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 begins with 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 epochs 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 0.8~2.3% accuracy. Similarly, our method achieves superior accuracy when compressing EfficientNet-B0 with the same FLOPs reduction as the previous methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV2</td>
<td>Baseline</td>
<td>300M</td>
<td>72.2%</td>
</tr>
<tr>
<td></td>
<td>LeGR [3]</td>
<td>220M</td>
<td>71.4%</td>
</tr>
<tr>
<td></td>
<td>GFS [33]</td>
<td>220M</td>
<td>71.6%</td>
</tr>
<tr>
<td></td>
<td>MetaPruning [22]</td>
<td>217M</td>
<td>71.2%</td>
</tr>
<tr>
<td></td>
<td>DMCP [6]</td>
<td>211M</td>
<td>71.6%</td>
</tr>
<tr>
<td></td>
<td>AMC [13]</td>
<td>210M</td>
<td>70.8%</td>
</tr>
<tr>
<td></td>
<td>PFS [31]</td>
<td>210M</td>
<td>70.9%</td>
</tr>
<tr>
<td></td>
<td>JointPruning [23]</td>
<td>206M</td>
<td>70.7%</td>
</tr>
<tr>
<td></td>
<td>CHEX-1</td>
<td>220M</td>
<td>72.0%</td>
</tr>
<tr>
<td></td>
<td>DMC [7]</td>
<td>162M</td>
<td>68.4%</td>
</tr>
<tr>
<td></td>
<td>GFS [33]</td>
<td>152M</td>
<td>69.7%</td>
</tr>
<tr>
<td></td>
<td>LeGR [3]</td>
<td>150M</td>
<td>69.4%</td>
</tr>
<tr>
<td></td>
<td>JointPruning [23]</td>
<td>145M</td>
<td>69.1%</td>
</tr>
<tr>
<td></td>
<td>MetaPruning [22]</td>
<td>140M</td>
<td>68.2%</td>
</tr>
<tr>
<td></td>
<td>CHEX-2</td>
<td>150M</td>
<td>70.5%</td>
</tr>
<tr>
<td>EfficientNet-B0</td>
<td>Baseline</td>
<td>390M</td>
<td>77.1%</td>
</tr>
<tr>
<td></td>
<td>PEEL [15]</td>
<td>346M</td>
<td>77.0%</td>
</tr>
<tr>
<td></td>
<td>CHEX-1</td>
<td>330M</td>
<td>77.4%</td>
</tr>
<tr>
<td></td>
<td>DSNet [17]</td>
<td>270M</td>
<td>75.4%</td>
</tr>
<tr>
<td></td>
<td>CHEX-2</td>
<td>270M</td>
<td>76.2%</td>
</tr>
<tr>
<td></td>
<td>CafeNet-R [29]</td>
<td>192M</td>
<td>74.5%</td>
</tr>
<tr>
<td></td>
<td>CHEX-3</td>
<td>192M</td>
<td>74.8%</td>
</tr>
</tbody>
</table>

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-50D [1]. OFA firstly trains a supernet, then applies progressive shrinking in four dimensions, including number of layers,
Table 2. Comparison with GrowEfficient and PruneTrain on CIFAR10 and ImageNet datasets. All methods train from scratch with the same number of epochs.

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

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 finetuning epochs of most of the previous methods. As shown in Table 4, our method achieves competitive 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 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.

---

\(^1\) [https://pytorch.org/vision/stable/models.html](https://pytorch.org/vision/stable/models.html)
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group pruning</td>
<td>1.4G</td>
<td>75.9%</td>
</tr>
<tr>
<td>Prune internal layers</td>
<td>1.0G</td>
<td>76.0%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic selection</td>
<td>1.0G</td>
<td>75.7%</td>
</tr>
<tr>
<td>Probabilistic sampling</td>
<td>1.0G</td>
<td>76.0%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>ResNet-50</th>
<th>ResNet-101</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOPs reduction</td>
<td>0%</td>
<td>50%</td>
</tr>
<tr>
<td>Throughput (img/s)</td>
<td>1328</td>
<td>2347</td>
</tr>
</tbody>
</table>

Table 8. Comparison of the actual inference runtime throughput.

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.

A3. Convergence analysis

A3.1. Problem formulation

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

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

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

A3.2. Notation

We clarify several notions to facilitate the convergence analysis.

- $W_t$ 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.
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 \( F(W) \) is \( L \)-smooth, i.e., it holds for any \( W_1, W_2 \in \mathbb{R}^d \) that
\[
\|\nabla F(W_1) - \nabla F(W_2)\| \leq L \|W_1 - W_2\| \tag{2}
\]
or equivalently,
\[
F(W_1) - F(W_2) \leq \langle \nabla F(W_2), W_1 - W_2 \rangle + \frac{L}{2} \|W_1 - W_2\|^2 \tag{3}
\]
Assumption 2 (Gradient noise). We assume
\[
\mathbb{E}\{\nabla f(W; x_t)\} = \nabla F(W), \tag{4}
\]
\[
\mathbb{E}\|\nabla f(W; x_t) - \nabla F(W)\|^2 \leq \sigma^2, \tag{5}
\]
where \( \sigma > 0 \) is a constant. 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 \( W \) and \( m_t \) that
\[
\|W - W \circ m_t\|^2 \leq \delta^2 \|W\|^2 \tag{6}
\]
\[
\|\nabla F(W) - \nabla F(W) \circ m_t\|^2 \leq \zeta^2 \|\nabla F(W)\|^2 \tag{7}
\]
where constants \( \delta \in [0, 1] \) and \( \zeta \in [0, 1] \).

With (7), we have
\[
\zeta \|\nabla F(W)\| \geq \|\nabla F(W) - \nabla F(W) \circ m_t\| \geq \|\nabla F(W)\| - \|\nabla F(W) \circ m_t\| \tag{8}
\]
which implies
\[
\|\nabla F(W) \circ m_t\|^2 \geq (1 - \zeta)^2 \|\nabla F(W)\|^2 \tag{9}
\]
for any \( W \) 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 \( \eta = \frac{\sqrt{\frac{\sigma}{2L}}}{\sqrt{T + 1}} \) in which \( C_0 = \mathbb{E}[F(W_0)] \), it holds that
\[
\frac{1}{T+1} \sum_{t=0}^{T} \mathbb{E}\|\nabla F(W_t) \circ m_t\|^2 \leq \frac{4\sigma\sqrt{TC_0}}{(1-\zeta)^2\sqrt{T+1}} + \frac{2L^2\delta^2}{(T+1)(1-\zeta)^2} \sum_{t=0}^{T} \mathbb{E}\|W_t\|^2 \tag{10}
\]
Proof. With inequality (3) and Line 6 in Algorithm 1, it holds that
\[
F(W_{t+1}) - F(W_t) \leq -\eta \langle \nabla F(W_t) \circ m_t; x_t \rangle \circ m_t \tag{11}
\]
With Assumption 2, it holds that
\[
\mathbb{E}\{\nabla F(W_t), \nabla f(W_t \circ m_t; x_t) \circ m_t\} = \mathbb{E}\{\nabla F(W_t) \circ m_t, W_t \circ m_t \circ m_t\}
\]
\[
= \frac{1}{2} \mathbb{E}\|\nabla F(W_t) \circ m_t\|^2 + \frac{1}{2} \mathbb{E}\|\nabla F(W_t) \circ m_t \circ m_t\|^2
\]
\[
\geq \frac{1}{2} \mathbb{E}\|\nabla F(W_t) \circ m_t\|^2 + \frac{1}{2} \mathbb{E}\|\nabla F(W_t \circ m_t \circ m_t)\|^2
\]
\[
- \frac{1}{2} \mathbb{E}\|\nabla F(W_t) - \nabla F(W_t \circ m_t)\|^2
\]
\[
\geq \frac{1}{2} \mathbb{E}\|\nabla F(W_t) \circ m_t\|^2 + \frac{1}{2} \mathbb{E}\|\nabla F(W_t \circ m_t \circ m_t)\|^2
\]
\[
- \frac{L^2}{2} \mathbb{E}\|W_t - W_t \circ m_t\|^2
\]
\[
\geq \frac{1}{2} \mathbb{E}\|\nabla F(W_t) \circ m_t\|^2 + \frac{1}{2} \mathbb{E}\|\nabla F(W_t \circ m_t \circ m_t)\|^2
\]
\[
- \frac{L^2}\mathbb{E}\|W_t\|^2
\]
\[
\geq \frac{1}{2} \mathbb{E}\|\nabla F(W_t) \circ m_t\|^2 + \frac{1}{2} \mathbb{E}\|\nabla F(W_t \circ m_t \circ m_t)\|^2
\]
\[
- \frac{L^2\delta^2}{2} \mathbb{E}\|W_t\|^2 \tag{12}
\]
Furthermore, with Assumption 2, it holds that
\[
\mathbb{E}\|\nabla f(W_t \circ m_t; x_t) \circ m_t\|^2 \leq \mathbb{E}\|\nabla F(W_t \circ m_t; x_t) \|^2
\]
\[
\leq E[\|\nabla F(W_t \odot m_t)\|^2 + \sigma^2]
\]

Substituting (12) and (13) into (11), we achieve
\[
E[F(W_{t+1}) - F(W_t)] \\
\leq - \frac{\eta(1 - \zeta)^2}{2} E[\|\nabla F(W_t)\|^2] \\
- \frac{\eta L^2 \sigma^2}{2} E[\|W_t\|^2] \\
+ \frac{\eta^2 L^2}{2} E[\|\nabla F(W_t \odot m_t)\|^2] \\
+ \frac{\eta L^2 \sigma^2}{2} E[\|W_t\|^2] + \frac{\eta^2 L \sigma^2}{2}
\]

where the last inequality holds by setting \(\eta \leq \frac{(1 - \zeta)^2}{2L}\). The above inequality will lead to
\[
\frac{1}{T + 1} \sum_{t=0}^{T} E[\|\nabla F(W_t)\|^2] + E[\|\nabla F(W_t \odot m_t)\|^2] \\
\leq \frac{4}{(1 - \zeta)^2} E[F(W_0)] \\
+ \frac{2L^2 \sigma^2}{1 - \zeta^2} \sum_{t=0}^{T} E[\|W_t\|^2] + 2L \sigma^2 \\
\leq \frac{4\sigma \sqrt{L C_0}}{(1 - \zeta^2) \sqrt{T + 1}} + \frac{2L^2 \sigma^2}{1 - \zeta^2} \sum_{t=0}^{T} E[\|W_t\|^2]
\]

where \(C_0 = E[F(W_0)] \) and the last equality holds when \(\eta = \frac{\sqrt{2\sigma \sqrt{L C_0}}}{\sigma \sqrt{L (T + 1)}}\). The above inequality will lead to (10). \(\square\)

### 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


