Appendix

A1. Implementation details

Image classification on ImageNet. To train ResNet with the CHEX method on ImageNet dataset, we use SGD optimizer with a momentum of 0.875, a mini-batch size of 1024, and an initial learning rate of 1.024. The learning rate is linearly warmed up for the first 8 epochs, and decayed to zero by a cosine learning rate schedule. The weight decay is set to 3e-5. Same as previous methods [8, 9, 22], we also use label smoothing with factor 0.1. We train the model for a total of 250 epochs. For data augmentation, we only use random resized crop to 224×224 resolution, random horizontal flip, and normalization.

Object detection on COCO2017. Following [21], we train SSD with the CHEX method on COCO train2017 split containing about 118k images and evaluate on the val2017 split containing 5k images. The input size is fixed to 300×300. We adopt SGD optimizer with a momentum of 0.9, a mini-batch size of 64, and a weight decay of 5e-4. We train the model for a total of 240k iterations. The initial learning rate is set to 1e-3, and is decayed by 10 at the 160k and 200k iteration. The SSD uses a ResNet-50 model pretrained on ImageNet dataset as the backbone.

Instance segmentation on COCO2014. We follow the standard practice as [10] to train Mask R-CNN with the CHEX method on COCO training split, and evaluate on the validation split. We train with a batch size of 32 for 160K iterations. We adopt SGD with a momentum of 0.9 and a weight decay of 1e-4. The initial learning rate is set to 0.04, which is decreased by 10 at the 100k and 140k iteration. The Mask R-CNN uses ResNet-50-FPN model as the backbone.

3D classification and segmentation on ModelNet40 and ShapeNet. Following [27], we train the PointNet++ model with the CHEX method using the Adam optimizer with a mini-batch size of 32. The learning rate begins with 0.001 and decays with a rate of 0.7 every 20 epochs. We train the model for a total of 200 epochs on the ModelNet40 for 3D shape classification, and 250 epcohs on the ShapeNet dataset for 3D part segmentation.

A2. More results

A2.1. CHEX on lightweight CNNs

We apply the CHEX method to compress compact CNN models MobileNetV2 and EfficientNet-B0. As shown in Table 1, our compressed MobileNetV2 model with around 30% FLOPs reduction achieves almost no accuracy loss compared to the unpruned baseline. With 50% FLOPs reduction, our compressed MobileNetV2 model outperforms previous state-of-the-art channel pruning methods by 0.8$\sim$2.3% accuracy. Similarly, our method achieves superior accuracy when compressing EfficientNet-B0 with the same FLOPs reduction as the previous methods.

Model	Method	FLOPs	Top-1
MobileNetV2	Baseline	300M	72.2%
	LeGR [3]	220M	71.4%
	GFS [33]	220M	71.6%
	MetaPruning [22]	217M	71.2%
	DMCP [8]	211M	71.6%
	AMC [13]	210M	70.8%
	PFS [31]	210M	70.9%
	JointPruning [23]	206M	70.7%
	CHEX-1	220M	**72.0%**
	DMC [7]	162M	68.4%
	GFS [33]	152M	69.7%
	LeGR [3]	150M	69.4%
	JointPruning [23]	145M	69.1%
	MetaPruning [22]	140M	68.2%
	CHEX-2	150M	**70.5%**
EfficientNet-B0	Baseline	390M	77.1%
	PEEL [15]	346M	77.0%
	CHEX-1	330M	**77.4%**
	DSNet [17]	270M	75.4%
	CHEX-2	270M	**76.2%**
	CafeNet-R [29]	192M	74.5%
	CHEX-3	192M	**74.8%**

Table 1. Results of MobileNetV2 and EfficientNet-B0 on ImageNet dataset.

A2.2. Comparison with GrowEfficient and PruneTrain

We follow GrowEfficient's settings [35] in choosing WideResNet-28-10 (on CIFAR10 dataset) and ResNet-50 (on ImageNet dataset) as the baseline models. For a fair comparison, we adopt the same training hyper-parameters as GrowEfficient, and train the models with CHEX from scratch. Both GrowEfficient and PruneTrain [25] sparsify the models using LASSO regularization during training. In contrast, the CHEX method incorporates explicit channel pruning and regrowing stages, and interleaves them in a repeated manner without any sparse regularization. As shown in Table 2, our method achieves noticeably higher accuracy than GrowEfficient and PruneTrain under same FLOPs reduction. Moreover, our method demonstrates effective training cost saving compared to the baseline model training without accuracy loss.

A2.3. Comparison with NAS

We also compare the CHEX method with the state-of-the-art NAS method, OFA [2] on ImageNet. For a fair comparison, we take ResNet-50D [11] as the baseline architecture to perform our CHEX method, by following OFA-ResNet-50 [1]. OFA firstly trains a supernet, then applies progressive shrinking in four dimensions, including number of layers,

Method	FLOPs reduction	Top-1	Training cost saving
WRN-28-10 on CIFAR10 (200 epochs)			
Baseline [35]	0%	96.2%	0%
GrowEfficient [35]	71.8%	95.3%	67.9%
CHEX	74.8%	**96.2%**	48.8%
ResNet-50 on ImageNet (100 epochs)			
Baseline [35]	0%	76.2%	0%
PruneTrain [25]	44.0%	75.0%	30.0%
GrowEfficient [35]	49.5%	75.2%	47.4%
CHEX	50.2%	**76.3%**	43.0%

Table 2. Comparison with GrowEfficient and PruneTrain on CIFAR10 and ImageNet datasets. All methods train from scratch with the same number of epochs.

number of channels, kernel sizes, and input resolutions, and finally finetunes the obtained sub-models. In contrast, our CHEX method only adjusts the number of channels via the periodic pruning and regrowing process, and we do not require training a supernet nor extra finetuning. As shown in Table 3, CHEX achieves superior accuracy under similar FLOPs constraints but with significantly less model parameters and training GPU hours than OFA.

Method	FLOPs	Params.	Top-1	Training cost (GPU hours)
OFA [1, 2]	900M	14.5M	76.0%	1200
OFA#25 [1, 2]	900M	14.5M	76.3%	1200
CHEX	980M	7.2M	**76.4%**	130
CHEX$_{2\times}$	980M	7.2M	**76.8%**	260

Table 3. Comparison with OFA using ResNet-50D model on ImageNet dataset. "CHEX$_{2\times}$" means doubling the training epochs in our method.

A2.4. CHEX from pretrained models

To further showcase the generality of our method, we apply CHEX to a pretrained model. For a fair comparison with other pretrain-prune-finetune methods, we use the pretrained ResNet models provided by the Torchvision model zoo [1]. In this setup, CHEX runs for 120 training epochs to match the finetuing epochs of most of the previous methods. As shown in Table 4, our method achieves competitive top-1 accuracy when reducing the same amount of FLOPs compared to previous state-of-the-art pretrain-prune-finetune methods.

A2.5. Comparison with gradual pruning

To further evidence the necessity of the channel exploration via the repeated pruning-and-regrowing approach, we compare CHEX with gradual pruning, where the channel exploration is changed to an iterative pruning-training process

[1]https://pytorch.org/vision/stable/models.html

Model	Method	FLOPs	Top-1	Epochs
ResNet-18	Baseline	1.81G	69.4%	90
	PFP [18]	1.27G	67.4%	90+180
	SCOP [30]	1.10G	69.2%	90+140
	SFP [12]	1.04G	67.1%	100+100
	FPGM [14]	1.04G	68.4%	100+100
	CHEX	1.04G	69.2%	90+120
ResNet-34	Baseline	3.7G	73.3%	90
	SFP [12]	2.2G	71.8%	100+100
	FPGM [14]	2.2G	72.5%	100+100
	GFS [33]	2.1G	72.9%	90+150
	DMC [7]	2.1G	72.6%	90+400
	NPPM [6]	2.1G	73.0%	90+300
	SCOP [30]	2.0G	72.6%	90+140
	CHEX	2.0G	72.7%	90+120
ResNet-50	Baseline	4.1G	76.2%	90
	SFP [12]	2.4G	74.6%	100+100
	FPGM [14]	2.4G	75.6%	100+100
	GBN [34]	2.4G	76.2%	90+260
	LeGR [3]	2.4G	75.7%	90+60
	GAL [20]	2.3G	72.0%	90+60
	Hrank [19]	2.3G	75.0%	90+480
	SRR-GR [32]	2.3G	75.8%	90+150
	Taylor [26]	2.2G	74.5%	90+25
	C-SGD [4]	2.2G	74.9%	-
	SCOP [30]	2.2G	76.0%	90+140
	DSNet [17]	2.2G	76.1%	150+10
	EagleEye [16]	2.0G	76.4%	120+120
	CHEX	2.0G	76.8%	90+120
ResNet-101	Baseline	7.6G	77.4%	90
	SFP [12]	4.4G	77.5%	100+100
	FPGM [14]	4.4G	77.3%	100+100
	PFP [18]	4.2G	76.4%	90+180
	AOFP [5]	3.8G	76.4%	-
	NPPM [6]	3.5G	77.8%	90+300
	DMC [7]	3.3G	77.4%	90+400
	CHEX	3.0G	78.2%	90+120

Table 4. Compress ResNets starting from the pretrained models. All models are trained on the ImageNet dataset. "Epochs" are reported as: pretraining epochs plus all subsequent training epochs needed to obtain the final pruned model.

with gradually increased channel sparsity (but without regrowing). For a fair comparison, we apply the CSS pruning criterion, determine the number of channels in each layer by the batch-norm scaling factors, and use the single training pass from scratch when perform gradual pruning. As shown in Table 5, CHEX outperforms gradual pruning by 1.1% accuracy under the same training setup.

Method	FLOPs	Top-1
Gradual pruning	1.0G	74.9%
CHEX	1.0G	**76.0%**

Table 5. Comparison with gradual pruning. Results are based on pruning ResNet-50 by 75% FLOPs on ImageNet.

A2.6. Prune models with shortcut connections

We have experimented two strategies to deal with the shortcut connections: (1) Prune internal layers (e.g., the first two convolution layers in the bottleneck blocks of ResNet-50), leaving the layers with residual connections unpruned as [24, 36]; (2) Use group pruning as [34], where the channels connected by the shortcut connections are pruned simultaneously by summing up their CSS scores. As shown in Table 6, the first strategy gave better accuracy at less FLOPs, thus we adopted the first strategy in our CHEX method when pruning models with shortcut connections.

Method	FLOPs	Top-1
Group pruning	1.4G	75.9%
Prune internal layers	1.0G	76.0%

Table 6. Comparison of two strategies for pruning models with shortcut connections, using ResNet-50 on ImageNet as an example.

A2.7. Probabilistic sampling vs. deterministic selection for channel regrowing

In channel regrowing stages, the channels to regrow are sampled according to a probabilistic distribution derived from the channel orthogonality. The importance sampling encourages exploration in the regrowing stages, as the randomness gives a chance to sample channels other than the most orthogonal ones. Our ablation in Table 7 shows that the probabilistic sampling achieves better accuracy. than deterministic selection based on channel orthogonality.

Method	FLOPs	Top-1
Deterministic selection	1.0G	75.7%
Probabilistic sampling	1.0G	76.0%

Table 7. Comparison of two schemes for determining the channels to regrow. Results are based on pruning ResNet-50 by 75% FLOPs on ImageNet.

A2.8. Actual inference runtime acceleration

In the main paper, we chose FLOPs as our evaluation criteria in order to compare with the prior arts, the majority of which evaluate in terms of FLOPs and accuracy. Meanwhile, we have compared the actual inference throughput (images per second) of our compressed models against the unpruned models in Table 8. CHEX method achieves $1.8\times \sim 2.5\times$ actual runtime throughput accelerations on PyTorch with one NVIDIA V100 GPU in float32.

Model	ResNet-50			ResNet-101		
FLOPs reduction	0%	50%	75%	0%	50%	75%
Throughput (img/s)	1328	**2347**	**3259**	840	**1536**	**2032**

Table 8. Comparison of the actual inference runtime throughput.

A2.9. Comparison of training cost

In Figure 1, we compare the total training FLOPs of CHEX versus prior arts for pruning ResNet-50 on ImageNet. CHEX achieves higher accuracy at a fraction of the training cost. This is because CHEX obtains the sub-model in one training pass from scratch, circumventing the expensive pretrain-prune-finetune cycles.

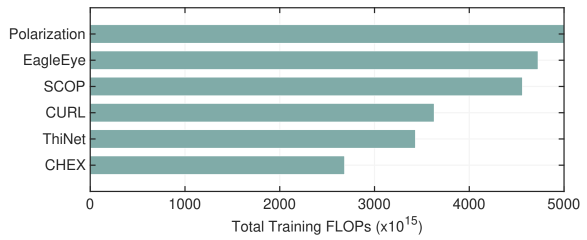

Figure 1. Comparison of the total training FLOPs for pruning ResNet-50 on ImageNet.

A3. Convergence analysis

A3.1. Problem formulation

Training deep neural networks can be formulated into minimizing the following problem:

$$\min_{\mathbf{W}\in\mathbb{R}^d} \quad F(\mathbf{W}) = \mathbb{E}_{x\sim\mathcal{D}}[f(\mathbf{W};x)] \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^d$ is the model parameter to be learned, and $f(\mathbf{W};x)$ is a non-convex loss function in terms of $\mathbf{W}$. The random variable x denotes the data samples that follow the distribution $\mathcal{D}$. $\mathbb{E}_{x\sim\mathcal{D}}[\cdot]$ denotes the expectation over the random variable x.

A3.2. Notation

We clarify several notions to facilitate the convergence analysis.

- $\mathbf{W}_t$ denotes the complete model parameter at the t-th iteration.
- $m_t \in \mathbb{R}^d$ is a mask vector at the t-th iteration.
- x_t is the data sampled at the t-th iteration.

A3.3. Algorithm formulation

With notations introduced in the above subsection, the proposed CHEX method can be generalized in a way that is more friendly for convergence analysis, see Algorithm 1.

Algorithm 1: CHEX (A math-friendly version)

```
1 Input: Initialize $\mathbf{W}_0$ and $m_0$ randomly ;
2 for iteration $t = 0, 1, \cdots, T$ do
3     Sample data $x_t$ from distribution $\mathcal{D}$ ;
4     Generate a mask $m_t$ following some rules ;
5     Update $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla f(\mathbf{W}_t \odot m_t; x_t) \odot m_t$
6 Output: The pruned model parameter $\mathbf{W}_T \odot m_T$;
```

Remark 1. Note that Algorithm 1 is quite general. It does not specify the rule to generate the mask m_t. This implies the convergence analysis established in Sec. 6 does not rely on what specific m_t is utilized. In fact, it is even allowed for m_t to remain unchanged during some period.

A3.4. Assumptions

We now introduce several assumptions on the loss function and the gradient noise that are standard in the literature.

Assumption 1 (SMOOTHNESS)**.** We assume $F(\mathbf{W})$ is L-smooth, i.e., it holds for any $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^d$ that

$$\|\nabla F(\mathbf{W}_1) - \nabla F(\mathbf{W}_2)\| \leq L\|\mathbf{W}_1 - \mathbf{W}_2\| \quad (2)$$

or equivalently,

$$\begin{aligned} &F(\mathbf{W}_1) - F(\mathbf{W}_2) \\ \leq\ &\langle \nabla F(\mathbf{W}_2), \mathbf{W}_1 - \mathbf{W}_2 \rangle + \frac{L}{2}\|\mathbf{W}_1 - \mathbf{W}_2\|^2 \end{aligned} \quad (3)$$

Assumption 2 (GRADIENT NOISE)**.** We assume

$$\mathbb{E}\{\nabla f(\mathbf{W}; x_t)\} = \nabla F(\mathbf{W}), \quad (4)$$

$$\mathbb{E}\|\nabla f(\mathbf{W}; x_t) - \nabla F(\mathbf{W})\|^2 \leq \sigma^2, \quad (5)$$

where $\sigma > 0$ is a constent. Moreover, we assume the data sample x_t is independent of each other for any t.

This assumption implies that the stochastic filter-gradient is unbiased and has bounded variance.

Assumption 3 (MASK-INCURRED ERROR)**.** It holds for any $\mathbf{W}$ and m_t that

$$\|\mathbf{W} - \mathbf{W} \odot m_t\|^2 \leq \delta^2 \|\mathbf{W}\|^2 \quad (6)$$

$$\|\nabla F(\mathbf{W}) - \nabla F(\mathbf{W}) \odot m_t\|^2 \leq \zeta^2 \|\nabla F(\mathbf{W})\|^2 \quad (7)$$

where constants $\delta \in [0, 1]$ and $\zeta \in [0, 1]$.

With (7), we have

$$\begin{aligned} &\zeta \|\nabla F(\mathbf{W})\| \\ \geq\ &\|\nabla F(\mathbf{W}) - \nabla F(\mathbf{W}) \odot m_t\| \\ \geq\ &\|\nabla F(\mathbf{W})\| - \|\nabla F(\mathbf{W}) \odot m_t\| \end{aligned} \quad (8)$$

which implies

$$\|\nabla F(\mathbf{W}) \odot m_t\|^2 \geq (1-\zeta)^2 \|\nabla F(\mathbf{W})\|^2 \quad (9)$$

for any $\mathbf{W}$ and m_t.

A3.5. Convergence analysis

Now we are ready to establish the convergence property.

Theorem 1 (CONVERGENCE PROPERTY)**.** *Under Assumptions 1 – 3, if learning rate* $\eta = \frac{\sqrt{2C_0}}{\sigma\sqrt{L(T+1)}}$ *in which* $C_0 = \mathbb{E}[F(\mathbf{W}_0)]$, *it holds that*

$$\begin{aligned} &\frac{1}{T+1}\sum_{t=0}^{T} \mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 \\ \leq\ &\frac{4\sigma\sqrt{LC_0}}{(1-\zeta)^2\sqrt{T+1}} + \frac{2L^2\delta^2}{(T+1)(1-\zeta)^2}\sum_{t=0}^{T}\mathbb{E}\|\mathbf{W}_t\|^2 \end{aligned} \quad (10)$$

Proof. With inequality (3) and Line 6 in Algorithm 1, it holds that

$$\begin{aligned} &F(\mathbf{W}_{t+1}) - F(\mathbf{W}_t) \\ \leq &-\eta\langle \nabla F(\mathbf{W}_t), [\nabla f(\mathbf{W}_t \odot m_t; x_t)] \odot m_t \rangle \\ &+ \frac{\eta^2 L}{2}\|[\nabla f(\mathbf{W}_t \odot m_t; x_t)] \odot m_t\|^2 \end{aligned} \quad (11)$$

With Assumption 2, it holds that

$$\begin{aligned} &\mathbb{E}\langle \nabla F(\mathbf{W}_t), \nabla f(\mathbf{W}_t \odot m_t; x_t) \odot m_t \rangle \\ \overset{(4)}{=}\ &\mathbb{E}\langle \nabla F(\mathbf{W}_t), \nabla F(\mathbf{W}_t \odot m_t) \odot m_t \rangle \\ =\ &\mathbb{E}\langle \nabla F(\mathbf{W}_t) \odot m_t, \nabla F(\mathbf{W}_t \odot m_t) \odot m_t \rangle \\ =\ &\frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t) \odot m_t\|^2 + \frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t) \odot m_t\|^2 \\ &- \frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t) \odot m_t - \nabla F(\mathbf{W}_t \odot m_t) \odot m_t\|^2 \\ \geq\ &\frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t) \odot m_t\|^2 + \frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t) \odot m_t\|^2 \\ &- \frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t) - \nabla F(\mathbf{W}_t \odot m_t)\|^2 \\ \overset{(2)}{\geq}\ &\frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t) \odot m_t\|^2 + \frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t) \odot m_t\|^2 \\ &- \frac{L^2}{2}\mathbb{E}\|\mathbf{W}_t - \mathbf{W}_t \odot m_t\|^2 \\ \overset{(6)}{\geq}\ &\frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t) \odot m_t\|^2 + \frac{1}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t) \odot m_t\|^2 \\ &- \frac{L^2\delta^2}{2}\mathbb{E}\|\mathbf{W}_t\|^2 \\ \overset{(9)}{\geq}\ &\frac{(1-\zeta)^2}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t)\|^2 + \frac{(1-\zeta)^2}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 \\ &- \frac{L^2\delta^2}{2}\mathbb{E}\|\mathbf{W}_t\|^2 \end{aligned} \quad (12)$$

Furthermore, with Assumption 2, it holds that

$$\begin{aligned} &\mathbb{E}\|[\nabla f(\mathbf{W}_t \odot m_t; x_t)] \odot m_t\|^2 \\ \leq\ &\mathbb{E}\|\nabla f(\mathbf{W}_t \odot m_t; x_t)\|^2 \end{aligned}$$

$$\overset{(5)}{\leq} \mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 + \sigma^2 \tag{13}$$

Substituting (12) and (13) into (11), we achieve

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{W}_{t+1}) - F(\mathbf{W}_t)] \\
\leq & -\frac{\eta(1-\zeta)^2}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t)\|^2 \\
& -\frac{\eta(1-\zeta)^2}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 \\
& +\frac{\eta L^2\delta^2}{2}\mathbb{E}\|\mathbf{W}_t\|^2 \\
& +\frac{\eta^2 L}{2}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 + \frac{\eta^2 L\sigma^2}{2} \\
\leq & -\frac{\eta(1-\zeta)^2}{4}\mathbb{E}\|\nabla F(\mathbf{W}_t)\|^2 \\
& -\frac{\eta(1-\zeta)^2}{4}\mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 \\
& +\frac{\eta L^2\delta^2}{2}\mathbb{E}\|\mathbf{W}_t\|^2 + \frac{\eta^2 L\sigma^2}{2}
\end{aligned} \tag{14}$$

where the last inequality holds by setting $\eta \leq \frac{(1-\zeta)^2}{2L}$. The above inequality will lead to

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\|\nabla F(\mathbf{W}_t)\|^2 + \mathbb{E}\|\nabla F(\mathbf{W}_t \odot m_t)\|^2 \\
\leq & \frac{4}{\eta(1-\zeta)^2(T+1)}\mathbb{E}[F(\mathbf{W}_0)] \\
& + \frac{2L^2\delta^2}{(1-\zeta)^2(T+1)}\sum_{t=0}^{T}\mathbb{E}\|\mathbf{W}_t\|^2 + \frac{2\eta L\sigma^2}{(1-\zeta)^2} \\
\leq & \frac{4\sigma\sqrt{LC_0}}{(1-\zeta)^2\sqrt{T+1}} + \frac{2L^2\delta^2}{(T+1)(1-\zeta)^2}\sum_{t=0}^{T}\mathbb{E}\|\mathbf{W}_t\|^2
\end{aligned} \tag{15}$$

where $C_0 = \mathbb{E}[F(\mathbf{W}_0)]$ and the last equality holds when $\eta = \frac{\sqrt{2C_0}}{\sigma\sqrt{L(T+1)}}$. The above inequality will lead to (10). □

A.4. Societal impact

Our method can effectively reduce the computation cost of diverse modern CNN models while maintaining satisfactory accuracy. This can facilitate the deployment of CNN models to real-world applications, such as pedestrian detection in autonomous driving and MRI image segmentation in clinic diagnosis. Moreover, our method does not increase the training cost compared to standard CNN model training. We provide a more affordable and efficient solution to CNN model compression, which is of high value for the community and society to achieve Green AI [28]. On the other hand, our method cannot prevent the possible malicious usage, which may cause negative societal impact.

A.5. Limitation

CHEX tends to work better for more over-parameterized CNN models. When the model has substantial redundancy, our method can obtain efficient sub-models that recover the original accuracy well. When compressing already under-parameterized CNN models, our method will still have noticeable accuracy loss, though such loss may still be less than the comparable methods (See Table 1 in Appendix A2.1).

CHEX is primarily evaluated on diverse computer vision (CV) tasks in this paper. More evaluations are required to verify the broader applicability of CHEX to other domains, such as natural language processing, and we leave it as one of our future works.

Finally, CHEX reflects the layer importance based on the scaling factors in the batch-norm layers. Although this technique can provide meaningful guidance in allocating the number of channels in the sub-models, deeper understanding on why this mechanism works is still an open question.

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. OFA-ResNet-50. `https://github.com/mit-han-lab/once-for-all`, 2020. 1, 2

[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. 1, 2

[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, 2

[4] 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

[5] 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. 2

[6] 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. 2

[7] 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. 1, 2

[8] 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. 1

[9] 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. 1

[10] 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. 1

[11] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 1

[12] 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. 2

[13] 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. 1

[14] 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. 2

[15] Yuenan Hou, Zheng Ma, Chunxiao Liu, Zhe Wang, and Chen Change Loy. Network pruning via resource reallocation. *arXiv preprint arXiv:2103.01847*, 2021. 1

[16] 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. 2

[17] 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. 1, 2

[18] 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. 2

[19] 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. 2

[20] 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. 2

[21] 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. 1

[22] 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. 1

[23] Zechun Liu, Xiangyu Zhang, Zhiqiang Shen, Zhe Li, Yichen Wei, Kwang-Ting Cheng, and Jian Sun. Joint multi-dimension pruning. *arXiv preprint arXiv:2005.08931*, 2020. 1

[24] 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. 3

[25] Sangkug Lym, Esha Choukse, Siavash Zangeneh, Wei Wen, Sujay Sanghavi, and Mattan Erez. Prunetrain: fast neural network training by dynamic sparse model reconfiguration. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2019. 1, 2

[26] 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. 2

[27] 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. 1

[28] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020. 5

[29] 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. 1

[30] 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. 2

[31] Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. *Proceedings of AAAI*, 2020. 1

[32] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14913–14922, 2021. 2

[33] 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, 2

[34] 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. 2, 3

[35] 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. 1, 2

[36] 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. 3