jing Li, and Alan JX Guo. Group-based network pruning via nonlinear relationship between convolution filters. *Applied Intelligence*, 52(8):9274–9288, 2022. 8, 1, 9, 11, 12, 15

[58] Guanqun Zhang, Shuai Xu, Jing Li, and Alan J. X. Guo. Group-based network pruning via nonlinear relationship between convolution filters. *Applied Intelligence*, 2022. 2, 4

[59] Kaiqi Zhao, Animesh Jain, and Ming Zhao. Automatic attention pruning: Improving and automating model pruning using attentions, 2023. 1, 9, 13

[60] Shaochen Zhong, Guanqun Zhang, Ningjia Huang, and Shuai Xu. Revisit kernel pruning with lottery regulated grouped convolutions. In *International Conference on Learning Representations*, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15

[61] Shaochen Henry Zhong, Zaichuan You, Jiamu Zhang, Sebastian Zhao, Zachary LeClaire, Zirui Liu, Daochen Zha, Vipin Chaudhary, Shuai Xu, and Xia Hu. One less reason for filter pruning: Gaining free adversarial robustness with structured grouped kernel pruning. *Advances in Neural Information Processing Systems*, 36, 2024. 11

[62] Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from

scratch. In *International Conference on Learning Representations*, 2021. 2

[63] Hao Zhou, José Manuel Álvarez, and Fatih Murat Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, 2016. 2

[64] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems*, 31, 2018. 1, 9, 12