REVISITING DYNAMIC CONVOLUTION VIA MATRIX DECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent research in dynamic convolution shows substantial performance boost for efficient CNNs, due to the adaptive aggregation of K static convolution kernels. It has two limitations: (a) it increases the number of convolutional weights by K-times, and (b) the joint optimization of dynamic attention and static convolution kernels is challenging. In this paper, we revisit it from a new perspective of matrix decomposition and reveal the key issue is that dynamic convolution applies dynamic attention over channel groups after projecting into a higher dimensional intermediate space. To address this issue, we propose dynamic channel fusion to replace dynamic attention over channel groups. Dynamic channel fusion not only enables significant dimension reduction of the intermediate space, but also mitigates the joint optimization difficulty. As a result, our method is easier to train and requires significantly fewer parameters without sacrificing accuracy.

1 INTRODUCTION

Recently, dynamic convolution (Yang et al., 2019; Chen et al., 2020c) has achieved a significant performance improvement in light-weight networks (Howard et al., 2017; Zhang et al., 2018b) with negligible computational cost. It has been soon adopted and extended for multiple vision tasks (Su et al., 2020; Chen et al., 2020b; Ma et al., 2020; Tian et al., 2020). Compared to its static counterpart, dynamic convolution aggregates multiple convolution kernels dynamically based upon their input dependent attention. Mathematically, the weight matrix of dynamic convolution is represented as:

$$\boldsymbol{W}(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{x}) \boldsymbol{W}_k \quad \text{s.t.} \quad 0 \leq \pi_k(\boldsymbol{x}) \leq 1, \sum_{k=1}^{K} \pi_k(\boldsymbol{x}) = 1, \tag{1}$$

where K convolution kernels $\{\boldsymbol{W}_k\}$ are aggregated linearly by using attention scores $\{\pi_k(\boldsymbol{x})\}$.

Dynamic convolution has two limitations: (a) it is not compact due to using K kernels, and (b) the joint optimization between the attention scores $\{\pi_k(\boldsymbol{x})\}$ and static kernels $\{\boldsymbol{W}_k\}$ is challenging. The small attention score π_k makes its corresponding kernel $\boldsymbol{W}_k$ hard to learn, especially in early training epochs. Chen et al. (2020c) shows that the training converges slowly when using softmax to generate $\{\pi_k\}$ that is near one-hot. Therefore, Yang et al. (2019) and Chen et al. (2020c) use additional constraints to mitigate these limitations. For instance, Chen et al. (2020c) uses a large temperature in softmax to encourage near-uniform attention.

In this paper, we revisit these two limitations via matrix decomposition. Accordingly, the dynamic convolution is reformulated by using singular value decomposition (SVD) as follows:

$$\boldsymbol{W}(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{x}) \boldsymbol{W}_0 + \sum_{k=1}^{K} \pi_k(\boldsymbol{x}) \Delta \boldsymbol{W}_k = \boldsymbol{W}_0 + \boldsymbol{U} \boldsymbol{\Pi}(\boldsymbol{x}) \boldsymbol{S} \boldsymbol{V}^T, \tag{2}$$

where $\boldsymbol{W}_0 = \sum_k \boldsymbol{W}_k / K$ is the average kernel. $\Delta \boldsymbol{W}_k = \boldsymbol{W}_k - \boldsymbol{W}_0$ is the residual and its SVD is denoted as $\Delta \boldsymbol{W}_k = \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T$. The singular values and singular vectors are then grouped as $\boldsymbol{U} = [\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K]$, $\boldsymbol{S} = diag(\boldsymbol{S}_1, \ldots, \boldsymbol{S}_K)$, $\boldsymbol{V} = [\boldsymbol{V}_1, \ldots, \boldsymbol{V}_K]$. $\boldsymbol{\Pi}(\boldsymbol{x})$ stacks attention scores diagonally as $\boldsymbol{\Pi}(\boldsymbol{x}) = diag(\pi_1(\boldsymbol{x})\boldsymbol{I}, \ldots, \pi_K(\boldsymbol{x})\boldsymbol{I})$, where $\boldsymbol{I}$ is an identity matrix. The decomposition is illustrated in Figure 1. Dynamic convolution is essentially adding a dynamic residual to a static convolution. The dynamic residual projects the input $\boldsymbol{x}$ to a higher dimensional space $\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}$

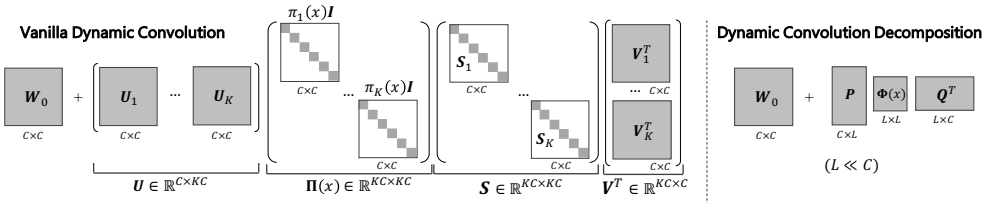


Figure 1: Dynamic convolution via matrix decomposition. **Left**: Reformulating the vanilla dynamic convolution by matrix decomposition (see Eq. 2). It applies ***dynamic attention*** $\boldsymbol{\Pi}(\boldsymbol{x})$ ***over channel groups*** in a ***high dimensional space*** ($\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x} \in \mathbb{R}^{KC}$). **Right**: proposed dynamic convolution decomposition, which applies ***dynamic channel fusion*** $\boldsymbol{\Phi}(\boldsymbol{x})$ in a ***low dimensional space*** ($\boldsymbol{Q}^T\boldsymbol{x} \in \mathbb{R}^L$, $L \ll C$), resulting in a more compact model.

(from C channels to KC channels), applies dynamic attention $\boldsymbol{\Pi}(\boldsymbol{x})$ over channel groups, and then reduces the dimension back to C channels by multiplying $\boldsymbol{U}$. *The major issue is the dynamic attention over channel groups.* It requires a high dimensional intermediate space and small attention values may suppress the learning of the corresponding channels.

To address this issue, we propose *dynamic convolution decomposition* (DCD), in which the key idea is to use *dynamic channel fusion* to replace dynamic attention over channel groups. Specifically, the dynamic channel fusion is represented as a full dynamic matrix $\boldsymbol{\Phi}(\boldsymbol{x})$, of which each element $\phi_{i,j}(\boldsymbol{x})$ is a function of input $\boldsymbol{x}$. As shown in Figure 1-(right), the dynamic residual is reformulated as the product of $\boldsymbol{\Phi}(\boldsymbol{x})$ and two static matrices as $\boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{Q}^T$, where $\boldsymbol{Q}$ squeezes the input into a low dimensional intermediate space, $\boldsymbol{\Phi}(\boldsymbol{x})$ dynamically fuses the channels in the intermediate space, and $\boldsymbol{P}$ expands the number of channels to the output space.

The key innovation is that dynamic channel fusion $\boldsymbol{\Phi}(\boldsymbol{x})$ enables significant dimension reduction in the intermediate space ($\boldsymbol{Q}^T\boldsymbol{x} \in \mathbb{R}^L$, $L \ll C$) without performance degradation. The number of parameters in $\boldsymbol{P}, \boldsymbol{Q}$ is significantly reduced, resulting in a more compact model. But the performance is not sacrificed as dynamic channel fusion retains representation power. Dynamic channel fusion also mitigates the joint optimization challenge in the vanilla dynamic convolution, as each column in matrices $\boldsymbol{P}, \boldsymbol{Q}$ is associated with multiple dynamic coefficients in $\boldsymbol{\Phi}(\boldsymbol{x})$. Thus, a few dynamic coefficients with small values do not suppress the learning of static matrices $\boldsymbol{P}, \boldsymbol{Q}$. Experimental results show that, without using additional constraints in Yang et al. (2019); Chen et al. (2020c), our method significantly reduces the number of parameters while achieving higher accuracy.

2 Related Work

Efficient CNNs: MobileNet (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019) decomposes $k \times k$ convolution into a depthwise and a pointwise convolution. ShuffleNet (Zhang et al., 2018b; Ma et al., 2018) uses group convolution and channel shuffle to further simplify pointwise convolution. Further improvement based upon these architectures has been investigated recently. EfficientNet (Tan & Le, 2019a; Tan et al., 2020) finds a proper relationship between input resolution and width/depth of the network. Tan & Le (2019b) mixes up multiple kernel sizes in a single convolution. Chen et al. (2020a) trades massive multiplications for much cheaper additions. Han et al. (2020) applies a series of cheap linear transformations to generate ghost feature maps. Zhou et al. (2020) flips the structure of inverted residual block to alleviate information loss. Yu et al. (2019) and Cai et al. (2019) train one network that supports multiple sub-networks with different complexities.

Matrix Decomposition: Lebedev et al. (2014) and Denton et al. (2014) use Canonical Polyadic decomposition (CPD) for convolution kernel to speed up networks, while Kim et al. (2015) investigates Tucker decomposition for the same purpose. More recently, Kossaifi et al. (2020) combines tensor decomposition with MobileNet to design efficient higher-order networks for video tasks, while Phan et al. (2020) proposes a stable CPD to deal with degeneracy issue of tensor decomposition during network training. Compared to these works using static decomposition, our method decomposes convolution kernel in a dynamic way by adapting the core matrix to the input.

Dynamic Neural Networks: Dynamic networks boost the representation capability by adapting parameters in convolution or activation function to the input. Ha et al. (2017) uses another network to

generate parameters for the main network. Hu et al. (2018) reweights channels by squeezing global context. Li et al. (2019) adapts attention over kernels with different sizes. Dynamic convolution (Yang et al., 2019; Chen et al., 2020c) aggregates multiple convolution kernels based on their attention. Ma et al. (2020) uses grouped fully connected layer to generate convolutional weights directly. Chen et al. (2020b) extends dynamic convolution from spatial agnostic to spatial specific. Su et al. (2020) proposes dynamic group convolution that adaptively selects input channels to form groups. Tian et al. (2020) applies dynamic convolution on instance segmentation. Chen et al. (2020d) adapts slopes and intercepts of two linear functions in ReLU (Nair & Hinton, 2010; Jarrett et al., 2009).

3 Our Method: Dynamic Convolution Decomposition

In this section, we introduce *dynamic convolution decomposition* to address the limitations of the vanilla dynamic convolution. For the sake of conciseness, we assume the convolution kernel $\boldsymbol{W}$ has the same number of input and output channels ($C_{in} = C_{out} = C$) and ignore the bias. We focus on 1×1 convolution in this section and will extend it to $k \times k$ convolution in the next section.

3.1 Revisiting Vanilla Dynamic Convolution

The vanilla dynamic convolution aggregates K convolution kennels $\{\boldsymbol{W}_k\}$ by using attention scores $\{\pi_k(\boldsymbol{x})\}$ (see Eq. 1). It can be reformulated as adding a dynamic residual on a static kernel (see Eq. 2), and the dynamic residual can be further decomposed by SVD (see Eq. 2 and Figure 1). The vanilla dynamic convolution has two limitations. Firstly, the model is not compact. *Essentially, it expands the number of channels by K times and applies dynamic attention over K channel groups.* The dynamic residual $\boldsymbol{U}\boldsymbol{\Pi}(\boldsymbol{x})\boldsymbol{S}\boldsymbol{V}^T$ is of shape $C \times C$, and its maximum rank is C. But it sums KC rank-1 matrices ($\pi_{\lceil i/C \rceil}(\boldsymbol{x})\boldsymbol{u}_i s_{i,i} \boldsymbol{v}_i^T$) as follows:

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{W}_0 + \boldsymbol{U}\boldsymbol{\Pi}(\boldsymbol{x})\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{W}_0 + \sum_{i=1}^{KC} \pi_{\lceil i/C \rceil}(\boldsymbol{x})\boldsymbol{u}_i s_{i,i} \boldsymbol{v}_i^T, \tag{3}$$

where $\boldsymbol{u}_i$ is the ith column vector of matrix $\boldsymbol{U}$, $\boldsymbol{v}_i$ is the ith column vector of matrix $\boldsymbol{V}$, $s_{i,i}$ is the ith diagonal entry of matrix $\boldsymbol{S}$ and $\lceil \cdot \rceil$ is ceiling operator. The static basis vector $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are not shared across different rank-1 matrices ($\pi_{\lceil i/C \rceil}(\boldsymbol{x})\boldsymbol{u}_i s_{i,i} \boldsymbol{v}_i^T$). This results in model redundancy.

Secondly, it is difficult to jointly optimize static matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and dynamic attention $\boldsymbol{\Pi}(\boldsymbol{x})$. This is because the small attention score $\pi_{\lceil i/C \rceil}$ may suppress the learning of its corresponding columns $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ in $\boldsymbol{U}$ and $\boldsymbol{V}$, especially in early training epochs (shown in Chen et al. (2020c)).

3.2 Dynamic Channel Fusion

We propose dynamic channel fusion to address the limitations of the vanilla dynamic convolution. Different from the diagonal attention matrix $\boldsymbol{\Pi}(\boldsymbol{x})$ that reweighs channel groups, dynamic channel fusion $\boldsymbol{\Phi}(\boldsymbol{x})$ is a full matrix, of which each element $\phi_{i,j}(\boldsymbol{x})$ is a function of input $\boldsymbol{x}$. $\boldsymbol{\Phi}(\boldsymbol{x})$ is of shape $L \times L$, dynamically fusing channels in the intermediate space $\mathbb{R}^L$. The key idea is that it enables significant dimension reduction in the intermediate space $L \ll C$, resulting in a more compact model. The dynamic convolution based on dynamic channel fusion is formulated as follows:

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{Q}^T = \boldsymbol{W}_0 + \sum_{i=1}^{L}\sum_{j=1}^{L} \boldsymbol{p}_i \phi_{i,j}(\boldsymbol{x}) \boldsymbol{q}_j^T, \tag{4}$$

where $\boldsymbol{Q}$ is of shape $C \times L$, squeezing the number of channels into a low dimensional space ($\boldsymbol{Q}^T\boldsymbol{x} \in \mathbb{R}^L$). These L channels are fused dynamically by using $\boldsymbol{\Phi}(\boldsymbol{x})$. $\boldsymbol{P}$ is of shape $C \times L$, expanding the number of channels from L to C as output. We name this new form of decomposition as **dynamic convolution decomposition** (DCD). The dimension of the intermediate space L is constrained as $L^2 < C$. The default value of L in this paper is set as $\lfloor \frac{C}{2^{\lfloor log_2 \sqrt{C} \rfloor}} \rfloor$ empirically, which means dividing C by 2 repeatedly until it is less than $\sqrt{C}$.

With this new design, the number of static parameters is significantly reduced (i.e. LC parameters in $\boldsymbol{P}$ or $\boldsymbol{Q}$ *v.s.* KC^2 parameters in $\boldsymbol{U}$ or $\boldsymbol{V}$, $L < \sqrt{C}$), thus resulting in a more compact model.

Mathematically, the dynamic residual $\boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{Q}^T$ sums L^2 rank-1 matrices $\boldsymbol{p}_i\phi_{i,j}(\boldsymbol{x})\boldsymbol{q}_j^T$, where $\boldsymbol{p}_i$ is the ith column vector of $\boldsymbol{P}$, and $\boldsymbol{q}_j$ is the jth column vector of $\boldsymbol{Q}$. This number (L^2) is much less than the counterpart (KC rank-1 matrices $\pi_{\lceil i/C \rceil}(\boldsymbol{x})\boldsymbol{u}_i s_{i,i}\boldsymbol{v}_i^T$) in the vanilla dynamic convolution (see Eq. 3) because L is constrained as $L^2 < C$. This demonstrates from another perspective that our method is more compact. Meanwhile, dynamic channel fusion also helps retain the representation power to prevent performance degradation.

Besides, DCD also mitigates the joint optimization difficulty. Since each column of matrix $\boldsymbol{P}$ (or $\boldsymbol{Q}$) is associated with multiple dynamic coefficients (e.g. $\boldsymbol{p}_i$ is related to $\phi_{i,1}, \ldots, \phi_{i,L}$), it is unlikely that the learning of $\boldsymbol{p}_i$ is suppressed by a few dynamic coefficients with small values.

3.3 More General Formulation

In the previous section, we focus on the dynamic residual and show that dynamic channel fusion provides a more compact solution. Here, we discuss the static kernel $\boldsymbol{W}_0$. Originally, it is multiplied by a dynamic scalar $\sum_k \pi_k(\boldsymbol{x})$, which is canceled in Eq. 2 as attention scores sum to one. If we relax the constraint $\sum_k \pi_k(\boldsymbol{x}) = 1$ and make it channel-wise, a more general form is achieved as:

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\Lambda}(\boldsymbol{x})\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{Q}^T, \tag{5}$$

where $\boldsymbol{\Lambda}(\boldsymbol{x})$ is a $C \times C$ diagonal matrix, of which each element $\lambda_{i,i}(\boldsymbol{x})$ is a function of input $\boldsymbol{x}$. $\boldsymbol{\Lambda}(\boldsymbol{x})$ provides channel-wise attention after the static kernel $\boldsymbol{W}_0$. Eq. 4 is a special case where $\boldsymbol{\Lambda}(\boldsymbol{x})$ is an identity matrix. Later, we will show that $\boldsymbol{\Lambda}(\boldsymbol{x})$ can provide additional improvement.

3.4 Dynamic Convolution Decomposition Layer

Implementation: Figure 2 shows the diagram of a dynamic convolution decomposition (DCD) layer. It has a light-weight dynamic branch to generate coefficients for both dynamic channel-wise attention $\boldsymbol{\Lambda}(\boldsymbol{x})$ and dynamic channel fusion $\boldsymbol{\Phi}(\boldsymbol{x})$. Similar to Squeeze-and-Excitation (Hu et al., 2018), the dynamic branch firstly squeezes the input $\boldsymbol{x}$ by average pooling. It is then followed by two fully connected (FC) layers with a ReLU between them. The former FC layer reduces the number of channels by r and the latter FC layer outputs $C+L^2$ coefficients (C for $\boldsymbol{\Lambda}$ and L^2 for $\boldsymbol{\Phi}$). Then we follow Eq. 5 to generate convolutional weights $\boldsymbol{W}(\boldsymbol{x})$. Similar to the static convolution, a DCD layer also has a batch normalization and an activation function (e.g. ReLU).

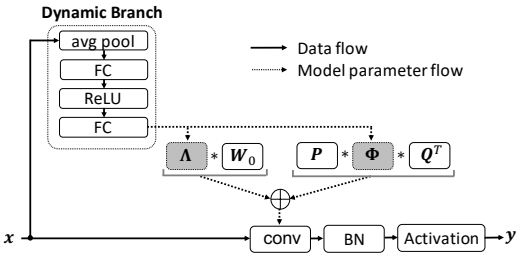


Figure 2: **Dynamic convolution decomposition layer**. The input $\boldsymbol{x}$ first goes through a dynamic branch to generate $\boldsymbol{\Lambda}(\boldsymbol{x})$ and $\boldsymbol{\Phi}(\boldsymbol{x})$, and then to generate the convolution matrix $\boldsymbol{W}(\boldsymbol{x})$ using Eq. 5. Finally, the convolution is applied on the input $\boldsymbol{x}$ and followed by batch normalization and activation function.

Parameter Complexity: DCD has similar FLOPs to the vanilla dynamic convolution. Here, we focus on the parameter complexity. The static convolution needs C^2 parameters and the vanilla dynamic convolution needs KC^2 parameters. DCD requires C^2, CL, and CL parameters for static matrices $\boldsymbol{W}_0$, $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. It also needs $(2C + L^2)\frac{C}{r}$ parameters in the dynamic branch to generate $\boldsymbol{\Lambda}(\boldsymbol{x})$ and $\boldsymbol{\Phi}(\boldsymbol{x})$, where r is the reduction rate of the first FC layer. The total complexity is $C^2 + 2CL + (2C + L^2)\frac{C}{r}$. Since L is constrained as $L^2 < C$, the complexity upper bound is $(1 + \frac{3}{r})C^2 + 2C\sqrt{C}$. When choosing $r = 16$, the complexity is about $1\frac{3}{16}C^2$. It is much less than that of the vanilla dynamic convolution ($4C^2$ in Chen et al. (2020c) and $8C^2$ in Yang et al. (2019)).

4 Extensions of Dynamic Convolution Decomposition

In this section, we extend the dynamic decomposition of 1×1 convolution (Eq. 5) in three ways: (a) sparse dynamic residual where $\boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{Q}^T$ is a diagonal block matrix, (b) $k \times k$ depthwise convolution, and (c) $k \times k$ convolution. Here, k refers to the kernel size.

4.1 DCD WITH SPARSE DYNAMIC RESIDUAL

The dynamic residual $\boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{Q}^T$ can be further simplified by making it sparse. An easy implementation is using a diagonal block matrix (see Figure 3). Each block is decomposed separately as $\boldsymbol{P}_b\boldsymbol{\Phi}_b(\boldsymbol{x})\boldsymbol{Q}_b^T, b \in [1, B]$ where B is the number of diagonal blocks. Mathematically, it is represented as:

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\Lambda}(\boldsymbol{x})\boldsymbol{W}_0 + \bigoplus_{b=1}^{B} \boldsymbol{P}_b\boldsymbol{\Phi}_b(\boldsymbol{x})\boldsymbol{Q}_b^T, \quad (6)$$

where $\bigoplus$ is matrix direct sum, i.e. $\bigoplus_{i=1}^{n} A_i = diag(A_1, \ldots, A_n)$. This form is more general, as Eq. 5 is a special case with $B = 1$. Note that the static kernel $\boldsymbol{W}_0$ is still a full matrix and only the dynamic residual is sparse. We will show later that keeping $\frac{1}{8}$ of entries in the dynamic residual non-zero ($B = 8$) only has slight performance degradation, but still significantly outperforms the static counterpart.

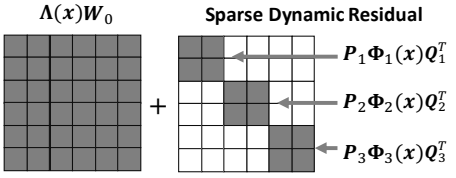


Figure 3: **Sparse dynamic residual**, which is represented as a diagonal block matrix. Each diagonal block is decomposed separately as $\boldsymbol{P}_b\boldsymbol{\Phi}_b\boldsymbol{Q}_b^T$. Note that the static kernel $\boldsymbol{W}_0$ is still a full size matrix.

4.2 DCD OF $k \times k$ DEPTHWISE CONVOLUTION

The weights of a $k \times k$ depthwise convolution kernel form a $C \times k^2$ matrix. Its dynamic decomposition can be easily extended from Eq. 5 by replacing matrix $\boldsymbol{Q}$ (that squeezes the number of channels) with matrix $\boldsymbol{R}$ (that squeezes the number of kernel elements) as follows:

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\Lambda}(\boldsymbol{x})\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{R}^T, \quad (7)$$

where $\boldsymbol{W}(\boldsymbol{x})$ and $\boldsymbol{W_0}$ are $C \times k^2$ matrices. $\boldsymbol{\Lambda}(\boldsymbol{x})$ is a diagonal $C \times C$ matrix with channel-wise attention. $\boldsymbol{R}$ is of shape $k^2 \times L_k$, reducing the number of kernel elements from k^2 to L_k. $\boldsymbol{\Phi}(\boldsymbol{x})$ is of shape $L_k \times L_k$ and performs dynamic fusion along L_k latent kernel elements. $\boldsymbol{P}$, with shape $C \times L_k$, is the weight matrix for depthwise convolution over L_k kernel elements. The default value of L_k is $\lfloor k^2/2 \rfloor$. Since depthwise convolution is channel separable, $\boldsymbol{\Phi}(\boldsymbol{x})$ does not fuse channels but fuse L_k latent kernel elements.

4.3 DCD OF $k \times k$ CONVOLUTION

Joint fusion of channels and kernel elements: A $k \times k$ convolution kernel forms a $C \times C \times k^2$ tensor. Its dynamic decomposition is achieved by extending Eq. 5 into a tensor form (see Figure 4):

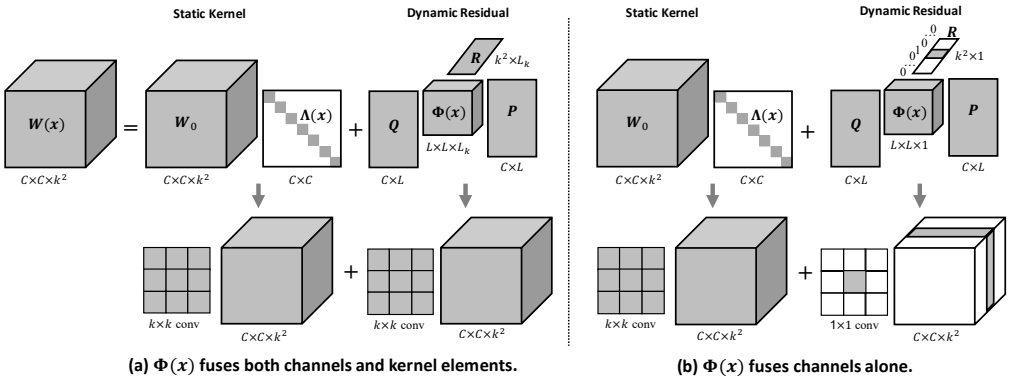
$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{W_0} \times_2 \boldsymbol{\Lambda}(\boldsymbol{x}) + \boldsymbol{\Phi}(\boldsymbol{x}) \times_1 \boldsymbol{Q} \times_2 \boldsymbol{P} \times_3 \boldsymbol{R}, \quad (8)$$

where $\times_n$ refers to n-mode multiplication (Lathauwer et al., 2000), and $\boldsymbol{W_0}$ is of shape $C \times C \times k^2$. $\boldsymbol{\Lambda}(\boldsymbol{x})$ is a diagonal $C \times C$ matrix with channel-wise attention. $\boldsymbol{Q}$ is of shape $C \times L$, reducing the number of input channels from C to L. $\boldsymbol{R}$ is of shape $k^2 \times L_k$, reducing the number of kernel elements from k^2 to L_k. $\boldsymbol{\Phi}(\boldsymbol{x})$ is of shape $L \times L \times L_k$, performing joint fusion of L channels over L_k latent kernel elements. $\boldsymbol{P}$ is of shape $C \times L$, expanding the number of channels from L to C as output. The number of latent channels L and the number of latent kernel elements L_k are constrained as $L_k < k^2$ and $L^2 L_k \leq C$. Their default values are set empirically as $L_k = \lfloor k^2/2 \rfloor$, $L = \lfloor \frac{C/L_k}{2^{\lfloor log_2 \sqrt{C/L_k} \rfloor}} \rfloor$.

Channel fusion alone: We found that the fusion of channels $\boldsymbol{\Phi}(\boldsymbol{x}) \times_1 \boldsymbol{Q}$ is more important than the fusion of kernel elements $\boldsymbol{\Phi}(\boldsymbol{x}) \times_3 \boldsymbol{R}$. Therefore, we reduce L_k to 1 and increase L accordingly. $\boldsymbol{R}$ is simplified as a one-hot vector $[0, \ldots, 0, 1, 0, \ldots, 0]^T$, of which '1' is located at the center (assuming k is an odd number). As illustrated in Figure 4-(b), the tensor of dynamic residual $\boldsymbol{\Phi}(\boldsymbol{x}) \times_1 \boldsymbol{Q} \times_2 \boldsymbol{P} \times_3 \boldsymbol{R}$ only has one non-zero slice, which is equivalent to a 1×1 convolution. Therefore, DCD of $k \times k$ convolution is essentially adding a 1×1 dynamic residual on a static $k \times k$ kernel.

5 EXPERIMENTS

In this section, the experimental results on ImageNet classification Deng et al. (2009) are presented. ImageNet has 1,000 classes and contains 1,281,167 images for training and $50,000$ images for validation. We also report ablation studies to analyze different components of our approach.



(a) $\Phi(x)$ fuses both channels and kernel elements. **(b) $\Phi(x)$ fuses channels alone.**

Figure 4: The dynamic convolution decomposition for $k \times k$ convolution.

We evaluate dynamic convolution decomposition (DCD) on two network architectures: ResNet (He et al., 2016) and MobileNetV2 (Sandler et al., 2018). DCD is used in all convolutional layers in ResNet and 1×1 convolutional layers in MobileNetV2. The reduction ratio r is set to 16 for ResNet and MobileNetV2 $\times 1.0$, and is set to 8 for smaller models (MobileNetV2 $\times 0.5$ and $\times 0.35$). All models are trained by using SGD with momentum 0.9. The batch size is 256. The other training parameters for each model are listed as follows:

ResNet: The learning rate starts from 0.1 and is divided by 10 every 30 epochs. The model is trained with 100 epochs. Dropout (Srivastava et al., 2014) 0.1 is used only for ResNet-50.

MobileNetV2: The initial learning rate is 0.05 and decays to 0 in 300 epochs by a cosine function. The weight decay is 2e-5 and dropout rate is 0.1. For MobileNetV2 $\times 1.0$, Mixup (Zhang et al., 2018a) and label smoothing are further added to avoid overfitting.

5.1 Ablations

We run a number of ablations to analyze dynamic convolution decomposition (DCD) by using MobileNet V2 $\times 0.5$. The default option is using the general form of DCD (Eq. 5) with a full size dynamic residual ($B = 1$) for all pointwise convolution layers. The default dimension of the intermediate space is $L = \lfloor \frac{C}{2^{\lfloor log_2 \sqrt{C} \rfloor}} \rfloor$. The goal is to answer the following two questions:

1. **How** do different components of DCD (e.g. dynamic channel fusion $\boldsymbol{\Phi}(\boldsymbol{x})$, dynamic channel-wise attention $\boldsymbol{\Lambda}(\boldsymbol{x})$, dimension of the intermediate space L) affect performance?
2. **How** do three DCD extensions perform?

Dynamic Channel Fusion $\boldsymbol{\Phi}(\boldsymbol{x})$ and Dynamic Channel-wise Attention $\boldsymbol{\Lambda}(\boldsymbol{x})$: Table 1-(a) shows the top-1 accuracy of different combinations of these two dynamic parts in DCD: $\boldsymbol{\Phi}(\boldsymbol{x})$ and $\boldsymbol{\Lambda}(\boldsymbol{x})$ in Eq. 5. Both parts significantly boost the accuracy (+2.8% and +3.8%) compared to the static baseline. Using the dynamic channel fusion alone ($\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{Q}^T$) has slightly more parameters and FLOPs, and achieves better performance than using the dynamic channel-wise attention alone ($\boldsymbol{\Lambda}\boldsymbol{W}_0$). The combination of the two parts provides additional improvement.

Dimension of the Intermediate Space L: Dynamic channel fusion matrix $\boldsymbol{\Phi}(\boldsymbol{x})$ is of shape $L \times L$ with L^2 dynamic coefficients. Thus, L controls the representation capacity and the parameter complexity of DCD. We adjust it by using different multipliers on the default value of L. Table 1-(b) shows the results of four different multipliers of L from $\times 1.0$ to $\times 0.25$. As L decreases, fewer parameters are required and the performance is degraded slowly. Even with a very low dimensional intermediate space $L \times 0.25$, our method still gains 3.3% top-1 accuracy over the static baseline.

Number of Diagonal Blocks in the Dynamic Residual B: Table 1-(c) shows the classification results for four different values of B. The dynamic residual is a full matrix when $B = 1$, while only $\frac{1}{8}$ of its entries are non-zero when $B = 8$. The accuracy degrades slowly as the dynamic residual becomes sparser with increasing B. The biggest performance drop happens when changing B from 1 to 2, as half of the weight matrix $\boldsymbol{W}(\boldsymbol{x})$ becomes static. However, the performance is still

Table 1: **Ablations of dynamic convolution decompostion (DCD)** evaluated on ImageNet classification. MobileNetV2 $\times 0.5$ is used.

Model	Params	MAdds	Top-1
$\boldsymbol{W}_0$ (static)	2.0M	97.0M	65.4
$\boldsymbol{\Lambda}\boldsymbol{W}_0$	2.4M	97.4M	68.2
$\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{Q}^T$	2.7M	104.4M	69.2
$\boldsymbol{\Lambda}\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{Q}^T$	2.9M	104.6M	**69.8**

(a) Dynamic channel fusion $\boldsymbol{\Phi}(\boldsymbol{x})$ and dynamic channel-wise attention $\boldsymbol{\Lambda}(\boldsymbol{x})$.

Model	L	Params	MAdds	Top-1
static	-	2.0M	97.0M	65.4
DCD	$\times 0.25$	2.4M	99.8M	68.7
	$\times 0.50$	2.5M	101.3M	69.0
	$\times 0.75$	2.6M	102.9M	69.6
	$\times 1.0$	2.9M	104.6M	**69.8**

(b) Dimension of the intermediate space L.

Network	B	Params	MAdds	Top-1
$\boldsymbol{W}_0$ (static)	-	2.0M	97.0M	65.4
$\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{Q}^T$	1	2.7M	104.4M	**69.2**
	2	2.6M	101.0M	68.5
	4	2.5M	99.1M	68.4
	8	2.5M	98.5M	68.3
$\boldsymbol{\Lambda}\boldsymbol{W}_0 + \boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{Q}^T$	1	2.9M	104.6M	**69.8**
	2	2.8M	101.3M	68.9
	4	2.7M	99.4M	68.8
	8	2.7M	98.8M	68.5

(c) Number of diagonal blocks B in the dynamic residual.

DW	PW	CLS	Params	MAdds	Top-1
			2.0M	97.0M	65.4
✓			2.4M	97.5M	68.3
	✓		2.9M	104.6M	69.8
		✓	2.2M	97.2M	66.6
✓		✓	2.6M	97.7M	69.0
✓	✓		3.3M	105.1M	69.6
	✓	✓	3.1M	104.8M	**70.2**
✓	✓	✓	3.5M	105.3M	70.0

(d) DCD at different layers. DW, PW, and CLS indicate depthwise convolution, pointwise convolution and classifier respectively.

Table 2: Comparing DCD with the vanilla dynamic convolution CondConv (Yang et al., 2019) and DY-Conv (Chen et al., 2020c). ✶indicates the dynamic model with the fewest parameters (static model is not included). CondConv contains $K = 8$ kernels and DY-Conv contains $K = 4$ kernels.

Width	Model	Params	MAdds	Top-1
$\times 1.0$	static	3.5M	300.0M	72.0
	DY-Conv	11.1M	312.9M	**75.2**
	CondConv	27.5M	329.0M	74.6
	DCD (ours)	✶5.5M	326.0M	**75.2**
$\times 0.5$	static	2.0M	97.0M	65.4
	DY-Conv	4.0M	101.4M	69.9
	CondConv	15.5M	113.0M	68.4
	DCD (ours)	✶3.1M	104.8M	**70.2**
$\times 0.35$	static	1.7M	59.2M	60.3
	DY-Conv	2.8M	62.0M	65.9
	DCD (ours)	✶2.3M	63.1M	**66.6**

(a) **MobileNetV2**.

Depth	Model	Params	MAdds	Top-1
ResNet-50	static	23.5M	3.8G	76.2
	DCD (ours)	30.7M	3.9G	**77.9**
ResNet-18	static	11.1M	1.81G	70.4
	DY-Conv	42.7M	1.85G	72.7
	DCD (ours)	✶14.0M	1.83G	**73.1**
ResNet-10	static	5.2M	0.89G	63.5
	DY-Conv	18.6M	0.91G	67.7
	DCD (ours)	✶6.5M	0.90G	**68.8**

(b) **ResNet**.

significantly better than the static baseline. Even when choosing $B = 8$, it still outperforms the static baseline by 2.9% (from 65.4% to 68.3%), demonstrating the representation power of the dynamic residual. Dynamic channel-wise attention $\boldsymbol{\Lambda}(\boldsymbol{x})$ provides additional improvement consistently.

DCD at Different Layers: Table 1-(d) shows the results for using DCD at three different types of layers (a) DW: depthwise convolution (Eq. 7), (b) PW: pointwise convolution (Eq. 5), and (c) CLS: fully connected classifier, which is a special case of pointwise convolution (the input resolution is 1×1). Using DCD in any type of layer boosts the performance compared to the static baseline (+2.9% for depthwise convolution, +4.4% for pointwise convolution, and +1.2% for classifier). Combining DCD in both pointwise convolution and classifier achieves the best performance (+4.8%). We notice a performance drop (from 70.2% to 70.0%) when using DCD in all three types of layers. We believe this is due to overfitting, as it has higher training accuracy.

Extension to 3×3 Convolution: We use ResNet-18 to study the extension of DCD in 3×3 convolution (see Section 4.3), as it stacks 16 layers of 3×3 convolution. Compared to the static baseline (70.4% top-1 accuracy), DCD with *joint fusion of channels and kernel elements* (Eq. 8)

gains 0.9% top-1 accuracy (71.3%). The top-1 accuracy is further improved by 1.8% (73.1%), when using DCD with *channel fusion alone* that transforms the dynamic residual as a 1×1 convolution matrix (see Figure 4-(b)). This demonstrates that dynamic fusion is more effective across channels than across kernel elements.

Summary: Based on the ablations above, the recommended setting is to use both dynamic channel fusion $\mathbf{\Phi}$ and dynamic channel-wise attention $\mathbf{\Lambda}$ with default dimension of the intermediate space L and a full size residual $B = 1$. DCD is recommended for pointwise convolution and classifier in MobileNetV2. When dealing with 3×3 convolution in ResNet, DCD with channel fusion alone is recommended. If slight performance drop is allowed, the model can be more compact by: (a) removing dynamic channel-wise attention $\mathbf{\Lambda}$, (b) reducing dimension of the intermediate space L, (c) making dynamic residual sparser (increasing B), (d) using DCD in depthwise convolution alone.

5.2 Main Results

We compare dynamic convolution decomposition (DCD) with the vanilla dynamic convolution (Yang et al., 2019; Chen et al., 2020c) in MobileNetV2 and ResNet. The recommended setting in the previous section is used. The results are shown in Table 2. DCD significantly reduces the number of parameters with even better performance for both network architectures. For instance in MobileNetV2 $\times 1.0$, DCD only requires 50% of parameters in Chen et al. (2020c) and 25% of parameters in Yang et al. (2019). For ResNet-18, DCD only requires 33% of parameters in Chen et al. (2020c) with 0.4% gain in top-1 accuracy. Although our method requires slightly more MAdds than Chen et al. (2020c), the increment is negligible. These results demonstate that our method is more compact and effective.

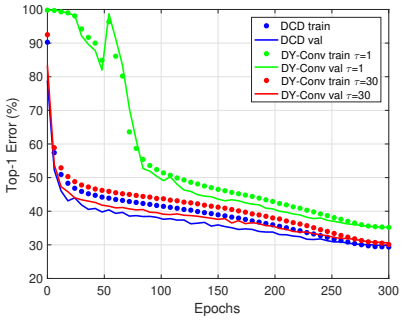


Figure 5: The comparison of training and validation error between DCD and DY-Conv on MobileNetV2 $\times 0.5$. τ is the temperature in softmax. Best viewed in color.

Figure 5 compares DCD with DY-Conv (Chen et al., 2020c) on training convergence. DY-Conv uses a large temperature in softmax to alleviate the joint optimization difficulty and make training more efficient. Without using any additional parameter tuning, our DCD converges even faster than DY-Conv with a large temperature and achieves higher accuracy.

5.3 Analysis of Dynamic Channel Fusion

We check how coefficients in dynamic channel fusion $\mathbf{\Phi}(\boldsymbol{x})$ evolve over layers. Different values of $\mathbf{\Phi}(\boldsymbol{x})$ over different images are expected to validate the *dynamic* property. We measure this by averaging the variance of each entry $\sigma_\Phi = \sum_{i,j} \sigma_{i,j}/L^2$, where $\sigma_{i,j}$ is the variance of $\phi_{i,j}(\boldsymbol{x})$ over all validation images. To compare σ_Φ across layers, we normalize it by the variance of the corresponding input feature map. Figure 6 shows the normalized variance σ_Φ across layers in MobileNetV2. Clearly, the dynamic coefficients have more variations at higher layers. We believe this is because the higher layers encode more context information, providing more clues to adapt convolution weights.

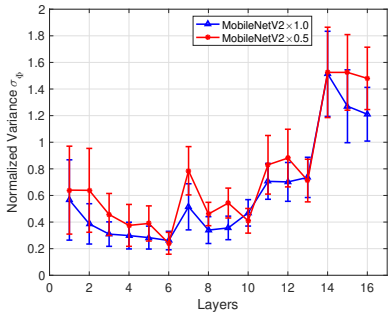


Figure 6: Normalized variance of dynamic coefficients σ_Φ across layers in MobileNetV2 $\times 0.5$ and $\times 1.0$.

6 Conclusion

In this paper, we revisit dynamic convolution via matrix decomposition and reveal that dynamic attention over channel groups is the key issue. It requires K times more parameters, and introduces joint optimization difficulty. We present dynamic convolution decomposition to address these issues. It applies dynamic channel fusion to enable significant dimension reduction of the intermediate space, resulting in a more compact model that is easier to learn with often improved accuracy. We hope our work provides deeper understanding to the recent work on dynamic convolution.

REFERENCES

Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *ArXiv*, abs/1908.09791, 2019.

Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020a.

Jyun-Ruei Chen, Xijun Wang, Zichao Guo, X. Zhang, and J. Sun. Dynamic region-aware convolution. *ArXiv*, abs/2003.12243, 2020b.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020c.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. *arXiv preprint arXiv:2003.10027*, abs/2003.10027, 2020d.

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

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

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. *ICLR*, 2017.

Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019. URL `http://arxiv.org/abs/1905.02244`.

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.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *The IEEE International Conference on Computer Vision (ICCV)*, 2009.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

Jean Kossaifi, Antoine Toisoul, Adrian Bulat, Yannis Panagakis, Timothy M Hospedales, and Maja Pantic. Factorized higher-order cnns with an application to spatio-temporal emotion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6060–6069, 2020.

L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. In *SIAM J. Matrix Anal. Appl*, 2000.

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

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 (CVPR)*, June 2019.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Ningning Ma, X. Zhang, J. Huang, and J. Sun. Weightnet: Revisiting the design space of weight networks. volume abs/2007.11823, 2020.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavsky, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. *arXiv preprint arXiv:2008.05441*, 2020.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Zhuo Su, Linpu Fang, Wen xiong Kang, D. Hu, M. Pietikäinen, and Li Liu. Dynamic group convolution for accelerating convolutional neural networks. In *ECCV*, August 2020.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pp. 6105–6114, Long Beach, California, USA, 09–15 Jun 2019a.

Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. In *30th British Machine Vision Conference 2019*, 2019b.

Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, August 2020.

Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1gMCsAqY7`.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018a. URL `https://openreview.net/forum?id=r1Ddp1-Rb`.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018b.

Daquan Zhou, Qi-Bin Hou, Y. Chen, Jiashi Feng, and S. Yan. Rethinking bottleneck structure for efficient mobile network design. In *ECCV*, August 2020.