ACDC: WEIGHT SHARING IN ATOM-COEFFICIENT DECOMPOSED CONVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNNs) are known to be significantly over-parametrized, and difficult to interpret, train, and adapt. In this work, we introduce a structural regularization across convolutional kernels in a CNN. In our approach, each convolution kernel is first decomposed into 2D dictionary atoms linearly combined by coefficients. The widely observed correlation and redundancy in a CNN hint a common low-rank structure among the decomposed coefficients, which is here further supported by our empirical observations. We then explicitly regularize CNN kernels by enforcing decomposed coefficients to be shared across sub-structures, while leaving each sub-structure with only its own dictionary atoms, a few hundreds of parameters typically, which leads to dramatic model reductions. We explore models with sharing across different sub-structures to cover a wide range of trade-offs between parameter reduction and expressiveness. Our proposed regularized network structures open the door to better interpreting, training, and adapting deep models. We validate the flexibility and compatibility of our method by image classification experiments on multiple datasets and underlying network structures, and show that CNNs now maintain performance with dramatic reduction in parameters and computations, e.g., only 5% parameters are used in a ResNet-18 to achieve comparable performance. Further experiments on few-shot classification show that faster and more robust task adaptation is obtained in comparison with models with standard convolutions.

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have achieved remarkable progresses on solving challenging tasks. The successes stimulate research directions that further improve CNNs from various angles, including network structures (He et al., 2016; Howard et al., 2017; Ma et al., 2018; Simonyan & Zisserman, 2014; Zagoruyko & Komodakis, 2016; Zhang et al., 2018), fast adaptations (Finn et al., 2017; Lopez-Paz & Ranzato, 2017; Shin et al., 2017), parameter efficiency (Cheng et al., 2017; Han et al., 2015a; Luo et al., 2017; Savarese & Maire, 2019; Yang et al., 2019; Wang et al., 2018b), and interpretability (Selvaraju et al., 2017; Zhou et al., 2016). With the trend of deeper and wider network structures with hundreds of millions of parameters, such investigations become even more pressing. The aforementioned challenges can be partially attributed to the under-regularized structures of convolutional kernels in a CNN, which are typically of very high dimensions and trained independently from random initialization. While recent works on efficient convolution operations (Chollet, 2017; Howard et al., 2017) alleviate the long recognized over-parametrization problem of deep CNNs, kernels across different convolutional layers are still modeled as isolated and independent groups of parameters, among which interactions only happen during feature and gradient propagation. Modeling kernels by shared structures has been empirically studied (Ha et al., 2016; Savarese & Maire, 2019), which sheds the light on explicitly modeling the underlying common structures across kernels, and confirms the widely observed redundancies in deep network parameters (Michel et al., 2019; Raghu et al., 2017). Studies on deep representations (Kornblith et al., 2019; Morcos et al., 2018; Raghu et al., 2017) suggest that, under certain linear transforms, deep features across layers are actually highly correlated. Such observations, together with the well recognized redundancies in parameters, motivate us to further exploit such correlation to enforce explicit structural regularizations over kernels. The work here presented provides a fundamental plug-and-play framework to introduce structure in convolution kernels via coefficient sharing within layers, resulting in significantly smaller and more interpretable networks with maintained or even improved performance.

We first perform atom-coefficient decomposition to convolution kernels, in which each kernel is decomposed as 2D dictionary atoms linearly combined by coefficients. A standard convolution layer can now be decomposed into two: a dictionary atom sub-layer involving spatial-only convolution with the dictionary atoms, followed by a coefficient sub-layer that linearly combines feature channels from the atom layer. Due to the underlying cross-layer correlations, after we properly align the outputs of both sub-layers across the network's multiple layers through canonical correlation analysis (CCA), we obtain a low rank structure for those dictionary coefficients. This observation hints us to enforce shared coefficients across sub-structures, e.g., layers. By sharing coefficients, each sub-structure is now left with only dictionary atoms, which typically include only a few hundreds of parameters and lead to dramatic model reduction. The focus of the paper is to introduce, derive, and fully explore such atom-coefficient decomposed convolution (ACDC) as a structural regularization to convolution kernels. The easily constructed variants, e.g., with different numbers of dictionary atoms and coefficients sharing across different substructures, enable a wide coverage of trade-offs between parameter reduction and model expressiveness. The explicitly regularized structures open the door to better interpreting, training, and adapting deep models.

We perform extensive experiments on standard image classification datasets, and show that, by using variants of ACDC as plug-and-play replacements to the standard convolution in various off-the-shelf network architectures, different degrees of model reductions are achieved with comparable or even better accuracy. Some ACDC variants can substantially reduce the overall computation of a deep CNN. Further experiments on few-shot classification demonstrate its fast adaptation across tasks.

Our main contributions are summarized as follows:

- We introduce ACDC, a plug-and-play replacement to the standard convolution that achieves a structural regularization for CNN kernels for better interpretability, training, and adaptations.
- Highlighting the remarkable flexibility, we introduce variants of ACDC constructed easily by sharing coefficient within different network sub-structures. Such variants lead to significant parameters reductions at negligent or not at all performance degradation.
- We validate the effectiveness of ACDC by plug-playing them into modern CNN architectures for various tasks.

2 Atom-Coefficient Decomposed Convolution

In this section, we start with a brief introduction of atom-coefficient decomposition motivated by dictionary learning and recent works on decomposed convolutional kernels. We then introduce the general idea of ACDC hinted by both the well recognized over-parametrization problem and the underlying cross-layer correlations of CNNs. Based on the idea of coefficients sharing enforced across network sub-structures, we describe in details how variants of ACDC are constructed as plug-and-play replacements to the standard convolution.

2.1 Convolutional Kernel Decomposition

Previous works have shown that a convolutional kernel in a CNN can be decomposed as a linear combination of pre-fixed basis (Qiu et al., 2018). In ACDC, we adopt a similar decomposition as shown in Figure 1, in which a convolutional kernel is represented as a linear combination of trainable 2D dictionary atoms. After decomposition, a convolution layer with c-channel output $\mathbf{Y}$ and c'-channel input $\mathbf{X}$ becomes

$$\mathbf{Y} = \mathbf{K} * \mathbf{X}, \quad \mathbf{K} = \mathbf{A}\mathbf{D}, \tag{1}$$

where * denotes the convolution operation. As illustrated in Figure 1, in (1), a convolutional kernel $\mathbf{K} \in \mathbb{R}^{c \times c' \times l \times l}$, which can be seen as a stack of $c \times c'$ 2D convolutional filters with the size of $l \times l$, is reconstructed by multiplying m 2D dictionary atoms $\mathbf{D} \in \mathbb{R}^{m \times l \times l}$ with the corresponding linear coefficients $\mathbf{A} \in \mathbb{R}^{c \times c' m}$. Note that square kernels are assumed here for simplicity, while all kernel shapes are supported. Since both convolution and tensor multiplication are commutative, a convolutional layer can now be decomposed into two:

- A dictionary *atom sub-layer* where each atom involves spatial-only convolution with the dictionary atoms, i.e., $\mathbf{Z} \in \mathbb{R}^{c'm \times h \times w} = \mathbf{D} * \mathbf{X}$;

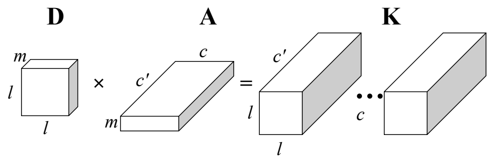

Figure 1: Illustration of the atom-coefficient decomposition. A convolutional kernel $\mathbf{K}$ with $c \times c'$ filters is reconstructed by multiplying m 2D dictionary Atoms with sizes $l \times l$ and coefficients $\mathbf{A} \in \mathbb{R}^{c \times c' \times m}$.

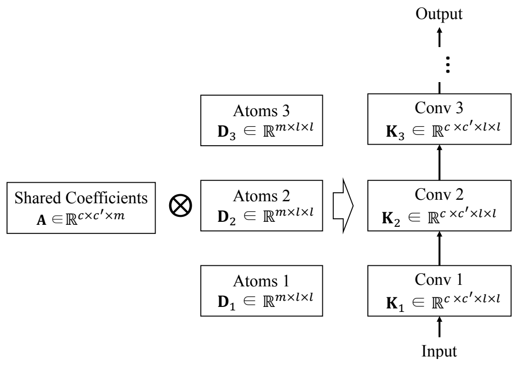
Output
Atoms 3 $\mathbf{D}_3 \in \mathbb{R}^{m \times l \times l}$
Conv 3 $\mathbf{K}_3 \in \mathbb{R}^{c \times c' \times l \times l}$
Shared Coefficients $\mathbf{A} \in \mathbb{R}^{c \times c' \times m}$
$\otimes$
Atoms 2 $\mathbf{D}_2 \in \mathbb{R}^{m \times l \times l}$
Conv 2 $\mathbf{K}_2 \in \mathbb{R}^{c \times c' \times l \times l}$
Atoms 1 $\mathbf{D}_1 \in \mathbb{R}^{m \times l \times l}$
Conv 1 $\mathbf{K}_1 \in \mathbb{R}^{c \times c' \times l \times l}$
Input

Figure 2: Illustration of *ACDC-net* deployed in a simple CNN. $\otimes$ denotes matrix multiplication. Intermediate features are omitted.

- A linear *coefficient sub-layer* that linearly combines feature channels from the *atom sub-layer*: $\mathbf{Y} \in \mathbb{R}^{c \times h \times w} = \mathbf{AZ}$. Note that $\mathbf{Z}$ here denotes *atom sub-layer* outputs, and stride 1 and same padding are assumed for the sake of discussion.

2.2 Correlation and Redundancy: The Motivation Behind

Deep CNNs are long recognized to be over-parametrized. The very deep layers in modern CNN structures (He et al., 2016; Huang et al., 2017; Zagoruyko & Komodakis, 2016) and the high-dimensional kernels with little structural regularizations lead to hundreds of millions of parameters. Such over-parametrization problem is also observed in the studies of deep representations (Raghu et al., 2017), and empirically alleviated by new network structures (Chollet, 2017; Howard et al., 2017), network compressions, and parameter reduction methods (Ha et al., 2016; Savarese & Maire, 2019).

Meanwhile, recent studies on deep representations (Kornblith et al., 2019; Morcos et al., 2018; Raghu et al., 2017) have shown that there exists obvious correlations for features across layers within a CNN after proper linear alignments. Such observations are also supported by the success of deep network with residual learning (He et al., 2016), which explicitly formulates the layers as learning residual functions with reference to the layer inputs. The correlation across features motivate us to explore and exploit correlations across kernels for structural regularizations.

We present here a motivating experiment on MNIST by applying CCA alignments as in (Raghu et al., 2017) to the *atom sub-layer* outputs and the *coefficient sub-layer* outputs of layer i and layer j. Note that no parameter sharing is imposed here, and the network reports the same testing accuracy before and after kernel decomposition. Formally, c, m, d, and hw denote the number of channels, number of dictionary atoms, test set size, and the 2D feature dimensions, respectively. The *atom sub-layer* outputs of the i-th and j-th layer, $\mathbf{Z}_i$ and $\mathbf{Z}_j \in \mathbb{R}^{cm \times dhw}$, are firstly aligned by linear transformations $\mathbf{P}_i$ and $\mathbf{P}_j \in \mathbb{R}^{cm \times cm}$ that maximize the correlation $\rho_z = \max\limits_{\mathbf{P}_i, \mathbf{P}_j} corr(\mathbf{P}_i \mathbf{Z}_i, \mathbf{P}_j \mathbf{Z}_j)$. And similarly, the *coefficient sub-layer* outputs of both layers, $\mathbf{Y}_i$ and $\mathbf{Y}_j \in \mathbb{R}^{c \times dhw}$, are aligned by $\mathbf{Q}_i$ and $\mathbf{Q}_j \in \mathbb{R}^{c \times c}$ that maximize the correlation $\rho = \max\limits_{\mathbf{Q}_i, \mathbf{Q}_j} corr(\mathbf{Q}_i \mathbf{Y}_i, \mathbf{Q}_j \mathbf{Y}_j)$. Omitting the layer indexes, the feed forwards of both layers can be rewritten as

$$\mathbf{Y} = \mathbf{QAP}^{-1}\mathbf{P}(\mathbf{D} * \mathbf{X}). \tag{2}$$

By merging the transform into the coefficients $\mathbf{A}$ by $\widetilde{\mathbf{A}} = \mathbf{QAP}^{-1}$, we obtain 'aligned coefficients' $\widetilde{\mathbf{A}}_i$ and $\widetilde{\mathbf{A}}_j$, that reside in a low rank structure reflected by the very similar effective ranks of $\widetilde{\mathbf{A}}_i$ and $[\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{A}}_j]$. For example, in our MNIST experiment using a 4-layer 32-channel CNN, out of the 6 possible (i, j) pairs, the average effective rank of $\widetilde{\mathbf{A}}_i$ and $[\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{A}}_j]$ are 31.98 and 38.56, respectively. Our observations agree with and further support recent studies on cross-layer feature correlations (Kornblith et al., 2019; Morcos et al., 2018; Raghu et al., 2017). Motivated by such empirical observations, we propose to enforce shared dictionary coefficients across layers, and we further extend the sharing to other network sub-structures, e.g., groups of filters within a layer.

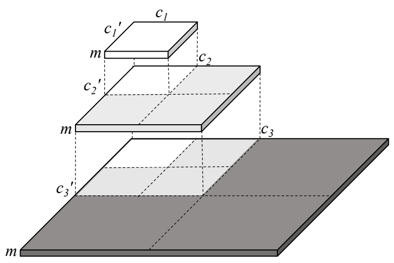

Figure 3: Illustration on how coefficients are shared across three layers with increasing numbers of channels. The shared coefficients are initialized with the largest dimensions required.

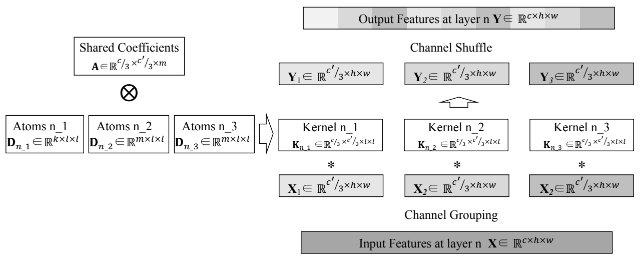
Output Features at layer n $\mathbf{Y} \in \mathbb{R}^{c \times h \times w}$
Shared Coefficients $\mathbf{A} \in \mathbb{R}^{c/3 \times c'/3 \times m}$
$\otimes$
Channel Shuffle
$\mathbf{Y}_1 \in \mathbb{R}^{c'/3 \times h \times w}$
$\mathbf{Y}_2 \in \mathbb{R}^{c'/3 \times h \times w}$
$\mathbf{Y}_3 \in \mathbb{R}^{c'/3 \times h \times w}$
Atoms n_1 $\mathbf{D}_{n\_1} \in \mathbb{R}^{k \times l \times l}$
Atoms n_2 $\mathbf{D}_{n\_2} \in \mathbb{R}^{m \times l \times l}$
Atoms n_3 $\mathbf{D}_{n\_3} \in \mathbb{R}^{m \times l \times l}$
Kernel n_1 $\mathbf{K}_{n\_1} \in \mathbb{R}^{c/3 \times c'/3 \times l \times l}$
Kernel n_2 $\mathbf{K}_{n\_2} \in \mathbb{R}^{c/3 \times c'/3 \times l \times l}$
Kernel n_3 $\mathbf{K}_{n\_3} \in \mathbb{R}^{c/3 \times c'/3 \times l \times l}$
$*$
$*$
$*$
$\mathbf{X}_1 \in \mathbb{R}^{c'/3 \times h \times w}$
$\mathbf{X}_2 \in \mathbb{R}^{c'/3 \times h \times w}$
$\mathbf{X}_2 \in \mathbb{R}^{c'/3 \times h \times w}$
Channel Grouping
Input Features at layer n $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$

Figure 4: *ACDC with grouping* with three groups at layer n. The input feature is first equally divided into groups (denoted as boxes with different grey scales), each of which is convolved with one group of filters reconstructed by multiplying the corresponding filter dictionary atoms and the shared coefficients. The output of three groups are combined by channel shuffle.

2.3 COEFFICIENTS SHARING ACROSS LAYERS

Based on the observations above and (1), networks with ACDC are constructed by directly sharing coefficients $\mathbf{A}$ within sub-structures as illustrated in Figure 2. We introduce two variants named *ACDC-net* and *ACDC-block*.

The simplest variant, *ACDC-net*, is constructed by enforcing common coefficients across all layers in a CNN. Formally, given a N-layers CNN, the n-th convolutional kernel is constructed by

$$\mathbf{K}_n = \mathbf{A}\mathbf{D}_n, \forall n = 1, \dots, N. \tag{3}$$

Assuming all layers have identical channel number with $c' = c$, the amount of parameters is reduced from $c^2 c^2 N$ to $c^2 m + Nkl^2$. An illustration of the *ACDC-net* is shown in Figure 2.

ACDC-block is a relaxed version of *ACDC-net*. Instead of enforcing common coefficients across all layers, we allow the sharing to happen among a few consecutive layers in a network. We refer a group of consecutive layers with identical number of output channels and identical feature sizes as a *block* in a deep network, and implement *ACDC-block* by enforcing coefficient sharing in each block. For example, adopting *ACDC-block* to a VGG16 network (Simonyan & Zisserman, 2014) is implemented by sharing coefficients within 5 groups, each of which consists of convolutional layers with 64, 128, 256, 512, 512 channels, respectively.

In practice, convolution layers within a network can have different numbers of channels. When sharing coefficients across layers with different channels numbers, we initialize the dimensions of the shared coefficients to be the largest dimensions needed by the corresponding layers. For example, given a N-layer CNN with convolutional kernels $\{\mathbf{K}_n \in \mathbb{R}^{c_n \times c'_n \times l \times l}; n = 1, \dots, N\}$, *ACDC-net* is constructed by initializing the shared coefficient as $\mathbf{A} \in \mathbb{R}^{c_{max} \times c'_{max} \times m}$, where $c_{max} = \max\{c_n; n = 1, \dots, N\}$ and $c'_{max} = \max\{c'_n; n = 1, \dots, N\}$. The kernels with fewer channels are reconstructed by multiplying the dictionary atoms with a subset of the shared coefficients $\mathbf{K}_n = \mathbf{A}[1 : c_n, 1 : c'_n, 1 : m]\mathbf{D}_n$. A 3-layer illustration with progressively increased channels is shown in Figure 3. Such a design choice is motivated by multi-scale decomposition (Mallat, 1999), and proves to be highly effective with our extensive experimental validation.

2.4 COEFFICIENTS SHARING ACROSS FILTER GROUPS

While both *ACDC-net* and *ACDC-block* remarkably reduce the number of parameters, the total number of coefficients largely depends on the highest channel number in a deep network. For example, a ResNet-18 or a VGG-16 network have up to 512 channels in the last few layers, which become the bottleneck of parameter efficiency. Meanwhile, observations in (Raghu et al., 2017) show that the representation learned at a layer is not fully determined by the number of neurons in the layer, and studies in (Michel et al., 2019) reveal the existence of redundancy within a single layer. Those observations stimulate us to explore weight sharing within a layer, where redundancy especially in high-dimensional layers can be further squeezed by sharing parameters within groups of filters in a convolutional kernel.

By breaking down the smallest sharing unit from a layer to part of a layer, we propose *ACDC with grouping*, in which a high-dimensional convolutional layer can now be separated into several groups

Table 1: Comparisons on CIFAR-10 with parameter sizes, parameter reduction rates, and test error. Those with higher than baseline accuracies but fewer parameters are marked in bold.

Architectures	m	s	VGG16 Size	VGG16 Error	ResNet18 Size	ResNet18 Error	WRN-40-4 Size	WRN-40-4 Error
Baseline	-	-	14.72M	6.2	11.17M	5.8	8.90M	4.97
ACDC-net	8	-	2.11M (85.7%↓)	**5.67**	2.28M (79.6%↓)	5.9	0.58M (93.5%↓)	**4.85**
	16	-	4.21M (71.4%↓)	**5.44**	4.38M (60.8%↓)	**5.4**	1.11M (87.5%↓)	**4.42**
ACDC-block	8	-	4.89M (66.8%↓)	**5.47**	2.96M (73.5%↓)	**5.5**	0.74M (91.7%↓)	**4.46**
	16	-	9.78M (33.6%↓)	**5.40**	5.76M (48.4%↓)	**4.9**	1.43M (83.9%↓)	**4.38**
ACDC-g-net	8	32	0.03M (99.8%↓)	10.24	0.20M (98.2%↓)	7.3	0.07M (99.2%↓)	8.20
	16	64	0.08M (99.5%↓)	9.87	0.26M (97.7%↓)	7.9	0.13M (98.5%↓)	6.85
ACDC-g-block	8	32	0.06M (99.6%↓)	9.71	0.22M (98.0%↓)	5.35	0.09M (99.0%↓)	8.92
	16	64	0.35M (97.6%↓)	6.63	0.45M (96.0%↓)	7.2	0.26M (97.1%↓)	6.88
ACDC-g-layer	8	32	0.13M (99.1%↓)	6.68	0.89M (92.0%↓)	5.2	0.36M (96.0%↓)	5.02
	16	64	0.80M (94.6%↓)	**5.67**	0.60M (94.6%↓)	6.2	1.98M (77.8%↓)	**4.23**

with identical sizes, and sharing coefficient is imposed across groups. Formally, given a convolutional layer with c' input channels and c output channels, respectively, we divide input channels into g identical-sized groups, and each group is convolved with a convolution kernel $\mathbf{K}_j \in \mathbb{R}^{\frac{c}{g} \times \frac{c'}{g} \times l \times l}, j = 1, \dots, g$. After grouping, we decompose $\{\mathbf{K}_j; j = 1, \dots, g\}$ into shared coefficients $\mathbf{A} \in \mathbb{R}^{\frac{c}{g} \times \frac{c'}{g} \times m}$, and g independent sets of dictionary atoms $\{\mathbf{D}_j \in \mathbb{R}^{m \times l \times l}; j = 1, \dots, g\}$. In this way, the number of shared coefficients is reduced by g^2 times, and the number of dictionary atoms is increased by g times. Since dictionary atoms have orders of magnitude smaller dimension comparing to the coefficients, applying *ACDC with grouping* achieves further parameter reduction. And since each $\mathbf{K}_j$ only convolve with a subset of the input feature, this method reduces the overall computations.

However, directly deploying this sharing mechanism breaks the network into several paralleled subnetworks with no feature passing and gradient propagation among them. To remedy this without adding any additional parametric components, we utilize *channel shuffle* (Zhang et al., 2018) that enables information to be efficiently propagated among groups in a non-parametric way. An illustration of the proposed *ACDC with grouping* is presented in Figure 4. Since the size of the shared coefficient now does not depend on the largest feature dimension of a network but the size of the groups, further parameter reduction is achieved by *ACDC with grouping* as shown in Section 3.

3 Experiments

In this section, we apply variants of ACDC as plug-and-play replacements to the standard convolution, and perform extensive experiments to validate the effectiveness of ACDC as a structural regularization of CNNs. Inspired by dropout (Srivastava et al., 2014), we adopt *atom-drop* as a regularization to the dictionary atoms by randomly dropping an atoms with probability p. We set $p = 0.1$ for all experiments. Details for implementations can be found in the supplementary material Section A. *ACDC with grouping* leads to three variants of ACDC, which are constructed by allowing coefficients to be shared within the entire network, blocks within a network, and layers within a network, and are named as *ACDC-g-net*, *ACDC-g-block*, and *ACDC-g-layer*, respectively. We use m and s to denote number of dictionary atoms and grouping size, respectively

3.1 Image Classification

Self-comparison on CIFAR-10. We first report on CIFAR-10 extensive self-comparison on variants of ACDC constructed with different numbers of dictionary atoms as well as grouping sizes. We present performance in terms of both parameter size and classification error in Table 1. VGG16 (Simonyan & Zisserman, 2014), ResNet18 (He et al., 2016), and Wide ResNet (WRN) (Zagoruyko & Komodakis, 2016) are adopted as the underlying network architectures in order to show the remarkable compatibility of ACDC. ACDC enhances deep CNNs with great flexibilities reflected by the wide range of parameter reduction from as low as 98% reduction with comparable performance, to about 70% reduction with even higher accuracy.

Table 2: Classification performances on CIFAR-10, CIFAR-100, and *Tiny*ImageNet datasets. Performance on state-of-the-art light CNN architectures are listed in the upper block. The middle block shows the performance of plug-and-play methods with parameter sharing in CNNs. Performance obtained by our reproductions are marked with $*$.

Methods	Size	CIFAR-10	CIFAR-100	*Tiny*ImageNet
SqueezeNet (Iandola et al., 2016)	2.36M	6.98*	29.56*	48.22*
ShuffleNet (Zhang et al., 2018)	0.91M	7.89*	29.94*	54.72*
ShuffleNet-V2 (Ma et al., 2018)	1.3M	8.96*	29.68*	51.13*
MobileNet-V2 (Sandler et al., 2018)	2.36M	5.52*	30.02*	48.22*
NASNet (Zoph et al., 2018)	3.1M	3.59	21.77*	47.17*
LegoNet-VGG16-w(o=2,m=0.5) (Yang et al., 2019)	3.7M	6.77	29.45*	48.42*
LegoNet-VGG16-w(o=4,m=0.25) (Yang et al., 2019)	0.9M	8.65	30.11*	55.22*
WRN-40-1 HyperNets (Ha et al., 2016)	0.10M	8.02	-	-
WRN-40-2 HyperNets (Ha et al., 2016)	2.24M	7.23	-	-
SWRN 28-10-1 (Savarese & Maire, 2019)	12M	4.01	19.73	43.05*
SWRN 28-10-2 (Savarese & Maire, 2019)	17M	3.75	18.37	41.12*
WRN-40-1 *ACDC-block* $m8$	0.043M	7.19	30.23	51.47
WRN-40-1 *ACDC-block* $m24$	0.114M	7.02	28.14	49.05
WRN-40-4 *ACDC-g-layer* $m16$ $s32$	0.67M	4.38	20.04	45.87
WRN-28-10 *ACDC-g-block* $m24$ $s160$	2.27M	4.25	19.64	41.24
WRN-28-10 *ACDC-net* $m12$	5.21M	3.52	18.81	39.96
WRN-28-10 *ACDC-block* $m24$	13.20M	**3.26**	**17.85**	**38.74**

Image classification with comparisons. We further present experiment results on CIFAR-10, CIFAR-100, and *Tiny*ImageNet. We compare exampled variants of ACDC against HyperNetworks (Ha et al., 2016) and Soft Parameter Sharing (Savarese & Maire, 2019), both of which serve as plug-and-play replacements to standard convolutions as well. Though HyperNetworks (Ha et al., 2016) achieves remarkable parameter reduction, ACDC is able to achieve higher accuracies with even fewer parameters. The parameter reductions in Soft Parameter Sharing (Savarese & Maire, 2019) are highly restricted by the large scale elements in the filter bank. For example, SWRN 28-10-1, as the smallest variant of Soft Parameter Sharing on WRN, adopts a single template per sharing group, and can only achieve 66% of parameter reduction. By adopting *ACDC-net* and *ACDC-block* to WRN, we are able to achieve both higher parameter reductions and accuracy. We also compare state-of-the-art light CNN architectures (Iandola et al., 2016; Zhang et al., 2018; Ma et al., 2018; Sandler et al., 2018) and architecture based on neural architecture search (Zoph et al., 2018). Additional comparisons against network compression and pruning methods are in supplementary material Section B. ACDC demonstrates high performance in terms of both parameter and accuracy, and show great flexibility, that in practice, allows model selection based on real-world applications.

Large-scale image classification. To fully demonstrate the effectiveness of ACDC, we perform further experiments on the large-scale ImageNet dataset (Deng et al., 2009). We use ResNet34 and ResNet50 as baseline models, and compare ACDC against LegoNet (Yang et al., 2019), Versatile (Wang et al., 2018b), network pruning method ChPrune (He et al., 2017), and recently proposed Structured Conv (Bhalgat et al., 2020). Results are presented in Table 3. ACDC achieves better parameter reductions on ResNet34. One variant of ACDC with grouping in layers uses only 1.66 million parameters with acceptable performance decrease. For ResNet50, since a large proportion of the parameters are in the 1×1 conv layers, ACDC achieves fair parameter reductions with maintained performance.

Table 3: Performance on ImageNet. Parameters, top-1 and top-5 errors are reported. Numbers obtained by our reproductions are marked with $*$. We use ResNet34 and ResNet50 as baseline models.

Methods	Size	Top-1	Top-5
ResNet34	21.28M	27.42*	9.02*
Structured Conv A	9.82M	27.19	-
Structured Conv B	5.60M	30.56	-
ACDC-layer $m12s64$ (ours)	1.66M	32.82	12.14
ACDC-net $m12$ (ours)	3.34M	30.85	11.25
ACDC-stage $m16$ (ours)	5.77M	27.78	9.72
ResNet50	25.26M	24.17*	7.82*
ChPrune	17.89M	27.7	9.2
LegoNet-w	9.3M	-	8.7
Versatile	11.0M	25.5	8.2
Structured Conv A	13.49	24.35	-
Structured Conv B	8.57	26.59	-
ACDC-net m8 (ours)	14.29M	25.38	8.08
ACDC-stage m8 (ours)	16.37M	24.04	7.68

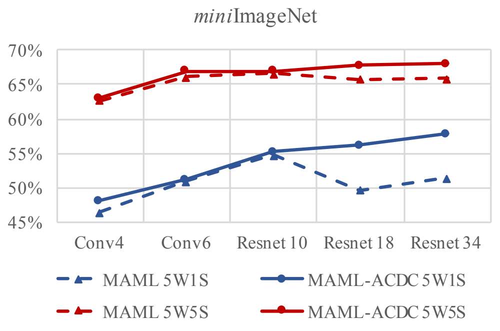

Figure 5: Few-shot image classification with deeper CNN architectures. 5W1S and 5W5S denote 5-way 1-shot and 5-way 5-shot experiments, respectively.

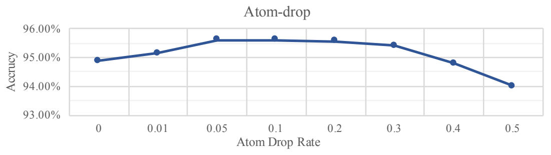

Figure 6: Accuracy with different atom drop rate p.

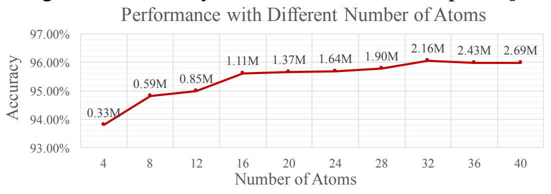

Figure 7: Accuracy with different number of dictionary atoms m. Parameter sizes are denoted as $\#M$.

Table 4: Comparisons of FLOPs with different variants of ACDC with grouping.

Networks	Baseline	*ACDC m8 s64*	*ACDC m16 s64*	*ACDC m8 s32*	*ACDC m16 s32*
VGG16	125.66B	64.15B	64.17B	32.29B	32.30B
ResNet18	222.4B	116.76B	116.78B	60.13B	60.14B

3.2 Adaptation Experiments

We further demonstrate that the proposed ACDC improves the adaptation of deep networks on novel tasks with limited supervisions, which is validated by few-shot classification using commonly adopted experimental settings. Specifically, we adopt *ACDC-net* on the model-agnostic meta-learning (MAML) (Finn et al., 2017) algorithm, which is a method that adapts to a novel task by tuning the entire network from a learned initialization. Although MAML is designed to be model-agnostic, we consistently observe that it struggles for further performance improvements when using deeper and wider networks. Same observations are reported in (Chen et al., 2019). We show that such limitation can be alleviated by structural regularizations with ACDC. We follow the same experimental settings as (Chen et al., 2019) and perform both 5-way 1-shot and 5-way 5-shot image classifications on *mini*ImageNet dataset. The comparisons are shown in Figure 5. Though adopting ResNet10 with MAML achieves improvements over simple network with few stacked layers, the performance drops with more residual layers as shown by the results on ResNet18. By using *ACDC-net* with deeper ResNets, performance is not only maintained but also improved when more layers are used.

3.3 Computation Efficiency

ACDC enjoys another merit of being computationally efficient when using *sharing with grouping*. Since after grouping, each group of convolutional filters only convolves with a subset of input features, *ACDC-g-block* and *ACDC-g-net* substantially reduce the number of FLOPs. We report comparisons with ResNet18 and VGG16 in Table 4. All numbers are obtained by feeding the network a typical batch with 100 64×64 images. It is clearly shown that by using small groups, the computation can be reduced dramatically, and larger number of dictionary atoms only effects the total FLOPs slightly.

3.4 Ablation Studies

Training with ACDC introduces two hyperparameters, which are the number of dictionary atoms m per sub-structure, and the atom drop rate p. We present here ablation studies on the impacts to the network accuracy with different p and m. Both experiments are conducted using ResNet18 with *ACDC-net* and trained on CIFAR-10. As shown in Figure 6, atom-drop improves generalization when $p \leq 0.1$. Higher values of p potentially degrade the performance as the training becomes unstable. Thus we use $p = 0.1$ as the default setting. As shown in Figure 7, having more dictionary atoms in each sub-structure leads to performance improvements that saturate at $m = 32$. More dictionary atoms also result in larger parameter sizes, which are unfavourable.

4 Related Work

CNN architectures. The tremendous success of applying convolutional neural networks (CNNs) on numerous tasks has stimulated rapid developments for more effective and efficient network architectures in both hand-crafted (Chen et al., 2017; He et al., 2016; Howard et al., 2017; Iandola et al., 2016; Sandler et al., 2018) and automatically discovered (Elsken et al., 2018; Liu et al., 2018; Pham et al., 2018; Zoph & Le, 2016) manners. We consider our work orthogonal to such topology-based methods, as the plug-and-play property of the proposed ACDC allows it to be added to all the aforementioned methods as a replacement to the standard convolution. Besides efforts on studying efficient network architectures, methods for network compression and pruning (Han et al., 2015a; 2016; 2015b; He et al., 2017; Li et al., 2016; Luo et al., 2017) have been extensively studied for decreasing the model size by pruning the inconsequential connections and weights of a network. Methods (Ha et al., 2016; Savarese & Maire, 2019) align with our direction as they are also insensitive to network topologies. And as shown in the experiments, ACDC can achieve higher performance in terms of parameter reduction and classification accuracy with greater flexibility.

Kernel decomposition in CNNs Convolutional kernel decomposition has been studied for various objectives. (Sosnovik et al., 2019) utilizes kernel decomposition as a tool for constructing same kernel with multiple receptive fields. DCFNet (Qiu et al., 2018) is proposed as a principle way of regularizing the convolutional filter structures by decomposing convolutional filters in CNN as a truncated expansion with pre-fixed bases.

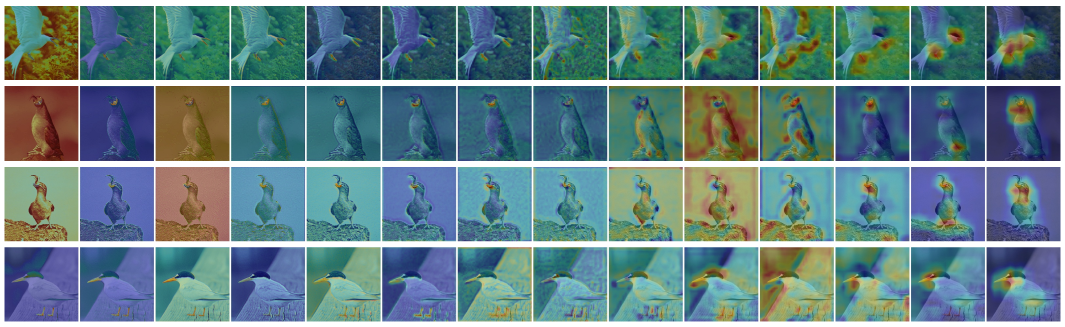

Figure 8: Illustration on extending CAM from the last layer to all layers with *ACDC-net*. In CAM, the visualized heatmap explains the importance of image regions. Each row shows the class activation maps of a sample from the first convolution layer (left) to the final convolution layer (right).

5 Conclusion and Interpretability Discussion

In this paper, we introduced atom-coefficient decomposed convolution, a plug-and-play replacement to the standard convolution by imposing structural regularization to kernels in a CNN. We presented observations that, due to the underlying cross-layer correlations, coefficients in the decomposed convolution layers reside in a low rank structure. We explicitly exploited such observations by enforcing coefficients to be shared within sub-structures of CNNs, and achieved significant parameter reductions. Variants of ACDC can be constructed with different sharing structures, number of atoms, and grouping sizes. We reported extensive experiment results that show the effectiveness of ACDC on standard image classification and adaptations.

The structural regularization with ACDC has the potential for better interpretability of CNNs, due to the cross-layer shared coefficients. We close our paper with an illustration that extends class activation mapping (CAM) (Zhou et al., 2016), which is originally proposed to explain the importance of image regions but only at the final convolution layer. CAM is calculated by weighted averaging the features of the final convolution layer by the weight vector of a particular class. Since now with *ACDC-net*, features across layers are generated with the same coefficients, we exploit the potential correspondence to extend CAM to all the preceding layers using the same weighted sum, and visualize the activation maps for all layers in Figure 8. We use a VGG network with 13 conv layers trained with CUB-200 (Wah et al., 2011) high-resolution bird classification dataset. It is clearly shown that, while the activation maps for shallow layers are inevitably noisy due to the limited receptive fields, features in deeper layers are progressively refined to the discriminative regions. This shows the great potential for better interpretability with ACDC, and we will keep this as a direction of future effort.

REFERENCES

Yash Bhalgat, Yizhe Zhang, Jamie Lin, and Fatih Porikli. Structured convolutions for efficient neural network design. *arXiv preprint arXiv:2008.02454*, 2020.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Advances in Neural Information Processing Systems*, pp. 4467–4475, 2017.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. Ieee, 2009.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. JMLR. org, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015b.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

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*, pp. 770–778, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *International Conference on Learning Representations*, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision*, pp. 19–34, 2018.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, 2017.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision*, pp. 116–131, 2018.

Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pp. 14014–14024, 2019.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, pp. 5727–5736, 2018.

Anton Obukhov, Maxim Rakhuba, Stamatios Georgoulis, Menelaos Kanakis, Dengxin Dai, and Luc Van Gool. T-basis: a compact representation for neural networks. *International Conference on Machine Learning*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8026–8037, 2019.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Qiang Qiu, Xiuyuan Cheng, Robert Calderbank, and Guillermo Sapiro. DCFNet: Deep neural network with decomposed convolutional filters. *International Conference on Machine Learning*, 2018.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pp. 6076–6085, 2017.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Pedro Savarese and Michael Maire. Learning implicitly recurrent cnns through parameter sharing. *arXiv preprint arXiv:1902.09701*, 2019.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. *arXiv preprint arXiv:1910.11093*, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018a.

Yunhe Wang, Chang Xu, XU Chunjing, Chao Xu, and Dacheng Tao. Learning versatile filters for efficient convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1608–1618, 2018b.

Zhaohui Yang, Yunhe Wang, Chuanjian Liu, Hanting Chen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. LegoNet: Efficient convolutional neural networks with lego filters. In *International Conference on Machine Learning*, pp. 7005–7014, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.

APPENDIX

A IMPLEMENTATION DETAILS

All experiments are performed using PyTorch (Paszke et al., 2019).

Initialization. We train every ACDC network from scratch. We use orthogonal initialization for all dictionary atoms, and Kaiming normal initialization for all coefficients.

Training details. All networks for CIFAR10 and CIFAR100 are trained for 350 epochs. The initial learning rate is set to be 0.1. The learning rate decays by a factor of 10 at the 150-th and the 250-th epoch. All ACDC networks are trained with a weight decay of 10^{-4}. The reported numbers are averaged over 5 runs. Networks for ImageNet are trained for 90 epochs. The initial learning rate is 0.1 and decays by a factor of 10 every 30 epochs. The weight decay for ImageNet experiments are set to be 2.5×10^{-5}. We consistently observe that lower values for weight decay yield better results on ACDC networks. We believe the reason is that the structural regularization of ACDC already reduces the risk of overfitting.

Regularization by atom-drop. To improve the robustness of the dictionary atoms and the corresponding reconstructed kernels, we further propose a structural regularization to dictionary atoms named *atom-drop* inspired by widely used dropout. Specifically, when training the network, we randomly drop a dictionary atom with probability p, which is referred as *atom drop rate*, by temporarily setting values of the dropped dictionary atoms to 0, and the values of all other remained dictionary atoms are multiplied by $\frac{1}{1-p}$ in order to maintain consistent scales of the reconstructed convolutional kernels. At test time, all dictionary atoms are presented with no dropping.

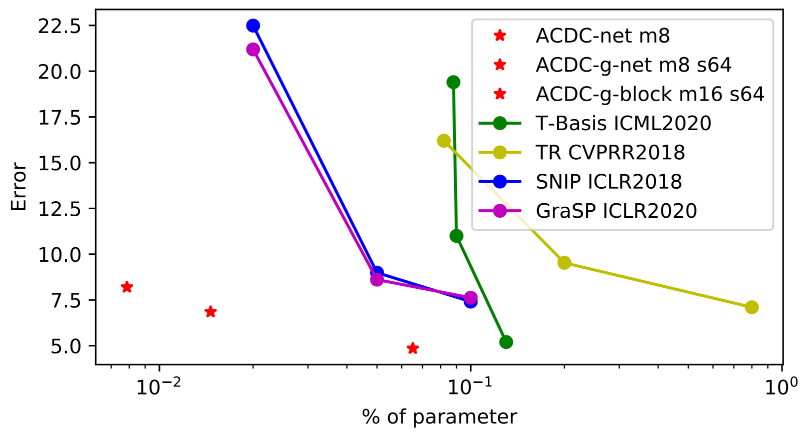

Figure 9: Comparisons against network compression and pruning methods on CIFAR-10 dataset. Performance is measured with error rates and proportion of remained parameters (both lower the better).As a plug-and-play method, ACDC can outperform network compression and pruning methods in terms of both parameter reduction rate and prediction accuracy.

B COMPARISONS AGAINST NETWORK COMPRESSION AND PRUNING METHODS

We present here further comparisons against network compression and pruning methods. Performance is measured with error rates and proportion of remained parameters (both lower the better). Note that different from post-processing using compression and pruning, ACDC is primarily proposed as a structural regularization and a plug-and-play replacement to standard convolutional layers, so the networks with ACDC remain trained end-to-end. We include SNIP (Lee et al., 2018), TR (Wang et al., 2018a), and recently proposed T-Basis (Obukhov et al., 2020) and GraSP (Wang et al., 2020) into the comparisons. Results shown in Figure 9 clearly demonstrate that ACDC can achieve even smaller parameter size with negligent scarification to accuracy.