CHEX: CHannel EXploration for CNN Model Compression

Zejiang Hou[1*] Minghai Qin[2] Fei Sun[2] Xiaolong Ma[3†] Kun Yuan[2] Yi Xu[4‡]
Yen-Kuang Chen[2] Rong Jin[2] Yuan Xie[2] Sun-Yuan Kung[1]
[1]Princeton University [2]Alibaba Group [3]Northeastern University [4]Dalian University of Technology

Abstract

Channel pruning has been broadly recognized as an effective technique to reduce the computation and memory cost of deep convolutional neural networks. However, conventional pruning methods have limitations in that: they are restricted to pruning process only, and they require a fully pre-trained large model. Such limitations may lead to sub-optimal model quality as well as excessive memory and training cost. In this paper, we propose a novel Channel Exploration methodology, dubbed as CHEX, to rectify these problems. As opposed to pruning-only strategy, we propose to repeatedly prune and regrow the channels throughout the training process, which reduces the risk of pruning important channels prematurely. More exactly: From intra-layer's aspect, we tackle the channel pruning problem via a well-known column subset selection (CSS) formulation. From inter-layer's aspect, our regrowing stages open a path for dynamically re-allocating the number of channels across all the layers under a global channel sparsity constraint . In addition, all the exploration process is done in a single training from scratch without the need of a pre-trained large model. Experimental results demonstrate that CHEX can effectively reduce the FLOPs of diverse CNN architectures on a variety of computer vision tasks, including image classification, object detection, instance segmentation, and 3D vision. For example, our compressed ResNet-50 model on ImageNet dataset achieves 76% top-1 accuracy with only 25% FLOPs of the original ResNet-50 model, outperforming previous state-of-the-art channel pruning methods. The checkpoints and code are available at here .

1. Introduction

Albeit the empirical success of deep convolutional neural networks (CNN) on many computer vision tasks, the excessive computational and memory cost impede their deployment on mobile or edge devices. Therefore, it is vital to

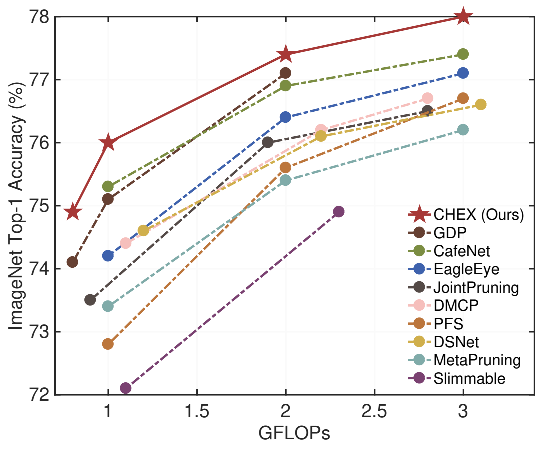

Figure 1. Comparison of the accuracy-FLOPs Pareto curve of the compressed ResNet-50 models on ImageNet. CHEX shows the top-performing Pareto frontier compared with previous methods. And we obtain the sub-models without pre-training a large model.

explore model compression, which reduces the redundancy in the model while maximally maintaining the accuracy.

Among various model compression approaches, channel pruning has been recognized as an effective tool to achieve practical memory saving and inference acceleration on general-purpose hardware. To derive a sub-model, channel pruning removes the redundant channels along with all the associated filters connected to those channels.

Most existing channel pruning methods [3, 13, 24, 25, 27, 28, 39, 53, 54, 57, 70, 72, 75, 80, 82] adopt a progressive pruning-training pipeline: pre-training a large model until convergence, pruning a few unimportant channels by the pre-defined criterion, and finetuning the pruned model to restore accuracy. The last two stages are usually executed in an interleaved manner repeatedly, which suffers from long training time. Various attempts have been made to improve the pruning efficiency. Training-based channel pruning methods [14, 40, 41, 45, 48, 81] impose sparse regularization such as LASSO or group LASSO to the model parameters during training. However, these commonly adopted regularization may not penalize the parameters to exactly zero. Removing

[*]Work done during an internship at DAMO Academy, Alibaba Group.
[†]Work partially done during an internship at Alibaba Group.
[‡]Work partially done during working period at Alibaba Group.

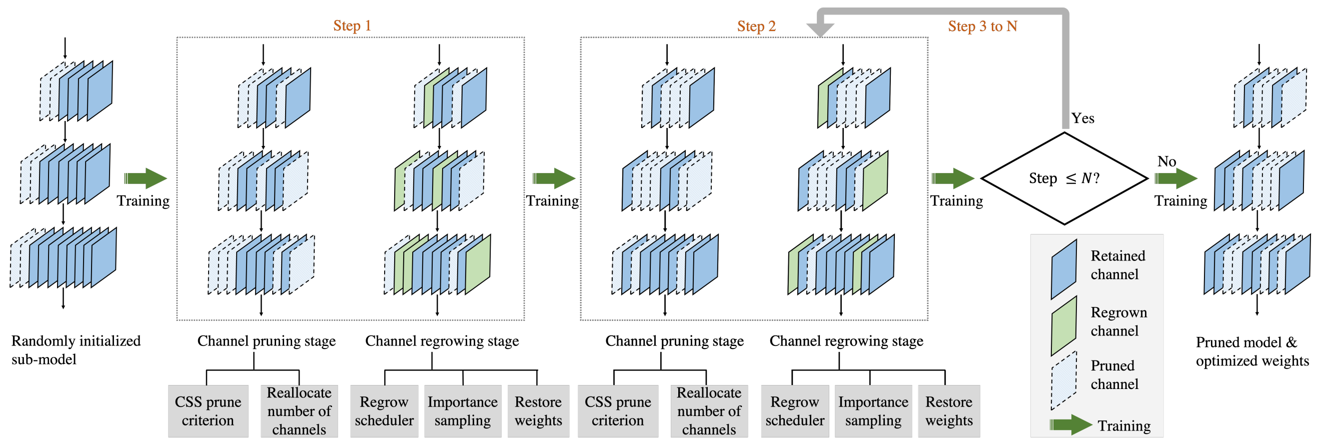

Figure 2. An illustration of our CHEX method, which jointly optimizes the weight values and explores the sub-model structure in one training pass from scratch. In CHEX, both retained and regrown channels in the sub-model are active, participating in the training iterations.

many small but non-zero parameters inevitably damages the model accuracy. Although this problem can be addressed by applying specialized optimizers [5, 32] or model reparameterization tricks [7], these methods require a well-pretrained large model, which counters our goal of improving pruning efficiency. Sampling-based methods [11, 30, 34, 51, 58, 78] directly train sparse models. However, these methods may suffer from training instability and converge to sub-optimal points [17]. [36, 66, 67, 74] shorten or eliminate the pre-training phase, and extract the sub-model at an early stage or even from random initialization. These methods are incapable to recover prematurely pruned important channels, which limits the model capacity and leads to unacceptable accuracy degradation.

To rectify the aforementioned limitations, we propose a *channel exploration* methodology called **CHEX** to obtain high accuracy sub-models without pre-training a large model or finetuning the pruned model. In contrast to the traditional pruning approaches that permanently remove channels [39, 53, 76, 82], we dynamically adjust the importance of the channels via a *periodic pruning and regrowing process*, which allows the prematurely pruned channels to be recovered and prevents the model from losing the representation ability early in the training process. From intra-layer's perspective, we re-formulate the channel pruning problem into the classic *column subset selection* (CSS) problem in linear algebra, leading to a closed-form solution. From inter-layer's perspective, rather than sticking to a fixed or manually designed sub-model structure, our approach re-evaluates the importance of different layers after each regrowing stage. This leads to a *sub-model structure exploration* technique to dynamically re-allocate the number of channels across all the layers under a given budget. With only one training pass from scratch, our obtained sub-models yield better accuracy than the previous methods under the same FLOPs reductions. Moreover, the simplicity of our method allows us to derive a theoretical analysis on the training behaviour of CHEX, to provide further insights and interpretation.

The contributions of this paper are highlighted as follows:

- We propose a channel exploration method CHEX with three novel features: (1) a periodic channel pruning and regrowing process, (2) a pruning criterion (i.e., leverage score) based on column subset selection, and (3) a sub-model structure exploration technique.
- Our method obtains the sub-model in one training pass from scratch, effectively reducing the training cost, because it circumvents the expensive pretrain-prune-finetune cycles.
- Experimentally, CHEX exhibits superior accuracy under different FLOPs constraints (as shown in Figure 1), and is applicable to a plethora of computer vision tasks. For image classification on ImageNet, our compressed ResNet-50 model yields $4\times$ FLOPs reduction while achieving 76% top-1 accuracy, improving previous best result [63] by 0.7%. For object detection on COCO dataset, our method achieves $2\times$ FLOPs reduction on the SSD model while improving 0.7% mAP over the unpruned baseline. For instance segmentation on COCO dataset and 3D point cloud segmentation on ShapeNet dataset, our method achieves $2\times$ and $11\times$ FLOPs reductions on the Mask R-CNN and PointNet++ models with negligible quality loss compared to the unpruned models, respectively.
- We provide a theoretical convergence guarantee of our CHEX method from the view of non-convex optimization setting, which is applicable to deep learning.

2. Methodology

Our method takes an existing CNN model as our channel exploration space, which provides the maximum number of explored channels. As illustrated in Figure 2, we describe the training flow of CHEX using a 3-layer CNN model as an example, in which we set the target channel sparsity[1] to 50%. From top to bottom, the three layers contain 6, 8, and 10 channels, respectively, denoted as Conv $-$ 6 $-$ 8 $-$ 10. Our method starts with a randomly initialized sub-model. During training, at periodically spaced training iterations (e.g., pre-determined ΔT iterations), a channel pruning and regrowing process is performed, which is referred to as one step in CHEX. A total of N steps are applied, and each step is composed of the following stages:

- A pruning stage removes the unimportant channels to the target sparsity. The number of channels are re-allocated across all different layers via the sub-model structure exploration technique. For example, in Figure 2, the pruned sub-model in step 1 has an architecture Conv $-$ 3 $-$ 4 $-$ 5, retaining 12 out of 24 channels. It may be adjusted later to Conv$-$2$-$3$-$7 in step 2. Such adaptations continue in the loop across the N steps.

- Immediately after pruning, a channel regrowing stage regrows back a fraction of the previously pruned channels, whose weight values are restored to their most recently used values before being pruned. Note that the regrown channels may be pruned multiple steps before. A decay scheduler is adopted to gradually reduce the number of regrown channels across the N steps. For example, in Figure 2, the channel regrowing stage in step 1 re-activates six channels, while it only re-activates four channels in step 2.

Our method interleaves the model training and the periodic pruning-regrowing stages. Since the total number of regrown channels is reduced at each step, the sub-model under training enjoys gradually decreased computation cost and converges to the target channel sparsity in the end. As an algorithm guideline, the pseudo-code of CHEX is provided in Algorithm 1.

2.1. Channel pruning stage

Suppose a CNN contains L layers with parameters $\{\mathbf{w}^1, ..., \mathbf{w}^L\}$, with $\mathbf{w}^l \in \mathbb{R}^{H^l W^l C^{l-1} \times C^l}$ denoting the reshaped[2] convolution weight matrix. $C^{l-1}, C^l, H^l \times W^l$ represent the number of input channels, the number of output channels, and the kernel size, respectively. The j-th channel in the l-th layer is denoted as $\mathbf{w}^l_{:,j}$. That is, a column in $\mathbf{w}^l$ represents one channel in the convolution. For notation simplicity, we use $K^l = H^l W^l C^{l-1}$ in the following text, i.e., $\mathbf{w}^l \in \mathbb{R}^{K^l \times C^l}$.

[1]We follow the notion of structured sparsity introduced in [68]. Our sub-model is a slimmer dense CNN model.

[2]For ease of derivation, we convert the convolution weight tensor of size $H^l \times W^l \times C^{l-1} \times C^l$ to the matrix of size $H^l W^l C^{l-1} \times C^l$, where the channels are listed as columns in the reshaped weight matrix.

Algorithm 1: Overview of the CHEX method.

```
1  Input: An $L$-layer CNN model with weights
     $\mathbf{W} = \{\mathbf{w}^1, ..., \mathbf{w}^L\}$; target channel sparsity $S$; total
     training iterations $T_{\text{total}}$; initial regrowing factor $\delta_0$;
     training iterations between two consecutive steps $\Delta T$;
     total pruning-regrowing steps $T_{\max}$; training set $\mathcal{D}$ ;
2  Output: A sub-model satisfying the target sparsity $S$ and
     its optimal weight values $\mathbf{W}^*$;
3  Randomly initialize the model weights $\mathbf{W}$;
4  for each training iteration $t \in [T_{total}]$ do
5      Sample a mini-batch from $\mathcal{D}$ and update the model
         weights $\mathbf{W}$ ;
6      if $Mod(t, \Delta T) = 0$ and $t \leq T_{max}$ then
7          Re-allocate the number of channels for each layer
             in the sub-model $\{\kappa^l, l \in [L]\}$ by Eq.(4) ;
8          Prune $\{\kappa^l C^l, l \in [L]\}$ channels by CSS-based
             pruning in Algorithm 2 ;
9          Compute the channel regrowing factor by a decay
             scheduler function ;
10         Perform importance sampling-based channel
             regrowing in Algorithm 3 ;
```

Suppose κ^l denotes the channel sparsity of the l-th layer. For each layer, channel pruning identifies a set of important channels with index set $\mathcal{T}^l$ ($|\mathcal{T}^l| = \lceil (1-\kappa^l) C^l \rceil$), which retains the most important information in $\mathbf{w}^l$, such that the remaining ones $\{\mathbf{w}^l_{:,j}, j \notin \mathcal{T}^l\}$ may be discarded with minimal impact to model accuracy. In other words, channel pruning selects the most "representative" columns from $\mathbf{w}^l$ that can reconstruct the original weight matrix with minimal error. From this perspective, channel pruning can be naturally represented as the *Column Subset Selection* (CSS) problem in linear algebra [12]. This provides us a new theoretical guideline for designing channel pruning criterion in a principled way, rather than depending on heuristics. To rigorously characterize the most "representative" columns of a matrix, we formally define CSS as follows:

Definition 2.1 (Column Subset Selection)**.** Given a matrix $\mathbf{w}^l \in \mathbb{R}^{K \times C^l}$, let $c \leq C^l$ be the number of columns to select. Find c columns of $\mathbf{w}^l$, denoted by $\mathbf{w}^l_c$, that would minimize:

$$\|\mathbf{w}^l - \mathbf{w}^l_c \mathbf{w}^{l\dagger}_c \mathbf{w}^l\|^2_F \quad \text{or} \quad \|\mathbf{w}^l - \mathbf{w}^l_c \mathbf{w}^{l\dagger}_c \mathbf{w}^l\|^2_2, \tag{1}$$

where $\dagger$ stands for the Moore-Penrose pseudo-inverse, $\|\cdot\|_F$ and $\|\cdot\|_2$ represent matrix Frobenius norm and spectral norm, respectively.

Since channel pruning and CSS share the same goal of best recovering the full matrix by a subset of its columns, we can leverage the rich theoretical foundations of CSS to derive a new pruning criterion. Our channel pruning stage is conducted periodically during training, thus we employ a computationally efficient deterministic CSS algorithm, referred to as the *Leverage Score Sampling* [60]. The core

Algorithm 2: CSS-based channel pruning.

```
1 Input: Model weights $\mathbf{w}^l$; pruning ratios $\kappa^l$ ;
2 Output: The pruned layer $l$ ;
3 Compute the number of retained channels
    $\tilde{C}^l = \lceil (1-\kappa^l) C^l \rceil$ ;
4 Compute the top $\tilde{C}^l$ right singular vectors $\mathbf{V}^l_{\tilde{C}^l}$ of $\mathbf{w}^l$ ;
5 Compute the leverage scores for all the channels in layer $l$
    $\psi^l_j = \|[\mathbf{V}^l_{\tilde{C}^l}]_{j,:}\|_2^2$ for all $j \in [C^l]$ ;
6 Retain the important channels identified as
    $\mathcal{T}^l = \text{ArgTopK}(\{\psi^l_j\}; \tilde{C}^l)$ ;
7 Prune channels $\{\mathbf{w}^l_{:,j}, j \notin \mathcal{T}^l\}$ from layer $l$ ;
```

Algorithm 3: Sampling-based channel regrowing.

```
1 Input: Indices of active channels $\mathcal{T}^l$ in the sub-model;
    regrowing factor $\delta_t$;
2 Output: The regrown layer $l$ ;
3 Compute the importance sampling probabilities by Eq.(2)
    $p^l_j = \exp(\epsilon^l_j)/\sum_{j'} \exp(\epsilon^l_{j'})$ for all $j \in [C^l] \setminus \mathcal{T}^l$ ;
4 Compute the number of regrown channels $k^l = \lceil \delta^t C^l \rceil$ ;
5 Perform importance sampling $\mathcal{G}^l$=Multinomial$(\{p^l_j\}; k^l)$ ;
6 Restore the MRU weights of the chosen channels
    $\{\hat{\mathbf{w}}^l_j, j \in \mathcal{G}^l\}$ ;
7 Regrow channels $\{\hat{\mathbf{w}}^l_j, j \in \mathcal{G}^l\}$ to layer $l$ ;
```

of this algorithm involves the leverage scores of matrix $\mathbf{w}^l$, which are defined as follows:

Definition 2.2 (Leverage Scores)**.** Let $\mathbf{V}_c \in \mathbb{R}^{C^l \times c}$ be the top-c right singular vectors of $\mathbf{w}^l$ (c represents the number of selected columns from $\mathbf{w}^l$). Then, the leverage score of the j-th column of $\mathbf{w}^l$ is given as: $\psi_j = \|[\mathbf{V}_c]_{j,:}\|_2^2$, where $[\mathbf{V}_c]_{j,:}$ denotes the j^{th} row of $\mathbf{V}_c$.

The leverage score sampling algorithm samples c columns of $\mathbf{w}^l$ that corresponds to the largest c leverage scores of $\mathbf{w}^l$. Despite its simplicity, theoretical analysis in [60] has shown that this deterministic solution provably obtains near optimal low-rank approximation error for Eq.(1).

Based on the above analysis, we propose a CSS-based channel pruning method with leverage score sampling, as shown in Algorithm 2. When given a pruning ratio κ^l for layer l, we need to select and retain $\lceil (1-\kappa^l) C^l \rceil$ important channels. We first compute the top $\lceil (1-\kappa^l) C^l \rceil$ right singular vectors of the weight matrix $\mathbf{w}^l$. Then, we calculate the leverage scores of all the channels in this layer as Definition 2.2, and rank them in descending order. Finally, we identify the set of important channels to retain as $\mathcal{T}^l = \text{ArgTopK}(\{\psi^l_j\}; \lceil (1-\kappa^l) C^l \rceil)$, which gives the indices of channels with the top $\lceil (1-\kappa^l) C^l \rceil$ leverage scores of $\mathbf{w}^l$. The remaining bottom-ranking channels are pruned.

2.2. Channel regrowing stage

Since the method trains from a randomly initialized model and the pruning stage may be based on the weights that are not sufficiently trained. In the early stage of training, the pruning decisions may not be optimal and some important channels are prematurely pruned. Therefore, after each pruning stage, our method regrows *a fraction* of the previously pruned channels back to the model. The regrown channels are updated in the subsequent training. If they are important to the model, they may survive the future pruning stages after a number of iterations of training. Moreover, the channel regrowing stage enables the model to have better representation ability during training, since the model capacity is not permanently restricted as the one-shot pruning methods.

To complete the regrowing stage, we need to assign proper weight values to the newly activated channels. One straightforward choice is to assign zero values for stable training, since the regrown channels do not affect the output of the model. However, we find that regrowing channels with zeros would receive zero gradients in the subsequent training iterations. This is undesirable because the regrown channels would remain deactivated and the method degenerates to the one-shot early-bird pruning [74]. Based on our ablations, we find the best scheme is that the newly activated channels restore their *most recently used* (MRU) parameters, which are the last values before they are pruned. We constantly maintain a copy of the weight values of the pruned channels, in case that they may get regrown back in the future regrowing stages. Note that the regrowing stage may regrow channels that are pruned multiple steps before, instead of just re-activating what are pruned at the pruning stage immediately before the current regrowing stage.

To determine the channels to regrow, a naive way is to perform uniform sampling from the candidate set $\{j | j \in [C^l] \setminus \mathcal{T}^l\}$. However, uniform sampling does not consider the possible inter-channel dependency between the active channels survived in the sub-model and the candidate channels to regrow. Instead, we propose an *importance sampling* strategy based on channel orthogonality for regrowing, as shown in Algorithm 3. Channel orthogonality automatically implies linear independency, which helps avoid trivial regrowing where the newly regrown channels lie in the span of the active channels. Channel orthogonality also encourages the channel diversity and improves model accuracy [1]. Formally, we denote the active channels in layer l by matrix $\mathbf{w}^l_{\mathcal{T}^l}$. The orthogonality ϵ^l_j of a channel $\mathbf{w}^l_j$ in the candidate set with respect to the active channels can be computed by the classic orthogonal projection formula [26]:

$$\epsilon^l_j = \|\mathbf{w}^l_j - \mathbf{w}^l_{\mathcal{T}^l} (\mathbf{w}^{l^T}_{\mathcal{T}^l} \mathbf{w}^l_{\mathcal{T}^l})^{\dagger} \mathbf{w}^{l^T}_{\mathcal{T}^l} \mathbf{w}^l_j\|_2^2. \quad (2)$$

A higher orthogonality value indicates that the channel is harder to approximate by others, and may have a better chance to be retained in the CSS pruning stage of the future steps. Thus, the corresponding channel may be sampled with a relatively higher probability. We use the orthogonality values to design our importance sampling distribution, and the probability p_j^l to regrow a channel $\mathbf{w}_j^l$ is given as:

$$p_j^l = \exp(\epsilon_j^l) / \sum\nolimits_{j'} \exp(\epsilon_{j'}^l). \tag{3}$$

Then, the channels to regrow are sampled according to the distribution Multinomial$(\{p_j^l | j \in [C^l] \setminus \mathcal{T}^l\}; \lceil \delta^t C^l \rceil)$ without replacement, where δ^t is the regrowing factor introduced as follows.

In the regrowing stage, we employ a cosine decay scheduler to gradually reduce the number of regrown channels so that the sub-model converges to the target channel sparsity at the end of training. Specifically, the regrowing factor at t-th step is computed as: $\delta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t \cdot \pi}{T_{\max}/\Delta T}\right)\right)\delta_0$, where δ_0 is the initial regrowing factor, $T_{\max}$ denotes the total exploration steps, and ΔT represents the frequency to invoke the pruning-regrowing steps.

2.3. Sub-model structure exploration

The starting model architecture may not have balanced layer distributions. Some layers are more important to the model accuracy and more channels need to be preserved, while some other layers may contain excessive number of channels. To better preserve model accuracy, our method dynamically re-distributes the surviving channels across different layers in each pruning stage. Such re-distribution is called sub-model structure exploration.

Inspired by [48], we use the learnable scaling factors in batch normalization (BN) [3] [33] to reflect the layer importance. Denote the BN scaling factors of all channels across all layers by $\Gamma = \{\boldsymbol{\gamma}^1, ..., \boldsymbol{\gamma}^L\}$, $\boldsymbol{\gamma}^l \in \mathbb{R}^{C^l}$, and the overall target channel sparsity by S. We calculate the layer-wise pruning ratios by ranking all scaling factors in descending order and preserving the top $1 - S$ percent of the channels. Then, the sparsity κ^l for layer l is given as:

$$\kappa^l = \left(\sum\nolimits_{j \in [C^l]} \mathbb{1}_{\{\gamma_j^l \le q(\Gamma, S)\}}\right) / C^l, l \in [L], \tag{4}$$

where $\mathbb{1}_{\{\gamma_j^l \le q(\Gamma, S)\}}$ is 0 if $\gamma_j^l > q(\Gamma, S)$ and 1 if $\gamma_j^l \le q(\Gamma, S)$. $q(\Gamma, S)$ represents the S-th percentile of all the scaling factors Γ. Accordingly, the number of channels in each layer of the sub-model is obtained as $\lceil (1 - \kappa^l) C^l \rceil$.

[48] relies on LASSO regularization to identify the insignificant channels, it extracts the sub-model from a fully pre-trained model in one-shot manner, and the subsequent finetuning procedure fixes the architecture without adaptation. In contrast, our method proposes a CSS-based pruning criterion without requiring any sparse regularization. And we advocate a repeated pruning and regrowing paradigm as opposed to the pruning-only strategy. We use BN scaling factors only for re-allocating the number of channels. We perform such re-allocation repeatedly at each step to take into account the changing layer importance during training. Thanks to our regrowing stages which help maintain the exploration space, our method can dynamically re-distribute channels from the less crucial layers to the more important ones, leading to a better sub-model structure.

[3] BN applies affine transformations to standardized input feature-maps: $X_{out} = \gamma \tilde{X}_{in} + \beta$, where γ / β are learnable scaling / shifting factors.

3. Theoretical Justification

We now provide the convergence guarantee for the CHEX method. Let $F(\mathbf{W}) = \mathbb{E}_{x \sim \mathcal{D}}[f(\mathbf{W}; x)]$ be the loss function of the deep learning task where x is the data following a distribution $\mathcal{D}$ and $\mathbb{E}[\cdot]$ is the expectation. In addition, let $\mathbf{W}_t \in \mathbb{R}^d$ be the model parameter at the t-th training iteration, and $m_t \in \{0, 1\}^d$ be a binary channel mask vector for $t = 1, \cdots, T$. Apparently, quantity $\mathbf{W}_t \odot m_t \in \mathbb{R}^d$ is a sub-model pruned from $\mathbf{W}_t$ where $\odot$ denotes the element-wise product. The following proposition shows that the CHEX will converge to a neighborhood around the stationary solution at rate $O(1/\sqrt{T})$ when learning rate is set properly. Due to the space limitation, we put its proof in the Appendix.

Proposition 1 (Convergence Guarantee)**.** Suppose the loss function $F(\mathbf{W})$ is L-smooth, the sampled stochastic gradient is unbiased and has bounded variance, and the relative error introduced by each mask is bounded, i.e., $\|\mathbf{W} - \mathbf{W} \odot m_t\|^2 \le \delta^2 \|\mathbf{W}\|^2$ and $\|\nabla F(\mathbf{W}) - \nabla F(\mathbf{W}) \odot m_t\|^2 \le \zeta^2 \|\nabla F(\mathbf{W})\|^2$ for constants $\delta \in [0, 1]$ and $\zeta \in [0, 1]$. If learning rate $\eta = \frac{\sqrt{2C_0}}{\sigma\sqrt{L(T+1)}}$ in which $C_0 = \mathbb{E}[F(\mathbf{W}_0)]$, the sub-models obtained by CHEX will converge as follows:

$$\begin{aligned} &\frac{1}{T+1}\sum_{t=0}^{T} \mathbb{E}[\|\nabla F(\mathbf{W}_t \odot m_t)\|^2] \\ \le\ & \frac{4\sigma\sqrt{LC_0}}{(1-\zeta)^2\sqrt{T+1}} + \frac{2L^2\delta^2}{(T+1)(1-\zeta)^2}\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{W}_t\|^2]. \end{aligned} \tag{5}$$

Remark. If there is no pruning (i.e., $m_t = \mathbf{1} \in \mathbb{R}^d$) in the training process, it holds that $\delta = 0$ and $\zeta = 0$. Substituting it into (5), we find that the CHEX method can converge exactly to the stationary solution, i.e., $\mathbb{E}[\|\nabla F(\mathbf{W}_T)\|^2] \to 0$ as T increases to infinity. When a sparse channel mask is utilized, it holds that $\delta \neq 0$ and $\zeta \neq 0$. In this scenario, the mask-induced error will inevitably influence the accuracy of the trained sub-model, i.e., constants δ and ζ will influence the magnitude of the second term in the upper bound (5).

4. Experiments

To evaluate the efficacy and generality of CHEX, we experiment on a variety of computer vision tasks with diverse CNN architectures. All experiments run on PyTorch framework based on DeepLearningExamples [59] with NVIDIA

Method	PT	FLOPs	Top-1	Epochs
ResNet-18				
PFP [42]	Y	1.27G	67.4%	270
SCOP [65]	Y	1.10G	69.2%	230
SFP [22]	Y	1.04G	67.1%	200
FPGM [24]	Y	1.04G	68.4%	200
DMCP [15]	N	1.04G	69.0%	150
CHEX	N	1.03G	**69.6%**	250
ResNet-34				
Taylor [57]	Y	2.8G	72.8%	-
SFP [22]	Y	2.2G	71.8%	200
FPGM [24]	Y	2.2G	72.5%	200
GFS [72]	Y	2.1G	72.9%	240
DMC [11]	Y	2.1G	72.6%	490
NPPM [10]	Y	2.1G	73.0%	390
SCOP [65]	Y	2.0G	72.6%	230
CafeNet [63]	N	1.8G	73.1%	300
CHEX	N	2.0G	**73.5%**	250
ResNet-101				
SFP [22]	Y	4.4G	77.5%	200
FPGM [24]	Y	4.4G	77.3%	200
PFP [42]	Y	4.2G	76.4%	270
AOFP [6]	Y	3.8G	76.4%	-
NPPM [10]	Y	3.5G	77.8%	390
DMC [11]	Y	3.3G	77.4%	490
CHEX-1	N	3.4G	**78.8%**	250
CHEX-2	N	1.9G	**77.6%**	250

Method	PT	FLOPs	Top-1	Epochs
ResNet-50				
GBN [75]	Y	2.4G	76.2%	350
LeGR [3]	Y	2.4G	75.7%	150
SSS [32]	N	2.3G	71.8%	100
TAS [8]	N	2.3G	76.2%	240
GAL [45]	Y	2.3G	72.0%	150
Hrank [43]	Y	2.3G	75.0%	570
Taylor [57]	Y	2.2G	74.5%	-
C-SGD [5]	Y	2.2G	74.9%	-
SCOP [65]	Y	2.2G	76.0%	230
DSA [58]	N	2.0G	74.7%	120
CafeNet [63]	N	2.0G	76.9%	300
CHEX-1	N	2.0G	**77.4%**	250
SCP [34]	N	1.9G	75.3%	200
Hinge [41]	Y	1.9G	74.7%	-
AdaptDCP [82]	Y	1.9G	75.2%	210
LFPC [21]	Y	1.6G	74.5%	235
ResRep [7]	Y	1.5G	75.3%	270
Polarize [81]	Y	1.2G	74.2%	248
DSNet [38]	Y	1.2G	74.6%	150
CURL [52]	Y	1.1G	73.4%	190
DMCP [15]	N	1.1G	74.1%	150
MetaPrune [49]	N	1.0G	73.4%	160
EagleEye [37]	Y	1.0G	74.2%	240
CafeNet [63]	N	1.0G	75.3%	300
CHEX-2	N	1.0G	**76.0%**	250

(a)

Method	FLOPs reduction	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
SSD object detection							
Baseline [47]	0%	25.2	42.7	25.8	7.3	27.1	40.8
DMCP [15]	50%	24.1	41.2	24.7	6.7	25.6	39.2
CHEX-1	50%	**25.9**	43.0	26.8	7.8	27.8	41.7
CHEX-2	75%	**24.3**	41.0	24.9	7.1	25.6	40.1
Mask R-CNN object detection							
Baseline [18]	0%	37.3	59.0	40.2	21.9	40.9	48.1
CHEX	50%	**37.3**	58.5	40.4	21.7	39.0	49.5
Mask R-CNN instance segmentation							
Baseline [18]	0%	34.2	55.9	36.2	15.8	36.9	50.1
CHEX	50%	**34.5**	55.7	36.7	15.9	36.1	51.2

(b)

Method	#Points	FLOPs reduction	Quality
3D shape classification (Quality is accuracy)			
Baseline [61]	1k	0%	92.8%
CHEX	1k	87%	**92.9%**
3D part segmentation (Quality is class/instance mIoU)			
Baseline [61]	2k	0%	82.5%/85.4%
ADMM [62]	2k	90%	77.1%/84.0%
CHEX	2k	91%	**82.3%/85.2%**

(c)

Table 1. **(a)**: Results of ResNet on ImageNet dataset. "PT": require pre-training. "Y": Yes, "N": No. **(b)**: Results of SSD on COCO2017 and Mask R-CNN on COCO2014. For objection detection, we evaluate the bounding box AP. For instance segmentation, we evaluate the mask AP. **(c)**: Results of PointNet++ for 3D point clouds classification on ModelNet40 dataset and segmentation on ShapeNet dataset.

Tesla V100 GPUs. We set $\delta_0 = 0.3$ and $\Delta T = 2$ epoch, where δ_0 means the initial regrowing factor, and ΔT is the number of training iterations between two consecutive pruning-regrowing steps. To keep our method simple and generic, the above hyper-parameters are kept constant for our experiments. We set the rest of the hyper-parameters in the default settings and specify them in the Appendix. In this paper, FLOPs is calculated by counting multiplication and addition as one operation by following [20].

4.1. Image recognition

For image recognition, we apply CHEX to ResNet [20] with different depths on ImageNet [4]. The baseline ResNet-18/34/50/101 models have 1.8/3.7/4.1/7.6 GFLOPs with 70.3%/73.9%/77.8%/78.9% top-1 accuracy, respectively.

As shown in Table 1(a), CHEX achieves noticeably higher accuracy than the state-of-the-art channel pruning methods under the same FLOPs. For example, compared with MetaPruning [49], DCMP [15] and CafeNet [63], our pruned ResNet-50 model with 4$\times$ FLOPs reduction achieves 2.6%, 1.6%, and 0.7% higher top-1 accuracy, respectively. On the other hand, at the same target accuracy, CHEX achieves higher FLOPs reduction. For example, CHEX achieves 4$\times$ FLOPs reduction on ResNet-101 model with 77.6% top-1 accuracy, compared to the latest work NPPM [10] which yields 2.2$\times$ FLOPs reduction.

The results in Table 1(a) also show an interesting property of our CHEX method. When we search a sub-model with a small FLOPs target, it is better to start our method on a larger model than on a smaller one. For instance, pruning a ResNet-50 model to 1 GFLOPs yields an accuracy 6.6% higher than pruning a ResNet-18 model to 1 GFLOPs. This indicates that CHEX performs training-time structural exploration more effectively when given a larger parametric space. To illustrate this point further, we conduct an additional experiment by applying CHEX to a model with twice the number of channels as the ResNet-50 model, with the goal of reducing its FLOPs to the same level as the original ResNet-50 model. Notably, this sub-model achieves 78.9% top-1 accuracy at 4.1 GFLOPs. This suggests that CHEX has the potential to optimize existing CNNs.

4.2. Object detection

For object detection, we apply CHEX to the SSD model [47] on COCO2017 dataset [46]. Table 1(b) summarizes the performance with different FLOPs reductions. Our pruned model outperforms the baseline model by 0.7% AP (25.9 vs. 25.2) while achieving 2$\times$ FLOPs reduction.

Method	Top-1
Baseline	73.9%
+ Periodic pruning and regrowing	74.8%
+ Dynamic sub-model structure exploration	75.6%
+ Importance sampling-based regrowing	76.0%

Table 2. Ablation study of different components in CHEX.

Compared with previous SOTA channel pruning method [15], our method achieves 1.8% higher AP (25.9 vs. 24.1) with 2× FLOPs reduction. Moreover, our method achieves 4× FLOPs reduction with less than 1% AP loss.

4.3. Instance segmentation

For instance segmentation, we apply CHEX to the Mask R-CNN model [18] on COCO2014 dataset. Since Mask R-CNN is a multi-task framework, we evaluate both the bounding box AP for object detection and the mask AP for instance segmentation. As shown in Table 1(b), the models pruned using our method achieve 2× FLOPs reduction without AP loss, even for the challenging instance segmentation task where the model needs to detect all objects correctly in an image while precisely segmenting each instance.

4.4. 3D point cloud

Apart from 2D computer vision problems, we also apply CHEX to compress PointNet++ [61] for 3D shape classification on ModelNet40 [69] dataset and 3D part segmentation on ShapeNet [73] dataset. The results are shown in Table 1(c). On shape classification, the model pruned using our method achieves around 7.5× FLOPs reduction while improving the accuracy slightly compared to the unpruned baseline. On part segmentation, the model pruned using our method achieves 11× FLOPs reduction while maintaining similar mIoU compared with the unpruned baseline.

5. Ablation Analysis

We investigate the effectiveness of different components in the CHEX method through ablation studies. All the following results are based on pruning the ResNet-50 model to 1 GFLOPs (4× reduction) on ImageNet dataset.

Ablation study. In Table 2, we study the effectiveness of different components in CHEX, namely the periodic channel pruning and regrowing process, the sub-model structure exploration technique, and the importance sampling-based regrowing. The baseline is one-shot early-stage pruning [74], where the model is pruned by CSS very early after 6% of the total training epochs, and there is no regrowing stage. The sub-model architecture is based on the BN scaling factors and kept fixed in the subsequent training. The periodic pruning and regrowing process repeatedly samples the important channels and prevents the channels from being pruned prematurely. This brings in 0.9% accuracy improvement. When the number of channels in each layer is also

Prune criterion	BN [48]	Magnitude [39]	CSS (Ours)
Pretrain-prune-finetune	73.1%	73.8%	74.6%
CHEX	75.3%	75.7%	76.0%

Table 3. Influence of the pruning criterion to the top-1 accuracy in pretrain-prune-finetune framework and our CHEX method.

Initialization of regrown channels					
Design choices	Zero	Random	EMA	MRU	
Top-1	74.1%	74.5%	75.5%	76.0%	
Regrowing factor					
Design choices	$\delta_0 = 0.1$	$\delta_0 = 0.2$	$\delta_0 = 0.3$	$\delta_0 = 0.4$	Full
Top-1	75.2%	75.6%	76.0%	75.9%	76.0%
Scheduler for channel regrowing					
Design choices	Constant	Linear decay		Cosine decay	
Top-1	75.3%	75.6%		76.0	
Pruning-regrowing frequency					
Design choices	$\Delta T = 1$	$\Delta T = 2$	$\Delta T = 5$	$\Delta T = 10$	$\Delta T = 20$
Top-1	75.2%	76.0%	75.8%	75.3%	74.9%

Table 4. Compare different design choices in the regrowing stages of the CHEX method.

dynamically adjusted instead of sticking to the fixed structure determined very early in training, we can obtain a more optimal sub-model structure, which further improves the accuracy by 0.8%. Finally, using the importance sampling strategy described in Eq.(3) instead of uniform sampling in the regrowing stages improves the top-1 accuracy to 76%.

Influence of pruning criterion. We study the influence of pruning criterion to the final accuracy by comparing CSS versus filter magnitude [39] and BN [48] pruning in Table 3. As suggested by [28, 39, 53, 57], pruning criterion is an important factor in the traditional pretrain-prune-finetune (PPF) approach. Indeed, when the PPF approach is used, our proposed CSS shows the best accuracy, outperforming magnitude and BN pruning by 0.8% and 1.5%, respectively. On the other hand, we observe that CHEX is more robust to the pruning criterion, as it improves the accuracy for all three criteria and the accuracy gap among them becomes smaller: CSS outperforms magnitude and BN pruning by 0.3% and 0.7%, respectively. We suspect that this is because CHEX can dynamically adjust the channel importance and can recover from sub-optimal pruning decisions.

Design choices for the regrowing stage. In Table 4, we investigate the impact of different schemes in the channel regrowing stage of CHEX:

(1) Different initialization schemes for the regrown channels affect model quality critically. We have experimented several methods by initializing the regrown channels with random normal distribution [19], zero weights, the exponential moving average (EMA) weights, and most recently used (MRU) weights. The results show that MRU yields the best top-1 accuracy, outperforming the other three initialization schemes by $0.5\% \sim 1.9\%$.

(2) The initial regrowing factor δ_0 also affects the model quality. Intuitively, a large regrowing factor can maintain

relatively larger model capacity in the early stage of training, and involve more channels into the exploration steps. As shown, the accuracy is improved by 0.8% as the δ_0 increases from 0.1 to 0.3, at the price of more training cost. However, we did not observe further improvement when the regrowing factor increases from $\delta_0 = 0.3$ to full model size, in which case one step in CHEX consists of pruning to target channel sparsity and regrowing all pruned channels. This suggests that regrowing *a fraction* of the previously pruned channels may be enough for a comprehensive exploration of the channels if we also aim more training cost savings.

(3) We decay the number of regrown channels in each regrowing stage to gradually restrict the scope of exploration. We compare several decay schedulers, including constant, linear, and cosine. The cosine decay scheduler performs better than the other two schemes by 0.7% and 0.4% accuracy, respectively. With a decay scheduler, the sub-model under training will enjoy gradually decreased computation cost.

(4) The training iterations between two consecutive pruning-regrowing steps, ΔT, also affects the model quality. A smaller ΔT incurs more frequent pruning-regrowing steps in CHEX, leading to better accuracy in general. We use $\Delta T = 2$ epoch, as it gives the best accuracy.

6. Related Works

Channel pruning methods can be roughly categorized into three classes.

(1) *Pruning after training* approaches follow a three-stage pipeline: pre-training a large model, pruning the unimportant channels, and finetuning the pruned model. Prior arts in this category mainly focus on comparing different pruning criteria. Exemplary metrics for evaluating channel importance include weight norm [3, 21, 22, 39, 70], geometric median [24], Taylor expansion of cross-entropy loss [44, 57, 75], discrepancy of final response layer [76], feature-maps reconstruction error [25, 53, 82], feature-maps rank [43], KL-divergence [52], greedy forward selection with largest loss reduction [72], feature-maps discriminant information [27, 28, 35]. Our method differs substantially from these approaches that we do not pre-train a large model. Instead, the compact sub-model is explored during a normal training process from scratch.

(2) *Pruning during training* approaches [14, 16, 30, 32, 34, 40, 41, 45, 48, 55, 71, 77, 81] perform channel selection and model training jointly, usually by imposing sparse regularizations to the channel-wise scaling factors and adopting specialized optimizers to solve it. Although these methods usually yield good acceleration due to joint optimization, many of them [14, 40, 41, 45, 48, 79, 81] perform the sparse regularization training on fully pre-trained large models. Therefore, these approaches still suffer from the expensive training cost. In contrast, our method does not rely on sparse regularization. Instead, the periodic channel pruning and regrowing processes adaptively explore the channel importance during training.

(3) *Pruning at early stage* methods compress the model at random initialization or early stage of training [9, 36, 50, 56, 64, 66, 67, 74]. Although the training cost is reduced, one-shot pruning based on under-trained model weights may not properly reflect the channel importance and the model representation ability is prematurely reduced at the very beginning, resulting in significant accuracy loss. In contrast, the proposed method can recover prematurely pruned channels and better maintain the model capacity.

AutoML or NAS based pruning methods automatically search the optimal number of channels in each layer of CNN. [23] uses reinforcement learning to search a compression policy in a layer-wise manner. MetaPruning [49] pre-trains a hyper-network to predict the weights for candidate pruned models, and adopts evolutionary search to find a good candidate. NetAdapt [70] progressively adapts a pre-trained model to a mobile platform until a resource budget is met. DMCP [13] proposes a differentiable search method by modeling channel pruning as a Markov process. CafeNet [63] proposes locally free weight sharing for the one-shot model width search. [29] proposes a Gaussian process search for optimizing the multi-dimensional compression policy. These approaches usually rely on a pre-trained supernet, and require a separate search process involving substantial search-evaluation iterations. In contrast, CHEX performs sub-model structure learning together with weights optimization, making it more efficient than the search-based methods. Channel pruning has also been incorporated into NAS to further tune the searched architecture for different latency targets [2, 31]. Since CHEX does not increase the training time, it can be potentially used as a step in AutoML.

7. Conclusion

We propose a novel channel exploration methodology, CHEX, to reduce the computation cost of deep CNNs in both training and inference. CHEX (1) dynamically adjusts the channel importance based on a periodic pruning and regrowing process, which prevents the important channels from being prematurely pruned; (2) dynamically re-allocates the number of channels under a global sparsity constraint to search for the optimal sub-model structure during training. We design the components in the pruning and regrowing process by proposing a column subset selection based criterion for pruning and a channel orthogonality based importance sampling for regrowing. This enables us to obtain a sub-model with high accuracy in only one training pass from scratch, without pre-training a large model or requiring extra fine-tuning. Experiments on multiple deep learning tasks demonstrate that our method can effectively reduce the FLOPs of diverse CNN models while achieving superior accuracy compared to the state-of-the-art methods.

Acknowledgment This work was supported by Alibaba Group through Alibaba Research Intern Program.

References

[1] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns? *Proceedings of Advances in Neural Information Processing Systems*, 2018. 4

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *Proceedings of International Conference on Learning Representations*, 2020. 8

[3] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 6, 8

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 6

[5] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4943–4953, 2019. 2, 6

[6] Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive cnn width optimization. *Proceedings of International Conference on Machine Learning*, 2019. 6

[7] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Lossless cnn channel pruning via decoupling remembering and forgetting. *Proceedings of the IEEE/CVF International Conference on Com-puter Vision*, 2021. 2, 6

[8] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. *Proceedings of Advances in Neural Information Processing Systems*, pages 759–770, 2019. 6

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Proceedings of International Conference on Learning Representations*, 2019. 8

[10] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9270–9280, 2021. 6

[11] Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1899–1908, 2020. 2, 6

[12] Ming Gu and Stanley C Eisenstat. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869, 1996. 3

[13] Jinyang Guo, Wanli Ouyang, and Dong Xu. Channel pruning guided by classification loss and feature importance. *Proceedings of AAAI*, 2020. 1, 8

[14] Jinyang Guo, Wanli Ouyang, and Dong Xu. Multi-dimensional pruning: A unified framework for model compression. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1508–1517, 2020. 1, 8

[15] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1539–1547, 2020. 6, 7

[16] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Proceedings of Advances in Neural Information Processing Systems*, pages 1379–1387, 2016. 8

[17] Yi Guo, Huan Yuan, Jianchao Tan, Zhangyang Wang, Sen Yang, and Ji Liu. Gdp: Stabilized neural network pruning via gates with differentiable polarization. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5239–5250, 2021. 2

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6, 7

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 7

[20] 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*, pages 770–778, 2016. 6

[21] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2009–2018, 2020. 6, 8

[22] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *Proceedings of International Joint Conference on Artificial Intelligence*, 2018. 6, 8

[23] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. *Proceedings of the European Conference on Computer Vision*, pages 784–800, 2018. 8

[24] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 1, 6, 8

[25] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 1, 8

[26] Roger A Horn and Charles R Johnson. Matrix analysis. 2012. 4

[27] Zejiang Hou and Sun-Yuan Kung. Efficient image super resolution via channel discriminative deep neural network pruning. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3647–3651, 2020. 1, 8

[28] Zejiang Hou and Sun-Yuan Kung. A feature-map discriminant perspective for pruning deep neural networks. *arXiv preprint arXiv:2005.13796*, 2020. 1, 7, 8

[29] Zejiang Hou and Sun-Yuan Kung. Multi-dimensional model compression of vision transformer. *arXiv preprint arXiv:2201.00043*, 2021. 8

[30] Zejiang Hou and Sun-Yuan Kung. Multi-dimensional dynamic model compression for efficient image super-resolution. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 633–643, 2022. 2, 8

[31] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 8

[32] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. *Proceedings of the European Conference on Computer Vision*, pages 304–320, 2018. 2, 6, 8

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of International Conference on Machine Learning*, pages 448–456, 2015. 5

[34] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. *Proceedings of International Conference on Machine Learning*, pages 5122–5131, 2020. 2, 6, 8

[35] Sun-Yuan Kung and Zejiang Hou. Augment deep bp-parameter learning with local xai-structural learning. *Communications in Information and Systems*, 20(3):319–352, 2020. 8

[36] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *Proceedings of International Conference on Learning Rrepresentations*, 2019. 2, 8

[37] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. *Proceedings of European Conference on Computer Vision*, pages 639–654, 2020. 6

[38] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 6

[39] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *Proceedings of International Conference on Learning Representations*, 2017. 1, 2, 7, 8

[40] Tuanhui Li, Baoyuan Wu, Yujiu Yang, Yanbo Fan, Yong Zhang, and Wei Liu. Compressing convolutional neural networks via factorized convolutional filters. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3977–3986, 2019. 1, 8

[41] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8018–8027, 2020. 1, 6, 8

[42] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. *Proceedings of International Conference on Learning Representations*, 2020. 6

[43] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 6, 8

[44] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2425–2432, 2018. 8

[45] 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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019. 1, 6, 8

[46] 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. *Proceedings of European conference on computer vision*, pages 740–755, 2014. 6

[47] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. *Proceedings of European conference on computer vision*, pages 21–37, 2016. 6

[48] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 1, 5, 7, 8

[49] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305, 2019. 6, 8

[50] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *Proceeding of International Conference on Learning Representations*, 2019. 8

[51] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *Proceedings of International Conference on Learning Representations*, 2018. 2

[52] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1458–1467, 2020. 6, 8

[53] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 1, 2, 7, 8

[54] Xiaolong Ma, Sheng Lin, Shaokai Ye, Zhezhi He, Linfeng Zhang, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, et al. Non-structured dnn weight pruning–is it beneficial in any platform? *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2021. 1

[55] Xiaolong Ma, Minghai Qin, Fei Sun, Zejiang Hou, Kun Yuan, Yi Xu, Yanzhi Wang, Yen-Kuang Chen, Rong Jin, and Yuan Xie. Effective model sparsification by scheduled grow-and-prune methods. In *International Conference on Learning*

Representations, 2022. 8

[56] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, et al. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 8

[57] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 1, 6, 7, 8

[58] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: More efficient budgeted pruning via differentiable sparsity allocation. *Proceedings of European Conference on Computer Vision*, pages 592–607, 2020. 2, 6

[59] NVIDIA. DeepLearningExamples. https://github.com/NVIDIA/DeepLearningExamples. 5

[60] Dimitris Papailiopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable deterministic leverage score sampling. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 997–1006, 2014. 3, 4

[61] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 6, 7

[62] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 925–938, 2019. 6

[63] Xiu Su, Shan You, Tao Huang, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Locally free weight sharing for network width search. *Proceedings of International Conference on Learning Representations*, 2021. 2, 6, 8

[64] Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020. 8

[65] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *Proceedings of Advances in Neural Information Processing Systems*, 2020. 6

[66] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020. 2, 8

[67] Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. *Proceedings of AAAI*, 2020. 2, 8

[68] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Processing of Advances in Neural Information Processing Systems*, pages 2074–2082, 2016. 3

[69] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 7

[70] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. *Proceedings of European Conference on Computer Vision*, pages 285–300, 2018. 1, 8

[71] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *Proceedings of International Conference on Learning Representations*, 2018. 8

[72] Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. *Proceedings of International Conference on Machine Learning*, pages 10820–10830, 2020. 1, 6, 8

[73] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6):1–12, 2016. 7

[74] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *Proceedings of International Conference on Learning Representations*, 2020. 2, 4, 7, 8

[75] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *Proceedings of Advances in Neural Information Processing Systems*, 2019. 1, 6, 8

[76] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 2, 8

[77] Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 8

[78] Xin Yuan, Pedro Henrique Pamplona Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. *Proceedings of International Conference on Learning Representations*, 2021. 2

[79] Tianyun Zhang, Xiaolong Ma, Zheng Zhan, Shanglin Zhou, Caiwen Ding, Makan Fardad, and Yanzhi Wang. A unified dnn weight pruning framework using reweighted optimization methods. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 493–498. IEEE, 2021. 8

[80] Tianyun Zhang, Shaokai Ye, Xiaoyu Feng, Xiaolong Ma, Kaiqi Zhang, Zhengang Li, Jian Tang, Sijia Liu, Xue Lin, Yongpan Liu, et al. Structadmm: Achieving ultrahigh efficiency in structured pruning for dnns. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2021. 1

[81] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning

using polarization regularizer. *Proceedings of Advances in Neural Information Processing Systems*, 2020. 1, 6, 8

[82] 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. *Proceedings of Advances in Neural Information Processing Systems*, pages 883–894, 2018. 1, 2, 6, 8