ResRep: Lossless CNN Pruning via Decoupling Remembering and Forgetting

Xiaohan Ding 1,2  Tianxiang Hao 1,2  Jianchao Tan 3  Ji Liu 3  
Jungong Han 4  Yuchen Guo 1  Guiguang Ding 1,2 *

1 Beijing National Research Center for Information Science and Technology (BNRist)  
2 School of Software, Tsinghua University, Beijing, China  
3 AI platform department, Seattle AI lab, and FeDA lab, Kwai Inc.  
4 Computer Science Department, Aberystwyth University, SY23 3FL, UK

dxh17@mails.tsinghua.edu.cn beyondhtx,tanjianchaoustc,ji.liu.uwisc@gmail.com  
jungonghan77,yuchen.w.guo@gmail.com dinggg@tsinghua.edu.cn

Abstract

We propose ResRep, a novel method for lossless channel pruning (a.k.a. filter pruning), which slims down a CNN by reducing the width (number of output channels) of convolutional layers. Inspired by the neurobiology research about the independence of remembering and forgetting, we propose to re-parameterize a CNN into the remembering parts and forgetting parts, where the former learn to maintain the performance and the latter learn to prune. Via training with regular SGD on the former but a novel update rule with penalty gradients on the latter, we realize structured sparsity. Then we equivalently merge the remembering and forgetting parts into the original architecture with narrower layers. In this sense, ResRep can be viewed as a successful application of Structural Re-parameterization. Such a methodology distinguishes ResRep from the traditional learning-based pruning paradigm that applies a penalty on parameters to produce sparsity, which means all the parameters of some channels become small in magnitude. Ideally, if the parameters of pruned channels are small enough, the pruned model may deliver the same performance as before (i.e., after training but before pruning), which we refer to as perfect pruning.

Since both the training and pruning may degrade the performance, we evaluate a training-based pruning method from two aspects. 1) Resistance. The training phase tends to decrease the accuracy (referred to as training-caused damage) as it introduces some desired properties such as structured sparsity into the model, which may be harmful because the objective of optimization is changed and the parameters are deviated from the optima. We say a model has high resistance if the performance maintains high during training. 2) Prunability. When we prune the model into a smaller one after training, the properties obtained (e.g.,

1. Introduction

The mainstream techniques to compress and accelerate convolutional neural network (CNN) include sparsification [10, 18, 20], channel pruning [26, 27, 35], quantization [4, 5, 45, 68], knowledge distillation [28, 32, 41, 61], etc. Channel pruning [27] (a.k.a. filter pruning [34] or network slimming [42]) reduces the width (i.e., number of output channels) of convolutional layers to effectively reduce the number of floating-point operations (FLOPs) and memory footprint, which is complementary to the other model compression methods as it produces a thinner model of the original architecture with no custom structures or operations. However, as CNN’s representational capacity depends on the width of conv layers, it is difficult to reduce the width without performance drops. On practical CNN architectures like ResNet-50 [32] and large-scale datasets like ImageNet [6], lossless pruning with high compression ratio has long been considered challenging. For reasonable trade-off between compression ratio and performance, a typical paradigm (Fig. 1.A) [2, 3, 9, 36, 39, 62, 63] trains the model with magnitude-related penalty loss (e.g., group Lasso [57, 60]) on the conv kernels to produce structured sparsity, which means all the parameters of some channels become small in magnitude. Ideally, if the parameters of pruned channels are small enough, the pruned model may deliver the same performance as before (i.e., after training but before pruning), which we refer to as perfect pruning.

1) Resistance. The training phase tends to decrease the accuracy (referred to as training-caused damage) as it introduces some desired properties such as structured sparsity into the model, which may be harmful because the objective of optimization is changed and the parameters are deviated from the optima. We say a model has high resistance if the performance maintains high during training. 2) Prunability. When we prune the model into a smaller one after training, the properties obtained (e.g.,
many channels being close to zero) will reduce the pruning-caused damage. If the model endures a high pruning ratio with low performance drop, we say it has high prunability.

We desire both high resistance and prunability, but the traditional penalty-based paradigm naturally suffers from a resistance-prunability trade-off. For example, a strong group Lasso achieves high sparsity with great training-caused damage, while a weak penalty maintains the performance but results in low sparsity, hence great pruning-caused damage. Sect. 3.3 presents detailed analysis.

In this paper, we propose ResRep to address the above problem, which is inspired by the neurobiology research on remembering and forgetting. 1) Remembering requires the brain to potentiate some synapses but depotentiate the others, which resembles the training of CNN that makes some parameters large and some small. 2) Synapse elimination via shrinkage or loss of spines is one of the classical forgetting mechanisms [56] as a key process to improve efficiency in both energy and space for biological neural network, which resembles pruning. Neurobiology research reveals that remembering and forgetting are independently controlled by Rutabaga adenylyl cyclase-mediated memory formation mechanism and Rac-regulated spine shrinkage mechanism, respectively [16, 21, 59], indicating it is more reasonable to learn and prune by two decoupled modules.

Inspired by such independence, we propose to decouple the “remembering” and “forgetting”, which are coupled in the traditional paradigm as the conv parameters are involved in both the “remembering” (objective function) and “forgetting” (penalty loss) in order for them to achieve a trade-off. That is, the traditional methods force every channel to “forget”, and remove the channels that “forgot the most”. In contrast, we first re-parameterize the original model into “remembering parts” and “forgetting parts”, then apply “remembering learning” (i.e., regular SGD with the original objective) on the former to maintain the “memory” (original performance), and “forgetting learning” on the latter to “eliminate synapses” (zero out channels).

ResRep comprises two key components: Convolutional Re-parameterization (Rep, the methodology of decoupling and the corresponding equivalent conversion) and Gradient Resetting (Res, the updatae rule for “forgetting”). Specifically, we insert a compactor, which is a $1 \times 1$ conv, after the original conv layer we desire to prune. During training, we add penalty gradients to only the compactors, select some compactor channels and zero out their gradients derived from the objective function. Such a training process makes some channels of compactors very close to zero, which are removed with no pruning-caused damage. Then we equivalently convert the compactor together with the preceding conv into a single conv with fewer channels through some linear transformations (Eq. 8, 9). Note that this method readily generalizes to the common case where the original conv is followed by batch normalization (BN) [31]. In this case, we append the compactor after BN, and convert the conv-BN-convector sequence after training by first equivalently fusing the conv-BN into a conv with bias (Eq. 4).

Eventually, the resultant model has the same architecture as the original (i.e., no compactors) but narrower layers (Fig. 1.B). As the equivalent conversion from the training-time model into the pruned model relies on the equivalent conversion of the parameters, ResRep can be viewed as an application of Structural Re-parameterization [11, 15, 14, 13]. The other Structural Re-parameterization works improve the VGG-like architectures [15], basic conv layers [11, 13] or an MLP-style building block [14] with different well-designed structures all but via an ordinary training process. In contrast, ResRep not only constructs extra structures (i.e., compactors) that can be equivalently converted back (Rep) but also uses a custom training strategy (Res). As will be shown in Sect. 4.3, Rep and Res are both essential: Rep constructs some structures for Res to apply on without losing the original information; Res zeros out some channels so that Rep can make the resultant conv layer narrower. Notably, ResRep can also prune a fully-connected layer because it is equivalent to a $1 \times 1$ conv [46].

**Algorithm 1 Pipeline of ResRep channel pruning.**

1. **Input:** well-trained model $\mathcal{W}$
2. Construct the re-parameterized model $\hat{\mathcal{W}}$ with compactors. Initialize the compactors as identity matrices and the other parts with the original parameters of $\mathcal{W}$
3. for $i = 0$ to max training iterations do
4. Forward a batch through $\hat{\mathcal{W}}$, compute the loss using the original objective function, derive the gradients
5. Apply Gradient Resetting on the gradients of compactors only (Eq. 14)
6. Update $\hat{\mathcal{W}}$ with the reset gradients of compactors and the original gradients of the other parameters
7. end for
8. Delete the rows of compactors which are close to zero (e.g., with norm $\leq 10^{-5}$) in $\hat{\mathcal{W}}$. Equivalently convert the parameters of $\hat{\mathcal{W}}$ into $\mathcal{W}'$ (Eq. 4, 8, 3). Now $\mathcal{W}'$ has the same architecture as $\mathcal{W}$ but narrower layers
9. **Output:** pruned model $\mathcal{W}'$

ResRep features: 1) High resistance. To maintain the performance, ResRep does not change the loss function, update rule or any training hyper-parameters of the original model (i.e., the conv-BN parts). 2) High prunability. The compactors are driven by the penalty gradients to make many channels small enough to realize perfect pruning, even with a mild penalty strength. 3) Given the required global reduction ratio of FLOPs, ResRep automatically finds the appropriate eventual width of each layer with no prior knowledge, making it a powerful tool for CNN.
Figure 1: Traditional penalty-based pruning vs. ResRep. We prune a $3 \times 3$ layer with one input channel and four output channels for illustration. For the ease of visualization, we ravel the kernel $K \in \mathbb{R}^{4 \times 1 \times 3 \times 3}$ into a matrix $W \in \mathbb{R}^{4 \times 9}$. A) To prune some channels of $K$ (i.e., rows of $W$), we add a penalty loss on the kernel to the original loss, so that the gradients will make some rows smaller in magnitude, but not small enough to realize perfect pruning. B) ResRep constructs a compactor with kernel matrix $Q \in \mathbb{R}^{4 \times 4}$. Driven by the penalty gradients, the compactor selects some of its channels and generates a binary mask, which resets some of the original gradients of $Q$ to zero. After multiple iterations, those compactor channels with reset gradients become infinitely close to zero, which enables perfect pruning. Finally, the conv-BN-compactor sequence is equivalently converted into a regular conv layer with two channels. Blank rectangles indicate zero values.

structure optimization. 4) End-to-end training and easy implementation (Alg. 1). We summarize our contributions as:

- Inspired by the neurobiology research, we proposed to decouple “remembering” and “forgetting” for pruning.
- We proposed two techniques, Rep and Res, to achieve high resistance and prunability. They can be used separately and the combination delivers the best performance.
- We achieved state-of-the-art results on common benchmark models, including real lossless pruning on ResNet-50 on ImageNet with a pruning ratio of 54.5%.

2. Related Work

Pruning may refer to removing any parameters or structures from a network. Unstructured pruning [10, 18, 19, 20] can reduce the number of non-zero parameters but cannot realize speedup on common computing frameworks. Structured pruning removes some whole structures (e.g., neurons of fully-connected layers, 2D kernels, and channels), which is more friendly to hardware [40, 63]. Channel pruning is especially practical as it reduces not only the model size, the actual computations, but also the memory footprint. Pruning is related to the lottery ticket hypothesis [17]. For example, one may use ResRep to find the “winning” channels before training. Except for generic model pruning, pruning in specific contexts (e.g., the limited-data scenario [50]) has also attracted much attention.

Most of the channel pruning methods can be categorized into two families. **Pruning-then-finetuning** methods identify and prune the unimportant channels from a well-trained model by some measurements [1, 30, 34, 47, 51, 52, 53, 67], which may cause significant accuracy drop, and finetune it afterwards. Some methods repeat pruning-finetuning iterations to measure the importance and prune progressively [8, 37]. A major drawback is that the pruned models can be easily trapped into bad local minima, and sometimes cannot even reach a similar level of accuracy with a counterpart of the same structure trained from scratch [44]. This discovery highlights the significance of perfect pruning, which eliminates the need for finetuning. In this family,
PCAS [66] is the most related to ResRep, which identifies the unimportant channels by training attention modules appended after conv layers. Unlike ResRep, PCAS performs imperfect pruning and requires finetuning after removing the unimportant channels. Moreover, PCAS discards the attention modules after training, which results in more structural damage, while ResRep uses a mathematically equivalent transformation to obtain the final model structure without any performance drop. Learning-based pruning methods utilize a custom learning process to reduce the pruning-caused damage. Apart from the above-mentioned penalty-based paradigm to zero out some of the channels [2, 3, 9, 36, 39, 62, 63], some other methods prune via making some filters identical [12], meta-learning [43], adversarial learning [38], etc.

3. ResRep for Lossless Channel Pruning

3.1. Formulation and Background

We first introduce the formulation of conv and channel pruning. Let and be the output and input channels, be the kernel size, be the kernel parameter tensor, be the optional bias, and be the convolution operator, and which replicates and corresponding entries in the bias or following BN, if any, succeeding layer (conv). Let be the broadcast function for the index set of channels which survive pruning. Let and be the transpose function and be the channel pruning set by . In our experiments, we use which is found to be small enough to realize perfect pruning. After pruning, the compactor has fewer rows than columns, i.e., . In our experiments, we use to convert every conv-BN-compactor sequence into a conv layer with bias and no following BN, the “BN” in our notation can be safely viewed as a bias. After training with Gradient Resetting, we prune the resulting close-to-zero channels of compactors and convert the model into , which has the same architecture as but narrower layers. Concretely, for a specific compactor with kernel , we prune the channels with norm smaller than a threshold . Formally, we obtain the to-be-pruned set by . In our experiments, we use , which makes no difference.

Given Eq. 1 and the homogeneity of conv, we can verify

(5)

Then we seek for the formula to construct and so that

(6)

With the additivity of convolution, we arrive at

(7)

We note that very channel of is a constant matrix, thus every channel of is also a constant matrix. As the 1 × 1 conv with on only performs cross-channel recombination, we can merge into by recombining the entries in . Let be the transpose function (e.g., ) is , we present the formulas to construct and , which can be easily verified.

(8)

(9)

In practice, we convert and save the weights of the trained re-parameterized model, construct a model with the original architecture but narrower layers without BN, and use the saved weights for testing and deployment.

1Gradient Resetting will make some channels infinitely close to zero, as shown in Fig. 4, so setting or makes no difference.
3.3. Gradient Resetting

We describe how to produce structured sparsity in compactors while maintaining the accuracy, beginning by discussing the traditional usage of penalty on a specific kernel $K$ to make the magnitude of some channels smaller, i.e., $\|K_{P,:,:,:}\| \to 0$. Let $\Theta$ be the universal set of parameters, $X, Y$ be the data examples and labels, $L_{\text{perf}}(X, Y, \Theta)$ be the performance-related objective function (e.g., cross-entropy for classification). The traditional paradigm adds a penalty term $P(K)$ by a pre-defined strength factor $\lambda$,

$$L_{\text{total}}(X, Y, \Theta) = L_{\text{perf}}(X, Y, \Theta) + \lambda P(K), \quad (10)$$

where the common forms of $P$ include $L_1$ [34], $L_2$ [9], and group Lasso [39, 63]. Specifically, group Lasso is effective in producing channel-wise structured sparsity. In the following discussions, we denote a specific channel in $K$ by $F^{(j)} = K_{j,:,:,:}$. Then the group Lasso loss is formulated as

$$P_{\text{Lasso}}(K) = \sum_{j=1}^{D} ||F^{(j)}||_E,$$  \hspace{1cm} (11)

where $||F^{(j)}||_E$ is the Euclidean norm

$$||F||_E = \sqrt{\sum_{c=1}^{C} \sum_{p=1}^{K} \sum_{q=1}^{K} F^{2}_{c,p,q}}.$$  \hspace{1cm} (12)

With $G(F)$ as the gradient, we take the derivative,

$$G(F) = \frac{\partial L_{\text{total}}(X, Y, \Theta)}{\partial F} = \frac{\partial L_{\text{perf}}(X, Y, \Theta)}{\partial F} + \lambda \frac{F}{||F||_E}.$$  \hspace{1cm} (13)

The training dynamics of a specific channel $F$ are quite straightforward. Beginning from a well-trained model, $F$ resides near the local optima, thus the first term of Eq. 13 is close to 0 but the second is not, so $F$ is pushed closer to 0. If $F$ is important to the performance, the objective function will intend to maintain its magnitude, i.e., the first gradient term will compete against the second, thus $F$ will end up smaller than it used to be, depending on $\lambda$. Otherwise, taking the extreme case for example, if $F$ does not influence $L_{\text{perf}}$ at all, the first term will be 0, so $F$ will keep growing towards 0 by the second term. In other words, the performance-related loss and the penalty loss compete so that the resulting value of $F$ will reflect its importance, which we refer to as competence-based importance evaluation for convenience. However, we face a dilemma. **Problem A:** The penalty deviates the parameters of every channel from the optima of the objective function. Notably, a mild deviation may not bring negative effects, e.g., $L_2$ regularization can also be viewed as a mild deviation. However, with a strong penalty, though some channels are zeroed out for pruning, the remaining channels are also made too small to maintain the representational capacity, which is an undesired side-effect. **Problem B:** With mild penalty for the high resistance, we cannot achieve high prunability, because most of the channels merely become closer to 0 than they used to be, but not close enough for perfect pruning.

We propose to achieve high prunability with a mild penalty by resetting the gradients derived from the objective function. We introduce a binary mask $m \in \{0, 1\}$, which indicates whether we wish to zero out $F$. For the ease of implementation, we add no terms to the objective function (i.e., $L_{\text{total}} = L_{\text{perf}}$), simply derive the gradients as usual, and then manually apply the mask, add the penalty gradients and use the resultant gradients for SGD update:

$$G(F) \leftarrow \frac{\partial L_{\text{perf}}(X, Y, \Theta)}{\partial F} m + \lambda \frac{F}{||F||_E}.$$  \hspace{1cm} (14)

We will describe how to decide which channels to zero out (i.e., set mask values for multiple channels) in the next section. In this way, we have solved the above two problems. **A)** Though we add Lasso gradients to the objective-related gradients of every channel, which is equivalent to deviating the optima by adding Lasso loss to the original loss, the deviation is mild ($\lambda = 10^{-4}$ in our experiments) hence harmless to the performance. **B)** With $m = 0$, the first term no longer exists to compete against the second, thus even a mild $\lambda$ can make $F$ steadily move towards 0.

3.4. The Remembering Parts Remember Always, the Forgetting Parts Forget Progressively

If directly used on conv kernels, $Res$ brings a problem: some objective-related gradients encode the supervision information for maintaining the performance but are discarded. Intuitively, the parameters are forced to “forget” some useful information (gradients). Fortunately, $Rep$ is exactly the solution, which allows us to prune the compactors only, not the original conv layers. $ResRep$ only forces the compactors to “forget”, and all the other layers still focus on “remembering”, so we will not lose the information encoded in the gradients of the original kernels.

To combine $Res$ with $Rep$, we need to decide which channels of $Q$ to be zeroed out. When training the re-parameterized model, we add the Lasso gradients to the compactors only. After a few epochs, $||Q_j||$ will reflect the importance of channel $j$ (competence-based importance evaluation discussed in the Sect. 3.3), so we start to perform channel selection based on the value of $Q$. Let $n$ be the number of compactors, $m^{(i)}$ (a $D^{(i)}$-dimensional binary vector) be the mask for the $i$-th compactor, we define $t_j^{(i)} \in \mathbb{R}^{D^{(i)}}$ as the metric vector,

$$t_j^{(i)} = ||Q_j^{(i)}||_2, \quad \forall 1 \leq j \leq D^{(i)}.\quad (15)$$

For each time of channel selection, we calculate the metric values for every channel in every compactor and organize them as a mapping $M = \{(i, j) \rightarrow t_j^{(i)} | \forall 1 \leq i \leq n, 1 \leq j \leq D^{(i)}\}$. Then we sort the values of $M$ in ascending order, start to pick one at a time from the smallest,
and set the corresponding mask $m_j^{(i)}$ to 0. We stop picking when the reduced FLOPs \(^2\) reaches our target, or we have already picked $\theta$ (named the channel selection limit) channels. The mask values of unpicked channels are set to 1. The motivation is straightforward: following the discussions of competence-based importance evaluation, just like the traditional usage of penalty loss to compete against the original loss and select the channels with smaller norms, we use the penalty gradients to compete with the original gradients. Even better, all the metric values are 1 at the beginning (because every compactor kernel is initialized as an identity matrix), making it fair to compare them among different layers. We initialize $\theta$ as a small number, increase $\theta$ every several iterations and re-select channels to "forget" progressively, avoiding zeroing out too many channels at once. As Fig. 4 shows, those mask-0 channels will become very close to 0, thus the structured sparsity emerges in compactors.

4. Experiments

4.1. Datasets, Models and Settings

We use ResNet-50 and MobileNet [29] on ImageNet-1K. For the reproducibility, we follow the data augmentation of PyTorch official example [54] including random cropping and flipping. For ResNet-50, we use the official torchvision base model (76.15\% top-1 accuracy) [55] for the fair comparison with most competitors. For MobileNet, we train from scratch with an initial learning rate of 0.1, batch size of 512 and cosine learning rate annealing for 70 epochs. The top-1 accuracy is 70.78\%, slightly higher than that reported in the original paper. We use ResNet-56/110 on CIFAR-10 [33] with the standard data augmentation [22]: padding to $40 \times 40$, random cropping and flipping. We train the base models with batch size of 64 and the common learning rate schedule which is initialized as 0.1, multiplied by 0.1 at epoch 120 and 180, and terminated after 240 epochs. We count the FLOPs as multiply-adds, which is 4.09G for ResNet-50 [55], 569M for MobileNet, and 126M/253M for ResNet-56/110.

4.2. Pruning Results on ImageNet and CIFAR-10

We apply ResRep on ResNet-50 and MobileNet with the same hyper-parameters: $\lambda = 10^{-4}$, batch size=256, initial learning rate=0.01 and cosine annealing for 180 epochs. We set the channel selection limit $\theta = 4$ and $\theta \leftarrow \theta + 4$ every 200 batches and the first channel selection begins after 5 epochs. That is, after a 5-epoch “warm-up”, we pick up 4 channels with the lowest $t$ values among all the layers and then pick 4 more channels every 200 batches, until we reach the FLOPs reduction target. For the ease of comparison, we experiment with ResNet-50 for three times with FLOPs

\(^2\)We count the theoretical FLOPs of the model without current mask-0 channels as current FLOPs. Reduced FLOPs = original - current FLOPs.

![Figure 2: Training loss (left) and validation accuracy (right) of pruned-finetuned and ResRep on ResNet-50.](image)

reduction target of 54.5\% (1\% higher than FPGM [26]), 56.1\% (SASL [58]) and 62.1\% (HRank [35]), respectively, and MobileNet with 73.9\% to compare with MetaPruning [43]. Following most competitors, we prune the first $(1 \times 1)$ and second $(3 \times 3)$ conv layers in every residual block of ResNet-50, and every non-depthwise conv of MobileNet. Inspired by a prior work [10] which zeros out some gradients and utilizes momentum and weight decay for CNN sparsification, we raise the SGD momentum coefficient $\gamma$ from 0.9 (the default setting in most cases) to 0.99. Intuitively, the mask-0 channels continuously grow in the same direction (i.e., towards zero), and such a tendency accumulates in the momentum, thus the zeroing-out process can be accelerated by a larger momentum coefficient. For ResNet-56/110, the target layers include the first layers of residual block, and we use the same hyper-parameters as ImageNet except batch size of 64 and 480 training epochs.

Table. 1, 2 show the superiority of ResRep. Our results are average of 3 runs on ImageNet and 5 runs on CIFAR. On ResNet-50, ResRep achieves 0.00\% top-1 accuracy drop, which is the first to realize lossless pruning with such high pruning ratio (54.5\%), to the best of our knowledge. In terms of top-1 accuracy drop, ResRep outperforms SASL by 0.82\%, HRank by 3.32\% and all the other recent competitors by a large margin. On MobileNet, ResRep outperforms MetaPruning by 1.77\%. On ResNet-56/110, ResRep also outperforms, even though the comparison on accuracy drop is biased towards other methods, as our base models have higher accuracy (i.e., it is more challenging to prune a higher-accuracy model without accuracy degradation).

The final width of each target layer (Fig. 3) shows that ResRep discovers the appropriate final structure without any prior knowledge, given the desired global pruning ratio. Notably, ResRep chooses to preserve more channels at higher-level layers of ResNet-50 and MobileNet, but prunes aggressively on the last blocks of ResNet-56. An explanation is that rich higher-level features are essential for maintaining the fitting capacity on difficult task like ImageNet, while ResNet-56 suffers from over-fitting on CIFAR-10.

We construct a series of baselines and variants of ResRep for comparison (Table. 3). 1) The base model is further fine-tuned with the same learning rate schedule for 180 epochs, and the accuracy is merely lifted by 0.04\%, suggesting the performance of ResRep is not simply due to the effect of training settings. 2) We construct a uniformly shrunk base-
Table 1: Pruning results of ResNet-50 and MobileNet on ImageNet.

<table>
<thead>
<tr>
<th>Model</th>
<th>Result</th>
<th>Base</th>
<th>Pruned</th>
<th>Top-1</th>
<th>Top-5</th>
<th>FLOPs ↓%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Base</td>
<td>Pruned</td>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
<td></td>
</tr>
<tr>
<td>SFP [23]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>GAL-0.5 [38]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>NISP [67]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>Taylor-FO-BN [51]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>Channel Pr [27]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>HP [64]</td>
<td>76.01</td>
<td>76.01</td>
<td>92.93</td>
<td>92.93</td>
<td>76.01</td>
<td>92.93</td>
</tr>
<tr>
<td>MetaPruning [43]</td>
<td>76.6</td>
<td>76.6</td>
<td>92.93</td>
<td>92.93</td>
<td>76.6</td>
<td>92.93</td>
</tr>
<tr>
<td>ResRep</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>C-SGD (extension) [12]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>DCP [69]</td>
<td>76.01</td>
<td>76.01</td>
<td>92.93</td>
<td>92.93</td>
<td>76.01</td>
<td>92.93</td>
</tr>
<tr>
<td>C-SGD [7]</td>
<td>75.33</td>
<td>75.33</td>
<td>92.56</td>
<td>92.56</td>
<td>75.33</td>
<td>92.56</td>
</tr>
<tr>
<td>ThiNet [48]</td>
<td>75.30</td>
<td>75.30</td>
<td>92.20</td>
<td>92.20</td>
<td>75.30</td>
<td>92.20</td>
</tr>
<tr>
<td>SASL [58]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>ResRep</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>TRP [65]</td>
<td>75.90</td>
<td>75.90</td>
<td>92.70</td>
<td>92.70</td>
<td>75.90</td>
<td>92.70</td>
</tr>
<tr>
<td>LFPC [23]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>HRank [35]</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>ResRep</td>
<td>76.15</td>
<td>76.15</td>
<td>92.87</td>
<td>92.87</td>
<td>76.15</td>
<td>92.87</td>
</tr>
<tr>
<td>MetaPruning [43]</td>
<td>70.6</td>
<td>70.6</td>
<td>66.1</td>
<td>66.1</td>
<td>70.6</td>
<td>66.1</td>
</tr>
<tr>
<td>ResRep</td>
<td>70.78</td>
<td>70.78</td>
<td>89.78</td>
<td>89.78</td>
<td>70.78</td>
<td>89.78</td>
</tr>
</tbody>
</table>

Table 2: Pruning results of ResNet-56/110 on CIFAR-10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Result</th>
<th>Base</th>
<th>Pruned</th>
<th>Top-1</th>
<th>Top-5</th>
<th>FLOPs ↓%</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMC [25]</td>
<td>92.8</td>
<td>92.8</td>
<td>91.9</td>
<td>91.9</td>
<td>92.8</td>
<td>92.8</td>
</tr>
<tr>
<td>FPGM [26]</td>
<td>93.59</td>
<td>93.59</td>
<td>93.26</td>
<td>93.26</td>
<td>93.59</td>
<td>93.59</td>
</tr>
<tr>
<td>SFP [24]</td>
<td>93.59</td>
<td>93.59</td>
<td>93.35</td>
<td>93.35</td>
<td>93.59</td>
<td>93.59</td>
</tr>
<tr>
<td>LFPC [23]</td>
<td>93.59</td>
<td>93.59</td>
<td>93.24</td>
<td>93.24</td>
<td>93.59</td>
<td>93.59</td>
</tr>
<tr>
<td>ResRep</td>
<td>93.71</td>
<td>93.71</td>
<td>93.71</td>
<td>93.71</td>
<td>93.71</td>
<td>93.71</td>
</tr>
<tr>
<td>TRP [65]</td>
<td>93.14</td>
<td>93.14</td>
<td>91.62</td>
<td>91.62</td>
<td>93.14</td>
<td>93.14</td>
</tr>
<tr>
<td>ResRep</td>
<td>93.71</td>
<td>93.71</td>
<td>92.66</td>
<td>92.66</td>
<td>93.71</td>
<td>93.71</td>
</tr>
</tbody>
</table>

Table 3: ResNet-50 baselines and variants of ResRep.

<table>
<thead>
<tr>
<th>Model</th>
<th>Result</th>
<th>Top-1 acc</th>
<th>FLOPs ↓%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Base model finetuned</td>
<td>76.19</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2) Uniformly shrunk baseline</td>
<td>74.39</td>
<td>55.4</td>
<td></td>
</tr>
<tr>
<td>3) Pruned-finetuned</td>
<td>74.66</td>
<td>54.5</td>
<td></td>
</tr>
<tr>
<td>4) Vector re-parameterization</td>
<td>75.57</td>
<td>54.5</td>
<td></td>
</tr>
<tr>
<td>5) Momentum on compactors as 0.9</td>
<td>75.05</td>
<td>45.1</td>
<td></td>
</tr>
</tbody>
</table>

ResRep is 74.39%, which is 1.58% lower than our 56.1%-pruned model.

We compare ResRep against a classic pruning-then-finetuning method by sorting all the channels of base model by their Euclidean norm, pruning from the smallest until 54.5% reduction ratio, and finetuning the resultant model with the same 180-epoch setting. Not surpris-
ingly, the finetuned model is hardly better than the shrunk baseline trained from scratch. The training loss and validation accuracy (Fig. 2) shows the pruned model recovers quickly but cannot reach a comparable accuracy as the 54.5%-pruned ResRep model. This observation is consistent with a prior study that a pruned model may be easily trapped into bad local minima [44]. 4) A straightforward alternative of re-parameterization to verify the significance of Convolutional Re-parameterization. It changes the form of compactor from $1 \times 1$ conv to a $D$-dimensional trainable vector (i.e., a channel-wise scaling layer) initialized as 1, and the Lasso penalty naturally degrades to L1. After training with the same settings as the 54.5%-pruned model for 180 epochs, the final accuracy is 75.57%, which is 0.58% lower. Intuitively, re-parameterization with $1 \times 1$ conv folds the original kernel into a lower-dim kernel (i.e., re-combines the channels), but a vector simply deletes some channels. 5) We verify the necessity of 0.99-momentum by setting the momentum of compactors as 0.9, and fewer channels end up close to zero (below $\epsilon = 10^{-5}$). Though a higher momentum reduces parameters faster, it is not necessary if longer training time is acceptable.

### 4.3. Ablation Studies

We then perform controlled experiments with the same training configurations as described above on ResNet-56 to evaluate Rep and Res separately. As the baseline, we adopt the traditional paradigm by directly adding Lasso loss (Eq. 11) on all the target layers. With $\lambda \in \{0.3, 0.03, 0.003, 0.001\}$, we obtain four models with different final accuracy: 69.81%, 87.09%, 92.65%, 93.69%. To realize perfect pruning on each trained model, we attain the minimal structure by removing the channels one at a time until the accuracy drops below the original (i.e., pruning any one more channel of the minimal structure will decrease the accuracy). Then we record the FLOPs reduction of the minimal structures: 81.24%, 71.94%, 57.56%, 28.31%. We test Rep but no Res by applying Lasso loss on the compactors with varying $\lambda$ to achieve comparable FLOPs reduction as baselines. And with Res but no Rep, we directly apply Gradient Resetting on the original conv kernels, targeting at the same FLOPs reduction as the four baseline models. Then we experiment with the full-featured ResRep. As shown in the left of Fig. 4 (the baseline data point of (81.24%, 69.81%) is ignored for better readability), Res and Rep deliver better final accuracy than the baselines, and perform even better when combined.

We investigate into the training process by saving the parameters of the $\lambda = 0.03$ baseline every 5 epochs. After training, we obtain the minimal structure, turn back to prune each saved model into the minimal structure, and report the accuracy before and after pruning. For ResRep, we do the same but on the compactors instead of the original conv layers. Fig. 4 (middle) shows that the baseline accuracy drops drastically because of the side-effects brought by strong Lasso, which implies low resistance. In contrast, the original accuracy of ResRep maintains on a high level. The pruning-caused damage (original accuracy minus pruned accuracy) is great for both the baseline and ResRep models at the beginning but reduces as the sparsity emerges. The pruned accuracy of baseline improves slowly and unstably due to the competence of two losses.

For each saved model, we also collect the quadratic sum of parameters which survive at last as well as the quadratic sum of those finally pruned, according to the final minimal structure. Fig. 4 (right, note the logarithmic scale) shows that the parameters of baseline soon become too small to maintain the performance, which explains the poor resistance. For ResRep, the magnitude of survived parameters decreases but maintains on a high level due to the mild penalty, and those to-be-pruned (i.e., mask-0) parameters drop steadily and soon become very close to zero, which explains the high resistance and high prunability.

### 5. Conclusion

The effectiveness of ResRep suggests that decomposing the traditional learning-based pruning into “performance-oriented learning” and “pruning-oriented learning” may be a promising research direction. As a successful application of Structural Re-parameterization, ResRep uses the methodology of constructing extra structures that can be converted back, which enables to adopt some custom techniques (an update rule on the compactors only, in this case). Such a methodology may be useful in other research areas.
References


