Learning Lightweight Neural Networks via Channel-Split Recurrent Convolution

Anonymous authors
Paper under double-blind review

Abstract

Lightweight neural networks refer to deep networks with small numbers of parameters, which are allowed to be implemented in resource-limited hardware such as embedded systems. To learn such lightweight networks effectively and efficiently, in this paper we propose a novel convolutional layer, namely *Channel-Split Recurrent Convolution (CSR-Conv)*, where we split the output channels to generate data sequences with length T as the input to the recurrent layers with shared weights. As a consequence, we can construct lightweight convolutional networks by simply replacing (some) linear convolutional layers with CSR-Conv layers. We prove that under mild conditions the model size decreases with the rate of $O(\frac{1}{T^2})$. Empirically we demonstrate the state-of-the-art performance using VGG-16, ResNet-50, ResNet-56, ResNet-110, DenseNet-40, MobileNet, and EfficientNet as backbone networks on CIFAR-10 and ImageNet. ***Please refer to our demo code in the supplementary file.***

1 Introduction

Convolutional neural networks (CNNs) (Krizhevsky et al., 2012) have revolutionized computer vision by achieving state-of-the-art performance on many applications (He et al., 2016; Girshick, 2015; Jia et al., 2014). The impressive improvement usually comes with a substantial increase in the number of parameters (*i.e.,* model size) which is undesirable for the model applicability in many real-world applications (Gong et al., 2014; Gui et al., 2019), such as embedded systems where the computing resources (*e.g.,* processor and memory) in the hardware are limited. Therefore, how to design/learn lightweight neural networks, *i.e.,* reducing storage requirement in terms of parameters while achieving good performance, is becoming increasingly demanded (Sandler et al., 2018; Denton et al., 2014; Liu et al., 2015; Zhou et al., 2016; Li et al., 2016; Singh et al., 2019).

Generally speaking, there are two families of approaches for learning lightweight networks in the literature: (1) network architecture design/search, and (2) network compression. Typical works in the former family include SqueezeNet (Iandola et al., 2016), MobileNet (Sandler et al., 2018), ShuffleNet (Zhang et al., 2018), EfficientNet (Tan & Le, 2019a), MnasNet (Tan et al., 2019), and ProxylessNAS (Cai et al., 2018). Such approaches focus on developing network architectures (*e.g.,* using small convolutional filters) to satisfy certain requirements such as model size while achieving good performance for the applications. The latter family includes the approaches such as compression with learning (Wen et al., 2016; Li et al., 2019a; 2020; Kim et al., 2019; Eban et al., 2020; Zhao et al., 2019c; Prabhu et al., 2020; Hu et al., 2018; Zhang et al., 2018) or after learning (Han et al., 2015a; Kusupati et al., 2018a), whose basic ideas are to remove the network redundancy by imposing some structural assumptions on the convolutional filters. Nice surveys on this topic can be found in (Cheng et al., 2017; Zhang et al., 2019; Choudhary et al., 2020; Deng et al., 2020; Neill, 2020).

Motivation. Intuitively, reducing the number of parameters in each convolutional layer can significantly compact a given network. However, this may lead to poor generalization of the network, as wider networks have been shown to effectively improve the performance (Zagoruyko & Komodakis, 2016; Tan & Le, 2019a). In order to compensate for the performance loss due to model size reduction (*i.e.,* lightweight networks), we are mainly motivated by the following works:

- *Deeper Networks:* In complement to the universal approximation theorem (Cybenko, 1989), recent works such as (Lu et al., 2017) have shown that with the increase of network depth, the number

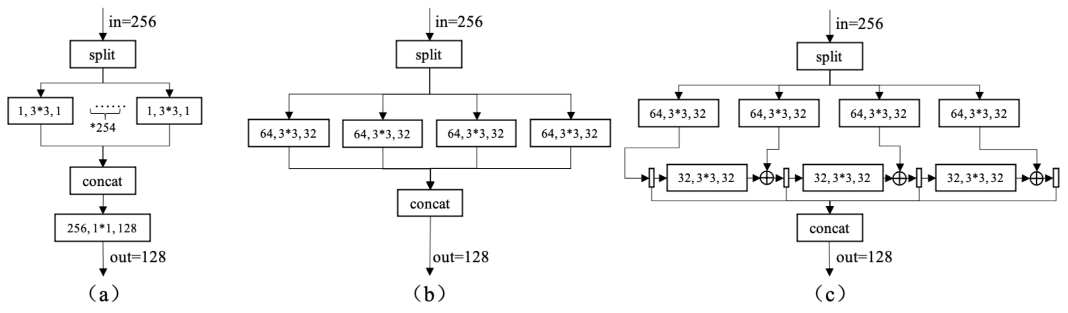


Figure 1: Comparison with 256 input channels and 128 output channels among **(a)** depth-wise separable convolution, **(b)** group convolution, and **(c)** our channel-split recurrent convolution (CSR-Conv) using vanilla RNNs. The linear convolutional operation is denoted as (#input channels, filter size, #output channels) and vertical small rectangles in **(c)** denote ReLU activation functions.

of hidden neurons can be dramatically reduced to approximate a function with similar expressive power. This motivates us to consider constructing a deeper network using narrow networks.

- *Visual Transformer (ViT):* Recently Dosovitskiy et al. (2021) demonstrated excellent performance on image recognition using ViT that are designed to handle sequential input data, similar to recurrent neural networks (RNNs). In their work, each image is divided into 16×16 patches in the spatial domain, and then fed into ViT as an input data sequence. This motivates us to consider exploring RNNs (not ViT due to its large model size) in different ways to learn lightweight networks.

Our proposed approach and contributions. Based on consideration above, we propose a novel convolutional layer, namely *Channel-Split Recurrent Convolution (CSR-Conv)*, as illustrated in Fig. 1(c) where the 256 input channels are equally split into 4 groups, fed into a recurrent convolutional layer (implemented using a vanilla RNN in the figure as demonstration) as input, and the hidden states in the RNN are concatenated to generate the 128 output channels. Compared with depth-wise separable convolution (used in MobileNet (Sandler et al., 2018)) and group convolution (used in ShuffleNet (Zhang et al., 2018) and ResNeXt (Xie et al., 2017)) in Fig. 1(a-b), respectively, we can see clearly that our key difference is to replace each linear convolution with a recurrent convolution. As a result, if imaging each figure as a graph where all the linear convolutions are denoted by nodes, then the depths (*i.e.,* the longest paths) between node "split" and node "concat" in the figures are different: in Fig. 1(a-b) the depths are both 2, while in Fig. 1(c) the depth is 5. In other words, recurrent convolution can lead to deeper network architectures, which could be beneficial for learning lightweight convolutional networks.

We are aware that the integration of RNNs with convolution for deep learning has been explored in the literature such as (Wang & Hu, 2017; Liang & Hu, 2015; Kim et al., 2016; Tai et al., 2017; Shi et al., 2015; Ondruska et al., 2016; Spoerer et al., 2017). However, to the best of our knowledge, *we are the first to explore the applicability of recurrent deep models (*e.g., *RNNs, GRUs, LSTM) as general recurrent convolutional layers to learn lightweight CNNs.* Given a backbone network such as VGGNet (Simonyan & Zisserman, 2014) or ResNet (He et al., 2016), we can easily replace its linear convolutional layers using our CSR-Conv to reduce the model size, achieving a deeper network as well as preserving its performance[1]. We analyze the relationship between model size and CSR-Conv to show that the model compression rate is fairly controllable. We also demonstrate the state-of-the-art performance of our approach based on seven existing network architectures on CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009) datasets.

2 Related Works

Recurrent neural networks. RNNs have achieved significant success in learning complex patterns for sequential input data, and have been widely used in computer vision (Zhao et al., 2019a; Zhong

[1]Certainly we can design new networks using our standalone CSR-Conv layers, but this is beyond the scope of this paper. In this paper, we only focus on learning lightweight networks given certain backbone networks.

et al.; Zhao et al., 2020; Zhang & Zuo, 2020; Corona et al., 2020; Pato et al., 2020). At each time step, an RNN updates the state vector based on the current state and input data. Subsequently, RNNs output the predictions as a function of the hidden states. The model parameters are learned by minimizing an empirical loss. In the literature, there are significant amount of works on developing RNNs such as, just to name a few, long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), gated recurrent unit (GRU) (Cho et al., 2014), UGRNN (Collins et al., 2016), FastGRNN (Kusupati et al., 2018b), unitary RNNs (Arjovsky et al., 2016), antisymmetric RNN (Chang et al., 2019), incremental RNN (Kag et al., 2020), exponential RNN (Lezcano-Casado & Martınez-Rubio, 2019), Lipschitz RNN (Erichson et al., 2020).

Recurrent convolutional neural networks (RCNN). (Liang & Hu, 2015) proposed incorporating the recurrent connections in each convolutional layer to generate features with different resolutions. (Wang & Hu, 2017) added a gate to the recurrent connections in RCNN to control context modulation and balance the feed-forward information and the recurrent information. (Kim et al., 2016) imposed very deep recursive layers to improve performance without introducing new parameters for additional convolutions. (Tai et al., 2017) developed a recursive convolutional neural network with the residual connection. (Shi et al., 2015) replaced vanilla RNN architecture with an LSTM structure in RCNN. (Ondruska et al., 2016) used dilated convolution in the RCNN to reduce computational complexity.

Variants of convolutional operator for compression. (Denil et al., 2013) proposed using a linear combination of basis functions to predict parameters for compression. (Bagherinezhad et al., 2017) proposed encoding convolutions by few lookups to a dictionary trained to cover the space of weights in CNNs. (Wu et al., 2018) presented a parameter-free, FLOP-free "shift" operation as an alternative to spatial convolutions. (Gao et al., 2018) proposed channel-wise convolutions, which replace dense connections among feature maps with sparse ones in CNNs. (Wen et al., 2016) proposed a Structured Sparsity Learning (SSL) method to regularize the structures such as filters, channels, filter shapes, and layer depth of CNNs. (Liao & Yuan, 2019) proposed an efficient CircConv operator based on the presumed circulant structures of convolutional filters where Fast Fourier Transform (FFT) can be used to compute the filter responses in feed-forward and inverse FFT can be applied in back-propagation.

Network compression. Weight pruning (Han et al., 2015a;b; Li et al., 2021) aims at reducing non-significant weights to reduce computation and memory usage of the model. Other than weight pruning, filter level pruning which leads to the removal of the corresponding feature maps is also studied intensively (He et al., 2018; Li et al., 2019b). Regularization constraints are also introduced in pruning (Alvarez & Salzmann, 2016; Liu et al., 2017; Huang & Wang, 2018). Low-rank factorization (Lebedev et al., 2014; Tai et al., 2015; Jaderberg et al., 2014; Zhang et al., 2015; Yu et al., 2017) aims to decompose the large weight matrices in the convolutional layers into smaller matrices with fewer parameters. Knowledge distillation (Hinton et al., 2015; Lan et al., 2018; Park et al., 2019) aims to force a smaller student network to fit specific features from a larger teacher network for knowledge transfer.

3 Our Approach

3.1 Problem Definition

In this paper we only focus on learning lightweight convolutional networks by replacing some linear convolutions with CSR-Conv in a given backbone network such as VGGNet or ResNet, so that the model size can satisfy certain requirements. We will not design or propose new network architectures.

Specifically, given a backbone network with L convolutional layers, a desirable model compression rate ρ_M (this constraint is optional depending on the applications/users), and a training dataset $\{\mathbf{x}, y\} \subseteq \mathcal{X} \times \mathcal{Y}$ with sample $\mathbf{x} \in \mathcal{X}$ and label $y \in \mathcal{Y}$, we propose the following optimization problem as our objective for learning lightweight networks:

$$\min_{\omega, \mathcal{T} \in \mathbb{Z}^L} \mathbb{E}_{(\mathbf{x},y)} \ell\Big(f(\mathbf{x}; \omega, \mathcal{T}), y\Big), \text{ s.t. } \frac{M_C}{M_B} \approx 1 - \rho_M, \tag{1}$$

where f denotes the modified network with CSR-Conv parametrized by ω, $\mathcal{T}$ denotes a set of the sequence lengths as input to the recurrent convolutional layers in CSR-Conv (if the length is equal to 1, there will be no change to the linear convolution), ℓ denotes the loss function, $\mathbb{E}$ denotes the expectation operation, and M_C, M_B denote the numbers of parameters in the modified and backbone networks, respectively. In case that achieving the exact compression rate ρ_M may be impossible, we instead try to search for the best network architectures with similar compression rates.

Grid-search solver with CSR-Conv. In contrast to network architecture search (NAS) that is optimized in the network architecture space, in this paper we simply use grid-search to determine $\mathcal{T}$, same as EfficientNet (Tan & Le, 2019a), because our search space is much smaller than NAS given the compression rate and backbone network. To accelerate our training, in our implementation we further reduce our search space to $\mathcal{T} \in \{1, T\}^L$, that is, a linear convolutional layer is either unchanged or split into T groups of channels. We then determine $T > 1$ using grid-search as well as learning ω. We list an exemplar of our network implementation in Table 1 where the bold parts are the filter sizes in CSR-Conv. We restrict our grid search so that the number of channels in the backbone network is approximately preserved by the RNNs.

Table 1: Illustration of our CSR-Conv-4 architecture in Table 2 for VGG-16 with $T = 5$, where the parameters in the 6th-13th convolutional layers are converted to the parameters $\mathbf{U}, \mathbf{V}$ in CSR-Conv with the same spatial sizes.

Layer	VGG-16	Ours	#Param (ρ_M)
Conv1	[3×64]	[3×64]	1,728(0.0%)
Conv2	[64×64]	[64×64]	36,864(0.0%)
Conv3	[64×128]	[64×128]	73,728(0.0%)
Conv4	[128×128]	[128×128]	147,456(0.0%)
Conv5	[128×256]	[128×**260**]	299,520(-1.6%)
Conv6	[256×256]	**[52×52]** **[52×52]**	48,672(91.9%)
Conv7	[256×256]	**[52×52]** **[52×52]**	48,672(91.9%)
Conv8	[256×512]	**[52×103]** **[103×103]**	134,685(87.8%)
Conv9	[512×512]	**[103×103]** **[103×103]**	190,962(91.9%)
Conv10	[512×512]	**[103×103]** **[103×103]**	190,962(91.9%)
Conv11	[512×512]	**[103×103]** **[103×103]**	190,962(91.9%)
Conv12	[512×512]	**[103×103]** **[103×103]**	190,962(91.9%)
Conv13	[512×512]	**[103×103]** **[103×103]**	190,962(91.9%)
FC	/	/	267,264(0.0%)
Total			2,022,489(86.5%)

3.2 CSR-Conv: Channel-Split Recurrent Convolution

We illustrate the general architecture of CSR-Conv in Fig. 2, where "Split" and "Concat" denote the channel split and concatenation operations, respectively. The in-between recurrent convolutional layer takes the split data sequence as input and outputs the hidden states over time. It can be implemented using an RNN, GRU, LSTM, *etc.* Recall that Fig. 1 illustrates our customized implementation based on a vanilla RNN, where the input and output are 3D features and the network weights are 4D. For simplicity, we represent all the input and output data as vectors, and network weights as matrices. Specifically, we denote $\mathbf{x}_l \in \mathbb{R}^{d_l}, \forall l \in [L]$ as a d_l-dim input for the l-th convolutional/recurrent layer ($\mathbf{x}_l = \mathbf{x}$, *i.e.,* the input data to the network, when $l = 0$). We will explain the architecture based on a vanilla RNN as well.

Channel split. The goal of this step is to generate data sequence based on the input channels for further process in the recurrent layer. Imagining that we need a sequence with length T at the l-th convolutional layer, then we reshape $\mathbf{x}_l$ to a matrix $\mathbf{X}_l = [\mathbf{x}_{l,t}]_{t\in[T]} \in \mathbb{R}^{\lceil \frac{d_l}{T} \rceil \times T}$ where $\lceil \cdot \rceil$ denotes the ceiling operator and $[\cdot]_{t\in[T]}$ denotes the vector concatenation operator. This new matrix will be fed into the recurrent layer column-by-column sequentially.

Vanilla RNN based recurrent convolution. We follow the simplest RNN formulation (*i.e.,* vanilla RNN) as below to implement the recurrent layer:

$$\mathbf{h}_{l,t} = \sigma\Big(\mathbf{U}_l^T \mathbf{h}_{l,t-1} + \mathbf{V}_l^T \mathbf{x}_{l,t}\Big), \mathbf{h}_{l,0} = \mathbf{0}, \forall t \in [T], \quad (2)$$

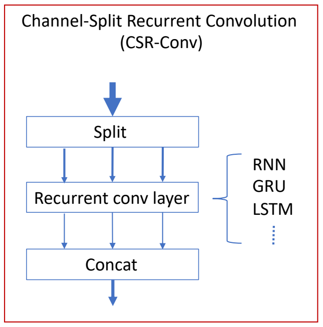


Figure 2: General architecture.

where at the layer l and time step t, $\mathbf{h}_{l,t} \in \mathbb{R}^{D_l}$ denotes the hidden state vector, $\mathbf{U}_l \in \mathbb{R}^{D_l \times D_l}$, $\mathbf{V}_l \in \mathbb{R}^{\lceil \frac{d_l}{T} \rceil \times D_l}$ denote the shared state and data transition matrices in the RNN, σ denotes the activation function such as ReLU, and $(\cdot)^T$ denotes the matrix transpose operator. Here we do not take the bias term into account, because in practice we do not observe any significant improvement with the bias term but introducing more parameters. Note that the recurrent layer defined in Eq. 2 can be viewed as the generalization of the traditional linear convolution, because both will be equivalent when $T = 1$. For other implementations, one can replace the formula in Eq. 2 with the corresponding formula to construct the recurrent layer.

Channel concatenation. Once we have the collection of the hidden state vectors, we simply concatenate them into a $(D_l \times T)$-dim vector $\mathbf{h}_l = [\mathbf{h}_{l,t}^T]^T$ that will be used in further process.

Table 2: Summary of our results on **(2nd block)** CIFAR-10 and **(3rd block)** ImageNet, where "#C-C" denotes the number of CSR-Conv modules used in the networks for learning compact networks, and "ρ_M" denotes the model size compression rate.

Network	Top-1 Err.(%)	$\rho_M(\downarrow)$	#Param.	#C-C	T
VGG-16	6.04±0.05	0.0%	14.98M	0	1
CSR-Conv-1	**5.89±0.06**	39.3%	9.09M	5	2
CSR-Conv-2	6.01±0.10	49.0%	7.64M	4	3
CSR-Conv-3	6.16±0.10	67.2%	4.91M	6	3
CSR-Conv-4	6.35±0.08	86.5%	2.02M	9	5
CSR-Conv-5	7.08±0.12	95.0%	0.75M	12	9
ResNet-56	6.74±0.14	0.0%	0.85M	0	1
CSR-Conv-1	**6.12±0.11**	21.8%	0.66M	4	2
CSR-Conv-2	6.69±0.12	61.0%	0.33M	11	3
CSR-Conv-3	6.83±0.10	70.3%	0.25M	17	3
CSR-Conv-4	7.93±0.19	78.8%	0.18M	15	4
CSR-Conv-5	9.15±0.13	88.9%	0.09M	22	5
ResNet-110	6.50±0.05	0.0%	1.74M	0	1
CSR-Conv-1	5.72±0.07	17.2%	1.44M	7	2
CSR-Conv-2	**5.55±0.05**	36.4%	1.11M	14	2
CSR-Conv-3	6.12±0.11	61.3%	0.67M	22	3
CSR-Conv-4	7.06±0.15	79.6%	0.35M	31	4
CSR-Conv-5	8.57±0.18	87.1%	0.22M	33	5
DenseNet-40	5.19±0.04	0.0%	1.06M	0	1
CSR-Conv-1	5.19±0.12	15.9%	0.89M	19	2
CSR-Conv-2	5.13±0.09	35.2%	0.69M	11	3
CSR-Conv-3	**5.09±0.14**	50.3%	0.53M	22	3
CSR-Conv-4	6.01±0.13	63.7%	0.38M	23	4
CSR-Conv-5	8.30±0.15	82.4%	0.19M	34	5
MobileNet-V2	5.53±0.15	0.0%	2.24M	0	1
CSR-Conv-1	5.21±0.13	26.4%	1.65M	4	3
CSR-Conv-2	**5.08±0.14**	34.3%	1.47M	7	3
CSR-Conv-3	5.37±0.09	44.1%	1.25M	17	3
CSR-Conv-4	5.84±0.12	51.4%	1.09M	18	3
CSR-Conv-5	6.10±0.21	57.3%	0.95M	18	4
ResNet-50	23.85±0.23	0.0%	25.56M	0	1
CSR-Conv-1	**23.51±0.27**	35.7%	16.43M	10	3
CSR-Conv-2	24.61±0.24	70.3%	7.59M	15	4
EfficientNet-B0	22.90±0.23	0.0%	5.28M	0	1
CSR-Conv-1	**22.34±0.31**	18.9%	4.28M	3	3
CSR-Conv-2	27.59±0.31	26.3%	3.89M	4	5
MobileNet-V2	27.80±0.29	0.0%	3.50M	0	1
CSR-Conv-1	**27.65±0.32**	14.0%	3.01M	2	4
CSR-Conv-2	29.45±0.32	29.5%	2.47M	12	4

3.3 Analysis

Proposition 1 (Model Size)**.** *Suppose that the numbers of input and output channels in each convolutional layer of the backbone network are equal to those from CSR-Conv with sequence length $T (T > 1)$. Then we can compute the model size ratio, λ_M, between CSR-Conv in Eq. 2 and the corresponding linear convolution as follows:*

$$\lambda_M = \frac{k^2 D(D+d)}{k^2 DdT^2} = \left(1 + \frac{d}{D}\right)\frac{1}{T^2} = O\left(\frac{1}{T^2}\right). \tag{3}$$

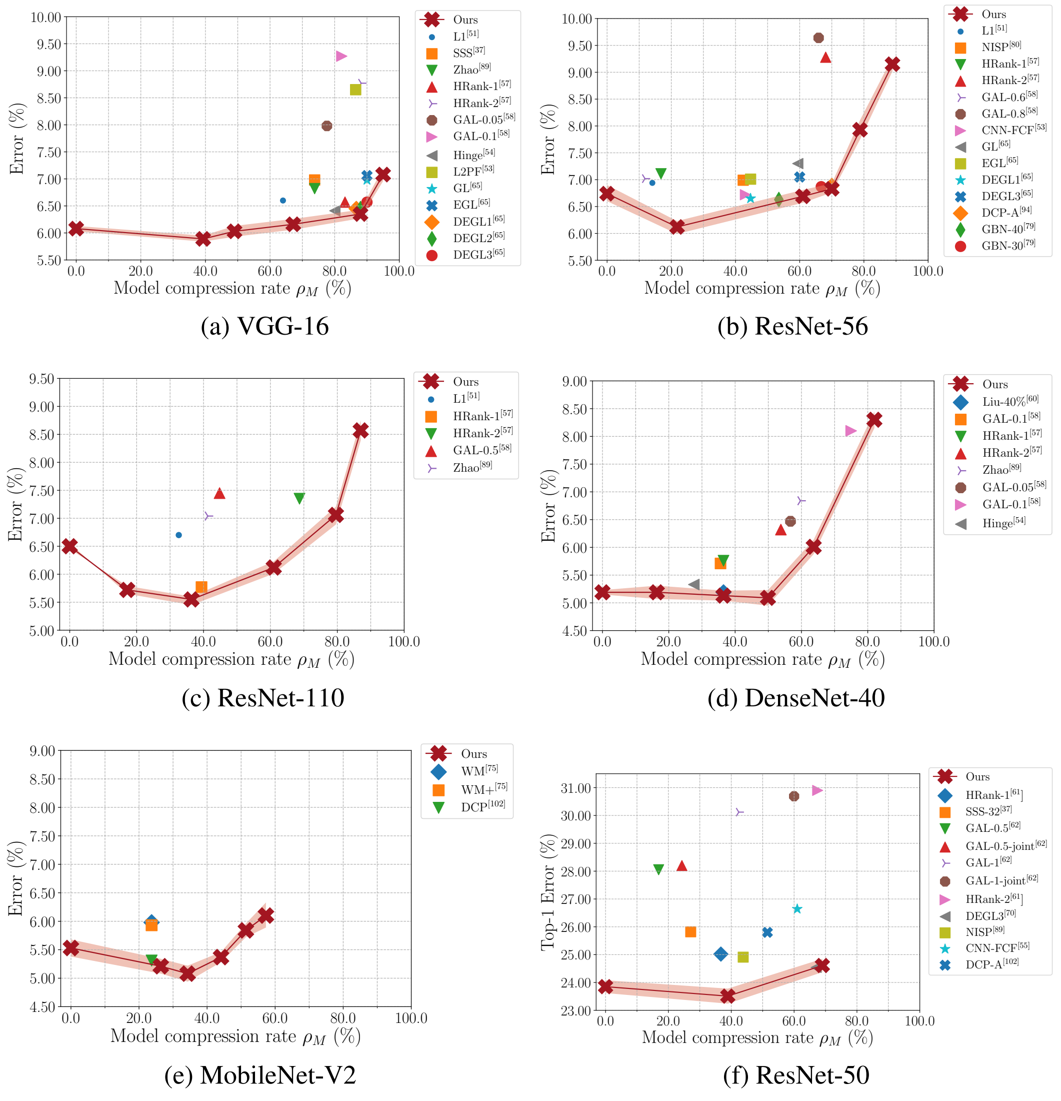
Often empirically $d \leq D \Leftrightarrow 0 < \frac{d}{D} \leq 1$ holds. Meanwhile, given the fact that the number of parameters in unchanged sub-networks is trivial, the compression rate will be heavily dominated by the number of the duplicate networks T.

Proposition 2 (FLOPs)**.** *Suppose that (1) the computational complexity of add, multiplication, and σ is a unit operation with one FLOP, and (2) the input and output dimension for the backbone network can be represented as dT and DT, respectively. Then we can compute the FLOP ratio, λ_F, between CSR-Conv in Eq. 2 and the corresponding linear convolution as follows:*

$$\lambda_F = \frac{k^2 WHDT(1+2D+2d)}{k^2 WHDT(1+2dT)} = \frac{1+2D+2d}{1+2dT} \leq \left(1+\frac{D}{d}\right)\frac{1}{T}, \tag{4}$$

where the equation holds if and only if $T = 1 + \frac{D}{d}$ that leads to $\lambda_F = 1$.

The upper-bound in Eq. 4 indicates that the FLOPs of CSR-Conv tends to decrease *w.r.t.* T approximately. For instance, empirically our CSR-Conv-1 for ResNet-56 in Table 2 has the same FLOPs as ResNet-56, even with better performance and smaller model size, because we set $T = 2$ and $d = D$ in CSR-Conv in our implementation. Differently, CSR-Conv-5 can achieve 42.0% of FLOP compression rate, compared with ResNet-56.



(a) VGG-16 (b) ResNet-56

(c) ResNet-110 (d) DenseNet-40

(e) MobileNet-V2 (f) ResNet-50

Figure 3: Comparison of error *vs.* compression rate on **(a-e)** CIFAR-10 and **(f)** ImageNet.

4 EXPERIMENTS

Datasets. Following the literature, we evaluate our approach comprehensively on CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009) for the image classification task. CIFAR-10 consists of 50k training images and 10k testing images from 10 classes. ImageNet is a large dataset, which contains over 1m training images and 50k testing images from 1000 categories.

Backbone networks & baseline approaches. We conduct experiments based on five main stream CNNs, *i.e.,* VGGNet (Simonyan & Zisserman, 2014)[2], ResNet (He et al., 2016)[2], DenseNet (Huang et al., 2017)[2], MobileNet (Sandler et al., 2018)[3], and EfficientNet (Tan & Le, 2019a)[4]. To better demonstrate the effectiveness of our approach in learning lightweight networks, we mainly compare it with state-of-the-art (1) lightweight networks and (2) network compression methods, including L1 (Li et al., 2016), SSS (Huang & Wang, 2018), Variational Pruning (Zhao et al., 2019b), HRank (Lin

[2]https://pytorch.org/docs/stable/torchvision/models.html
[3]https://github.com/tonylins/pytorch-mobilenet-v2
[4]https://github.com/lukemelas/EfficientNet-PyTorch

et al., 2020),NISP (Yu et al., 2018), GAL (Lin et al., 2019), Hinge (Li et al., 2020), CNN-FCF (Li et al., 2019b), Group Lasso (Oyedotun et al., 2020), L2PF (Huang et al., 2018), EGL (Oyedotun et al., 2020) and DEGL (Oyedotun et al., 2020), DCP-A (Zhuang et al., 2018), Slimming (Liu et al., 2017) and GBN (You et al., 2019).

Implementation. We use PyTorch to implement our network architecture. Following the literature as well as the original code for each network, in our experiments we use the SGD optimizers with the cross-entropy loss and set the initial learning rate, momentum, and decay as 0.05, 0.9, and 0.0005, respectively. The learning rate is divided by 2 every 30 epochs on CIFAR-10 and by 10 every 10 epochs on ImageNet. We use Top-1 error as our performance measure for both datasets. We report our results based on three random trials in terms of mean and standard deviation.

4.1 Results Summary

We summarize our results in Table 2 based on seven classic network architectures. In general, we use grid search to determine which convolutional layers in the backbone network should be replaced by CSR-Conv layer. Overall, CSR-Conv can be used to learn smaller but better lightweight networks based on different backbones. Specifically,

- CSR-Conv can effectively learn lightweight networks using less than half of the model sizes of the backbone networks with no, or only $< 1\%$ performance loss. On CIFAR-10, CSR-Conv can even achieve $\rho_M > 80\%$ with $1\% \sim 3\%$ performance loss.
- CSR-Conv seems to be able to improve the performance by $0.1\% \sim 1\%$ when $\rho_M < 50\%$.
- CSR-Conv performs stably, as the standard deviation ranging from 0.04% to 0.31%.
- Often more CSR-Conv layers are needed to learn more lightweight networks. Meanwhile, deeper RNNs are desirable for better performance. This validates our motivation.

4.2 Comparison with Lightweight Networks

We also compare our CSR-Conv based networks with the state-of-the-art lightweight networks. The comparison results are listed in Table 3, where we show 3 CSR-Conv based networks with EfficientNet and MobileNet as our backbones. It is clear that CSR-Conv with EfficientNet has the lowest error among all the networks. The "lighter" models with the MobileNet backbone also have similar or better performance comparing to the networks with similar model sizes such as MUXNet-s and DY-MobileNetV2 x0.35. Note that the DY-MobileNetV2 x0.35 model also uses MobileNetV2 as the backbone network, and our model can achieve significantly better performance with even less parameters. This also validates the effectiveness of our CSR-Conv layer. Since these competitors are based on standard linear convolutions, we strongly believe that our CSR-Conv layer can further reduce the model sizes of such networks while preserving (even improving) their performance. Also, post-processing such as pruning can be applied to our networks to achieve smaller networks. See Table 6 later for example.

Table 3: Lightweight network comparison on ImageNet in terms of the number of parameters and top-1 error. Numbers are cited from https://paperswithcode.com/sota/image-classification-on-imagenet. All the networks with model sizes smaller than 5M are included.

Networks	#Param.	Err. (%)
MUXNet-xs (Lu et al., 2020)	1.8M	33.3
MUXNet-s (Lu et al., 2020)	2.4M	28.4
Ours-1 (MobileNet-V2)	**2.5M**	**29.5**
DY-MobileNetV2 x0.35 (Chen et al., 2020)	2.8M	35.1
Ours-2 (MobileNet-V2)	**3.0M**	**27.7**
ECA-Net (Wang et al., 2020)	3.3M	27.4
PVTv2-B0 (Wang et al., 2021)	3.4M	29.5
MUXNet-m (Lu et al., 2020)	3.4M	24.7
MnasNet-A1 (Tan et al., 2019)	3.9M	24.7
DY-MobileNetV2 x0.5 (Chen et al., 2020)	4.0M	30.6
Proxyless (Cai et al., 2018)	4.0M	25.4
MUXNet-l (Lu et al., 2020)	4.0M	23.4
MixNet-S (Tan & Le, 2019b)	4.1M	24.2
Ours-3 (Efficient-B0)	**4.3M**	**22.3**
GreedyNAS-C (You et al., 2020)	4.7M	23.8
DY-MobileNetV3-Small (Chen et al., 2020)	4.8M	30.3
MnasNet-A2 (Tan et al., 2019)	4.8M	24.4
ViTAE-T-Stage (Xu et al., 2021)	4.8M	23.2
PiT-Ti (Heo et al., 2021)	4.9M	25.4

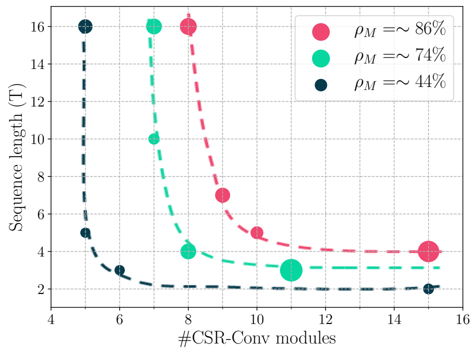


Figure 4: VGG-16 error (dot size) on CIFAR-10. Each curve indicates similar ρ_M values.

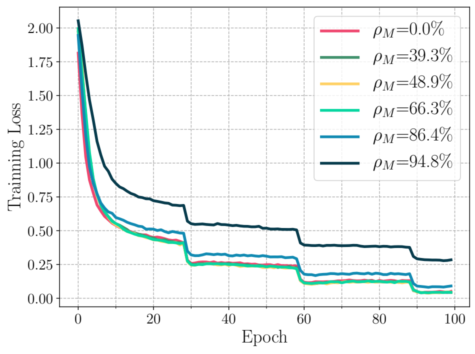


Figure 5: Training loss comparison using the VGG-16 backbone on CIFAR-10.

4.3 Comparison with Network Compression

Fig. 3 illustrates our comparison with the state-of-the-art on both CIFAR-10 and ImageNet, where methods towards the bottom right corner are preferred. We can see the performance trends as discussed above for Table 2. Surprisingly, our approach forms "lower-bound" curves in each subfigure, indicating that given similar model compression rates CSR-Conv often works best. This is because of the overparameterization in the neural networks so that we have a sufficiently large parameter space to identify a better yet lightweight architecture. Thanks to our design, CSR-Conv has the flexibility of exploring the performance with a specific compression rate. In summary, CSR-Conv can manage to learn lightweight networks effectively, consistently and robustly using different backbone networks on large-scale complicated datasets.

4.4 Ablation Study

Impacts of the hidden state transition in vanilla RNNs. The hidden state transition helps construct deeper networks, compared with the backbones, to compensate for the performance loss when learning lightweight networks. To verify this usage, we compare our model with a baseline model with shared weights in group convolutions (denoted as "s-GroupConv"), as illustrated in Fig. 1(b), to replace our CSR-RNN layers. We then tune such networks so that the model size compression ratios are approximately the same as ours. We list some results in Table 4, where we can see that in all the cases our results are consistently better than this baseline, demonstrating the need of the hidden state transition.

Table 4: Top-1 error (%) comparison on CIFAR-10 using VGG-16.

Networks	$\rho_M(\downarrow)$	Ours	s-GroupConv
CSR-Conv-1	39.4%	5.89	6.23
CSR-Conv-2	49.0%	6.01	6.56
CSR-Conv-3	67.2%	6.16	7.03
CSR-Conv-4	86.5%	6.35	7.27
CSR-Conv-5	95.0%	7.08	7.82

Impacts of the number of CSR-Conv layers and input sequence length. Recall that we use grid search to seek a lightweight network architecture to meet a certain model size compression rate, if required. We take VGG-16 for example to demonstrate their impacts on the performance, as illustrated in Fig. 4. Note that we select convolutional layers in the VGG-16 architecture in descending order. We can see that:

- The model compression rates towards the bottom left corner are lower and lower.
- Given the same model size compression rate, the networks form nice "U" shaped contours where more CSR-Conv layers need short sequence length.
- Lightweight networks with small errors, given compression rates, are distributed along the valley.

These observations are very useful as guidance for our approach on how to search for a lightweight network architecture effectively.

RNN variants, GRU, and LSTM as the recurrent layer. Overall, we do not observe any significant performance improvement over the vanilla RNN implementation. For instance, to learn lightweight networks based on VGG-16 with a model compression rate of $\sim$87% on CIFAR-10, vanilla RNN, incremental RNN, and FastRNN achieve 6.35%, 6.45%, and 7.87% in terms of classification error, respectively. Using the same amount of parameters Lipschitz RNN achieves 6.55% error with a model compression rate of $\sim$78%. Similarly, we replace vanilla RNNs with GRUs and LSTMs to learn lightweight networks based on ResNet-56 that achieve (10.9%, 7.53%) and (-18.8%, 7.80%) in terms of (ρ_M, error) on CIFAR-10, respectively. These results are worse than vanilla RNNs as well, probably due to the short sequence length. Therefore, by default we utilize vanilla RNNs as the recurrent layer in our CSR-Conv.

FLOP reduction. As demonstration, we verify the FLOP reduction of our approach using ResNet-56 in practice and list our results in Table 5. Recall that our main focus of the paper is to learn lightweight networks, and FLOPs tend to decrease as well with the increase of sequence length, in general. For CSR-Conv-1, the input and output dimensions are the same so that $T = 1 + \frac{D}{d}$ holds, and thus no drop in FLOPs exists. Such results in Table 5 also verify Prop. 2 properly.

Table 5: Comparison on CIFAR-10.

Networks	Err.(%)	FLOPs(↓)	Param(↓)
ResNet-56	6.74	0.0%	0.0%
CSR-Conv-1	6.12	0.0%	21.8%
CSR-Conv-2	6.69	12.8%	61.0%
CSR-Conv-3	6.83	17.2%	70.3%
CSR-Conv-4	7.93	29.6%	78.8%
CSR-Conv-5	9.15	43.5%	88.9%

Running time. Recall that our CSR-Conv layer leads to deeper networks that need to be optimized/inferred sequentially. Therefore, our running time is heavily dependent on the numbers of CSR-Conv layers in the networks, as well as the bottleneck computation in the backbone networks. For instance, the training time is 0.3ms per batch on a Quadro RTX 6000 GPU when we run ResNet-56 on CIFAR-10 dataset. Under the same setting, CSR-Conv-1 (CSR-Conv-5) involves 4 (22) CSR-Conv layers and runs for 0.56ms (1.31ms), with the increase of compression rate from 21.8% to 88.9%. Differently, on ImageNet the MobileNet-V2 architecture takes 1.068s to train per batch and CSR-Conv-1 (CSR-Conv-2) takes 1.071s (1.096s) that involves 2 (7) CSR-Conv layers.

Training curves. It is critical to make sure that our lightweight networks are easy to train even with a small portion of parameters and RNNs that share parameters. To demonstrate this, we illustrate our training curves of VGG-16 in Fig. 5 where $\rho_M = 0$ denotes the backbone network and the rest are the variants of our approach. For simplicity, we only plot the training curves of the first 100 epochs. As expected, the networks with a higher compression rate are more difficult to train, leading to larger training losses and test errors. Note that the trends of loss are very similar to each other, indicating that our lightweight yet deeper networks can be trained as easily as backbone networks.

Further compression with existing methods. Note that the learned filters in our CSR-Conv layers are still dense, and thus we can apply network compression methods as post-processing to further reduce the model size. We list some results in Table 6 using the classic compression algorithm in (Han et al., 2015a) to prune our learned lightweight networks. It is apparent that the pruning algorithm can further reduce $\sim$ 5% of model sizes with marginal $\sim$ 0.06% error increase. These results show that our CSR-Conv layer can be considered as being orthogonal to the literature of network compression.

Table 6: Pruning results on CIFAR-10 based on our learned CSR-Conv networks. Here, VGG-16 and ResNet-56 are two backbone networks.

Network	Err.(%)	ρ_M(↓)
CSR-Conv + VGG-16	6.35	86.5%
CSR-Conv + VGG-16 + (Han et al., 2015a)	6.40	91.9%
CSR-Conv + ResNet-56	6.83	70.3%
CSR-Conv + ResNet-56 + (Han et al., 2015a)	6.90	75.3%

5 Conclusion

In this paper, we aim to address the problem of learning lightweight networks by proposing a novel CSR-Conv layer that replaces traditional linear convolution with channel-split recurrent convolution. The hidden state transition in the vanilla RNNs leads to deeper networks, given backbones, to compensate for the performance loss while reducing the model sizes. Essentially our CSR-Conv can be viewed as the generalization of linear convolution. We show that the model size of a lightweight network decreases *w.r.t.* the number of the duplicate networks with the rate of $O(\frac{1}{T^2})$. We then conduct comprehensive experiments to evaluate our CSR-Conv on CIFAR-10 and ImageNet. We demonstrate state-of-the-art performance on learning lightweight networks in terms of accuracy *vs.* model size. We can even further improve our results by integrating existing network compression algorithms such as pruning.

REFERENCES

Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pp. 2270–2278, 2016. 3

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128, 2016. 3

Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Lcnn: Lookup-based convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7120–7129, 2017. 3

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 1, 7

Bo Chang, Minmin Chen, Eldad Haber, and Ed H. Chi. AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryxepo0cFX. 3

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11039, 2020. 7

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. 1

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 3

Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, pp. 1–43, 2020. 1

Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. Capacity and Trainability in Recurrent Neural Networks. *arXiv e-prints*, November 2016. 3

Enric Corona, Albert Pumarola, Guillem Alenya, and Francesc Moreno-Noguer. Context-aware human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. 1

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 2, 6

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4): 485–532, 2020. 1

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pp. 2148–2156, 2013. 3

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pp. 1269–1277, 2014. 1

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy. 2

Elad Eban, Yair Movshovitz-Attias, Hao Wu, Mark Sandler, Andrew Poon, Yerlan Idelbayev, and Miguel A Carreira-Perpinán. Structured multi-hashing for model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11903–11912, 2020. 1

N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, and Michael W Mahoney. Lipschitz recurrent neural networks. *arXiv preprint arXiv:2006.12070*, 2020. 3

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. In *Advances in Neural Information Processing Systems*, pp. 5197–5205, 2018. 3

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015. 1

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 1

Shupeng Gui, Haotao N Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*, pp. 1285–1296, 2019. 1

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. 1, 3, 9

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. 3

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. 1, 2, 6

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018. 3

Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021. 7

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 3

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018. 1

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. 6

Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 709–718. IEEE, 2018. 7

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 304–320, 2018. 3, 6

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. 1

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 3

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, 2014. 1

Anil Kag, Ziming Zhang, and Venkatesh Saligrama. Rnns incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In *International Conference on Learning Representations*, 2020. 3

Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung. Efficient neural network compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12569–12577, 2019. 1

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1637–1645, 2016. 2, 3

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 6

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012. 1

Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*, pp. 9017–9028, 2018a. 1

Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*, 2018b. 3

Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In *Advances in neural information processing systems*, pp. 7517–7527, 2018. 3

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 3

Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pp. 3794–3803, 2019. 3

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1, 6

Jiashi Li, Qi Qi, Jingyu Wang, Ce Ge, Yujian Li, Zhangzhang Yue, and Haifeng Sun. Oicsr: Out-in-channel sparsity regularization for compact deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7046–7055, 2019a. 1

Tuanhui Li, Baoyuan Wu, Yujiu Yang, Yanbo Fan, Yong Zhang, and Wei Liu. Compressing convolutional neural networks via factorized convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3977–3986, 2019b. 3, 7

Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 7

Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6438–6447, June 2021. 3

Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3367–3375, 2015. 2, 3

Siyu Liao and Bo Yuan. Circconv: A structured convolution with low complexity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4287–4294, 2019. 3

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2020. 6

Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, 2019. 7

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 806–814, 2015. 1

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017. 3, 7

Zhichao Lu, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Muxconv: Information multiplexing in convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12044–12053, 2020. 7

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6231–6239, 2017. 1

James O' Neill. An overview of neural network compression. *arXiv preprint arXiv:2006.03669*, 2020. 1

Peter Ondruska, Julie Dequaire, Dominic Zeng Wang, and Ingmar Posner. End-to-end tracking and semantic segmentation using recurrent neural networks. *arXiv preprint arXiv:1604.05091*, 2016. 2, 3

Oyebade Oyedotun, Djamila Aouada, and Bjorn Ottersten. Structured compression of deep neural networks with debiased elastic group lasso. In *The IEEE Winter Conference on Applications of Computer Vision*, pp. 2277–2286, 2020. 7

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019. 3

Lourenco V. Pato, Renato Negrinho, and Pedro M. Q. Aguiar. Seeing without looking: Contextual rescoring of object detections for ap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

Anish Prabhu, Ali Farhadi, Mohammad Rastegari, et al. Butterfly transform: An efficient fft based neural architecture design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12024–12033, 2020. 1

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. 1, 2, 6

Xingjian Shi, Zhourong Chen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 2015:802–810, 2015. 2, 3

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 6

Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. *arXiv preprint arXiv:1905.04446*, 2019. 1

Courtney J Spoerer, Patrick McClure, and Nikolaus Kriegeskorte. Recurrent convolutional neural networks: a better model of biological object recognition. *Frontiers in psychology*, 8:1551, 2017. 2

Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015. 3

Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3147–3155, 2017. 2, 3

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019a. 1, 4, 6

Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *arXiv preprint arXiv:1907.09595*, 2019b. 7

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019. 1, 7

Jianfeng Wang and Xiaolin Hu. Gated recurrent convolution neural network for ocr. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 334–343, 2017. 2, 3

Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 7

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021. 7

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016. 1, 3

Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9127–9135, 2018. 3

Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *arXiv preprint arXiv:2106.03348*, 2021. 7

Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1999–2008, 2020. 7

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 2133–2144, 2019. 7

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018. 7

Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7370–7379, 2017. 3

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(3):2224–2287, 2019. 1

Jun Zhang and Hongquan Zuo. A deep rnn for ct image reconstruction. In *Medical Imaging 2020: Physics of Medical Imaging*, volume 11312, pp. 113124N. International Society for Optics and Photonics, 2020. 3

Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015. 3

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. 1, 2

Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Cam-rnn: Co-attention model based rnn for video captioning. *IEEE Transactions on Image Processing*, 28(11):5552–5565, 2019a. 2

Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Tth-rnn: Tensor-train hierarchical recurrent neural network for video summarization. *IEEE Transactions on Industrial Electronics*, 2020. 3

Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2780–2789, 2019b. 6

Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Building efficient deep neural networks with unitary group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11303–11312, 2019c. 1

Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. 2

Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pp. 662–677. Springer, 2016. 1

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 875–886, 2018. 7