

# SD-Conv: Towards the Parameter-Efficiency of Dynamic Convolution

Shwai He<sup>1,2</sup>Chenbo Jiang<sup>3</sup>Daize Dong<sup>2</sup>Liang Ding<sup>1\*</sup><sup>1</sup>JD Explore Academy<sup>2</sup>University of Electronic Science and Technology of China<sup>3</sup>Nanjing University of Science and Technology Zijin Collegeshwai.he@gmail.com, cbjiang@foxmail.com,  
dzdong2019@gmail.com, dingliang1@jd.com

## Abstract

*Dynamic convolution achieves better performance for efficient CNNs at the cost of negligible FLOPs increase. However, the performance increase can not match the significantly expanded number of parameters, which is the main bottleneck in real-world applications. Contrastively, mask-based unstructured pruning obtains a lightweight network by removing redundancy in the heavy network. In this paper, we propose a new framework, **Sparse Dynamic Convolution** (SD-CONV), to naturally integrate these two paths such that it can inherit the advantage of dynamic mechanism and sparsity. We first design a binary mask derived from a learnable threshold to prune static kernels, significantly reducing the parameters and computational cost but achieving higher performance in Imagenet-1K. We further transfer pretrained models into a variety of downstream tasks, showing consistently better results than baselines. We hope our SD-Conv could be an efficient alternative to conventional dynamic convolutions.*

## 1. Introduction

There have been rich discussions on the representation power of deep neural networks in two opposite directions [36, 45]. From the perspective of increasing the model capacity, more layers and channels with specialized infrastructure (e.g. dynamic convolution [7]) can achieve higher performance with less overfitting [1, 41]. In the view of model compression, network pruning and quantization of complex networks can induce smaller models possibly at the expense of minor accuracy loss [12, 14]. Regarding the trade-off between cost and gain in these two opposite approaches, what will happen when we combine them for infrastructure optimization? Especially, can we combine the

advances of dynamic convolution and sparsity towards the best of both worlds – achieving a desirable trade-off between complexity and performance?

Dynamic convolution (DY-Conv) [7] achieves significant performance gains over static convolution with negligible computational cost but relatively high memory cost. Specifically, it utilizes an input-based attention mechanism to generate dynamic attention weights to combine multiple parallel static kernels, boosting the performance at the cost of increased convolutional parameters.

However, during inference, this parameter increase does not match the model performance improvement completely. For example, DY-ResNet-18 [7] is 2.3% higher than ResNet-18 and 4% lower than ResNet-50 in Top-1 accuracy on Imagenet-1K [8], while its parameter amount is about four times of ResNet-18 and twice of ResNet-50. In addition, large-scale DNNs require huge storage and deployment cost, which becomes the main bottleneck of the real-world deployment [13, 26, 30]. These phenomena raise the problem of parameter efficiency when we try to adopt dynamic convolution more efficiently.

A possible routine to improve the parameter efficiency in dynamic convolution is to refine the method of kernel combination. For example, Li *et al.* [34] reformulate the linear combination of dynamic convolution into a summation of the static kernel and sparse dynamic residual. Another scheme is to increase the sparsity to build compact parameter-efficient networks. One can prune task-unrelated neurons that usually have small absolute values [13, 17, 56] or contribute little to the decrease of loss function [32, 33, 44].

In recent studies, some sparse networks not only decrease storage and computational requirements but also achieve higher inference scores than dense networks [10], suggesting the potential utility of sparse structure in alleviating overfitting problems [25, 51]. In terms of the representation power of subnetworks, the Lottery Ticket Hypothe-

\*Corresponding author

sis [13] shows that there consistently exists lightweight sub-networks that can be trained from scratch with competitive learning speed as their larger counterparts, while maintaining comparable test accuracy. Based on this hypothesis, we assume that we can find the subnetworks for dynamic convolution in the training process and achieve a compact and efficient dynamic convolution network.

In this work, we re-examine the parameter efficiency for dynamic convolution. We first simply prune out half of the parameters in the  $k$  parallel kernels for pretrained dynamic convolution layers. Surprisingly, we find that this pruning operation has minimal effect on the numerical feature of dynamic parameters and a negligible impact on performance.

Based on this discovery, we further propose to integrate dynamic convolution with sparsity, namely sparse dynamic convolution, which enjoys natural complementarity. Technically, we present a new algorithm to train the dynamic convolution modes via iterative pruning. Specifically, we set a learnable threshold for each convolutional layer and prune the neurons whose magnitudes are below the threshold. We also propose a penalty term to explicitly regulate the  $L_0$ -norm of maintained parameters to guarantee the total parameters under an overall budget without additional hyperparameters. Considering that the computational kernel is a linear combination of static kernels, the masked kernels can then be integrated into a sparsely computing kernel with reduced FLOPs.

To validate the efficiency of sparse dynamic convolution, we execute our methods on both ImageNet-1K [8] and downstream tasks, and demonstrate the promotion that arises from both dynamic convolution and sparsity: Dynamic mechanism improves the representation power with negligible extra computational cost; Sparsity reduces the redundancy of dynamic convolution and promotes the performance during inference. In short, our main contributions are as follows:

- We propose the Sparse Dynamic Convolution (SD-Conv) to improve the parameter efficiency of dynamic convolution by marrying the dynamic convolution and sparsity to maintain the advantage of both worlds.
- We propose a novel  $L_0$ -norm based pruning method with an optimization policy to train sparse dynamic convolution networks efficiently.
- Our experiments on both upstream tasks and downstream tasks have shown the complementarity between sparsity and dynamic convolution.

## 2. Related Work

Both dynamic convolution and sparsity are often considered separately to promote neural networks. We delve into the combination of them and briefly review them as follows:

**Dynamic Networks** Dynamic networks adapt input-based parameters or activation functions to boost representation power. HyperNetworks [16] use a secondary network to generate parameters for the main network. SENet [23] applies channel-wise attention to channels. DRConv [6] transfers the increasing channel-wise filters to a spatial dimension with a learnable instructor. CondConv [52] and Dynamic Convolution [7] each proposed a new convolution operator to improve the representation capability with negligible extra FLOPs. Instead of using a single static convolution kernel per layer, they use the linear combination of a set of  $k$  parallel static kernels  $\{W_i, b_i\} (i = 1, 2, \dots, k)$ , where the linear scale is dynamically aggregated via a function of individual inputs. Dynamic convolution [7] utilizes an attention function to formulate the linear score:

$$\begin{aligned} \hat{W} &= \sum_{i=1}^k \pi_i \cdot W_i \\ \text{s.t. } \sum_{i=1}^k \pi_i &= 1, \quad 0 \leq \pi_i \leq 1, \end{aligned} \quad (1)$$

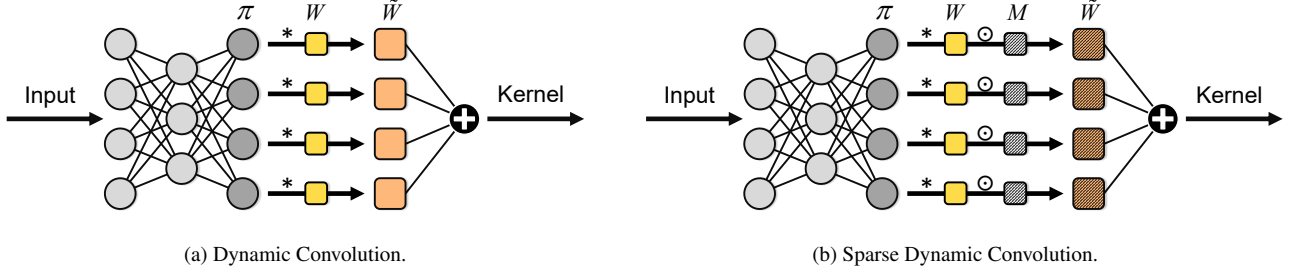
where  $\pi_k$  is the attention score of the  $k$ -th kernel. Dynamic convolution only introduces two negligible additional computations: 1) Computing the attention scores  $\pi_i (i = 1, 2, \dots, k)$ . 2) Aggregating parameters based on attention scores  $\sum_{i=1}^k \pi_i(x) * W_i$ . This linear combination significantly promotes the representation power of dynamic convolution and improves the performance in mainstream computer vision tasks.

However, towards the use of  $k$  parallel kernels in dynamic convolution, Li *et al.* [34] have proposed that it lacks compactness, and further utilized a matrix deposition method to improve this problem. Similarly, our work investigates the parameter efficiency of dynamic convolution and utilizes network pruning methods to improve it.

**Sparsity** Sparsity has been widely studied to compress deep neural networks in resource-constrained environments. It can be generally categorized into two groups: STRUCTURED and UNSTRUCTURED sparsity. Structured sparsity prunes blocks of sub-networks in a neural network, while unstructured fine-grained sparsity prunes multiple individual weights distributed across the whole neural network. Between the two sparsity types, unstructured sparsity usually achieves significantly higher compression ratios while maintaining relatively better performance [15, 17], which therefore leaves as our default sparsity type.

Unstructured sparsity usually detects unimportant parameters and utilizes a threshold to prune them. On the one hand, many previous works compute the threshold based on different importance-based criteria, including magnitude [13, 17, 56], Hessian-based heuristics [32, 33] and connection sensitivity [31, 37]. On the other hand, sparse training

Figure 1. Illustration of convolution kernel generation process for Dynamic Convolution (Left) and our proposed Sparse Dynamic Convolution architecture (Right).



with differential thresholds has also been widely explored. Kusupati *et al.* [30] and Manessi *et al.* [40] propose to learn layer-wise thresholds automatically using a soft thresholding operator or a close variant of it. As the learning-based thresholds contribute to the minimization of task-specific loss, the differential thresholds-based sparse method [30] contributes to high performance. Besides, sparsity learned during training with approximate  $L_0$ -norm regulation has also been used in several works [2, 35], because it controls the overall sparsity directly. To make the  $L_0$ -norm of thresholds differentiable, Louizos *et al.* [35] set a collection of non-negative stochastic gates to determine the weights to be pruned, while Azarian *et al.* [2] propose an approximate form of  $L_0$ -norm to estimate the gradient. Considering both performance and controllability, we adopt a threshold-based  $L_0$ -norm in our sparse method.

### 3. Methodology

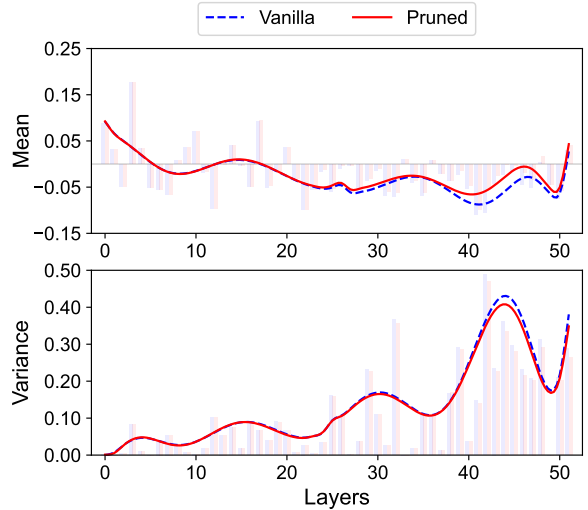
In this section, we first present our motivation for sparsity in dynamic convolution, then illustrate our efficient architecture, namely Sparse Dynamic Convolution (SD-CONV).

#### 3.1. Motivation

In conventional dynamic convolution, each convolutional layer prepares  $k$  parallel kernels to aggregate the dynamic kernel, leading to a nearly  $k$  times larger model and potential parameter redundancy. For example, the total parameters of dynamic ResNet-50 (DY-ResNet-50) are about 100.9M (with 4 kernels) compared to about 23.5M for static ResNet-50. For this phenomenon, we raise two questions: (1) *Is it necessary to pay the cost of enormous parameters and computations, e.g. 329% in DY-ResNet-50, to aggregate the dynamic kernels?* (2) *Is it necessary to deploy all of these parameters to maintain the slight performance improvement, e.g. 1.1% in DY-ResNet-50?*

To answer these questions, we turn to analyze the pre-trained DY-ResNet-50 [19] model from the view of network pruning. Specifically, we prune out 50% parameters of it

Figure 2. The scaled layer-wise mean and variance values of the aggregated kernel weights of ResNet-50. “Vanilla” and “Pruned” denote dynamic convolution networks before and after pruning, respectively. The mean and variance values keep nearly unchanged after pruning 50% parameters.



with the lowest magnitude on CIFAR-100 dataset<sup>1</sup> [29]. We measure the mean and variance values of the parameters in aggregated kernels as the proxy of the dynamic property: given different input samples, each dynamic convolution layer aggregates the computational kernel dynamically. By iterating over the entire validation dataset, we compute the layer-wise mean and variance of parameters in the aggregated kernel, which is shown in Figure 2. Clearly, The change curves of the vanilla and the pruned networks almost coincide, with only some small divergences in the upper layers. Therefore, network pruning has little impact on the numeric features of the dynamic property.

We also conduct a preliminary experiment to investigate the performance gap caused by network pruning on CIFAR-100 using ResNet [19] as backbones. We can see from

<sup>1</sup><https://github.com/weiaicunzai/pytorch-cifar100>

Table 1. Preliminary results for network pruning in dynamic convolution (DY-Conv). After pruning 50% parameters “w/ Pruned”, dynamic convolutions still maintain comparable performance and the advantage over static convolutions “Static”. \* indicates the dynamic models with the best performance, the fewest parameters, and the fewest FLOPs (“Static” models are excluded).

Depth	Method	Param.	FLOPs	Top-1 (%)
ResNet-10	Static	0.3M	29.9M	66.0
	DY-Conv	1.2M	34.8M	68.9
	w/ Pruned	*0.6M	*27.1M	68.1(-0.8)
ResNet-18	Static	0.7M	35.6M	67.6
	DY-Conv	2.8M	43.4M	72.4
	w/ Pruned	*1.4M	*31.9M	71.9(-0.5)
ResNet-50	Static	1.5M	122.3M	72.2
	DY-Conv	6.2M	143.4M	75.2
	w/ Pruned	*3.3M	*108.5M	74.6(-0.6)

results in Table 1 that pruned networks still maintain almost equally competitive performance: DY-ResNet families (with dynamic convolution layers) only encounter less than 1% of performance drop and still outperform static networks by more than 2% in accuracy. The above observations motivate us to explore effective and efficient sparse dynamic convolution structures.

### 3.2. Sparse Dynamic Convolution

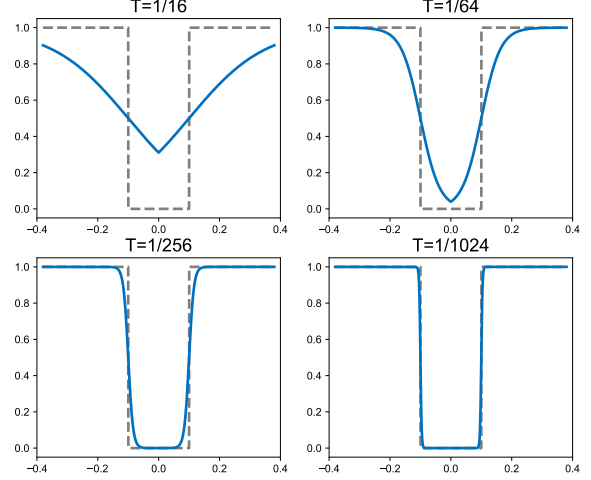
In this section, we propose Sparse Dynamic Convolution, which utilizes parallel sparse kernels to aggregate dynamic kernels. We use binary masks  $M$  to sparsify the kernels by pruning out unimportant parameters. Generally, a binary mask is a 0/1 matrix indexing the pruned weights in the parallel kernels. To make the binary mask trainable, we define a magnitude score  $S = \|W\|$  and a threshold  $\tau$ . The mask is then rounded to 1 if the score is greater than the threshold, and vice versa, given by

$$M_i = \begin{cases} 1, & \text{if } S_i \geq \tau \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

The major challenge for training binary masks is that Eq. (2) is non-differentiable, impeding the calculation of gradients and blocking the updating process. To solve this problem, Piggyback [39] utilizes the Straight-Through Estimator (STE) [3] (where the gradient is directly passed to its input  $\frac{\partial M}{\partial W} = 1$ ) to enable gradient estimation so that the gradient descent can update parameters.

According to Zhou *et al.* [55], the values of  $\tilde{M}$  are not restricted in binary values 0/1 strictly, which may cause an unstable training process and accuracy drop. Inspired by Yang *et al.* [53], we adopt the softmax function to approximate  $\tilde{M}$  into binary values 0/1 for better gradient calculation:

Figure 3. Effect of the hyperparameter  $T$  on the binary function Eq. (3 and 4). It is easily observed that this hyperparameter contributes to the sharpness. By decreasing  $T$ , we observe that output values gradually follow an approximate 0/1 distribution.



$$\psi_1 = \sigma(S - \tau), \quad (3)$$

$$\tilde{M} = \frac{\exp(\psi_1/T)}{\exp(\psi_1/T) + \exp(\psi_0/T)}, \quad (4)$$

where  $\sigma(\cdot)$  is the sigmoid function and  $\tau$  denotes the threshold.  $\psi_0 = 1 - \psi_1$  is the complement of  $\psi_1$ ,  $\tilde{M}$  is the generated mask following an approximate 0/1 distribution.  $T$  is the hyperparameter controlling the sharpness of the function. For example,  $T = \frac{1}{1024}$  encourages the output to be either 0 or 1, which is shown in Figure 3. Then we transform  $\tilde{M}$  into binary values  $M$  using STE to generate and update the binary masks:

$$M = \text{round}(\tilde{M}), \quad \nabla M = \nabla \tilde{M}. \quad (5)$$

By utilizing binary masks to  $k$  kernels, we transform the dynamic convolution into sparse dynamic convolution. In this layer, we first sparsify the  $k$  parallel static convolution kernels  $W_i$  ( $i = 1, 2, \dots, k$ ) and then combine them dynamically, given by

$$\tilde{W}_i = M_i \odot W_i, \quad \hat{W} = \sum_i \pi_i * \tilde{W}_i. \quad (6)$$

Here,  $\tilde{W}_i$  and  $\hat{W}$  denote the sparsified parameters of the static kernel and the aggregated sparse kernel, respectively.

### 3.3. Loss Function

The binary mask  $M$  is determined by the magnitude of parameters  $W$  and the threshold  $\tau$ . In general situations,  $\tau$  is a hyperparameter as the threshold that controls the global

sparsity. A naive way to set the threshold is to maintain the uniform sparsity of all layers. However, many experiments have indicated that setting multiple thresholds to control the layer-wise non-uniform sparsity performs much better [2, 30, 35]. Existing methods to acquire layer-wise sparsity are often dependent on hyperparameters and require iterative trials [13].

To address this problem, we propose a learning-based strategy to obtain layer-wise thresholds. Specifically, we first transform  $\tau$  into learnable parameters and utilize it to generate differential masks:

$$\frac{\partial M}{\partial \tau} = \frac{\partial M}{\partial \tilde{M}} \frac{\partial \tilde{M}}{\partial \psi_1} \frac{\partial \psi_1}{\partial \tau}. \quad (7)$$

The gradient backpropagated to  $M$  indicates two directions: contributing to the performance improvement and constraining the overall sparsity. To constrain the sparsity,  $L_0$ -norm regularization has been widely researched in model sparsity [2, 35], for it directly regulates the overall parameter budget. Therefore, given the overall sparse level  $s$ , the non-zero ratio of overall parameters is  $\bar{s} = 1 - s$ . We resort to  $L_0$ -norm penalty  $\mathcal{L}_s(\tau, s)$  to constrain the layer-wise non-uniform sparsity as followed:

$$\mathcal{L}_s(\tau, \bar{s}) = \text{ReLU}\left(\sum_l N_l \cdot (\|M^{(l)}\|_0 - \bar{s})\right), \quad (8)$$

where  $\mathcal{L}_s$  is the regulation loss that controls the global sparsity,  $N_l$  is the number of parameters in the  $l$ -th layer,  $\|M^{(l)}\|_0$  is the  $L_0$  norm of the mask  $M$  in the  $l$ -th layer. Note that we use the ReLU function to restrict the global sparsity under the setting value, this loss term only works when the network is denser than expected. Formally, we define our loss function  $\mathcal{L}$  as followed:

$$\mathcal{L} = \mathcal{L}_c(y, f(\mathbf{x}, W, \tau)) + \lambda_s \mathcal{L}_s(\tau, \bar{s}) + \lambda_r \|W\|_2, \quad (9)$$

where we represent our networks as  $f$  and the ground truth label as  $y$ .  $\mathcal{L}_c$  is the standard loss function, e.g., cross-entropy in image classification.  $\|W\|_2$  is the  $L_2$  weight regularization loss and  $\lambda_r$  is the weight decay rate.  $\lambda_s$  is a hyperparameter that determines the pruning speed.

### 3.4. Optimization Policy

We train the sparse dynamic convolution following an iterative pruning process [11, 13]. Notably, considering the time-consuming training process of dynamic networks, we restrain the total steps consistent with vanilla dynamic convolution and equally divide the total steps into  $n+1$  phases. Given the sparse level  $s$  and pruning iterations  $n$ , in the first  $n$  phases, we prune  $s^{\frac{1}{n}}$  percent of the parameters at the end of each phase and retrain the network in the next phase. The whole training policy is shown in Algorithm 1.

---

#### Algorithm 1: Sparse Dynamic Convolution

---

**Input:** Sparsity  $s$ , Total Steps  $T$ , Pruning Iterations  $n$ , Dynamic Convolution Network  $f$ .

**Output:** Sparse Dynamic Network  $f_s$ .

- 1: Initialize  $\bar{s} = 1 - s$ ,  $\bar{s}_0 = 0$ ,  $\Delta t = \frac{T}{n+1}$ .
  - 2: **for**  $t = 1$  to  $T$  **do**
  - 3:   Compute loss  $\mathcal{L} = \mathcal{L}_c + \lambda_s \mathcal{L}_s(\tau, \bar{s}_t) + \lambda_r \|W\|_2$ .
  - 4:   Update parameters  $W_{t+1} \leftarrow W_t - \eta_W \frac{\partial \mathcal{L}}{\partial W}$ .
  - 5:   Update thresholds  $\tau_{t+1} \leftarrow \tau_t - \eta_\tau \frac{\partial \mathcal{L}}{\partial \tau}$ .
  - 6:   **if**  $t \bmod \Delta t = 0$  **then**
  - 7:     Update variable  $\bar{s}_{t+1} = \bar{s}^{\frac{t}{\Delta t * n}}$ .
  - 8:   **else**
  - 9:     Pass variable  $\bar{s}_{t+1} = \bar{s}_t$ .
  - 10:   **end if**
  - 11: **end for**
- 

## 4. Experiment

In this section, we provide comprehensive experiments on both large-scale image recognition datasets and downstream tasks with different CNN architectures to validate the effectiveness of SD-Conv. Specifically, we compare the performance of sparse dynamic convolution with other convolution architectures, and further analyze the design of sparse dynamic convolution from the perspective of sparsity and reduced FLOPs.

### 4.1. Image Classification on ImageNet

Our main experiments are implemented on the ImageNet dataset [8], which is one of the most challenging image classification datasets with 1,000 classes, including 1,281,167 images for training and 50,000 images for validation.

**CNN Backbones.** We use ResNet [19] and MobileNetV2 [48] families for experiments, covering both light-weight CNN architectures and larger ones. Specifically, we choose ResNet-10, ResNet-18, ResNet-50 and MobileNetV2 ( $1.0\times$ ,  $0.75\times$ ,  $0.5\times$ ) as the backbones.

**Experimental Setup.** We validate the effectiveness of our method by replacing dynamic convolution for all convolution layers except the first layer. Each layer has  $k = 4$  experts with the reduce ratio as 16 for the attention block in dynamic convolution [7]. We use an SGD optimizer [47] with 0.9 momentum, following cosine learning rate scheduling and warmup strategy. The learning rate rises to the max learning rate linearly in the first 10 epochs and is scheduled to arrive at zero within a single cosine cycle. When generating binary masks, we set constant  $T = \frac{1}{1024}$  to ensure  $\bar{M}$  follows approximately 0/1 binary values. The scale factor  $\lambda_s$  of sparse penalty  $\mathcal{L}_s(\tau, s)$  is fixed as 0.01. We follow Zhou *et al.* [7]’s temperature annealing strategy to avoid the unstable output values of the softmax function in the first epoch. We train the ResNet models for



Table 2. Comparison for MobileNetV2 and ResNet between Sparse Dynamic Convolution and baselines, including static convolution, Condconv [52] and DY-Conv [7]. \* indicates the dynamic model with the fewest parameters or the fewest FLOPs (static models are not included). The best performance is **bold**.

Width	Method	Param.	FLOPs	Top-1 (%)	Depth	Method	Param.	FLOPs	Top-1 (%)
$\times 1.0$	Static	3.5M	300.0M	72.0	ResNet-10	Static	5.2M	0.89G	63.4
	CondConv	27.5M	329.0M	74.6		CondConv	36.7M	0.92G	66.8
	DY-Conv	11.8M	312.9M	75.2		DY-Conv	18.6M	0.91G	67.5
	SD-Conv	*7.7M	*261.9M	<b>75.3</b>		SD-Conv	*10.4M	*0.73G	<b>67.9</b>
$\times 0.75$	Static	2.6M	209.1M	69.3	ResNet-18	Static	11.1M	1.81G	70.4
	CondConv	17.5M	233.9M	71.8		CondConv	81.4M	1.89G	72.0
	DY-Conv	7.6M	220.1M	72.8		DY-Conv	42.7M	1.85G	72.7
	SD-Conv	*5.0M	*171.8M	<b>73.2</b>		SD-Conv	*23.2M	*1.51G	<b>73.3</b>
$\times 0.5$	Static	2.0M	97.0M	65.4	ResNet-50	Static	23.5M	3.8G	76.2
	CondConv	15.5M	113.0M	68.4		CondConv	129.9M	4.0G	76.8
	DY-Conv	4.4M	101.4M	69.9		DY-Conv	100.9M	4.0G	77.3
	SD-Conv	*3.1M	*81.5M	<b>70.3</b>		SD-Conv	*54.0M	*3.4G	<b>77.4</b>

(a) **MobileNetV2**

(b) **ResNet**

100 epochs, and the max learning rate is 0.1. For the MobileNetV2 models, we train them for 300 epochs, and the max learning rate is 0.05. The weight decay is  $4e-5$  for all models.

**Main Results.** Table 2a and 2b show the comparison between SD-Conv and other convolution architectures in two CNN architectures (ResNet and MobileNetV2). Our baselines include the static convolution, CondConv [52] and DY-Conv [7]. We set  $s = 50\%$  to make the overall sparsity over 50%. As shown, sparse dynamic convolution achieves significant performance improvement with a much smaller model size compared to vanilla dynamic convolution. For ResNet-18, sparse dynamic convolution has only 54.3% of the parameters of vanilla dynamic convolution. For MobileNetV2-1.0, our method only requires 53.5% of the parameters of dynamic convolution to achieve the same level of accuracy. The most prominent advantage of sparse dynamic convolution is its low computational cost. Owing to the sparse computational kernel  $\tilde{W}$ , our method requires much fewer FLOPs in the convolution operation that acts as the dominant part of the overall FLOPs. The computational cost of our method is even less than that of static convolution, while all the other dynamic networks introduce extra computational costs. For example, sparse dynamic convolution only has 87.3% of FLOPs of static convolution in MobileNetV2-1.0.

**Robustness.** Traditional network structures are robust to the images perturbed with small Gaussian noise, while networks pruned with random masks can even have higher robustness than normal ones [38]. To check whether SD-Conv also enjoys such property or even has better robustness in this scenario, we also consider the model’s endurance of noise attack. We conduct an robustness evalu-

Table 3. The robustness evaluation based on random noise attack. Setting different standard variance  $\sigma$ , we evaluate the performance of different models. The best performance is **bold**.

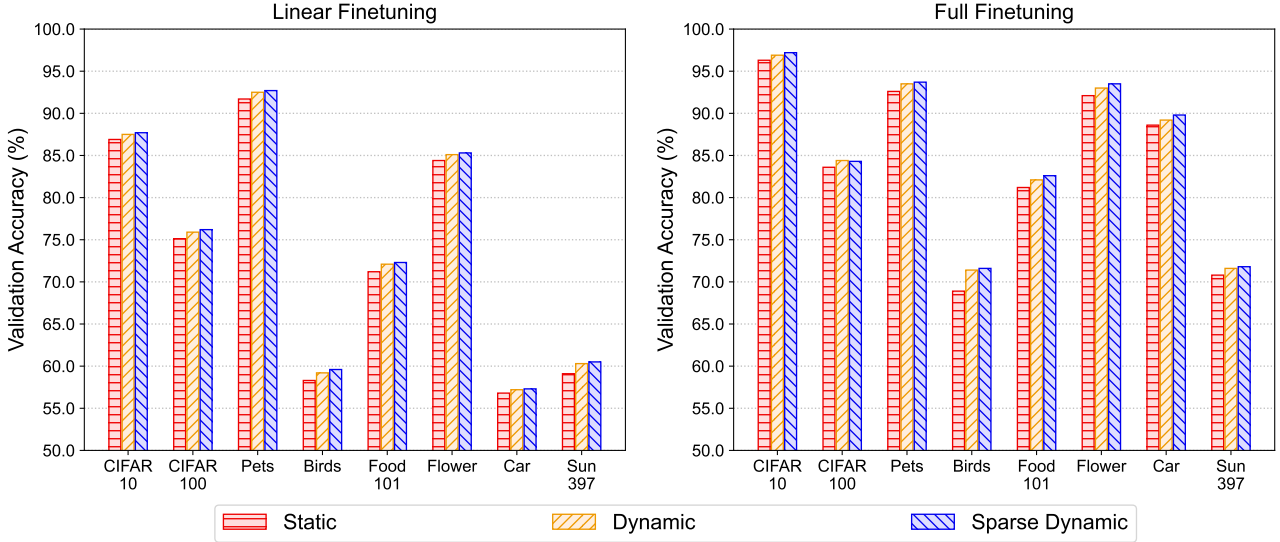
Model	Option	0.05	0.10	0.15	0.20
ResNet-50	Static	68.2	65.4	58.4	54.2
	DY-Conv	68.7	66.1	59.2	55.4
	SD-Conv	<b>69.1</b>	<b>66.5</b>	<b>59.5</b>	<b>55.9</b>
ResNet-18	Static	60.7	53.9	49.8	45.1
	DY-Conv	61.1	54.8	50.4	46.3
	SD-Conv	<b>61.3</b>	<b>55.2</b>	<b>50.5</b>	<b>46.7</b>

ation on ImageNet for ResNet-50 [19]. Inspired by Luo *et al.*’s work [38], we feed input images with Gaussian noises  $z \sim N(0, \sigma^2)$  to networks. Table 3 shows the robustness evaluation on random noise attack, we set the sparse ratio  $s$  as 20% and 80% separately for our model. Disturbed by the same intensity of noise, we can see that our networks have the highest accuracy and gain up to 0.5% improvement compared to dynamic architectures.

## 4.2. Transferring to Downstream Tasks

Network architectures measured against ImageNet [8] have fueled much progress in computer vision researches across a broad array of problems, including transferring to new datasets [9, 49], object detection [24], image segmentation [18] and perceptual metrics of images [27]. Many previous works have proved that better network architectures learn better features to be transferred across vision-based tasks [22, 49]. Therefore, we further evaluate the effectiveness of our network on downstream vision tasks, including CIFAR-10 [29], CIFAR-100 [29], Oxford-IIIT Pets [43],

Figure 4. Transferability comparison between Static, Dynamic, and our proposed SD-Conv from pretrained ResNet to different downstream tasks. We report two finetuning approaches: linear finetuning and full finetuning.



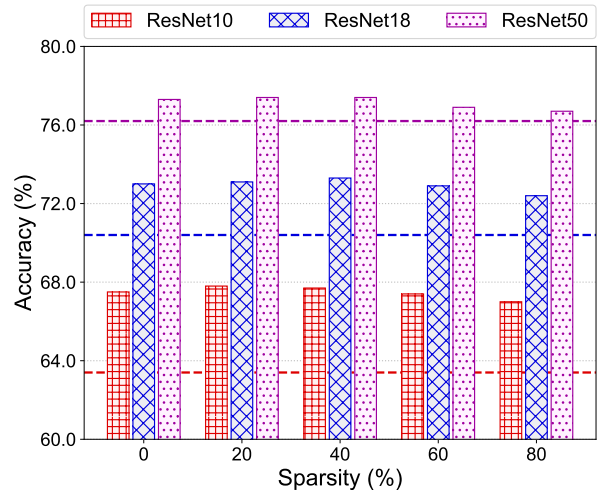
Birdsnap [4], Food-101 [5], Oxford 102 Flowers [42], Stanford Cars [28], SUN397 [50]. These tasks span several domains, difficulties, and data sizes.

We transfer all parameters of the upstream model except the last (fully connected) layer, which is adjusted to the number of classes in the downstream task, using Kaiming uniform initialization [20]. We finetune the pretrained ImageNet model following two strategies, linear finetuning and full finetuning. For linear finetuning, we only train the linear classifier “on the top” of a fixed representation on downstream tasks, while we re-initialize the final layer and train the whole model for full finetuning. For both strategies, we take top-1 classification accuracy as the metric to compare different structures, which is shown in Figure 4. The results clearly show that the sparse dynamic convolution achieves consistent improvement compared to dynamic convolution on downstream tasks, suggesting that pruning redundant information in the weights is beneficial to dynamic convolution architectures in transfer settings.

### 4.3. Further Analysis

**Further Increased Sparsity Can Still Maintain Superior Performance.** To further explore the impact of sparsity, we conduct an ablation study by investigating a series of sparse ratios (from 20% to 80%). Figure 5 shows the result of the ablation study on ImageNet classification [8] experiments for ResNet [19] in different depth, where we directly report the classification accuracy. For ResNet models with different depths, we can observe a consistent phenomenon that SD-Conv performs stably under different degrees of network pruning. At low sparse ratios (e.g.  $s \leq 40\%$ ), pruning out some unimportant parameters can lead to higher performance. When further increasing the sparse ratios, e.g.

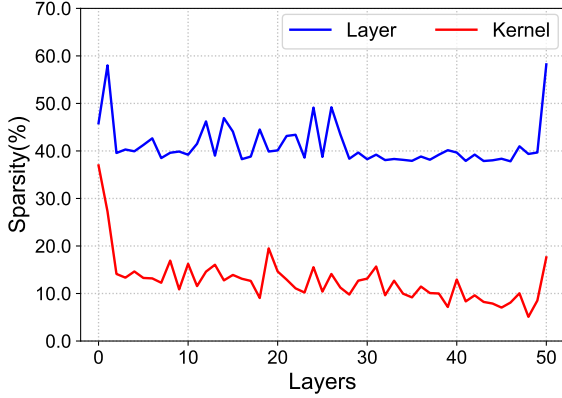
Figure 5. The ablation study on sparsity for ResNet. Dotted line represents the performance of static convolution.



60% and 80%, sparse dynamic convolution networks still maintain a significant performance advantage over static convolution networks. Considering the competitive performance, network pruning is an efficient way to simplify and promote dynamic convolution.

**Reduced FLOPs Come from the Sparse Aggregated Kernel.** As shown in Table 2a and Table 2b, the FLOPs of sparse dynamic convolution networks are even lower than static convolution. According to our observation, the reduced FLOPs come from the sparsity in propagated kernels  $\tilde{W}$ . Even so, we have to mention that the sparsity of  $\tilde{W}$  is not definitely dependent on the sparsity of parameter  $\tilde{W}_j$  ( $j = 1, 2, \dots, k$ ) but lies in the overlap between them: the

Figure 6. The curves of kernel and layer sparsity for ResNet-18.



$i$ -th element of  $\tilde{W}$  is zero only when all static kernels  $\tilde{W}_j$  ( $j = 1, 2, \dots, k$ ) have zero elements in the  $i$ -th position. Therefore, other than layer sparsity towards parameters (the proportion of zero-elements in  $\tilde{W}_j$ ), we also resort to the kernel sparsity towards propagated kernels (the proportion of zero-elements in  $\tilde{w}$ ). To investigate the distribution of pruned parameters in propagated kernels, we visualize the pruned ratio of propagated kernels  $\tilde{W}$  and  $k$  kernels  $W_k$  in Figure 6. We show that the kernel sparsity follows a similar trend to layer sparsity but maintains relatively smaller values. Even so, each propagated kernel still maintains a certain degree of sparsity, and the pruned weights contribute to the reduced FLOPs compared to dense convolution kernels.

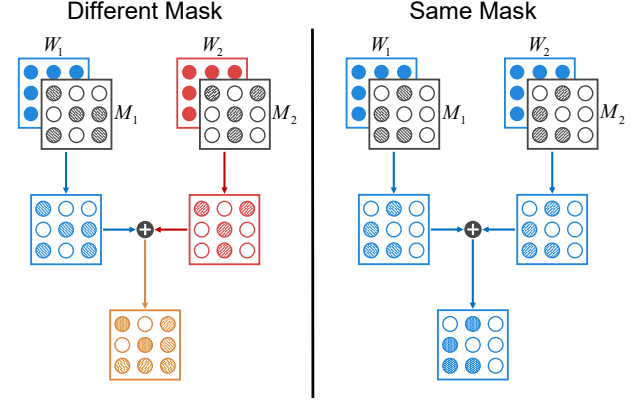
## 5. Discussion of Masking Strategy

Table 4. Comparison between two different masking strategies. We use “Diff” to denote the different masking strategy and “Same” to denote the same masking strategy. \* indicates the dynamic model with the fewest parameters or the fewest FLOPs.

Network	Method	Param	Flops	Acc(%)
ResNet-50	Static	2.35M	3.8G	76.2
	DY-Conv	100.9M	4.0G	77.3
	Diff	*63.3M	3.5G	<b>77.4</b>
	Same	*63.3M	*2.5G	76.6
MobilenetV2-1.0	Static	3.5M	300.0M	72.0
	DY-Conv	11.1M	312.9M	75.2
	Diff	*5.3M	271.9M	<b>75.3</b>
	Same	*5.3M	*192.3M	74.6

As aforementioned in Section 4.3, the kernel sparsity lies in the overlap region of  $k$  masks  $M_i$  ( $i = 1, 2, \dots, k$ ) and is usually lower than the parameter sparse ratio. Only when  $M_1 = M_2 = \dots = M_k$ , the kernel sparsity can be the highest and the FLOPs can be minimized. Therefore, we evaluate a strategy that directly applies the same mask to

Figure 7. Comparison between two masking strategies. The left one is the default setting in SD-Conv, which takes  $k$  different masks for each counterpart kernel. On the right, each kernel shares the same mask.



the static kernels and then compare it with our proposed method, as shown in Figure 7. We can see from the numeric results from Table 4 that utilizing the same mask to  $k$  kernels can cause a performance drop though it significantly reduces the FLOPs. In contrast, our learning-oriented thresholds lead to different masks among static kernels and obtain significantly better results. We believe that the sparser aggregated kernels cause the performance drop and there exists a trade-off between optimal FLOPs and performance in sparse dynamic convolution.

## 6. Conclusion

In this work, we systematically re-examine the parameter efficiency property of dynamic convolution networks through the lens of network pruning. Based on our findings, we propose a plug-in strategy, i.e. Sparse Dynamic Convolution, for existing dynamic convolution methods. Our method improves the performance of dynamic convolution both in upstream ImageNet classification and a variety of downstream tasks, with fewer parameters and FLOPs. Our study empirically indicates the effectiveness of sparsity in dynamic convolution and informs the potential to further promote sparse dynamic convolution in view of the trade-off between performance and FLOPs.

In future work, we would like to investigate the parameter efficiencies of other neural network models, especially for scenarios where high efficiency is required, e.g., Adapter [21], Prompt [54] and the distilled student [46].

## Acknowledgements

We are grateful to the anonymous WACV reviewers and the area chair for their insightful comments and suggestions.



## References

- [1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers, 2020. **1**
- [2] Kambiz Azarian, Yash Bhalgat, Jinwon Lee, and Tijmen Blankevoort. Learned threshold pruning, 2021. **3, 5**
- [3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. **4**
- [4] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2014. **7**
- [5] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101—mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014. **7**
- [6] Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun. Dynamic region-aware convolution, 2021. **2**
- [7] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels, 2020. **1, 2, 5, 6**
- [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. **1, 2, 5, 6, 7**
- [9] Nanqing Dong and Eric P Xing. Domain adaption in one-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 573–588. Springer, 2018. **6**
- [10] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020. **1**
- [11] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021. **5**
- [12] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Remi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression, 2021. **1**
- [13] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. **1, 2, 5**
- [14] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019. **1**
- [15] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns, 2016. **2**
- [16] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016. **2**
- [17] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. **1, 2**
- [18] 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. **6**
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. **3, 5, 6, 7**
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. **7**
- [21] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of EMNLP*, 2022. **8**
- [22] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. **6**
- [23] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019. **2**
- [24] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017. **6**
- [25] Shaoyi Huang, Dongkuan Xu, Ian Yen, Yijue Wang, Sung-En Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Sanguthevar Rajasekaran, Hang Liu, and Caiwen Ding. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 190–200, Dublin, Ireland, May 2022. Association for Computational Linguistics. **1**
- [26] Eugenia Iofinova, Alexandra Peste, Mark Kurtz, and Dan Alistarh. How well do sparse imagenet models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12266–12276, 2022. **1**
- [27] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. **6**
- [28] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. **7**
- [29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. **3, 6**
- [30] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity, 2020. **1, 3, 5**
- [31] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019. **2**
- [32] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017. **1, 2**
- [33] Li Li, Zhu Li, Yue Li, Birendra Kathariya, and Shuvra Bhattacharyya. Incremental deep neural network pruning based

- on hessian approximation. In *2019 Data Compression Conference (DCC)*, pages 590–590. IEEE, 2019. 1, 2
- [34] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Ye Yu, Lu Yuan, Zicheng Liu, Mei Chen, and Nuno Vasconcelos. Revisiting dynamic convolution via matrix decomposition, 2021. 1, 2
- [35] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through  $l_0$  regularization, 2018. 3, 5
- [36] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017. 1
- [37] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data, 2020. 2
- [38] Tiange Luo, Tianle Cai, Mengxiao Zhang, Siyu Chen, and Liwei Wang. Random mask: Towards robust convolutional neural networks. *arXiv preprint*, 2020. 6
- [39] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights, 2018. 4
- [40] Franco Manessi, Alessandro Rozza, Simone Bianco, Paolo Napoletano, and Raimondo Schettini. Automated pruning for deep neural network compression. In *2018 24th International conference on pattern recognition (ICPR)*, pages 657–664. IEEE, 2018. 3
- [41] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020. 1
- [42] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 7
- [43] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 6
- [44] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5113–5122. PMLR, 09–15 Jun 2019. 1
- [45] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR, 2017. 1
- [46] Jun Rao, Xv Meng, Liang Ding, Shuhan Qi, and Dacheng Tao. Parameter-efficient and student-friendly knowledge distillation. *ArXiv*, abs/2205.15308, 2022. 8
- [47] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 5
- [48] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. 5
- [49] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 6
- [50] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 7
- [51] Qi Xu, Ming Zhang, Zonghua Gu, and Gang Pan. Overfitting remedy by sparsifying regularization on fully-connected layers of cnns. *Neurocomputing*, 328:69–74, 2019. Chinese Conference on Computer Vision 2017. 1
- [52] Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference, 2020. 2, 6
- [53] Li Yang, Zhezhi He, Junshan Zhang, and Deliang Fan. Ksm: Fast multiple task adaption via kernel-wise soft mask learning, 2020. 4
- [54] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation. *ArXiv*, abs/2208.10160, 2022. 8
- [55] Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch, 2021. 4
- [56] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017. 1, 2