Parameter Efficient Dynamic Convolution via Tensor Decomposition

Zejiang Hou
zejiangh@princeton.edu
Sun-Yuan Kung
kung@princeton.edu

Department of Electrical Engineering
Princeton University
Princeton, NJ, USA

Abstract

Dynamic convolution has demonstrated substantial performance improvements for convolutional neural networks. Previous aggregation based dynamic convolution methods are challenged by the parameter/memory inefficiency, and the learning difficulty due to the scalar type attention for aggregation. To rectify these limitations, we propose a parameter efficient dynamic convolution operator (dubbed as PEDConv) that learns to discriminatively perturb the spatial, input and output filters of a shared base convolution weight, through a tensor decomposition based input-dependent reparameterization. Our method considerably reduces the number of parameters compared to prior arts and limit the computational cost to maintain inference efficiency. Meanwhile, the proposed PEDConv significantly boosts the accuracy when substituting standard convolutions on a plethora of prevalent deep learning tasks, including ImageNet classification, COCO object detection, ADE20K semantic segmentation, and adversarial robustness. For example, on ImageNet classification, PEDConv applied to ResNet-50 achieves 80.5% Top-1 accuracy at almost the same computation cost as static convolutional baseline, improving previous best dynamic convolution method by 1.9% accuracy. Moreover, the proposed method can be readily extended to both input and spatial dynamic regime with adaptive reparameterization at different spatial locations, in which case ResNet-50 achieves 79.3% Top-1 accuracy while reducing 44% FLOPs compared to the baseline model.

1 Introduction

Deep convolutional neural networks have made significant progress in a wide range of computer vision tasks including image classification, object detection, and semantic segmentation. The powerful representation ability of CNNs stems from that different convolution kernels are responsible for extracting diverse information encoded across channels and layers. However, current design of convolutional layers applies the same convolutional weight to process different input images. This deprives convolution of the ability to adapt to diverse visual patterns with respect to different images. Consequently, more effective information can only be captured when increasing the capacity of the network (e.g., adding more convolutional layers and increasing the kernel or channel size). The best-performing CNNs usually involve high computational cost and large memory footprint, impeding their deployment on environments with strict latency requirements and limited computing resources.

Different from static convolutions, to adapt to the diverse visual features, recent studies focus on using input-dependent convolution weights to process different inputs, i.e., dynamic convolution. Prior arts [4, 35] utilize a set of K parallel convolution weights $\{\mathbf{W}_k\}$ instead of a single static one. These weights are linearly combined $\mathbf{W}(\mathbf{x}) = \sum_k \alpha_k(\mathbf{x}) \cdot \mathbf{W}_k$ for each individual input $\mathbf{x}$ via the input-dependent attention $\alpha(\mathbf{x})$, so that the model capacity can be increased while maintaining the inference efficiency. However, aggregation based dynamic convolutions have several limitations: (1) The use of K (K can be up to 32 in [35]) convolution weights brings in considerable parameter and model memory overheads; (2) the over-parameterized network is prone to overfitting, thus requires meticulously designed training techniques [35]; (3) the scalar type attention α_k multiplied to the convolution weight $\mathbf{W}_k$ imposes optimization challenge: if the attention value is too small, the corresponding weight will be insufficiently trained. Hence, existing method [4] applies softmax normalization with very large temperature to the attention values for training dynamic convolutions.

Our contributions. To rectify the above limitations, we propose a novel parameter efficient dynamic convolution, PEDConv, which dynamically reparameterizes the weights of each different filters and kernels w.r.t. different inputs. The core component of PEDConv is a Canonical Polyadic tensor decomposition based reparameterization applied to the base weight, where the low-rank decomposition components are made conditioned on the input. In this way, PEDConv can capture both generic information shared among inputs in base weight optimization, and input specification in learning the dynamic reparameterization. In order to learn to generate the decomposition components, we propose to learn a conditional distribution w.r.t. the input, in conjunction with mutual information maximization between the generated components and input to enforce the input-dependency. PEDConv can be readily applied to substitute standard convolutions in diverse CNN architectures including ResNet, MobileNet, EfficientNet with negligible computation cost increase. To demonstrate the effectiveness and generality of PEDConv, we experiment with a wide variety of deep learning tasks including ImageNet classification, COCO object detection, ADE20K semantic segmentation, and adversarial robustness. PEDConv achieves superior accuracy while requiring fewer parameters and FLOPs compared to the state-of-the-art methods.

2 Related Works

This section mainly covers the spectrum of related works on dynamic neural networks, and we clarify the difference of PEDConv with previous dynamic convolution methods. Benefiting from the data dependency mechanism, dynamic neural networks can flexibly adapt their parameters to match the diverse visual patterns and boost the representation ability. [8] introduces a hyper-network to generate the parameters for the main network. Similarly, [29] uses auxiliary network to dynamically generate the convolution filters for the mask head in instance segmentation task. [11] re-weights different channel-wise feature-maps based on the global context for each block. [15] adaptively adjusts each channel's receptive field size based on softmax attention guided by multiple scales of input information. [5] adapts the slopes and intercepts of two linear functions in ReLU activation. [27] proposes dynamic group convolution that adaptively selects input channels to be connected within each group. [13] dynamically adjusts the filter numbers of the network with respect to different inputs. [7] directly re-samples from the original kernel space to adapt the effective receptive field for handling object deformations. For neural machine translation, [33] predicts separate convolution kernels based on the current time-step in order to determine the importance of context elements. Dynamic convolutions in [4, 35] aggregate multiple convolution kernels based on the input-dependent attention values. [20] uses grouped fully-connected layers to generate dynamic convolution weights directly. [16] proposes dynamic channel fusion to learn additive off-set matrices together with channel-wise attention. [3, 14, 40] propose spatial-specific dynamic convolution by making the filters adaptive to different spatial regions or locations. We highlight the difference of our method: (1) PEDConv entails a Canonical Polyadic tensor decomposition [30, 38] based reparameterization that captures both generic information (base weights are shared across inputs) and input specification (decomposition components varies for different inputs); (2) we propose to learn the conditional distribution with mutual information maximization, compared to deterministic weight generation in prior arts [4, 35], (3) PEDConv demonstrates better parameter-efficiency and superior accuracy compared to prior arts [4, 35].

3 PEDConv Methodology

The goal of this work is to design a dynamic convolution which dynamically reparameterizes the weights of each different filters and kernel positions w.r.t. different inputs, while being more parameter-efficient and less memory demanding. To this end, we reparameterize the base convolution weights as $\mathcal{A}(\mathbf{x}) \odot \mathbf{W}_{\text{base}}$, where each element of $\mathcal{A}(\mathbf{x}) \in \mathbb{R}^{C_{out} \times C_{in} \times k \times k}$ is dependent on input $\mathbf{x}$ and $\odot$ represents the Hadamard product. This formulation can be viewed as leveraging "slow" weights $\mathbf{W}_{\text{base}}$ that are shared across inputs and capture general information, and "fast" weights $\mathcal{A}(\mathbf{x})$ that learn specific information for adaptation to each individual input. Moreover, in our formulation, the direction and amount of perturbation for each element of the base convolution weight can vary per-input basis, instead of being simply distinct up to a scaling factor across different inputs when one uses scalar type attention. However, generating $\mathcal{A}(\mathbf{x})$ for each input can introduce considerable parameters and computation overheads, since $\mathcal{A}(\mathbf{x})$ has the same tensor shape as the convolution weight. To circumvent this problem, we propose a tensor decomposition based reparameterization for parameter-efficient dynamic convolution (PEDConv), which is formulated as:

$$\mathbf{W}(\mathbf{x}) = \mathbf{W}_{base} \times_1 \texttt{diag}(\gamma(\mathbf{x})) \times_2 \texttt{diag}(\phi(\mathbf{x})) \times_3 \texttt{diag}(\psi(\mathbf{x})) \tag{1}$$

where $\gamma(\mathbf{x}) \in \mathbb{R}^{C_{out}}$, $\phi(\mathbf{x}) \in \mathbb{R}^{C_{in}}$, $\psi(\mathbf{x}) \in \mathbb{R}^{k^2}$, $\mathbf{W}_{base} \in \mathbb{R}^{C_{out} \times C_{in} \times k \times k}$. The operator $\times_i$ denotes the i-mode product between a tensor and a matrix. $\texttt{diag}(\cdot)$ converts vector to diagonal matrix. Note that $\mathbf{W}(\mathbf{x})$ represents convolution weights dependent or conditioned on $\mathbf{x}$.

Eq.(1) performs the rank-1 decomposition of $\mathcal{A}(\mathbf{x})$ with three components $\gamma(\mathbf{x})$, $\phi(\mathbf{x})$, and $\psi(\mathbf{x})$, capturing the input dynamics for the input-side filters, output filters and kernels, respectively. Formally, the decomposition can be expressed as: $\mathcal{A}(\mathbf{x}) \approx \gamma(\mathbf{x}) \otimes \phi(\mathbf{x}) \otimes \psi(\mathbf{x})$, where $\otimes$ stands for the outer product of vectors. The kernel dimension is not decomposed since the kernel size is usually small in modern CNNs (e.g., 3×3). Therefore, our convolution weight reparameterization can be equivalently computed as the Hadamard product between the input-dependent decomposition and the base weight:

$$\mathbf{W}(\mathbf{x}) = \mathbf{W}_{base} \odot (\gamma(\mathbf{x}) \otimes \phi(\mathbf{x}) \otimes \psi(\mathbf{x})) \tag{2}$$

Due to the tensor decomposition, the parameters needed for generating $\mathcal{A}(\mathbf{x})$ can be greatly reduced. That is, we only need $C_{out} + C_{in} + k^2$ parameters to dynamically perturb each element of a base convolution weight of size $C_{out} \times C_{in} \times k^2$. To summarize, PEDConv with the proposed input-dependent tensor decomposition based reparameterization is given as:

$$Y_{t,w',h'} = \sum_{s=1}^{C_{in}} \sum_{j=1}^{k} \sum_{i=1}^{k} \Big(\gamma(\mathbf{x})_t \phi(\mathbf{x})_s \psi(\mathbf{x})_{ji} [\mathbf{W}_{\text{base}}]_{t,s,j,i} \Big) X_{s,w_j,h_i} \tag{3}$$

where $w_j = (w' - 1)\Delta + j - p$, $h_i = (h' - 1)\Delta + i - p$, Δ is stride and p is zero-padding. Following the standard design in CNN, we use batch normalization and an activation function (e.g., ReLU) after PEDConv when substituting it for static convolutions.

Learning the decomposition components. For notation simplicity, we combine the decomposition components as $\mathbf{g}(\mathbf{x}) = [\gamma(\mathbf{x}); \phi(\mathbf{x}); \psi(\mathbf{x})] \in \mathbb{R}^{C_{out}+C_{in}+k^2}$. To dynamically generate $\mathbf{g}(\mathbf{x})$ w.r.t. different $\mathbf{x}$, we propose to learn the conditional prior distribution denoted by $p(\mathbf{g}|\mathbf{x};\theta,\pi)$, where θ represents the base convolution weights and π refers to the weights of the generator that generates $\mathbf{g}$. In our case of training with the conditional prior, the marginal log-likelihood is obtained as: $\log p(\mathbf{Y}|\mathbf{X};\theta,\pi) = \sum_{i=1}^{N} \log \mathbb{E}_{\mathbf{g}_i \sim p(\mathbf{g}_i|\mathbf{x}_i;\theta,\pi)}[p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{g}_i;\theta)]$. In practice, we consider its lower bound derived by the variational inference method (see supplementary for details), which is given as the expected loss over the conditional prior:

$$\log p(\mathbf{Y}|\mathbf{X};\theta,\pi) \geq \sum_{i=1}^{N} \mathbb{E}_{\mathbf{g}_i \sim p(\mathbf{g}_i|\mathbf{x}_i;\theta,\pi)}[\log p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{g}_i;\theta)] \tag{4}$$

To evaluate the gradient on this lower bound objective w.r.t. θ and π, we use the reparametrization trick introduced by [12]. Specifically, we assume the conditional distribution $p(\mathbf{g}|\mathbf{x};\theta,\pi)$ has the form of Gaussian with mean μ and diagonal covariance $\mathtt{diag}(\sigma^2)$. At each convolution layer, the generator module (denoted by G_π) would take the preceding layer's output feature-maps (denoted by X) as input, and output the distribution parameters (mean μ and standard deviation σ) of the conditional distribution. The actual CP decomposition components $\mathbf{g}$ are sampled from this conditional distribution using the reparameterization trick:

$$[\mu_i, \sigma_i] = G_\pi(\mathbf{X}_i), \;\; \mathbf{g}_i \sim p(\mathbf{g}_i|\mathbf{x}_i;\theta,\pi) = \mathcal{N}(\mu_i, \mathtt{diag}(\sigma_i^2)) = \mu_i + \sigma_i \odot \varepsilon \tag{5}$$

where we use subscript i to emphasize per-input basis. $\varepsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ is for reparameterization trick so that our training objective (i.e., Monte Carlo estimate [12] of the expectation in Eq.(4)) is differentiable w.r.t. θ and π with the presence of sampling operation. Following [25], we perform a deterministic inference without sampling $\mathbf{g} = \mathbb{E}[\mathbf{g}|\mathbf{x}]$ at testing time.

Enforcing input-dependency via info-max. To enforce the input-dependency of PEDConv, we propose to maximize the mutual information (MI) between the generated $\mathbf{g}$ and the input data during training. This MI objective can be described as: $\max I((\mathbf{x}_i,\mathbf{y}_i);\mathbf{g}_i)$. Note that the information maximization occurs only at training time for learning the model parameters, which is why we can incorporate the label information into the MI objective. Based on the chain rule of MI, the objective can be rewritten as: $\max I(\mathbf{x}_i;\mathbf{g}_i) + I(\mathbf{y}_i;\mathbf{g}_i|\mathbf{x}_i)$. Directly computing the MI is intractable since the true posterior distributions are unknown. Therefore, we use Variational Information Maximization [1] to derive a lower bound of the MI objective (see supplementary for details), which is given as:

$$\max_{\theta,\pi} \mathbb{E}[\log p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{g}_i;\theta)] + \lambda \max_{\pi,\theta'} \log p(\mathbf{x}_i|\mathbf{g}_i;\theta')] \tag{6}$$

We omit the constant entropy terms and the subscript of the expectation for clarity.

The first term in Eq.(6) maximizes the log likelihood of label for the training data, which is equivalent to minimizing the cross-entropy loss in classification. This task loss optimizes the base convolution weights and the generator module weights. For the second term, $p(\mathbf{x}_i|\mathbf{g}_i;\theta')$ is the approximated posterior for deriving the variational lower bound of

MI, which is equivalent to the decoding term in the variational auto-encoder [12]. Following [12], we assume this decoding term takes the form of Gaussian. A known fact is that maximizing the log likelihood of Gaussian is equivalent to minimizing the MSE reconstruction loss. Specifically, we instantiate the decoding term $p(\mathbf{x}_i|\mathbf{g}_i;\theta')$ by a decoder MLP with weights θ'. This decoder MLP has two fully-connected layers with ReLU in-between, and it only appears at training time for MI maximization. The input of the decoder MLP is $\mathbf{g}_i$, which is mapped to the reconstructed GAP feature by the decoder MLP, because the GAP feature is the direct input to $p(\mathbf{g}_i|\mathbf{x}_i;\theta,\pi)$ (detailed out in following section on generator module design). The intuition is that the generated (more concretely, the sampled) CP decomposition components $\mathbf{g}_i$ should be able to reconstruct the inputs of the generator (which is the GAP feature) in order to achieve better input-dependency. During training, in addition to the cross-entropy loss, we also minimize the MSE loss between the reconstructed GAP feature and the original GAP feature. This MSE loss optimizes the generator module weights and decoder MLP weights. With this training objective, the generated CP decomposition components for PEDConv will be able to carry the information about each specific input data while optimizing the task loss.

Generator module design. The input-dependent CP decomposition components $\mathbf{g}(\mathbf{x})$ are generated based on the input feature-maps $\mathbf{X}$ for the corresponding layer (equivalently, the output feature-maps from preceding layer). To make the generator compact and efficient, we employ a bottleneck two-layer MLP. Firstly, global average pooling (GAP) is applied to obtain the global spatial information of the input feature-maps. Then, two fully connected layers with a ReLU activation function in-between are applied to generate the distribution parameters for the conditional distribution in Eq.(5):

$$[\mu,\sigma] = G_\pi(\mathbf{X}) = \mathbf{W}_{\text{fc1}} \cdot (\mathbf{W}_{\text{fc2}} \cdot \frac{1}{HW} \sum_{i\in H, j\in W} \mathbf{X}_{c,i,j})_+ \tag{7}$$

We denote the two-layer MLP generator by $G_\pi(\cdot)$ with weights π, that takes as input the GAP feature of corresponding layer, and output mean μ and std σ of $p(\mathbf{g}|\mathbf{x};\theta,\pi)$. π represents the learnable parameters $\mathbf{W}_{\text{fc1}} \in \mathbb{R}^{(C_{out}+C_{in}+k^2)\times C_{in}/r}$, $\mathbf{W}_{\text{fc2}} \in \mathbb{R}^{C_{in}/r\times C_{in}}$, where r here is the reduction factor in the bottleneck MLP. $\mathbf{X} \in \mathbb{R}^{C_{in}\times H\times W}$ denotes the input feature-maps, and $(\cdot)_+$ is ReLU activation. After computing the vector $\mathbf{g}(\mathbf{x})$ by Eq.(5),(7), we can obtain the three CP decomposition components $\gamma(\mathbf{x}), \phi(\mathbf{x}), \psi(\mathbf{x})$ in Eq.(1) by chunking $\mathbf{g}(\mathbf{x})$.

Complexity analysis. Given that the spatial dimension is reduced for generating the input-dependent decomposition components, the computation cost of PEDConv mainly comes from the convolution operation. Specifically, the FLOPs complexity of PEDConv can be calculated by $(HWC_{out}C_{in}k^2) + ((2C_{in}+C_{out}+k^2)C_{in}/r + 2C_{out}C_{in}k^2)$, where the first term counts the convolution operation while the second term counts the decomposition components generator and application. The second term has much less computation compared to the convolution operation, making PEDConv almost as efficient as the static convolution. In terms of parameter complexity, static convolution and vanilla dynamic convolution [4, 35] require $C_{out}C_{in}k^2$ and $KC_{out}C_{in}k^2$ ($K \geq 4$) parameters, respectively. In contrast, PEDConv requires $C_{out}C_{in}k^2$ for the base convolution weights, and an additional $(2C_{in}+C_{out}+k^2)C_{in}/r$ parameters are required by the bottleneck MLP for generating the decomposition components. Since the second term is relatively negligible, the proposed PEDConv has much less parameter complexity than the aggregation based dynamic convolutions that linearly combines multiple weights with scalar type attentions.

Generalized rank-*R* decomposition. Eq.(2) gives a special case of rank-1 decomposition of $\mathcal{A}(\mathbf{x})$. A generalized rank-R Canonical Polyadic (CP) tensor decomposition can be expressed as the sum of R rank-1 tensors that are formulated as the outer product of rank-1 components: $\mathcal{A}(\mathbf{x}) \approx \sum_{r=1}^{R} \gamma_r(\mathbf{x}) \otimes \phi_r(\mathbf{x}) \otimes \psi_r(\mathbf{x})$. R influences the parameter efficiency, where the smaller the R is, the more reduced the parameters. PEDConv with the rank-R CP decomposition is given as: $\mathbf{W}(\mathbf{x}) = \sum_{r=1}^{R} \mathbf{W}_{\text{base}} \times_1 \mathtt{diag}(\gamma_r(\mathbf{x})) \times_2 \mathtt{diag}(\phi_r(\mathbf{x})) \times_3 \mathtt{diag}(\psi_r(\mathbf{x}))$.

4 Experiments

To verify the efficacy and generality of PEDConv, we evaluate on a variety of deep learning tasks with diverse CNN architectures. Our method is implemented in PyTorch. All experiments are run on NVIDIA Tesla V100 GPUs. In terms of implementation, we use rank-1 CP decomposition for PEDConv in Eq.(1). We set the reduction factor in the bottleneck MLP to $r = 16$, and the loss balance factor in Eq.(6) to $\lambda = 1e^{-3}$. To keep our method simple and generic, these hyper-parameters are kept constant for all experiments. For other training hyper-parameters, we use the default settings and specify them in following subsections.

4.1 Image classification

We evaluate our method on ImageNet [6], which has 1000 classes and consists of 1.28M training images and 50K validation images. PED convolution is embedded into diverse advanced CNN architectures, including ResNet-18/50, MobileNetV1/V2, and EfficientNet, to substitute all standard convolution layers except the first layer. All models are trained by the SGD optimizer with a momentum of 0.9 and a batch-size of 512. For training ResNets, we set the weight decay to 1e-4. The initial learning rate is set to 0.512, and decays by a cosine scheduler for total 120 epochs. For training MobileNets, we set the weight decay to 4e-5. The initial learning rate is set to 0.1 and decays by cosine scheduler for total 250 epochs. For training EfficientNet, we follow the same procedure as [28]. We evaluate single-crop Top-1 accuracy, and use input resolution 224×224 for both training and testing. Standard data augmentations described in PyTorch official repository are adopted.

The results are reported in Table 1, including comparisons with state-of-the-art methods. As shown, with comparable FLOPs and parameters, PEDConv consistently boosts the accuracy over baseline models. For example, PEDConv-MobileNetV2/ResNet-50 acheives 3.5%/4.3% accuracy gain over MobileNetV2/ResNet-50 with only 2% extra FLOPs, respectively. Moreover, PEDConv requires noticeably fewer model parameters compared with the state-of-the-art dynamic convolutions. For MobileNetV2, PEDConv only requires 43% of the parameters of DY-Conv and 17% of the parameters of CondConv, while achieving better Top-1 accuracy with less FLOPs. For ResNet-18/50, PEDConv only requires 28% of the parameters of DY-Conv/CondConv, while improving the accuracy by 1.5%/1.9%, respectively. In addition, on more powerful NAS-based mobile network EfficientNet-B0, our method still achieves 1.1% accuracy gain over the strong baseline. These results demonstrate the effectiveness, efficiency and compactness of our method.

4.2 Object detection

We further verify the efficacy and generality of PEDConv applied to SSD [19] on COCO2017 [17] object detection. Following [19], we train SSD on the COCO train2017 split containing

Model	Method	Params	FLOPs	Top-1 (%)
MobileNetV1	Baseline	4.2M	569M	71.9
	DSNet [13]	-	565M	74.5
	DR-Conv [3]	-	610M	75.5
	CondConv [35]	33.0M	600M	73.7
	Ours	5.1M	579M	**75.9**
MobileNetV2	Baseline	3.5M	300M	72.0
	DK [7]	11.1M	760M	74.8
	CA [9]	4.0M	310M	74.3
	DCD [16]	5.5M	326M	75.2
	DY-Conv [4]	11.1M	313M	75.2
	CondConv [35]	27.5M	329M	74.6
	Ours	4.8M	307M	**75.5**
EfficientNet-B0	Baseline	5.3M	391M	77.1
	CA [9]	5.4M	400M	76.9
	CondConv [35]	13.3M	402M	77.8
	Ours	7.2M	400M	**78.2**

(a) Light-weight CNN models.

Model	Method	Params	FLOPs	Top-1 (%)
ResNet-18	Baseline	11.7M	1.81G	70.4
	DCD [16]	14.0M	1.83G	73.1
	DY-Conv [4]	42.7M	1.85G	72.7
	Ours	11.9M	1.83G	**74.2**
ResNet-50	Baseline	25.6M	4.1G	76.2
	DK [7]	81.2M	4.6G	78.5
	SK [15]	27.5M	4.5G	79.2
	DCD [16]	30.7M	4.2G	77.9
	WeightNet [20]	31.1M	4.2G	77.5
	CondConv [35]	104.8M	4.2G	78.6
	Ours	28.6M	4.2G	**80.5**

(b) Non-compact CNN models.

Table 1: Results of comparing PEDConv with state-of-the-arts on ImageNet. PEDConv outperforms previous dynamic convolutions [4, 35] in terms of higher accuracy and fewer parameters/FLOPs.

Detector	Backbone	Params	FLOPs	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{S}	AP^{bbox}_{M}	AP^{bbox}_{L}
SSD300	ResNet-50	22.9M	20.2G	25.2	42.7	25.8	7.3	27.1	40.8
	PEDConv-ResNet-50	24.4M	20.3G	29.3	48.2	30.4	9.4	31.9	47.9

Table 2: Results of PEDConv applied to SSD300 on COCO object detection.

about 118k images and test on the val2017 split containing 5k images. The input size is fixed to 300×300. We adopt SGD optimizer with a momentum of 0.9, a batch-size of 64, and a weight decay of 5e-4. The model is trained with a learning rate of 1e-3 for 160k iterations, which decays by 10 and 100 times, and continue to train for 40k iterations each.

The results of comparing PEDConv against standard convolution are shown in Table 2. As show, PEDConv achieves considerable performance gain across all evaluation metrics, e.g., 4.1% higher in bounding box AP with only 6% parameters overhead and nearly the same computational cost. In the taxonomy of different AP scores for small/medium/large objects, we notice that the most compelling improvement appears in AP_L, where PEDConv surpasses the baseline by 7%. We conjecture that this success may arise from the power of PEDConv in adapting the convolution filters to diverse visual patterns and prioritizing the most informative elements w.r.t. different inputs.

4.3 Semantic segmentation

We also conduct experiments on the semantic segmentation. We employ the segmentation frameworks UperNet [34] and PSPNet [36] to validate PEDConv on the ADE20K dataset [39], which contains around 25K images for training and 2K images for testing. We follow the same training procedure as [34, 36] to train UperNet and PSPNet. The performance of the models are evaluated by the mean of the Intersection over Union (IoU) averaged over all the 150 semantic categories in the testing set. As shown in Table 3, PEDConv consistently improves mean IoU over the baseline models for both segmentation frameworks, achieving 2.4% and 1.3% gains for UperNet and PSPNet, respectively.

4.4 Adversarial robustness

We further investigate the robustness of PEDConv models to adversarially perturbed images (i.e., adversarial examples). We follow the adversarial training method [24] to train robust

Segmentor	Backbone	Params	FLOPs	Mean IoU (%)
UperNet	ResNet-50	64.1M	156.2G	40.4
	PEDConv-ResNet-50	72.3M	156.3G	42.8
PSPNet	Dilated ResNet-50	51.5M	186.8G	41.3
	Dilated PEDConv-ResNet-50	59.7M	186.9G	42.6

Table 3: Results of PEDConv applied to UperNet and PSPNET on ADE20K semantic segmentation.

Training	Model	Method	Params	FLOPs	Evaluated Against		
					Natural Images (%)	PGD-10 (%)	PGD-50 (%)
Adversarial (Free $m = 4$ [24])	ResNet-50	Baseline	25.6M	4.1G	60.2	32.8	31.9
		PEDConv	28.6M	4.2G	64.2	35.6	34.8

Table 4: Validation accuracy and robustness of ResNet-50 with standard convolution and the proposed PEDConv. Both models are trained with adversarial training to resist ℓ_∞ $\varepsilon = 4$ attacks.

models for the large-scale ImageNet classification, given that adversarial training is one of the few defenses against adversarial attacks that withstands strong attacks. Table 4 summarizes the results for robust models with PEDConv defending the PGD-10/50 attacks with $\ell_\infty \varepsilon = 4$. As shown, PEDConv improves both the natural accuracy and the robustness to PGD attacks significantly compared with standard convolution models.

To explain the improvement of robustness, we conjecture that PEDConv enhances the model capacity since it can adapt different convolution weights to classify different input images. Our results in Table 1(b) show that PEDConv-ResNet-50 can achieve superior accuracy compared to much larger capacity ResNets. It has been studied in [21] that model capacity is a crucial factor for the robustness and the ability to train against strong adversaries in adversarial training, since separating adversarial examples would require more complicated decision boundary. Our empirical results suggest that dynamic convolution by PEDConv may be another efficient way to enhance model capacity and adversarial robustness.

4.5 Ablation study

We analyze the effect of different components in PEDConv via ablation study.

Different formulations. Table 5(a) summarizes the comparison of the default formulation of PEDConv in Eq.(1) versus three variants on ImageNet using ResNet-18/50. $\gamma(\mathbf{x})$ denotes perturbing the output-side filters only $\mathbf{W}(\mathbf{x}) = \mathbf{W}_{\text{base}} \times_1 \gamma(\mathbf{x})$. $\phi(\mathbf{x})$ denotes perturbing the input-side filters only $\mathbf{W}(\mathbf{x}) = \mathbf{W}_{\text{base}} \times_2 \phi(\mathbf{x})$. $\phi(\mathbf{x})$ denotes perturbing the kernel dimensions only $\mathbf{W}(\mathbf{x}) = \mathbf{W}_{\text{base}} \times_3 \psi(\mathbf{x})$. As observed, all three variants enhance the accuracy compared with static convolution. Perturbing the kernel dimensions only yields the best parameter efficiency, due to the small kernel sizes used in modern CNNs. The combination of three components brings in additional accuracy improvement so that PEDConv achieves the SOTA accuracy. These results necessitate our proposed fine-grained dynamics for each filter and kernel weights w.r.t. different inputs. Moreover, we also study the effect of learning the conditional distribution with Eq.(5) versus deterministic generation of the decomposition components. We find that incorporating stochasticity improve the accuracy noticeably, i.e., 0.5%/1% gain on ResNet-18/50, respectively.

Decomposition rank. We investigate the impact of tensor decomposition rank on ImageNet using ResNet-18. As shown in Table 5(b), increasing the rank value improves the accuracy, at the cost of slightly more model parameters and computation cost.

PEDConv at different layers. Table 5(c) reports the results on substituting PEDConv for different types of convolution layers (PW: pointwise 1×1 convolution, DW: depthwise con-

$\gamma(\mathbf{x})$	$\phi(\mathbf{x})$	$\psi(\mathbf{x})$	Variational	ResNet-18 Params.	ResNet-18 FLOPs	ResNet-18 Top-1 (%)	ResNet-50 Params.	ResNet-50 FLOPs	ResNet-50 Top-1 (%)
				11.7M	1.8G	70.4	25.6M	4.1G	77.0
✓				11.8M	1.8G	72.7	27.9M	4.1G	79.1
	✓			11.8M	1.8G	73.0	27.4M	4.1G	78.7
		✓		11.7M	1.8G	73.1	26.7M	4.1G	78.3
✓	✓	✓		11.9M	1.8G	73.7	28.6M	4.2G	79.5
✓	✓	✓	✓	11.9M	1.8G	74.2	28.6M	4.2G	80.5

(a)

Rank	Params.	FLOPs	Top-1 (%)
$R=1$	11.9M	1.8G	74.2
$R=4$	12.4M	1.9G	74.7
$R=8$	13.0M	2.0G	74.9

(b)

PW	DW	Params.	FLOPs	Top-1 (%)
✓		4.8M	579M	75.3
	✓	4.6M	569M	75.0
✓	✓	5.1M	579M	75.9

(c)

Layers	Params.	FLOPs	Top-1 (%)
Shallow	4.3M	570M	74.0
Deep	5.1M	578M	75.0
All	5.1M	579M	75.9

(d)

Table 5: (a) Study of different formulations of PEDConv applied to ResNet-18/50 on ImageNet. (b) Impact of decomposition rank for PEDConv applied to ResNet-18 on ImageNet. (c) PEDConv applied at different types of convolutions in MobileNetV1 on ImageNet. (d) PEDConv applied at different layer depth of MobileNetV1 on ImageNet.

volution) for MobileNetV1 on ImageNet. Applying PEDConv to any type of layer improves the accuracy over static baseline, while using it for all types of layers yields the best accuracy. We find that substituting PEDConv for pointwise layers yields better accuracy than for depthwise layers, while the latter case is more parameter efficient.

PEDConv at different depths. We analyze the effect of PEDConv at different depths (shallow layers vs. deep layers) on ImageNet using MobileNetV1. We can observe that applying PEDConv to deeper layers has more positive influence to the accuracy than shallower layers. We conjecture the underlying reason is that deeper layers encode more context information, providing more clues to adapt the convolution weights.

Effect of enforcing input-dependency via info-max. We analyze the effect of enforcing input-dependency via info-max on ImageNet using ResNet-50. When we do not use this mutual information maximization at training time, the resulting model achieves 79.9% Top-1 accuracy, a 0.6% decrease compared to default complete PEDConv in Table 1(b).

4.6 Extension to spatial dynamic

The proposed PEDConv can be readily extended to both input and spatial dynamic, which provides spatially varying convolution weights at different spatial locations w.r.t. different inputs. In spatial dynamic PEDConv, the CP decomposition components are generated from the channel-wise feature vector at each spatial location. Denote the input feature-maps for a layer as $X \in \mathbb{R}^{C\times H\times W}$. We consider each spatial location $(i,j), i \in [H], j \in [W]$, from where we extract the channel-wise feature vector $X_{:,i,j} \in \mathbb{R}^{C}$. There are a total of $H \times W$ such feature vectors, one for each spatial location. We apply the bottleneck two-layer MLP (cf. Eq.(7)) to generate one set of CP decomposition components for each spatial location (i,j) by taking $X_{:,i,j}$ at corresponding location as input. The generated decomposition components reparameterize the base convolution weight and yield spatially dynamic convolution for processing the feature at different spatial locations. Since $X_{:,i,j}$ at different spatial locations can be different, CP components generated from the feature vectors at different spatial locations become different. Using these different components to reparameterize the base convolution would yield different filter and kernel weights per spatial location. Spatial dynamic PEDConv can be implemented by the unfolding operation in PyTorch, which unfolds the input

Model	Params	FLOPs	Top-1 (%)
ResNet-50 (baseline)	25.6M	4.1G	76.2
SE-ResNet-50 [11]	28.1M	4.1G	77.5
CBAM-ResNet-50 [32]	28.1M	4.1G	77.3
AA-ResNet-50 [2]	25.8M	4.2G	77.7
LR-Net-50 [10]	23.3M	4.3G	77.3
Stand-Alone ResNet-50 [23]	18.0M	3.6G	77.6
SAN-19 [37]	17.6M	3.8G	77.4
Axial ResNet-S [31]	12.5M	3.3G	78.1
BoT-50 [26]	20.8M	3.8G	77.0
SCNet-50 [18]	25.6M	4.0G	78.2
Involution-50 [14]	15.5M	2.7G	78.4
DDF-ResNet-50 [40]	16.8M	2.3G	79.1
PEDConv-ResNet-50 (S.D.)	17.2M	2.3G	**79.3**

Model	Params	FLOPs	Top-1 (%)
ResNet-101 (baseline)	44.6M	7.8G	77.4
SE-ResNet-101 [11]	49.3M	7.8G	78.4
CBAM-ResNet-101 [32]	49.3M	7.8G	78.5
AA-ResNet-101 [2]	45.4M	8.0G	78.7
LR-Net-101 [10]	42.0M	8.0G	78.5
Axial ResNet-M [31]	26.5M	6.8G	79.2
RegNet [22]	26.2M	6.5G	79.2
Involution-101 [14]	25.6M	4.7G	79.1
DDF-ResNet-101 [40]	28.1M	4.1G	80.2
PEDConv-ResNet-101 (S.D.)	29.3M	4.1G	**80.3**

Table 6: Results of spatial dynamic PEDConv on ImageNet. PEDConv achieves the highest accuracy with least FLOPs compared to other SOTAs."S.D." denotes Spatial Dynamic.

feature-maps into patches, on which are apply the different convolution weights, and finally reshape to the output feature-maps. In order to control the computation cost incurred by the per-pixel dynamic generator, we apply spatial dynamic PEDConv with depth-wise base convolutions, and use it replace standard the 3×3 convolutions in ResNet-50/101.

The results are summarized in Table 6, where we compare with a series of state-of-the-art variants of ResNet including self-attention based methods [26, 37]. As shown, (1) spatial dynamic PEDConv improves the accuracy over baseline while achieving nearly 2x FLOPs reduction; (2) our method achieves superior accuracy whilst with the most parsimonious FLOPs compared with previous state-of-the-arts. We note that translation invariance no longer holds for spatial dynamic PEDConv. Although we use different convolution weights per spatial location, the weights of the CP component generator are shared across different spatial locations. We conjecture that this generator weight sharing may inject inductive bias for training spatial dynamic PEDConv. The accuracy improvements may stem from adapting the convolution weights across different spatial locations, thus the model treats visual elements bearing different informativeness discriminatively in the spatial domain.

Among various ResNet variants, DDF [40] is a concurrent work and a strong baseline that also proposes spatially-varying dynamic convolution. We highlight the differences between DDF and proposed PEDConv. (1) DDF decouples spatial and channel dynamic filters, whereas PEDConv proposes to use low-rank Canonical Polyadic tensor decomposition to factorize the input-dependent perturbation tensor to input filters, output filters, and kernels. (2) DDF directly generates the weights of dynamic filters, whereas PEDConv generates the CP components that are applied to dynamically reparameterize the base convolution weight. (3) To generate the CP components, we propose to use variational inference method to learn the conditional distribution with mutual information maximization, compared to the deterministic generation in DDF.

5 Conclusion

We introduce a novel parameter efficient dynamic convolution PEDConv. We propose a tensor decomposition based weight reparameterization to achieve input-dependent convolution. To enforce the input dependency of the convolution weight and the input, we propose a mutual information maximization objective while learning the conditional prior distribution for generating the decomposition components. PEDConv can serve as an effective substitute of static convolution in existing CNNs or NAS. Experiments on multiple deep learning tasks demonstrate the efficacy and generality of PEDConv, which significantly outperforms SOTA dynamic convolutions in terms of accuracy while requiring fewer parameters and FLOPs.

References

[1] David Barber Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16:201, 2004.

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3286–3295, 2019.

[3] Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun. Dynamic region-aware convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8064–8073, 2021.

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

[5] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *European Conference on Computer Vision*, pages 351–367. Springer, 2020.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE international conference on computer vision*, 2009.

[7] Hang Gao, Xizhou Zhu, Steve Lin, and Jifeng Dai. Deformable kernels: Adapting effective receptive fields for object deformation. *arXiv preprint arXiv:1910.02940*, 2019.

[8] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[9] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13713–13722, 2021.

[10] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019.

[11] 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.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8607–8617, 2021.

[14] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12321–12330, 2021.

[15] 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.

[16] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Ye Yu, Lu Yuan, Zicheng Liu, Mei Chen, and Nuno Vasconcelos. Revisiting dynamic convolution via matrix decomposition. *arXiv preprint arXiv:2103.08756*, 2021.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Changhu Wang, and Jiashi Feng. Improving convolutional networks with self-calibrated convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10096–10105, 2020.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[20] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. *arXiv preprint arXiv:2007.11823*, 2020.

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[22] 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.

[23] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[24] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

[25] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491, 2015.

[26] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16519–16529, 2021.

[27] Zhuo Su, Linpu Fang, Wenxiong Kang, Dewen Hu, Matti Pietikäinen, and Li Liu. Dynamic group convolution for accelerating convolutional neural networks. In *European Conference on Computer Vision*, pages 138–155. Springer, 2020.

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

[29] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. *arXiv preprint arXiv:2003.05664*, 2020.

[30] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[31] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020.

[32] 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.

[33] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

[34] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.

[35] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *arXiv preprint arXiv:1904.04971*, 2019.

[36] 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.

[37] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[38] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

[39] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[40] Jingkai Zhou, Varun Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6647–6656, 2021.