2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)

2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) | 978-1-6654-9346-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/WACV56688.2023.00385

Learning Lightweight Neural Networks via Channel-Split Recurrent Convolution

Guojun Wu, Xin Zhang, Ziming Zhang, Yanhua Li
Worcester Polytechnic Institute
{gwu, xzhang17, zzhang15, yli15}@wpi.edu

Xun Zhou
University of Iowa
xun-zhou@uiowa.edu

Christopher Brinton
Purdue University
cgb@purdue.edu

Zhenming Liu
College of William & Mary
zliu@cs.wm.edu

Abstract

Lightweight neural networks refer to deep networks with small numbers of parameters, which can be deployed 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. Codes can be found on https://github.com/tuaxon/CSR_Conv.*

1. Introduction

Convolutional neural networks (CNNs) have revolutionized computer vision by achieving state-of-the-art (SOTA) performance on many applications. In practice, there are many applications utilizing CNNs. As shown in Fig. 1, *Semantic Segmentation* aims to label each pixel of an image with a corresponding class of what is being represented [17], and *Classification Task* classifies images based on its main characters [31]. Besides, if there are more objects in one image, we need to detect those objects [51] and find corresponding pixels [43]. The impressive improvement usually comes with a substantial increase in the number of parameters (*i.e.,* model size) which is undesirable in many real-world applications [14, 15], such as embedded systems where the computing resources (*e.g.,* processor and memory) in the hardware are limited. We list some ad-hoc tasks in

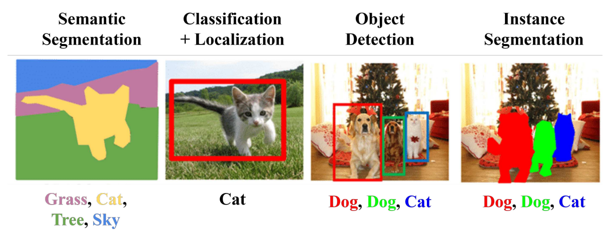

Figure 1: Illustration of various image processing tasks.

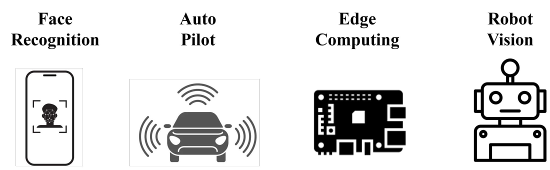

Figure 2: Applications that need lightweight networks.

Fig. 2 where lightweight models are needed, like face recognition on a cellphone, auto piloting with object detection on cars or trucks, pedestrians re-identification on CCTV cameras and vision-related tasks for a robot. Therefore, how to design/learn lightweight neural networks, *i.e.,* reducing storage requirement in parameters while achieving good performance, is becoming increasingly demanded [52, 34, 55].

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 [27], MobileNet [52], ShuffleNet [75], EfficientNet [60], MnasNet [59], and ProxylessNAS [2]. 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 approaches such as compression with learning [50, 23, 75] or after learning [16, 32], whose basic ideas are to remove the network redundancy by imposing some structural assumptions on the convolutional filters. Nice surveys on this topic are in [9, 46].

Motivation. Intuitively, reducing the number of parameters

2642-9381/23/$31.00 ©2023 IEEE
DOI 10.1109/WACV56688.2023.00385

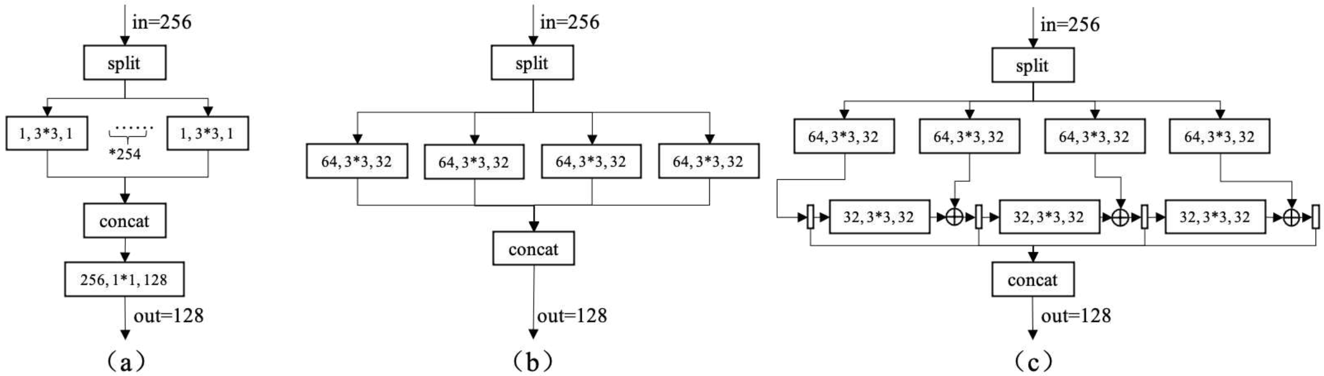

Figure 3: Comparison with 256 input channels and 128 output channels among **(a)** depth-wise separable convolution, **(b)** group convolution, and **(c)** our CSR-Conv using vanilla RNNs. The linear convolutional operation is denoted as (#input channels, filter size, #output channels) and vertical small rectangles in **(c)** denote ReLU activation functions.

in each convolutional layer can significantly compact a given network. However, this may lead to poor network generalization, as wider networks are shown to effectively improve the performance [73, 60]. To compensate for the performance loss due to model size reduction (*i.e.,* lightweight networks), we are motivated by the following works:

- *Deeper Networks:* In complement to the universal approximation theorem [7], recent works such as [45] have shown that with the increase of network depth, the number of hidden neurons can be dramatically reduced to approximate a function with similar expressive power. This motivates us to construct a deeper network using narrow networks.
- *Visual Transformer (ViT):* Recently [11] 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 explore RNNs (not ViT due to its large model size) in different ways to learn lightweight networks.

Our proposed approach and contributions. Based on considerations above, we propose a novel convolutional layer, namely *Channel-Split Recurrent Convolution (CSR-Conv)*, as illustrated in Fig. 3(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 demonstrated) 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 [52]) and group convolution (used in ShuffleNet [75] and ResNeXt [67]) in Fig. 3(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. 3(a-b) the depths are both 2, while in Fig. 3(c) the depth is 5. In other words, recurrent convolution can lead to deeper network architectures, which is beneficial for learning lightweight convolutional networks.

We are aware that the integration of RNNs with convolution for deep learning has been explored in literatures [62, 58, 53, 47, 56]. 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 such as VGGNet [54] or ResNet [18], we can easily replace its linear convolutional layers using our CSR-Conv to reduce the model size, achieving a deeper network while preserving its performance[1]. We analyze the relationship between model size and CSR-Conv to show its controllable model compression rate. We also demonstrate SOTA performance of our approach based on seven existing network architectures on CIFAR-10 [30] and ImageNet [8] datasets.

2. Related Works

Network compression. Weight pruning [16, 37] aims at reducing non-significant weights to reduce computation and memory usage of a model. Other than that, filter level pruning which leads to the removal of the corresponding feature maps is also studied intensively [19, 35]. Regularization constraints are also introduced in pruning [42, 26]. Low-rank factorization [57, 72] aims to decompose the large weight matrices in the convolutional layers into smaller matrices with fewer parameters. Knowledge distillation [21, 49] aims to force a smaller student network to fit specific features from a larger teacher network for knowledge transfer.

Variants of convolutional operator for compression. [10] proposed using a linear combination of basis functions to predict parameters for compression. [1] proposed encoding convolutions by few lookups to a dictionary trained to cover

[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.

the space of weights in CNNs. [66] presented a parameter-free, FLOP-free "shift" operation as an alternative to spatial convolutions. [13] proposed channel-wise convolutions, which replace dense connections among feature maps with sparse ones in CNNs. [39] 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.

Recurrent neural networks. RNNs have achieved significant success in learning complex patterns for sequential input data, and have been widely used in computer vision [74, 6]. 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) [22], gated recurrent unit (GRU) [5], FastGRNN [33], antisymmetric RNN [3], incremental RNN [28], and Lipschitz RNN [12].

Recurrent convolutional neural networks (RCNN). [38] proposed incorporating the recurrent connections in each convolutional layer to generate features with different resolutions. [62] added a gate to the recurrent connections in RCNN to control context modulation and balance the feed-forward information and the recurrent information. [29] imposed very deep recursive layers to improve performance without introducing new parameters for additional convolutions. [58] developed a recursive CNN with the residual connection. [53] replaced vanilla RNN architecture with an LSTM structure in RCNN. [47] used dilated convolution in the RCNN to reduce computational complexity.

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:

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

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.

Table 1: Illustration of our CSR-Conv-4 architecture in Tab. 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%)

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 [60], 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 Tab. 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.

3.2. CSR-Conv

We illustrate the architecture of CSR-Conv in Fig. 4, 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. 3 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.

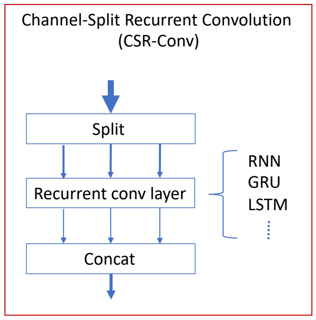

Figure 4: General architecture

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:

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

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 hidden state vectors, we concatenate them into a $(D_l \times T)$-dim vector $\mathbf{h}_l = [\mathbf{h}_{l,t}^T]^T$ to 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:*

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

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:*

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

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 Tab. 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.

4. Experiments

Datasets. Following the literature, we evaluate our approach comprehensively on CIFAR-10 [30] and ImageNet [8] 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 [54][2], ResNet [18][2], DenseNet [24][2], MobileNet [52][3], and EfficientNet [60][4]. To better demonstrate the effectiveness of our approach in learning lightweight networks, we mainly compare it with SOTA (1) lightweight networks and (2) network compression methods, including L1 [34], SSS [26], Variational Pruning [76], HRank [40], NISP [71], GAL [41], Hinge [36], CNN-FCF [35], Group Lasso [48], L2PF [25], EGL [48] and DEGL [48], DCP-A [77], Slimming [42] and GBN [70].

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 Tab. 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, i). 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. ii). CSR-Conv seems to be able to improve the performance by $0.1\% \sim 1\%$ when $\rho_M < 50\%$. iii). CSR-Conv performs stably, as the standard deviation ranging from 0.04% to 0.31%. iv). Often more CSR-Conv layers are needed to learn more lightweight networks. Meanwhile, deeper RNNs leads to better performance. This validates our motivation.

4.2. Comparison with Lightweight Networks

We also compare our CSR-Conv based networks with the SOTA lightweight networks. The comparison results are listed in Tab. 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 Tab. 7 later for example.

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

Table 3: Lightweight network comparison on ImageNet in terms of the number of parameters and top-1 error. Numbers are cited from [65]. All the networks with model sizes smaller than 5M are included.

Networks	#Param.	Err. (%)
MUXNet-xs [44]	1.8M	33.3
MUXNet-s [44]	2.4M	28.4
Ours-1 (MobileNet-V2)	**2.5M**	**29.5**
DY-MobileNetV2 x0.35 [4]	2.8M	35.1
Ours-2 (MobileNet-V2)	**3.0M**	**27.7**
ECA-Net [63]	3.3M	27.4
PVTv2-B0 [64]	3.4M	29.5
MUXNet-m [44]	3.4M	24.7
MnasNet-A1 [59]	3.9M	24.7
DY-MobileNetV2 x0.5 [4]	4.0M	30.6
Proxyless [2]	4.0M	25.4
MUXNet-l [44]	4.0M	23.4
MixNet-S [61]	4.1M	24.2
Ours-3 (Efficient-B0)	**4.3M**	**22.3**
GreedyNAS-C [69]	4.7M	23.8
DY-MobileNetV3-Small [4]	4.8M	30.3
MnasNet-A2 [59]	4.8M	24.4
ViTAE-T-Stage [68]	4.8M	23.2
PiT-Ti [20]	4.9M	25.4

Table 4: Top-1 error (%) comparison on CIFAR-10 via 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

4.3. Comparison with Network Compression

Fig. 5 illustrates our comparison with the SOTA 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 Tab. 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.

Table 5: Top-1 error(%) results on ImageNet via ResNet-50

Networks	$\rho_M(\downarrow)$	Ours	s-GroupConv
CSR-Conv-1	35.7%	23.51	24.81
CSR-Conv-2	70.3%	24.61	25.63

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, we compare our model with a baseline with shared weights in group convolutions (denoted as "s-GroupConv"), as illustrated in Fig. 3(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 Tab. 4 and Tab. 5, 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.

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. 6. Note that we select convolutional layers in the VGG-16 architecture in descending order. We can see that: i). the model compression rates towards the bottom left corner are lower and lower; ii). given the same model size compression rate, the networks form nice "U" shaped contours where more CSR-Conv layers need short sequence length; iii). lightweight networks with small errors, given compression rates, are distributed along the valley. These observations are useful guidance for our approach on searching 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 ~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 ~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. We verify the FLOP reduction of our approach using ResNet-56 in practice and list our results in

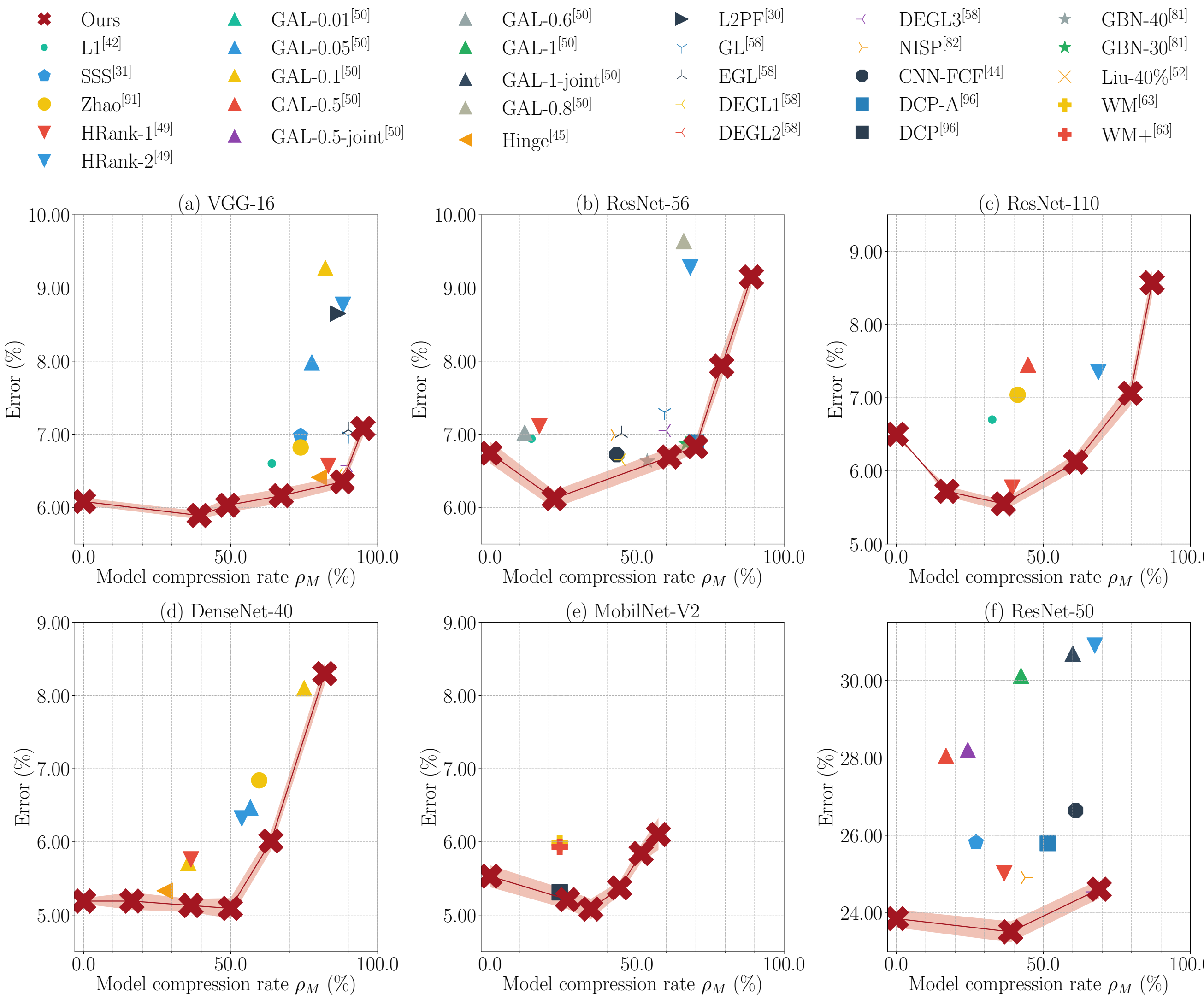

Figure 5: Comparison of error *vs.* compression rate on **(a-e)** CIFAR-10 and **(f)** ImageNet.

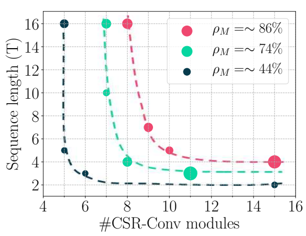

Figure 6: VGG-16 error (dot size) on CIFAR-10. Each curve indicates similar ρ_M.

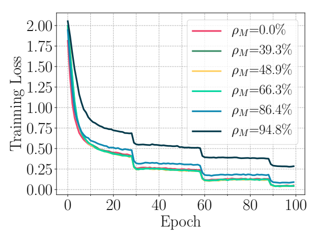

Figure 7: Training loss comparison using the VGG-16 backbone on CIFAR-10.

Tab. 6. 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 Tab. 6 also verify Prop. 2 properly.

Table 6: 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 number of CSR-Conv layers in the networks and the bottleneck computation in the backbones. 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

Table 7: Pruning results on CIFAR-10(VGG-16)/ImageNet(Resnet-50) based on our CSR-Conv.

Network	Err.(%)	$\rho_M(\downarrow)$
CSR-Conv + VGG-16	6.35	86.5%
CSR-Conv + VGG-16 + [16]	6.40	91.9%
CSR-Conv + ResNet-50	23.51	35.7%
CSR-Conv + ResNet-50 + [16]	23.90	30.4%

same setting, CSR-Conv-1 (CSR-Conv-5) involves 4 (22) CSR-Conv layers and runs for 0.56ms (1.31ms), with the compression rate increasing from 21.8% to 88.9%. Differently, on ImagetNet the MobileNet-V2 architecture takes 1.068s to train each 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. We therefore illustrate our training curves of VGG-16 in Fig. 7 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 Tab. 7 using the classic compression algorithm in [16] to prune our learned lightweight networks. It is apparent that the pruning algorithm can further reduce model sizes with marginal error increase on both datasets. These results show that our CSR-Conv layer can be considered as being orthogonal to the literature of network compression.

4.5. Future Work

In this work, we proposed the CSR-Conv architecture to construct lightweight CNNs as a replacement of vanilla convolution layers. With extensive experiments, we show the CSR-Conv layer's effectiveness compared to various baselines and its potential on constructing tiny neural networks. However, there remain many open questions to stress in future work. First, which recurrent unit backbones to use. In Fig. 4, we show that the recurrent conv layer can be implemented with different recurrent units. Though having tested on some structures, we are curious about whether the performance could be better if we use more "modern" architectures such as the attention mechanism, or whether GRU or LSTM can improve the performance on more complex tasks. Second, how to pick which layer to compress and set a proper compression rate. For now, we have some heuristics such as replacing latter layers rather than the first few layers. That is the reason why our model with more parameters can sometimes have worse performance than its "lighter" variants. However, it is interesting to see how this architecture cooperates with NAS methods to determine the best layer numbers and sequence length for each layer. Last but not least, how to apply beyond GPUs. We want to implement this architecture on edge devices like Raspberry Pi to validate its performance on real world tasks. However, we would argue that even lightweight network on GPU is critical to real world applications. For example, tech giants like Google and Facebook often deploy model with parameters consuming hundreds of GBs in storage. It would be beneficial if we can reduce the storage space with lightweight networks.

5. Conclusion

In this paper, we aim to address the problem of lightweight network by proposing a novel yet embarrassingly simple approach, CSR-Conv, that generalizes the conventional convolutional layers without any explicit structural assumption on filters. By taking the advantage of shared parameters in RNNs, we manage to replace linear convolutional layers with large number of parameters with small RNNs that can approximate more complicated nonlinear functions with fewer parameters. To train such RNNs, we divide the input and output channels of convolution into groups to generate input and output sequences. We unveil that the model and FLOPs compression rates in our approach depend on not only the network architecture but also the sequence length, *i.e.,* quadratically and linearly, respectively. We then conduct comprehensive experiments to evaluate our RNN-Conv approach to compress VGG-16, ResNet-56, ResNet-110, and DenseNet-40 on CIFAR-10, and ResNet-50 on ImageNet. We can achieve SOTA performance with similar training and inference speed to the original networks. We can even further improve our results by integrating with existing network compression algorithms such as pruning. We hope that our approach can provide a simple baseline for lightweight neural network research in the future.

6. Acknowledgement

Guojun Wu, Xin Zhang and Yanhua Li were supported in part by NSF grants IIS-1942680 (CAREER), CNS-1952085, CMMI1831140, and DGE-2021871. Ziming Zhang was supported by NSF grant CCF-2006738. Xun Zhou was funded in part by the SAFER-SIM University Transportation Center through a grant from the U.S. Department of Transportation (69A3551747131). Christopher Brinton was funded by NSF grant CNS-2146171, and Office of Naval Research (ONR) grant N000142212305. Zhenming Liu was supported by NSF grants NSF-2008557, NSF-1835821, and NSF-1755769.

References

[1] 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*, pages 7120–7129, 2017.

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

[3] 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.

[4] 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*, pages 11030–11039, 2020.

[5] 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.

[6] 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.

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

[8] 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*, pages 248–255. Ieee, 2009.

[9] 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.

[10] 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*, pages 2148–2156, 2013.

[11] 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.

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

[13] 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*, pages 5197–5205, 2018.

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

[15] 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*, pages 1285–1296, 2019.

[16] 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*, 2015.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] 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*, pages 770–778, 2016.

[19] 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.

[20] 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.

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

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

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[24] 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*, pages 4700–4708, 2017.

[25] 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)*, pages 709–718. IEEE, 2018.

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

[27] 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.

[28] 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.

[29] 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*, pages 1637–1645, 2016.

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages

1097–1105, 2012.
[32] 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*, pages 9017–9028, 2018.
[33] 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*, 2018.
[34] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.
[35] 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*, pages 3977–3986, 2019.
[36] 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.
[37] 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)*, pages 6438–6447, June 2021.
[38] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3367–3375, 2015.
[39] Siyu Liao and Bo Yuan. Circconv: A structured convolution with low complexity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4287–4294, 2019.
[40] 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*, pages 1529–1538, 2020.
[41] 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*, pages 2790–2799, 2019.
[42] 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*, pages 2736–2744, 2017.
[43] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.
[44] 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*, pages 12044–12053, 2020.
[45] 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, pages 6231–6239, 2017.
[46] James O' Neill. An overview of neural network compression. *arXiv preprint arXiv:2006.03669*, 2020.
[47] 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.
[48] 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*, pages 2277–2286, 2020.
[49] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.
[50] 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*, pages 12024–12033, 2020.
[51] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
[52] 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*, pages 4510–4520, 2018.
[53] 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.
[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[55] 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.
[56] 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.
[57] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.
[58] 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*, pages 3147–3155, 2017.
[59] 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*, pages 2820–2828, 2019.

[60] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[61] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *arXiv preprint arXiv:1907.09595*, 2019.

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

[63] 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.

[64] 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.

[65] Papers with Code. Image Classification on ImageNet. `https://paperswithcode.com/sota/image-classification-on-imagenet`, 2022.

[66] 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*, pages 9127–9135, 2018.

[67] 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.

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

[69] 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*, pages 1999–2008, 2020.

[70] 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*, pages 2133–2144, 2019.

[71] 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*, pages 9194–9203, 2018.

[72] 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*, pages 7370–7379, 2017.

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

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

[75] 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*, 2018.

[76] 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*, pages 2780–2789, 2019.

[77] 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*, pages 875–886, 2018.