Pick-or-Mix: Dynamic Channel Sampling for ConvNets

Ashish Kumar[†], Daneul Kim[‡], Jaesik Park[‡], Laxmidhar Behera[†]
[†] Indian Institute of Technology Kanpur, India
[‡] Seoul National University, Republic of Korea
{ashishkumar822@gmail.com, carpedkm@snu.ac.kr, jaesik.park@snu.ac.kr, lbehera@iitk.ac.in}

Abstract

Channel pruning approaches for convolutional neural networks (ConvNets) deactivate the channels, statically or dynamically, and require special implementation. In addition, channel squeezing in representative ConvNets is carried out via 1×1 convolutions which dominates a large portion of computations and network parameters. Given these challenges, we propose an effective multi-purpose module for dynamic channel sampling, namely Pick-or-Mix (PiX), which does not require special implementation. PiX divides a set of channels into subsets and then picks from them, where the picking decision is dynamically made per each pixel based on the input activations. We plug PiX into prominent ConvNet architectures and verify its multi-purpose utilities. After replacing 1×1 channel squeezing layers in ResNet with PiX, the network becomes 25% faster without losing accuracy. We show that PiX allows ConvNets to learn better data representation than widely adopted approaches to enhance networks' representation power (e.g., SE, CBAM, AFF, SKNet, and DWP). We also show that PiX achieves state-of-the-art performance on network downscaling and dynamic channel pruning applications.

Code: `https://github.com/ashishkumar822/PiX`

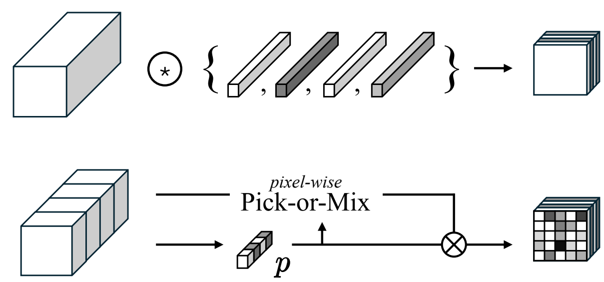

Figure 1. Conceptual overview of PiX in the context of channel reduction for ConvNets. Top: Traditional dense 1×1 convolution. Although not all channels are important, dense convolutions process all the channels equally. Bottom: PiX avoids dense convolution and samples the channels dynamically from the input by producing sampling probabilities with far fewer FLOPs. PiX is multipurpose without requiring specialized implementations.

1. Introduction

Convolutional neural networks (ConvNets) [11, 33] have been successfully applied to many machine vision tasks [18, 30]. With the introduction of larger models, a general trend is to make them faster via channel pruning. Prior works in channel pruning [8, 10, 12, 16] focus on making network lighter to accelerate the inference speed. However, some approaches require specialized convolution implementations and pre-trained models [8], or they are constrained by the baseline accuracy [10]. Moreover, whether static or dynamic, these channel pruning methods remove or deactivate the network channels, thus hindering the network from handling difficult inputs [8, 35].

It is a fundamental property of ConvNets that for a given spatial location or pixel in the ConvNets' feature map, any one channel may have stronger activation, thus of considerable importance, while for another pixel, the same channel might be less important. Therefore, it is crucial to allow the network to *prioritize channels differently per each pixel* instead of dropping a whole channel applied by pruning approaches. This inspires us to pick *neuron-specific output from the channels* instead of shutting down an entire channel.

In addition, we observe that standard ConvNet designs still have room for improvement, i.e., 1×1 convolution layers (or called channel squeezing layers) dominate in both number and computations without contributing to the receptive field due to their pixel-wise operation nature. For instance, ResNet-50 [11] consists of 16 such layers out of 50, accounting for $\sim 25\%$ (1.05B/4.12B) of overall FLOPs.

In this context, we introduce a novel module, namely Pick-or-Mix (PiX) that addresses the computational dominance of channel-squeezing layers by *dynamically sampling channels*, PiX transforms a feature map $X \in \mathbb{R}^{C \times H \times W}$ into another one $Y \in \mathbb{R}^{\lceil C/\zeta \rceil \times H \times W}$ (Figure 1). Essentially, our method picks or mixes $\lceil C/\zeta \rceil$ channels from the input C channels with a sampling factor ζ. It divides a set of channels into subsets and then outputs one channel from each subset via our Pick-or-Mix strategy. PiX samples the channel based on the *pixel-level runtime decisions* made by the preceding layers; thus, decisions of PiX are dynamic and input-dependent. In addition, Pick-or-Mix does not involve extensive pixel-wise convolution, making the network more efficient. The simple design allows us to plug PiX into representative ConvNets. We plug PiX into representative ConvNets for the purpose of faster channel squeezing, network downscaling, and dynamic channel pruning.

Our experiments show that PiX can reduce the computational cost of the vanilla channel squeezing layer (*i.e.*, 1×1 convolution layer) while maintaining or achieving even better performance, e.g., ResNet becomes $\sim 25\%$ faster without bells and whistles (Sec 3.5.1, Table 1). PiX can customize ConvNets in a controlled manner while being faster and more accurate than the baseline counterpart with similar parameters (Sec. 3.5.2, Table 3), e.g., PiX outperforms recent RepVGG [5] without a complicated training phase while having simple network design. We also observe similar accuracy but at reduced parameters (Table 7). PiX performs better by $\sim 3\%$ relative to various recent dynamic channel pruning approaches [1, 8, 28, 35] on ResNet18 with $\sim 2\times$ FLOPs saving. (Sec. 3.5.3, Table 6).

We compare the accuracy and FLOPs of PiX with other state-of-the-art approaches. We also conduct transfer learning on PiX-enhanced network on CIFAR-10, CIFAR-100 for classification, and CityScapes for semantic segmentation. We observe better performance relative to the baselines.

2. Related Work

Convolutional Neural Networks. The earlier ConvNets [11, 33] are accuracy-oriented but still dominant in the industry [5, 20], thanks to their high representation power, architectural simplicity, and customizability. EfficientNet [34] emerged with network architecture search, but due to its nature of AutoML, it is deep and branched compared to traditional ConvNets [11, 33]. Even after half a decade, ResNet continues to improve [3, 23], indicating its architectural significance, while VGG-like architecture continues as it is design-friendly with low-powered computing devices due to its shallow, easily scalable, and low latency design [21].

This is also visible from ResNet design space exploration [29] that provides a competitive alternative to the advanced ConvNets [34] while being simpler. SENet [15], CBAM [36], and ResNest [37], Attentional Feature Fusion [3] further depict the importance of older architectures by developing novel units to improve the accuracy of ResNet by adding parameters and marginal computational overhead. More recently, RepVGG [5] improves the inference of years old VGG [33] model. In this paper, we tackle the overhead of 1×1 layers in standard ConvNets and expand its application to state-of-the-art transformers.

Accelerated Inference. ConvNet acceleration begins with static pruning [22] or network compression [13]. These methods [13, 22] are model agnostic, but they require the additional overhead of pre-training and fine-tuning, thus increasing the training time [8].

Furthermore, by using more efficient convolutions such as depthwise separable convolution [32], MobileNets [14, 31, 38] address this issue at the network architecture level. In contrast, PiX, without any significant architectural modifications, enables faster inference by providing an alternative to channel squeezing 1×1 convolutions.

3. Pick-or-Mix (PiX)

Modern ConvNets [5, 11, 37] are essentially a stack of convolution layers, but the design of channel squeezing 1×1 convolution still has room for improvement. The main challenge is exploiting the cross-channel information appropriately and developing a suitable mixing strategy to ensure accurate model learning.

In this section, we introduce Pick-or-Mix (PiX) in detail.

Overview Consider a tensor $X = \{X^{[1]}, X^{[2]}, ..., X^{[C]}\}$, where $X^{[i]} \in \mathbb{R}^{H \times W}$ denotes i^{th} channel of X. We aim to produce $Y = \{Y^{[1]}, Y^{[2]}, ..., Y^{[\lceil C/\zeta \rceil]}\}$, such that $\mathrm{O}(\mathcal{F}_{pix}) \ll \mathrm{O}(\mathcal{F}_s)$, where $\mathcal{F}_{pix}$ is the PiX enhanced network and $\mathcal{F}_s$ is the original network. Here, $\zeta \in \mathbb{R}$ is the channel sampling factor which controls the dimensionality of the output Y. The proposed dynamic channel sampling approach (PiX) progressively infers intermediate 1D descriptors $z \in \mathbb{R}^C$, $p \in \mathbb{R}^{\lceil C/\zeta \rceil}$ from input feature map $X \in \mathbb{R}^{C \times H \times W}$ for channel sampling by using learnable parameter $\phi = \{\theta, \beta\}$. It then applies per-pixel dynamic channel sampling operator π for fusing a subset of channels and produces an output feature map $Y \in \mathbb{R}^{\lceil C/\zeta \rceil \times H \times W}$ of reduced dimensionality that is controllable by the sampling factor $\zeta \in \mathbb{R}_{\geq 1}$.

The PiX module is illustrated in Figure 2 and can be sectioned into three stages: (1) global context aggregation, which provides a channel-wise global spatial context in the form of z (Sec. 3.1) (2) cross-channel information blending that transforms z into p, referred to as PiX sampling probability (Sec. 3.2), and (3) channel sampling stage that utilizes p and X to produce Y. (Sec. 3.3)

3.1. Global Context Aggregation

We define a transformation of global context aggregation as $gca : \mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^C$ which gathers global context from the input X for each channel:

$$gca(X) = \frac{1}{H \times W}\Big[\mathrm{cc}(X^{[0]}), \mathrm{cc}(X^{[1]}), .., \mathrm{cc}(X^{[C-1]})\Big] \quad (1)$$

where, $\mathrm{cc} : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}$ reduces i^{th} channel $X^{[i]}$ of X to a scalar. We use l_1-norm for cc due to its computational efficiency and vectorized parallelization onto GPUs. l_1-norm

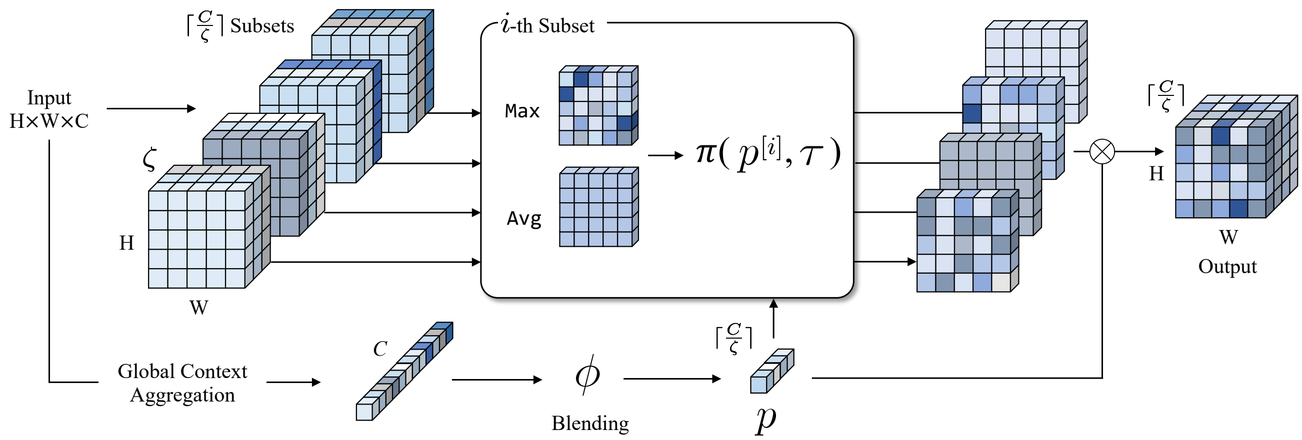

Figure 2. The proposed PiX module with its Pick-or-Mix dynamic channel sampling strategy. Each subset of input channels is picked (via max operator) or mixed (via average operator) to constitute the squeezed channels of the output. Interestingly, PiX can fuse channels differently for each pixel (please refer to Sec. 3.3).

of a channel is also known as global pooling, which is commonly employed [11, 15] to aggregate global spatial information.

3.2. Sampling Probability

Now the output of the previous step $z = gca(X)$ (Eq. 1) is passed through *sampling probability predictor* ϕ, serving two purposes. First, since each element of z consists of spatial information of only a single channel of X, the descriptor z lacks cross-channel information. ϕ mitigates this issue by blending the cross-channel information in the elements of z. Second, the fusion factor ζ, i.e., C to $\lceil C/\zeta \rceil$, reduces the input number of channels. We define $\phi(z) = z\theta + \beta$, where, $\theta \in \mathbb{R}^{\lceil C/\zeta \rceil \times C}$ and $\beta \in \mathbb{R}^{\lceil C/\zeta \rceil}$ are the weights and the biases, initialized with *Xavier* [9] and zero respectively.

After $\phi(z)$, we obtain *channel subset-wise sampling probability* $p \in \mathbb{R}_{\geq 0}^{\lceil C/\zeta \rceil}$ with sigmoid function.

3.3. Dynamic Channel Sampling

We introduce our computationally efficient dynamic channel sampling approach conditioned on p (Sec. 3.2). We can express the dynamic channel sampling with a functor $\mathcal{F}$: $\mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^{\lceil C/\zeta \rceil \times H \times W}$ such that $Y = \mathcal{F}(X; p)$.

Channel Space Partition. We partition X into $\lceil C/\zeta \rceil$ subsets. Each subset ($\Gamma^{[i]}$, where $i \in \{0, \cdots, \zeta - 1\}$) receives a maximum of ζ channels with the last one lesser than that in case C/ζ is non-integer.

Pixel-wise Channel Fusion. We devise a channel fusion strategy, namely *Pick-or-Mix* for each partitioned channel subset $\Gamma^{[i]}$. Specifically, for an arbitrary channel subset $\Gamma^{[i]}$, we then apply the channel fusion strategy to obtain a single channel feature map that constitutes one of the output channels. v is fused via the following equations:

$$\pi(\Gamma^{[i]}) = \begin{cases} p^{[i]} \times \texttt{Max}(\Gamma^{[i]}), & p^{[i]} \leq \tau \\ p^{[i]} \times \texttt{Avg}(\Gamma^{[i]}), & p^{[i]} > \tau, \end{cases} \tag{2}$$

where π is *Pick* (selecting the maximum) or *Mix* (averaging responses) channel function function, and $p^{[i]}$ is the pre-calculated sampling probability for a i-th subset (Sec. 3.2). τ is hyperparameter, set to 0.5 based on our ablations. In Eq. 2, the selection of a fusion operator is performed dynamically via the sampling probability p produced via the input, thus making PiX input adaptive.

To generalize this idea over the whole input feature map X, the functor $\mathcal{F}$ for this strategy can be given as:

$$\mathcal{F}(X; p) = \Big[\pi(\Gamma^{[0]}),\ \pi(\Gamma^{[1]}),\ \ldots,\ \pi(\Gamma^{[\lceil C/\zeta \rceil - 1]})\Big], \tag{3}$$

as depicted in Figure 2.

It is important to note that *channel sampling applies differently for each spatial location in PiX*. For example, when $\zeta > 1$ and $p^{[i]} \leq \tau$, with the help of `Max`, the selected channel index in a subset varies for each spatial location (or simply pixel) depending on channel values of that pixel. Moreover, each $\Gamma^{[i]}$ subset applies a different operator, i.e., some subgroup applies $\texttt{Max}(\cdot)$, and the other applies $\texttt{Avg}(\cdot)$. This subset-wise operation selection introduces $2^{\lceil C/\zeta \rceil}$ combinations, giving numerous ways to fuse the input channels.

Since fusion is done *on a pixel basis*, one pixel may prioritize any channel over another, demonstrating the capability of PiX. This degree of freedom to *fuse channels dynamically in a spatially varying manner* introduces a high level of non-linearity into the network, which helps to achieve PiX a competitive accuracy on various tasks with a simplified network structure. When $\zeta = 1$, since $\texttt{Max}(v) = \texttt{Avg}(v)$, PiX will act as global channel-wise attention as in SENet [15].

From the perspective of computation cost, note that π just refers to pre-computed $p^{[i]}$ for selecting lightweight operation (`Max` or `Avg`), and π does not involve expensive pixel-wise 1×1 convolution. Therefore, PiX can effectively save computation costs.

Our motivation to selectively utilize `Max` and `Avg` lies in the fundamentals of ConvNets [19] where max and avg. pooling are essentially summarization operations. The dynamic decision based on $p^{[i]}$ enables the ConvNets to learn rich representations and allows sub-sampling of the features. We also support our motivation empirically by employing the `Min` operator instead of `Max` or `Avg`. We observe a performance degradation by roughly 2% (see the supplement).

3.4. Computational Complexity

In PiX, the computation reduction primarily occurs due to collapsing the input tensor $X \in \mathbb{R}^{C\times H\times W}$ into $z \in \mathbb{R}^{C}$. In a naive channel squeezing operation, a 1×1 convolution is applied densely over $X \in \mathbb{R}^{C\times H\times W}$, having $C \times H \times W$ FLOPs. In contrast, in PiX, X is first collapsed into $z \in \mathbb{R}^{C}$, and then the sampling probability predictor is applied over z, *resulting in only $C \times (C/\zeta)$ FLOPs*. This is how PiX saves computations drastically.

Note that the *only learnable parameter in PiX is θ and β* as described in Sec. 3.2.

3.5. PiX Embodiment as a Multi-Purpose Module

The ability of PiX to perform channel sampling naturally translates to the underlying operations of different tasks, such as channel squeezing (Sec. 3.5.1), network scaling (Sec. 3.5.2), and dynamic channel pruning (Sec. 3.5.3).

We describe below in detail how PiX achieves these objectives despite keeping its structure the same. We also discuss the benefit of using PiX for these tasks. Note that it is the functionality of PiX that it can act as a network downscaler by controlling the channels. However, it is not a direct method of model compression.

3.5.1 Channel Squeezing

Prior works have conducted channel squeezing operations mostly with 1×1 layers in ResNet-like designs [11]. PiX maintains a similar level of accuracy to such approaches by utilizing channel sampling probability (Sec. 3.2) in conjunction with the pixel-wise dynamic channel sampling (Sec. 3.3). More importantly, PiX is *free from expensive dense 1×1 convolution*. Instead, by operating on a vector z, PiX effectively saves FLOPs and squeezes the channel faster.

To demonstrate our claims, we replace channel squeezing 1×1 layers in the representative ResNet [11] family (ResNet-50, -101, and -152) with PiX and evaluate the accuracy, FLOPs, and training and inference time. PiX speeds up the training and inference, which are empirically verified in Table 1 and Table 4 (see the supplement for the details).

Alternatively, channel squeezing can be done via depth-wise pooling in a non-parametric way [17]. However, it eliminates all the squeeze convolution layers, resulting in an accuracy drop, as shown in E4 in Table 7.

3.5.2 Network Downscaling

We can control ConvNets' parameters and computational complexity by adjusting the number of input or output channels. When conducting parameter reduction, it is called network downscaling. PiX can achieve this goal via its channel reduction capability. In our approach, the input feature map for each layer is squeezed by the PiX module with sampling factor $\zeta > 1$ and then sent to the next layers.

PiX module can be inserted into the existing layers, allowing it to downscale ConvNets by changing ζ. We use ResNet-18, ResNet-50, VGG-16, and MobileNet for the effectiveness of this application. Notably, PiX-downscaled network variant consistently outperforms the downscaled baseline. PiX-downscaled networks have the same parameters but lower FLOPs and higher accuracy (Table 3).

3.5.3 Dynamic Channel Pruning

When we plug PiX into a model, it uses ζ to determine the number of output channels. Thus, once ζ is set, the number of channels obtained from PiX is deterministic or *static*. However, as PiX selects channels on the fly, meaning that which channels will be sent to the next layer is not predetermined, it leads to a *dynamic reduction behavior*.

For this reason, we call PiX as static-dynamic channel pruner. This contrasts with the dynamic channel pruning approach, which keeps all the channels in the network intact but decides which ones to compute to save computations. This mandates the need for *specialized convolution implementation* to take advantage. On the other hand, the static-dynamic behavior of PiX is free of such necessity, which is of practical significance. The static behavior reduces the network's memory footprint and bandwidth while outperforming dynamic channel pruning approaches.

Please refer to the supplement for the procedure to embody PiX as a dynamic channel pruner. Table 6 shows a comparison with dynamic pruning approaches. We use ResNet-18 and VGG-16 for evaluation.

3.6. Relation With Existing Approaches

Using Global Context. We discuss representative approaches that are closest to the proposed PiX. The idea of using global context was introduced by SENet [15] aiming to improve network accuracy, which squeezes and expands a global context vector by using two convolution layers to

Table 1. PiX as a channel squeezer. We replace 1×1 channel squeezing layers in ResNet with PiX. We denote the channel squeezing factor of the vanilla network and our modification in the ζ column.

	Approach	ζ	#Params	FLOPs ↓	Top-1% ↑	Train Time Per-Iteration ↓	Train Time 120-Epochs ↓	Train Time 200-Epochs ↓
E0	● ResNet-50 [11]	4	25.5M	4.12B	76.30	575ms	4.0 Days	6.7 Days
	○ ResNet-50 + PiX	4	25.5M	**3.18B** (↓22.8%)	**76.77** (↑0.47%)	**359ms**	**2.5 Days**	**4.1 Days**
	○ ResNet-50 + PiX(`Avg`)	4	25.5M	**3.18B** (↓22.8%)	**76.58** (↑0.28%)	**359ms**	**2.5 Days**	**4.1 Days**
	○ ResNet-50 + PiX(`Max`)	4	25.5M	**3.18B** (↓22.8%)	**76.57** (↑0.27%)	**359ms**	**2.5 Days**	**4.1 Days**
E1	● ResNet-101	4	44.5M	7.85B	77.21	575ms	4.0 Days	6.7 Days
	○ ResNet-101 + PiX	4	44.5M	**6.05B** (↓22.9%)	**77.96** (↑0.45%)	**431ms**	**3.0 Days**	**5.0 Days**
E2	● ResNet-152	4	60.1M	11.58B	77.78	863ms	6.0 Days	10.0 Days
	○ ResNet-152 + PiX	4	60.1M	**8.91B** (↓23.0%)	**78.12** (↑0.44%)	**575ms**	**4.0 Days**	**6.7 Days**
E3	● ResNet-50	8	12.3M	1.85B	73.66	260ms	1.8 Days	3.0 Days
	○ ResNet-50 + PiX	8	12.3M	**1.39B** (↓24.8%)	**74.47** (↑0.81%)	**180ms**	**1.25 Days**	**2.0 Days**
E4	● ResNet-50 + SE [15]	4	28.0M	4.13B	76.85	575ms	4.0 Days	6.7 Days
	○ ResNet-50 + SE + PiX	4	28.0M	**3.19B** (↓22.8%)	**76.95** (↑0.10%)	**359ms**	**2.5 Days**	**4.1 Days**

Table 2. A functional comparison of PiX.

Method	No Finetuning	No Custom Convolutions	As a Channel Squeezer	As a Network Downscalar	As a Dynamic Pruner
● SE [15]	✓	✓	✗	✗	✗
● CBAM [36]	✓	✓	✗	✗	✗
● FBS [8]	✗	✗	✗	✗	✓
○ PiX	✓	✓	✓	✓	✓

predict channel saliency. CBAM [36] extends SENet, performing both max and avg. pooling during global context extraction then passes them through a shared MLP. FBS [8] uses global attention to predict channel saliency. FBS picks Top-K channels using the predicted channel saliency, and the suppressed channels are inhibited in the computations of the subsequent layer. PiX inherits the idea of using global context to generate sampling probability p. (Sec. 3.2)

Channel Pruning. PiX differs from existing channel pruning [8] approaches in both structure and functionality. PiX is not natively a channel pruner; it is the ability of PiX to sample channels on the fly, which can be utilized as a channel pruner. Therefore, PiX *does not require an architectural change* to behave as a channel pruner. On the other hand, FBS [8], for instance, is a channel pruner, and the design is not intended for other purposes, e.g., as a channel squeezer. For reference, we report the accuracy drop when FBS is modified to work as a channel squeezer in Sec. 4.5.

A functional comparison of PiX with prior work is shown in Table 2. We recommend referring to the supplement for visual differences between PiX and SENet, CBAM, and FBS. In the supplement, we also provide details on *the memory and FLOPs requirements* of PiX, SE, CBAM, and FBS. Note that PiX has the lowest FLOPs and memory consumption.

Group Convolution. Apart from the above modules, in terms of operation, the channel space partition should not be confused with group convolution (GC) [27, 38]. In GC, the input channels are divided into groups, and convolution is performed over each group, whereas we perform our PiX dynamic channel sampling operation onto each pixel. Moreover, the kernel size in GC is a hyperparameter, which does not exist in PiX. Also, GC requires the input number of channels to be exactly divisible by the number of groups, which is not the case with PiX. Please see the supplement for the visual differences between GC and PiX.

4. Experiments

We evaluate PiX by plugging it into various prominent ConvNets [11, 14, 33] and Transformers [25], and we compare against recent approaches [3, 5, 15, 23, 36]. We follow the tradition of training the models on ImageNet [4] with 1.28M training and 50K validation images over 1,000 categories for image classification task. For transfer learning, we use CIFAR-10 and CIFAR-100 datasets for image classification and CityScapes [2] for the downstream task of semantic segmentation. We use [7] for FLOP calculations, which aligns with our theoretical calculations.

Please see the supplement for training details, code snippets, ablations, and our theoretical FLOP calculations.

4.1. PiX as Channel Squeezer

Channel Squeezing (Sec. 3.5.1) aims to reduce FLOPs while maintaining accuracy and parameters (Table 1).

E0 - E2: PiX reduces FLOPs by 23% in ResNet family while having better accuracy. PiX achieves computationally efficient squeezing, as visible by the $\sim 23\%$ reduction in FLOPs in all of the PiX variants. Interestingly, ResNet-101 + PiX surpasses the baseline ResNet-152 with a significant FLOP difference of 47%. We argue that our conjecture on reusing the parameters of PiX works to maintain the non-

Table 3. PiX as a network downscaler. Increasing ζ in the networks where our PiX is applied decreases the number of parameters, working as a network downscaler. For a fair comparison with the baseline networks, we match the size of the ResNet, VGG, and MobileNet family to our downscaled networks. Baseline networks + PiX consistently shows better accuracy and reduced FLOPs with similar network parameters.

Approach	#Param	FLOPs ↓	Top-1% ↑	Approach	#Param	FLOPs ↓	Top-1% ↑
ResNet-18 × 1.050	12.80M	1.99B	71.71	VGG-16 × 1.05	16.72M	4.20B	73.25
ResNet-18 + PiX @$\zeta = 1$	12.80M	**1.84B**	**73.15**	VGG-16 + PiX @$\zeta = 1$	16.78M	**3.85B**	**74.53**
ResNet-18 × 0.756	6.77M	1.12B	69.37	VGG-16 × 0.63	8.67M	2.26B	70.53
ResNet-18 + PiX @$\zeta = 2$	6.77M	**0.99B**	**70.60**	VGG-16 + PiX @$\zeta = 2$	8.65M	**1.94B**	**72.47**
ResNet-18 × 0.631	4.78M	0.82B	67.55	VGG-16 × 0.75	5.97M	1.59B	69.12
ResNet-18 + PiX @$\zeta = 3$	4.77M	**0.72B**	**68.70**	VGG-16 + PiX @$\zeta = 3$	5.96M	**1.32B**	**70.78**
ResNet-18 × 0.555	3.74M	0.67B	66.10	VGG-16 × 0.54	4.59M	1.25B	67.56
ResNet-18 + PiX @$\zeta = 4$	3.74M	**0.57B**	**67.15**	VGG-16 + PiX @$\zeta = 4$	4.59M	**0.98B**	**69.32**
ResNet-50 × 1.051	28.09M	4.51B	76.57	MobileNet-v1 ×1.334	7.04M	0.97B	74.49
ResNet-50 + PiX @$\zeta = 1$	28.08M	**4.13B**	**77.65**	MobileNet-v1 + PiX @$\zeta = 1$	7.03M	**0.58B**	**74.53**
ResNet-50 × 0.732	14.09M	2.33B	75.62	MobileNet-v1 ×1.0	4.20M	0.58B	70.60
ResNet-50 + PiX @$\zeta = 2$	14.08M	**2.12B**	**76.65**	MobileNet-v1 + PiX @$\zeta = 2$	4.06M	**0.33B**	**72.27**
ResNet-50 × 0.657	11.52M	1.95B	75.11				
ResNet-50 + PiX @$\zeta = 3$	11.51M	**1.76B**	**75.70**				

Table 4. Speed analysis of PiX as a channel squeezer. PiX introduces speed gain on various entry-level or low-powered GPUs. We use @224 × 224 px., @FP32, and the reported numbers are the mean of 25 runs. 'FPS': Frames Per Second.

NVIDIA GPUs	Cores	Computing power	ResNet-50	ResNet-50 +PiX	ResNet-101	ResNet-101 +PiX	ResNet-152	ResNet-152 +PiX
A40	10752	37.00 TFLOPs	142 FPS	**166** FPS (17% ↑)	90 FPS	**100** FPS (11% ↑)	66 FPS	**71** FPS (8% ↑)
RTX-2080Ti	4352	13.45 TFLOPs	125 FPS	**166** FPS (32% ↑)	71 FPS	**83** FPS (17% ↑)	58 FPS	**66** FPS (14% ↑)
GTX-1080Ti	3584	11.45 TFLOPs	111 FPS	**142** FPS (28% ↑)	76 FPS	**83** FPS (10% ↑)	58 FPS	**66** FPS (14% ↑)
Jetson NX	384	1.00 TFLOPs	20 FPS	**25** FPS (25% ↑)	13 FPS	**16** FPS (23% ↑)	10 FPS	**12** FPS (20% ↑)

linearity of the network is verified. Also, the empirical result shows that PiX learns useful data representations (Sec. 3.5.1). Despite the reduction in FLOPs, PiX exhibited slight accuracy improvements.

E3: PiX with a higher squeezing factor. We analyze PiX for a higher squeezing factor, i.e., $\zeta = 8$, and observe that PiX performs better than the baseline while having almost 25% fewer FLOPs. Interestingly, the accuracy gap between ResNet@$\zeta = 4$ and $\zeta = 8$ is 2.64%, while this gap reduces to 2.30% for PiX at a notable 56% reduction in the FLOPs.

These empirical results demonstrate the robustness of PiX towards parameter reduction and its ability to learn to sample channels efficiently.

E4: PiX enabled squeeze-excitation (SE) networks [15] are more accurate. It is noticeable that PiX performs better than SE, especially in FLOPs, indicating that PiX improves the computational performance of SE-like modules. It is because PiX reduces the computations of the channel squeezing layer from the network equipped with SE-like modules. Hence, the network can take advantage of global attention weighting from SE-like modules and computationally efficient channel squeezing operation via PiX.

E0-E4: PiX reduces training time on ResNet. Table 1 also shows throughput analysis on 8× NVIDIA 1080Ti system. Noticeably, PiX has the lowest per-iteration time, which reduces the overall training duration. Since PiX reduces the computations of the channel squeezing 1×1 layers, this indicates that 1×1 squeeze layers are a computational bottleneck in ResNet.

4.2. Inference Latency

Since FLOPs are not an accurate measure of the actual speed [5], we conduct a latency analysis on four different types of GPUs (Table 4). The first three are entry-level desktop GPUs, while the last one is a low-powered (10W) embedded computing device that is far less powerful. The table shows that PiX brings a maximum of 32% speedup, which demonstrates the practicality of PiX for real-time applications.

4.3. PiX as Network Downscaler

Along with channel squeezing, PiX also offers simplified network downscaling (Sec. 3.5.2). By increasing ζ, we achieve a similar effect to that of network downscaling, outperforming the downscaled networks by other approaches. We used width scaling (increasing the number of channels in each conv layer) for the baseline.

The empirical result in Table 3 shows that our proposed PiX is seamlessly applicable for network downscaling regardless of network architectures (ResNet-18, ResNet-50,

Table 5. PiX + ViT. We replace the vanilla channel squeezing layer with PiX in the feed-forward network (FFN) of recent EfficientViT [25]. We observe that the utility of PiX also transfers to the Transformer models, as evidenced by the reduced runtime. *Note:* EfficientViT uses a squeezing factor of two in its FFN.

Approach	ζ	#Param	FLOPs	Top-1%	Training Hours
∘ EfficientViT-M5 [25]		12M	522M	76.8	36
∘ EfficientViT-M5 + PiX	2	12M	522M	76.9	**24**
∘ EfficientViT-M5 [25] ×0.5		3.2M	136M	67.8	32
∘ EfficientViT-M5 + PiX ×0.5	2	3.2M	136M	67.8	**24**

VGG-16, and even on MobileNet-v1), showing superior performance than all the baselines. It shows the diverse scope and applicability of PiX in low-powered devices for customizing a network for a dedicated purpose.

4.4. PiX into Vision Transformers (ViT)

Although our approach is designed for ConvNets, we go even further and apply PiX into ViT models to investigate the feasibility. We apply PiX to the feed-forward network (FFN) of the ViTs, which is essentially a stack of channel expansion 1×1 layer followed by a channel squeezing 1×1 layers. We experiment with the latest EfficientViTs [25]. We choose the EfficientViT-M5 variant.

Since FFN layers form only a small portion of Transformers, the parameter and FLOPs roughly remain the same, as shown in Table 5. However, the wall time of the PiX variant is smaller, reducing the training time from 36 hours to 24 hours and reducing the downscaled model's training time from 32 hours to 24 hours. Despite similar FLOPs, the functioning of PiX requires less memory access, which reduces the memory access cost (MAC) and hence latency [5].

We believe that with further improvement in the context of ViTs, the classification performance of PiX can be improved, which we leave as future work.

4.5. PiX as Dynamic Channel Pruner

The ability of PiX to pick channels dynamically is similar to dynamic pruning (Sec. 3.5.3). The difference is that PiX selects the channels dynamically while existing approaches turn off a few channels. We compare PiX with dynamic pruning approaches.

PiX *vs.* dynamic pruning approaches. Referring to Table 6, the PiX baseline (i.e., ResNet-18 + PiX $@\zeta = 1$, Top-1 Acc. 73.15%) and the downscaled (ResNet-18 + PiX $@\zeta = 3$, Top-1 Acc. 70.60% in Table 3), shows compelling performance than the state-of-the-art dynamic pruning approaches [1, 6, 8, 12, 16, 28, 35, 40]. Note that PiX does not require fine-tuning to obtain better performance, unlike other approaches, such as [8], leading to a simpler pipeline of PiX.

Following [8, 22, 24], we report ΔTop-5 error with the benefit of FLOP reduction using VGG-16 as a baseline. Table 6 shows that PiX offers a competitive performance than other approaches [8, 13, 22, 24, 26].

Table 6. PiX as a dynamic channel pruner. We compare our approach with representative dynamic or static channel pruning methods using ResNet-18 and VGG-16. Vanilla ConvNet + PiX shows compatible accuracy and FLOPs saving gain.

@ ResNet-18	Dynamic	Top-1% ↑ Baseline	Top-1% ↑ Downscaled	FLOPs Saving↑
∘ Soft Filter Pruning [12]		70.28	67.10	1.72×
∘ Discrimination-aware [40]		69.64	67.35	1.89×
∘ Collaborative Layers [6]	✓	69.98	67.33	1.53×
∘ Channel Gating [16]	✓	69.02	67.40	1.61×
∘ Boosting and Suppression [8]	✓	70.71	68.17	1.98×
∘ Storage Efficient Pruning [1]	✓	69.76	68.73	1.94×
∘ Manifold Reg. Pruning [35]	✓	69.76	68.88	2.06×
∘ Dynamic Struct. Pruning [28]	✓	69.76	68.38	2.56×
∘ PiX	✓	**73.15**	**70.60**	1.85×
@ VGG-16	**Dynamic**	**Δ Top-5 ↑**		**FLOPs Saving↑**
∘ Filter Pruning [22]		−8.6		4×
∘ Runtime Neural Pruning [24]	✓	−2.32		3×
∘ AutoML Compression [13]		−1.4		5×
∘ ThiNet-Conv [26]		−0.37		3×
∘ Boosting and Suppression [8]	✓	**−0.04**		3×
∘ PiX	✓	**−0.04**		3×

Existing dynamic channel pruning approach is not multi-purpose. To highlight the key advantage of PiX that it does not need to change its structure to serve different purposes, we customize FBS [8] for channel squeezing, although FBS is not intended to perform. FBS was chosen because of its strong resemblance with disabling channels via global attention. FBS picks top-k channels in its original operation and has the same input-output dimensions, i.e., $\in \mathbb{R}^{C \times H \times W}$. However. for this experiment, we configure FBS to output $\in \mathbb{R}^{\lceil C/k \rceil \times H \times W}$, where $k = \zeta$.

We then replace all the channel squeezing layers with this modified FBS module and train the model. We observe that FBS faces convergence issues. We identify the underlying cause is due to the drop-out of intermediate channels from the input X when selecting top-k channels. Also, the channels appearing in the output (Y) that lost position identity or channel index causes convergence issues. When Y is operated upon via subsequent convolutions, the approach is not intended to learn the relation between the channels, as the position or index of a given channel in X keeps changing in Y. This indicates that FBS-like pruning methods can not complement PiX, but vice-versa is possible, as demonstrated earlier, highlighting the utility of PiX.

4.6. PiX in the Wild

We compare PiX with prior works [3, 15, 23, 36] in improving ResNet accuracy and feature fusion via the attention mechanism [3, 15, 36]. We present the result in Table 7.

E0-E2: PiX *vs.* SE [15] and CBAM [36]. We compare PiX with the methods that aim to improve performance with the newly proposed layer. We observe that PiX performs

Table 7. PiX *vs*. existing approaches for enhancing the accuracy of the vanilla ConvNets. '⋆' denotes that PiX is applied only before the second layer of a ResNet-18 block (see the supplement).

	Approach	#Params ↓	FLOPs ↓	Top-1% ↑
E0	ResNet-18 [11]	11.60M	1.81B	70.40
	ResNet-18 + SE [15]	11.78M	1.81B	70.59
	ResNet-18 + CBAM [36]	11.78M	1.81B	70.73
	ResNet-18 + PiX⋆	11.88M	1.81B	**71.65**
	ResNet-18 + PiX	12.80M	1.84B	**73.15**
E1	ResNet-50	25.50M	4.12B	76.30
	ResNet-50 + SE [15]	28.09M	4.13B	76.85
	ResNet-50 + CBAM [36]	28.09M	4.13B	77.34
	ResNet-50 + PiX	28.08M	4.13B	**77.65**
E2	MobileNet [14]	4.23M	0.56B	68.61
	MobileNet + SE [15]	5.07M	0.57B	70.03
	MobileNet + CBAM [36]	5.07M	0.57B	70.99
	MobileNet + PiX	**4.06M**	**0.33B**	**72.27**
E3	ResNet-50 + AFF [3] @160 Epochs	30.30M	4.30B	79.10
	ResNet-50 + SKNet [23] @160 Epochs	27.70M	4.47B	79.21
	ResNet-50 + PiX @160 Epochs	28.08M	**4.13B**	**79.40**
E4	RepVGG-A0 [5]	9.10M	1.51B	72.41
	VGG-16 [33] + PiX	**8.65M**	1.94B	**72.47**
E5	ResNet-50 + DWP [17]	19.60M	2.82B	75.35
	ResNet-50 + PiX @$\zeta = 2$	**14.08M**	**2.12B**	**76.65**

better than SE and CBAM, even on MobileNet [14], while the proposed PiX has a simpler structure and multi-purpose utility.

E3: PiX *vs.* AFF [3] and SKNet [23]. Attentional Feature Fusion (AFF) fuses two feature maps adaptively, and SKNet improves accuracy by adaptively weighting the output of two convolutions with different kernel sizes. These models are trained for longer epochs. Therefore, we also train PiX at the same setting [3]. We observe that PiX outperforms these two methods while being architecturally simple.

E4: PiX + VGG *vs.* RepVGG [5]. RepVGG is a recent approach that speeds up VGG [33] via structural reparameterization (Sec. 2) during inference time only. We see that VGG-16 + PiX offers a competitive performance to RepVGG while being simpler at both train and test time.

E5: PiX *vs.* DWP [17]. Depth-wise pooling (DWP) is a comparable approach for channel squeezing. Hence, we trained ResNet-50 endowed with DWP.

As mentioned in Sec. 3.5.1, eliminating sampling probability predictor ϕ from the network removes all the squeezing layers, leading to parameter and accuracy loss. DWP is an example of this case, which eliminates all the 1×1 squeezing layers, facing a loss of accuracy (1.30%), compared to PiX used for channel squeezing.

Due to the parameter differences in ResNet-50 + PiX and ResNet-50 + DWP, we compare the latter with a downscaled variant of ResNet-50 + PiX. As a result, PiX surpasses DWP, verifying our hypothesis that in channel squeezing mode, PiX preserves the non-linearity that allows for maintaining accuracy.

Table 8. PiX *vs*. ResNet. Transfer learning evaluation for classification (E0) and semantic segmentation (E1) tasks.

	Approach	#Params	FLOPs ↓	CIFAR-10 ↑	CIFAR-100 ↑
E0	ResNet-50 [11]	25.5M	4.12B	95.57	81.60
	ResNet-50 + PiX	25.5M	**3.18B**	**95.67**	**82.22**
	Approach	**#Params**	**FLOPs ↓**	**CityScapes ↑**	
E1	ResNet-101 + [39]	44.5M	7.85B	78.4	
	ResNet-101 + [39] + PiX	44.5M	**6.05B**	**79.1**	

4.7. Transfer Learning

E0: PiX transfers better on image classification task. To analyze the generalization of PiX across datasets and tasks, we perform transfer learning from ImageNet to CIFAR-10 and CIFAR-100. Each of the datasets consists of 50K training and 10K test images. For training, we finetune the models pretrained over ImageNet. The training strategy for both datasets remains identical to that of ImageNet except for 200 epochs. From Table 8, it can be seen that PiX performs better at lower FLOP requirements.

E1: PiX transfers better on semantic segmentation task. We evaluate PiX for a challenging task of semantic segmentation. We use a prominent approach [39] and replace the backbone with ResNet-101+PiX. Consequently, PiX outperforms the baseline both in terms of FLOPs and accuracy by 0.7% units mIoU.

5. Conclusion

In this work, we introduce Pick-or-Mix (PiX) for dynamic channel sampling. It works by exploiting global spatial context by blending cross-channel information and then picking or mixing channels on *per-pixel basis*. The picked channels can be different for each pixel depending upon the operator selection. This capability allows PiX to maintain accuracy even by cutting down FLOPs. PiX can work as a computationally efficient channel squeezer, can downscale a given model, or function as a dynamic channel pruner. We show that PiX is easy to plug into the existing ConvNets or even ViT, without altering its structure, and we show that PiX outperforms state-of-the-art approaches.

Limitations. Currently, our approach is designed for discrete squeezing factors ζ. Future extensions of the proposed approach include developing a more generalized fusion approach that can sample channels at non-integer ζ.

Acknowledgment. This study was supported by the I-Hub Foundation for Cobotics (IHFC), Technology Innovation Hub of Indian Institute of Technology, Delhi (IIT Delhi) under the project grant IITM/IHFC/IITDELHI/LB/370. Danuel Kim and Jaesik Park were supported by IITP grant funded by the Korea government (MSIT) (No.2021-0-01343, AI Graduate School Program: Seoul National University, 5%) and NRF grant No.2023R1A1C200781211 (95%).

References

[1] Jianda Chen, Shangyu Chen, and Sinno Jialin Pan. Storage efficient and dynamic flexible runtime channel pruning via deep reinforcement learning. *Advances in neural information processing systems*, 33:14747–14758, 2020. 2, 7

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[3] Yimian Dai, Fabian Gieseke, Stefan Oehmcke, Yiquan Wu, and Kobus Barnard. Attentional feature fusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3560–3569, 2021. 2, 5, 7, 8

[4] 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. 5

[5] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021. 2, 5, 6, 7, 8

[6] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5840–5848, 2017. 7

[7] flops counting tool. *https://github.com/sovrasov/flops-counter.pytorch*. 5

[8] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018. 1, 2, 5, 7

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 3

[10] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589, 2020. 1

[11] 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. 1, 2, 3, 4, 5, 8

[12] 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. 1, 7

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018. 2, 7

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 5, 8

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2, 3, 4, 5, 6, 7, 8

[16] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 7

[17] Abid Hussain and Wang Hesheng. Depth-wise pooling: A parameter-less solution for channel reduction of feature-map in convolutional neural network. In *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 299–304. IEEE, 2019. 4, 8

[18] Yoonwoo Jeong, Seungjoo Shin, Junha Lee, Chris Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Perfception: Perception using radiance fields. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 1

[19] 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. 4

[20] Ashish Kumar and Laxmidhar Behera. Semi supervised deep quick instance detection and segmentation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8325–8331. IEEE, 2019. 2

[21] Ashish Kumar, Mohit Vohra, Ravi Prakash, and Laxmidhar Behera. Towards deep learning assisted autonomous uavs for manipulation tasks in gps-denied environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1613–1620. IEEE, 2020. 2

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. 2016. 2, 7

[23] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 510–519, 2019. 2, 5, 7, 8

[24] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017. 7

[25] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14430, 2023. 5, 7

[26] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 7

[27] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 5

[28] Jun-Hyung Park, Yeachan Kim, Junho Kim, Joon-Young Choi, and SangKeun Lee. Dynamic structure pruning for compressing cnns. *arXiv preprint arXiv:2303.09736*, 2023. 2, 7

[29] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 1

[31] 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. 2

[32] L Sifre and S Mallat. Rigid-motion scattering for image classification. arxiv 2014. *arXiv preprint arXiv:1403.1687*. 2

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 2, 5, 8

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 2

[35] Yehui Tang, Yunhe Wang, Yixing Xu, Yiping Deng, Chao Xu, Dacheng Tao, and Chang Xu. Manifold regularized dynamic network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5018–5028, 2021. 1, 2, 7

[36] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2, 5, 7, 8

[37] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2736–2746, 2022. 2

[38] 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*, pages 6848–6856, 2018. 2, 5

[39] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 8

[40] 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. *Advances in neural information processing systems*, 31, 2018. 7