Kernel Representation for Dynamic Networks

Yichen Zhou
https://orcid.org/0000-0002-0596-2087
Teck Khim Ng
https://www.comp.nus.edu.sg/~ngtk/

School of Computing, National University of Singapore, Singapore

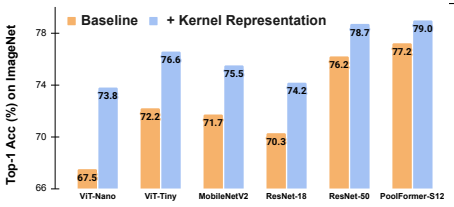
Abstract

Dynamic convolution enhances model capacity by combining multiple kernels based on input features, offering significant improvements over traditional convolution in various vision tasks without substantially increasing computational complexity. However, it solely relies on current input features to generate kernels, overlooking the generation process in previous layers. This results in sub-optimal kernel generation that limits the representational power of dynamic networks. To address these issues, we propose a separate yet coupled network to learn layer-wise kernel representation. The kernel representation, along with the feature representation, can be easily used to generate kernels by a small network and is updated layer-by-layer based on the kernel representation from the previous layer and the new feature representation of the current layer. To further complement the learning network, the initial kernel representation begins with low-frequency image features, and the final output kernel representation is concatenated with the feature representation for classification. Extensive experimental results show that the proposed kernel representation improves the network capacity and brings noticeable accuracy boost for various backbone architectures, e.g. +2.5%∼3.9% on ResNets, +3.8%∼6.0% on MobileNets, +0.8%∼6.3% on vision transformers and +1.8% on PoolFormers.

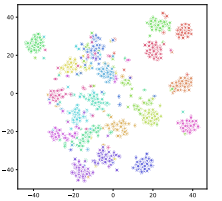
1 Introduction

In the past few years, there has been remarkable progress in neural network structures, including deep Convolutional Neural Networks (CNNs) [4, 5, 11, 17, 19, 20, 21, 27], Vision Transformers (ViTs) [3, 23], MLP-based networks [22, 24], and pooling-based networks [29]. These networks have been widely used as backbones in various vision tasks. Several studies have also focused on developing new modules and components to enhance the performance of existing backbones, especially CNNs. These modules include Squeeze-and-Excitation [7], CBAM [26], GE [6], and ECA [18], which improve the feature output of a standard convolution layer.

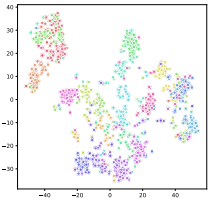
More recently, some research works [1, 12, 13, 14, 15, 28] study how to directly improve the convolution operation itself. One representative solution is dynamic convolution, which adaptively chooses and combines a set of kernels according to the input features of the current network layer. Conditionally Parameterized Convolutions [28] and Dynamic Convolution [1] are the pioneer works that first propose to linearly combine a set of convolutional



(a) Performance on ImageNet-1k.



(b) Dist. with KR



(c) Dist. w/o KR

Figure 1: (a) shows the classification improvements of our KR to different backbones on ImageNet. (b) and (c) show the T-SNE visualizations of feature distributions from 20 classes extracted from ViT models trained with and w/o kernel representation (KR). By contrast, KR has shown to be effective in clustering intra-class samples, demonstrating its superior capability in modeling semantic information.

kernels to generate a sample-adaptive kernel for the convolution at the current layer, where the coefficients depend on the input features of the current layer. These works are followed by several other ones [12, 13] to explore different ways of generating kernels as functions of input features to further improve the representation capacity. Instead of linear combination, WeightNet [15] uses a light-weight network with globally pooled features as input to directly generate the parameters for the kernels. Adaptive kernel generation has offered notable advancements over standard kernel convolution by enhancing model capacity with little extra computational cost. However, existing works use only current input features for kernel generation, without taking into account of what kernels were dynamically generated in previous layers. It results in sub-optimal kernel generation that employ a fixed kernel generation strategy on features obtained from dynamic kernels. This limits the precision and efficiency for dynamic kernel generation and hence representational power of dynamic networks.

In this paper, we introduce a novel kernel representation (KR) addressing challenges in dynamic convolution: the disconnect between input features and dynamic kernels, and insufficient information for kernel generation. Our KR is formulated by a tri-input network comprising: (1) parameters for current layer kernel generation, (2) globally pooled current input features, and (3) the kernel data from previous layers, encapsulating historical feature generation. This KR, combined with current feature data, guides the subsequent layer's kernel generation. For instance, if earlier layers predominantly chose dog-related kernels given a dog image, the current layer, informed by its KR, can more adeptly generate relevant kernels for dog-associated features. t-SNE visualizations in Fig. 1b and 1c demonstrate that our dynamic model with KR better clusters samples of the same class compared to its counterpart without KR. Moreover, our method integrates extra data beyond feature maps for kernel creation. With KR encompassing kernel and feature representations from preceding layers, the current layer can make more comprehensive decisions for kernel generation.

We integrated the kernel representation-based dynamic convolution into key backbone architectures like ResNets [9], Vision Transformers [3], MobileNetV2 [17], and PoolFormers [29], then assessed their performance on the ImageNet-1k [2] benchmark. Figure 1a displays the enhanced performance of models using kernel representation over fixed kernel counterparts. Results reveal dynamic networks with our proposed kernel representation significantly boost backbone network accuracy: increases range from +2.5% to 3.9% for ResNets, +3.8% to 6.0% for MobileNets, +0.8% to 6.3% for Vision Transformers, and +1.8% for PoolFormers. These also surpass other baseline dynamic convolution networks. comprehensive ablation studies confirms the effectiveness of our kernel representation both at the micro layer and overall network levels.

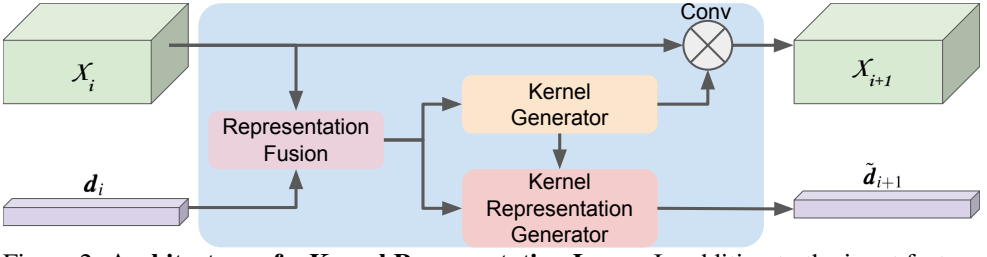


Figure 2: **Architecture of a Kernel Representation Layer.** In addition to the input feature representations, we use a kernel representation to preserve the semantic information of the dynamic features. We create a complete global representation of the input features with fusion of feature representation with the kernel representation. The generated kernels are applied to the input features, and the output kernel representation is passed to the next layer.

2 Related Works

Recent years have seen a surge in research on sample-adaptive convolution due to its efficiency in enhancing CNN model capacity. Early methods like Conditionally Parameterized Convolutions (CondConv)[28] utilize a basic routing function to compute dynamic coefficients for kernel aggregation, creating sample-adaptive kernels for convolution with input features. Another foundational method, Dynamic Convolution (DyConv)[1], employs a softmax function to normalize the coefficients for kernel combination, initializing the temperature high to boost multiple kernel training efficiency. DyNet [31] takes a similar approach in linearly combining kernel weights but allows independent coefficient generation for each output channel, enhancing flexibility.

Context-Gated Convolution (CGC)[14] dynamically adjusts convolution layer kernel weights through a module influenced by global context. WeightNet[15], rather than linearly combining learnable kernels, calculates kernels for each sample via a sub-network, enhancing kernel flexibility and bolstering the model's capacity. DCD [13] utilizes matrix decomposition and dynamic channel fusion to streamline sample-adaptive kernel generation. The more recent Omni-dimensional Dynamic Convolution (ODConv) [12] offers an advanced kernel generation approach. It determines the final convolutional kernel considering four dimensions: kernel size, input and output channels, and candidate kernels, boosting the prowess of dynamic convolutions and delivering competitive results.

Current dynamic convolution techniques only generate kernels for individual layers, overlooking inter-layer feature dependencies in deep neural networks, which limits their performance. We suggest a method that pass kernel representations with encoded feature information from one layer to another, guiding kernel creation. Our kernel representation is versatile, compatible with modern vision networks like CNNs, ViTs, MLP-based, and pooling-based systems, whereas earlier methods were limited to traditional CNNs.

3 Kernel Representation for Dynamic Networks

3.1 Preliminary on Dynamic Convolution

For the widely used standard convolution, it can be formulated as $\mathcal{X}_{i+1} = \text{Conv}(\mathcal{X}_i; \mathcal{W}_i)$ where $\mathcal{X}_i$ and $\mathcal{X}_{i+1}$ denote the input and output feature maps at the i-th layer respectively. After training, the convolution kernel $\mathcal{W}_i$ is fixed during inference. Differently, dynamic

convolution adaptively generates kernel $\mathcal{W}_i$ via a kernel generating function with current input feature $\mathcal{X}_i$ as input for both training and inference phases, i.e. $\mathcal{W}_i = G(\mathcal{X}_i; \boldsymbol{\alpha}_i)$, where $\boldsymbol{\alpha}_i$ denotes the parameters for generating kernel $\mathcal{W}_i$. In practice, many methods [1, 12, 15] implement G as the linear combination of a set of trainable kernels $\{\mathcal{W}_{i,k}\}_{k=1}^{n}$:

$$\mathcal{W}_i = G(\mathcal{X}_i; \boldsymbol{\alpha}_i) = \boldsymbol{\alpha}_{i,1}\mathcal{W}_{i,1} + ... + \boldsymbol{\alpha}_{i,n}\mathcal{W}_{i,n}, \tag{1}$$

where $\mathcal{W}_{i,k} \in \mathbb{R}^{c_{\text{in}} \times c_{\text{out}} \times H \times W}$, $\boldsymbol{\alpha}_{k,i} \in \mathbb{R}$, and $\boldsymbol{\alpha}_i = \{\boldsymbol{\alpha}_{k,i}\}_{k=1}^{n}$ denotes the predicted dynamic coefficients used to generate different kernel $\mathcal{W}_i$ for different samples.

To predict $\boldsymbol{\alpha}_i$, existing dynamic convolution methods firstly use global average pooling to summarize current input feature $\mathcal{X}_i$ into a global representation vector $\boldsymbol{h}_i$:

$$\boldsymbol{h}_i = \text{GlobalAveragePool}(\mathcal{X}_i) \in \mathbb{R}^{c_{\text{in}}}. \tag{2}$$

Then they feed $\boldsymbol{h}_i$ into a small network S to predict $\boldsymbol{\alpha}_i$:

$$\boldsymbol{\alpha}_i = S(\boldsymbol{h}_i) \in \mathbb{R}^{n}. \tag{3}$$

After training, the weights in the prediction network S are fixed, and current input feature $\mathcal{X}_i$ is used to generate sample-specific kernel $\mathcal{W}_i$ through Eqn. (2), Eqn. (3) and finally Eqn. (1).

3.2 Kernel Representation Layer

For the aforementioned dynamic convolution methods, one can observe that the predicted parameters $\boldsymbol{\alpha}_i$ solely depend on the input features $\mathcal{X}_i$, but ignores what dynamic kernels were applied to produce these features. This results in sub-optimal kernel generation that employs a fixed kernel generation strategy on features obtained from dynamic kernels. This might result in misalignment of kernels generated among different layers and limits the efficiency of feature processing, hence withholding the potential of dynamic networks.

To alleviate these issues, we design Kernel Representation (KR) Layer for Dynamic Networks. As shown in Fig. 2, besides the input features, a kernel representation layer takes an additional input kernel representation vector $\boldsymbol{d}_i$ as input, and also generates an output kernel representation vector $\boldsymbol{d}_{i+1}$ for the next layer. Here the kernel representation $\boldsymbol{d}_i$ mainly tells the layer the nature of kernels used for the current input sample and thus allows the model to know how the input feature map is generated, which will be introduced in detail below.

Representation Fusion (RF). In the RF module, we first uses global average pooling to gather the global feature representation $\boldsymbol{h}_i$ from the input feature map $\mathcal{X}_i$ via Eqn. (2). The RF module aims to fuse the information from both the kernel representation $\boldsymbol{d}_i$ and the feature representation $\boldsymbol{h}_i$. Therefore we first concatenate $\boldsymbol{d}_i$ and $\boldsymbol{h}_i$, and pass the features into a fusion network F:

$$\tilde{\boldsymbol{h}}_i = F(\text{cat}[\boldsymbol{d}_i, \boldsymbol{h}_i]) \tag{4}$$

where F is implemented as one simple linear projection. After that, the Kernel Generator receives $\tilde{\boldsymbol{h}}_i$ to generate the coefficients $\boldsymbol{\alpha}_i$ which is further used to generate sample-specific kernel $\mathcal{W}_i$ via Eqn. (1) for convolution. The Kernel Representation Generator takes both $\tilde{\boldsymbol{h}}_i$ and $\boldsymbol{\alpha}_i$ as input to produce the kernel representation $\boldsymbol{d}_{i+1}$ for the next layer.

Kernel Generator (KG). This module takes the global representation $\tilde{\boldsymbol{h}}_i$ as input, and predicts the coefficients $\boldsymbol{\alpha}_i$ via a simple MLP S:

$$\boldsymbol{\alpha}_i = S(\tilde{\boldsymbol{h}}_i). \tag{5}$$

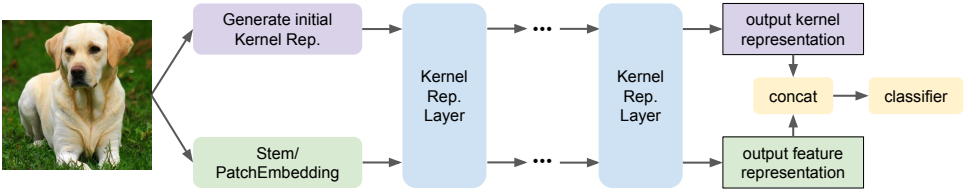


Figure 3: **Overall architecture of dynamic neural network with proposed kernel representation.** Every kernel representation layer takes in the feature representation and kernel representation from the previous layer as input, and outputs the processed features and a kernel representation to be used by the next layer.

Then KG module uses the $\boldsymbol{\alpha}_i$ to generate the dynamic kernel $\mathcal{W}_i$ via Eqn. (2) for performing convolution operation. Note that $\tilde{\boldsymbol{h}}_i$ contains the information of both current feature map $\boldsymbol{d}_i$ and also the kernel representation $\boldsymbol{h}_i$ which contains information on which kernels are used to generate current feature $\boldsymbol{d}_i$. Therefore KG has the information of the feature values $\boldsymbol{d}_i$ and also its generation process, thus making comprehensive decision to generate suitable kernels from the current kernel set. See more discussion at the end of this section.

Kernel Representation Generator (KRG). To generate a kernel representation $\boldsymbol{d}_{i+1}$ for the next $(i+1)$-th layer, the KRG module first uses a linear layer to encode the predicted $\boldsymbol{\alpha}_i$ into an embedding $\bar{\boldsymbol{\alpha}}_i$, and then feeds $\bar{\boldsymbol{\alpha}}_i$ and $\tilde{\boldsymbol{h}}_i$ into a linear layer K to generate the kernel representation $\tilde{\boldsymbol{d}}_{i+1}$:

$$\tilde{\boldsymbol{d}}_{i+1} = K(\text{cat}[\tilde{\boldsymbol{h}}_i, \bar{\boldsymbol{\alpha}}_i]). \tag{6}$$

To enhance the optimization, for each layer we concatenate the output feature kernel representation $\tilde{\boldsymbol{d}}_{i+1}$ with the input kernel representation $\boldsymbol{d}_i$ defined as

$$\boldsymbol{d}_{i+1} = \text{concatenate}[\boldsymbol{d}_i, \tilde{\boldsymbol{d}}_{i+1}], \tag{7}$$

and then feed it into the next layer. This can be considered as a form of skip connections [8] so that the gradient can directly flow to shallow layers of the network.

Discussion. Now we discuss the benefits of kernel representation $\boldsymbol{d}_i$ in the dynamic kernel generation by discussing the its three input variables $\boldsymbol{d}_{i-1}$, $\boldsymbol{h}_{i-1}$ and $\bar{\boldsymbol{\alpha}}_{i-1}$ (see Eqn. (6) and Eqn. (4)) For $\bar{\boldsymbol{\alpha}}_{i-1}$, it makes the i-th layer aware of the generation process of its input features output by the $(i-1)$-th layer. $\boldsymbol{h}_{i-1}$ provides the global semantic feature representation of input $\mathcal{X}_{i-1}$. For $\boldsymbol{d}_{i-1}$, it is the kernel representation which, through a recursive process, contains both 1) the information on kernel generation process through all historical kernel coefficients $\boldsymbol{\alpha}_k$ $(k = 0, \cdots, i-2)$, and 2) the information on semantic features through globally pooled features $\boldsymbol{h}_k$ $(k = 0, \cdots, i-2)$ of previous layers. With the comprehensive feature and kernel representations, the current layer can make more global and suitable kernel generation decisions via the parameter prediction network S in the KG module.

3.3 Kernel Representation Enhanced Networks

Figure 3 shows the overall framework of a deep network with our kernel representation. In this architecture, most parameterized layers are replaced with kernel representation layers (Figure 2), except for the first stem or Patch Embedding. For most convolutions or linear layers in the network, the kernel representation input for a layer comes from the previous layer's output. However, the network's initial input kernel representation requires special handling since no prior layer exists. We compute a vector that captures the image's semantic

information for this initial representation. To achieve this, we perform a 2-dimensional discrete Fourier transform on the real RGB color values of the input image. Subsequently, the magnitudes of 8×8 low-frequency components are extracted from all 3 channels, resulting in a vector of length 192 that represents the frequency domain of the image. Finally, we employ a trainable linear projection layer to map the 192-dimensional frequency representation to a 64-dimensional $\boldsymbol{d}_0$, which serves as the initial kernel representation to KR enhanced dynamic networks.

In standard deep neural networks, global feature activation from global pooling or a class token is passed to the classifier. But in networks using kernel representation, we combine this activation with the kernel representation for the overall feature before classification. This synchronizes optimization of the kernel representation with the backbone during training, producing more relevant kernels for specific image classification. Notably, networks enhanced with kernel representation don't need unique operators. Comprising only linear layers, normalizations, and activations, the whole network, inclusive of its original backbone and added kernel path, is trained end-to-end.

While previous dynamic kernel methods primarily addressed the micro-level, crafting sample-adaptive kernels for each layer, our method adds a macro-level perspective, optimizing sample-adaptive feature processing. Rather than just using input features for kernel creation, we employ kernel representations that inform about prior layer kernel processes. This continuity enhances the network's kernel suitability during forward passes. Furthermore, during training, kernel representation updates easily via backward propagation, enriching class-specific information and refining kernel generation. Fig. 1b and 1c visualizations highlight improved feature clustering within the same class using kernel representation, showcasing its ability to encode class semantics effectively. This differs from prior methods that rely solely on current input features for kernel generation.

As the kernel representation are only 1-dimensional vectors, there are few additional parameters and little computational cost incurred by employing them compared to the main dynamic convolutional layers in the backbone.

4 Experiments

4.1 Implementation and Settings

The proposed kernel representation is a general tool that can be used with most modern deep neural networks. In this work, we integrate kernel representation with representative architectures, including MetaFormer [29] architectures (ViTs [3] and PoolFormer [29]) and CNNs (ResNets [2] and MobileNets [17]). For all networks, the dimension for the first kernel representation $\boldsymbol{d}_0$ projected from the low frequency components of the input image is 64. For ViTs, we consider each residual block as one parameterized unit, and increase the number of dimensions of the kernel representation by 32 for each block, i.e. $dim(\tilde{\boldsymbol{d}}_{i+1}) = 32$. For ResNet-18, we set $dim(\tilde{\boldsymbol{d}}_{i+1}) = 32$ for each convolution layers with kernel representation. For ResNet-50 and MobileNets, as both networks are relatively deep, we set the $dim(\tilde{\boldsymbol{d}}_{i+1})$ to a smaller value of 16. For PoolFormer, we increase the number of dimensions of the kernel representation by 32 for each MLP and Patch Embedding block.

For our experiments with Vision Transformers (ViTs), drop-path is not employed during training for our dynamic models, as our proposed kernel representation relies on the feature dependencies between successive layers, which could be disrupted by the use of drop-path.

Models	FLOPs	Top-1 Acc (%)
ViT-Nano	0.61G	67.5
+ODConv (4×) [13]	0.62G	71.8 +4.3
+KR (ours) (1×)	0.61G	71.0 +3.5
+KR (ours) (4×)	0.62G	**73.8** +6.3
ViT-Tiny	1.26G	72.2
+ODConv (4×) [13]	1.28G	75.5 +3.3
+KR (ours) (1×)	1.27G	75.8 +3.6
+KR (ours) (4×)	1.29G	**76.6** +4.4
ViT-Small	4.61G	79.8
+ODConv (4×) [13]	4.70G	80.4 +0.6
+KR (ours) (1×)	4.64G	80.4 +0.6
+KR (ours) (4×)	4.71G	**80.6** +0.8

Table 1: Performances of ViTs on ImageNet

Models	FLOPs	CIFAR-10 Acc (%)	CIFAR-100 Acc (%)
ViT-Nano	168M	80.0	58.6
+KR (ours) (1×)	176M	87.4 +7.4	64.2 +5.6
+KR (ours) (4×)	182M	**88.4** +8.4	**65.2** +6.6
ViT-Tiny	367M	82.0	60.5
+KR (ours) (1×)	383M	89.0 +7.0	66.0 +5.5
+KR (ours) (4×)	396M	**90.3** +8.3	**68.4** +7.9

Table 2: Performances of ViTs on CIFAR

Models	FLOPs	Top-1 Acc (%)
PoolFormer-S12	1.82G	77.2
+KR (ours) (1×)	1.84G	78.2 +1.0
+KR (ours) (4×)	1.88G	**79.0** +1.8

Table 3: Performance on PoolFormer

We train the models on ImageNet-1k [2] or CIFAR [10] training set and evaluate on their corresponding validation set. For each backbone model, we use the same training settings as the baseline vanilla model with fixed kernels. For fair comparison, the number of epochs, batch-size, and data augmentations are the same for all model variants within a set of experiments. Specifically, for our experiments on the ImageNet-1k dataset, all models were trained for 300 epochs using the training recipes proposed by DeiT [23]. For the experiments conducted using the CIFAR-10 and CIFAR-100 datasets, we train all models for 200 epochs with a batch size of 256. We applied standard data augmentation techniques for CIFAR datasets, including a 4-pixel padding on each side followed by a random crop of a 32x32 region. We also included a random horizontal flip with a probability of 50%, and the images were normalized before feeding them into the network. We used the Adam optimizer with a weight decay of 0.00005. The learning rate was warmed up to 0.001 for the first 5 epochs and decayed using a cosine annealing schedule for the remainder of the training.

4.2 Results on MetaFormer Architectures

Results on ViTs. We first implement the proposed kernel representation on Vision Transformers. There are two linear layers in each residual block within the Transformer blocks, which are equivalent to 1×1 convolutions. We add kernel representation to each of the two linear layers to generate sample dependent kernels. Note that the actual self-attention calculation $\text{softmax}(QK^T)V$ remains unchanged, as there are no parameters involved in these matrix multiplications. We evaluate with ViT-Tiny and ViT-Small with 192 and 384 feature channels respectively. The results are shown in Table 1. When we employ kernel representation for the ViT-Tiny model, the performance can be improved by 4.4% to 76.6%. The results show that our kernel representation greatly enhances the feature representation power. We perform another set of experiments with models of a smaller size to further prove its merit. We reduce the number of channels of ViT-Tiny from 192 to 128 and name it "ViT-Nano". Results in in Table 1 show that kernel representation guided ViT-Nano can achieve an accuracy of 73.8%, which is higher than 72.2% by the vanilla ViT-Tiny, but with less than half the computational cost of vanilla ViT-Tiny. These results reveal the favorable improvement in efficiency by using our kernel representation. Moreover, we reproduced state-of-the-art dynamic kernel method ODConv on ViTs by replacing the linear layers to 1×1 dynamic ODConvs. Our proposed KR outperform ODConv consistently on different scales by 0.2% to 2.0%, further demonstrating the effectiveness of KR over conventional dynamic kernel methods.

Models	FLOPs	Acc (%)
ResNet-18 slim	1.35G	66.3
ResNet-18 slim +ODConv (1×)	1.37G	69.6 +3.3
ResNet-18 slim +ODConv (4×)	1.44G	71.6 +5.3
ResNet-18 slim +KR (1×)	1.38G	70.7 +4.4
ResNet-18 slim +KR (4×)	1.45G	**72.6** +6.3
ResNet-18	1.81G	70.3
+CondConv (8×) [23]	1.89G	72.0 +1.7
+DyConv (4×) [8]	1.86G	72.8 +2.5
+WeightNet [18]	1.83G	71.6 +1.3
+WE [14]	1.82G	71.0 +0.7
+DCD [13]	1.84G	72.3 +2.0
+ODConv (1×) [17]	1.84G	73.1 +2.8
+ODConv (4×) [17]	1.92G	73.9 +3.6
+KR (ours) (1×)	1.85G	73.3 +3.0
+KR (ours) (4×)	1.93G	**74.2** +3.9
ResNet-50	3.86G	76.2
+CondConv (8×) [23]	3.98G	76.7 +0.5
+DyConv (4×) [8]	3.97G	76.8 +0.6
+WeightNet [18]	3.89G	77.5 +1.5
+WE [14]	3.86G	77.1 +0.9
+DCD [13]	3.94G	76.9 +0.7
+ODConv (1×) [17]	3.92G	78.0 +1.8
+ODConv (4×) [17]	4.08G	78.5 +2.3
+KR (ours) (1×)	3.93G	78.3 +2.1
+KR (ours) (4×)	4.09G	**78.7** +2.5

Table 4: Performances of ResNets

Models	FLOPs	Acc (%)
MobileNetV2 (1.0×)	301M	71.7
+CondConv(8×) [23]	318M	74.1 +2.4
+DyConv(4×) [8]	317M	74.9 +3.2
+DCD [13]	318M	74.2 +2.5
+ODConv(1×) [17]	311M	74.8 +3.1
+ODConv(4×) [17]	327M	75.4 +3.7
+KR (ours) (1×)	315M	74.9 +3.2
+KR (ours) (4×)	331M	**75.5** +3.8
MobileNetV2 (0.75×)	209M	69.2
+CondConv(8×) [23]	239M	71.8 +2.6
+DyConv(4×) [8]	220M	72.8 +3.6
+DCD [13]	223M	71.9 +2.7
+ODConv(1×) [17]	217M	72.4 +3.2
+ODConv(4×) [17]	226M	73.8 +4.6
+KR (ours) (1×)	220M	72.8 +3.6
+KR (ours) (4×)	230M	**74.0** +4.8
MobileNetV2 (0.5×)	97M	64.3
+CondConv(8×) [23]	110M	67.2 +2.9
+DyConv(4×) [8]	103M	69.1 +4.8
+DCD [13]	106M	69.3 +5.0
+ODConv(1×) [17]	102M	68.3 +4.0
+ODConv(4×) [17]	106M	70.0 +5.7
+KR (ours) (1×)	105M	68.5 +4.2
+KR (ours) (4×)	109M	**70.3** +6.0

Table 5: Performances of MobileNetV2

ViT on CIFAR datasets. We further test the proposed KR's efficacy on the smaller CIFAR datasets [10], training all models for 200 epochs using consistent hyper-parameters. We set patch embedding dimensions for all models at 4×4. Table 2 shows that KR-equipped models consistently surpass the baselines. Notably, the KR-enhanced ViT-Nano model, with 182 MFLOPs, achieves 88.4% and 65.2% on CIFAR-10 and CIFAR-100, surpassing the standard ViT-Tiny model, which costs 2× the computation at 367 MFLOPs.

Results on PoolFormer. To further validate the effectiveness of kernel representation, we conducted experiments on PoolFormer, a pooling-based network without spatial convolutions or attention. All models are trained for 300 epochs with the same training settings as PoolFormer-S12. The results are shown in Table 3. The top-1 accuracy can be increased to 78.2% and 79.0% by using 1 and 4 times the number of kernel parameters, respectively, demonstrating the effectiveness of our kernel representation.

4.3 Results on CNN Architectures

Results on ResNets. Table 4 shows the results of different types of convolutions with the most popular CNN backbones ResNet18 and ResNet50. All models are trained for 100 epochs without advanced data augmentations nor Mixup [30]. By replacing standard convolutions with the various previous dynamic convolutions, the accuracy can be improved by 1.7% to 3.6% and 0.5% to 2.3% respectively on ResNet-18 and ResNet-50. In comparison, by integrating kernel representation into ResNets, the top-1 accuracy is boosted by 3.9% to 74.2% on ResNet-18, and by 2.5% to 78.7% on ResNet-50. The results show that our KR can bring performance improvements to ResNets backbones. Moreover, by encoding feature semantics, KR achieves comparable results to standard CNN networks but with fewer channels. This is verified with ResNet-18 slim, using channel widths of 64, 128, 192, and 256 across the four stages. Table 4 indicates KR's significant accuracy boost on ResNet-18 slim, even when compared to ODConv. Moreover, a KR-augmented ResNet-18 slim, with 20% less GFLOPs, performs 2.3% better than a vallina ResNet-18.

Method	Model	Top-1 Acc(%)
kernel representation networks	ViT-Nano	73.8
	ViT-Tiny	76.6
Dynamic kernels without kernel representation	ViT-Nano	71.1 -2.7
	ViT-Tiny	75.1 -1.5
kernel representation without low frequency embed	ViT-Nano	72.1 -1.7
	ViT-Tiny	75.6 -1.0
kernel representation without BatchNorm	ViT-Nano	72.8 -1.0
	ViT-Tiny	76.0 -0.6
Standard Linear Baseline	ViT-Nano	67.5 -6.3
	ViT-Tiny	72.2 -4.4

Table 6: Ablation Results

Hyper-parameters setting	FLOPs	Top-1 Acc (%)
Default	622M	73.8
Parameter multiplier $4 \Rightarrow 1$	612M	71.0
Parameter multiplier $4 \Rightarrow 2$	615M	71.8
Parameter multiplier $4 \Rightarrow 8$	635M	73.8
Reduction ratio $8 \Rightarrow 4$	627M	73.6
Reduction ratio $8 \Rightarrow 16$	619M	72.6
Dimensions of KR $64 \Rightarrow 32$	619M	72.2
Dimensions of KR $64 \Rightarrow 128$	628M	73.7

Table 7: Performance for various hyper-parameters settings of KR ViT-Nano.

Results on MobileNets. Table 5 shows the results evaluated on the ImageNet validation set with MobileNetV2 backbones using different width multipliers (1.0, 0.75 and 0.5) respectively. All models are trained for 150 epochs following the training settings of ODConv [12] for fair comparison. Many existing dynamic convolution methods focus on improving model capacity of the light-weight mobile-sized models, with noticeable improvements of 3% to 5% on these networks. By utilizing kernel representation guided kernel generation, we are able to obtain even higher accuracy on all scales of MobileNetV2. The kernel representation guided kernel generation provides greater representation capability for the models, and can offer accuracy gains of 6.0%, 4.8% and 3.8% on MobileNetV2 with width multipliers of 0.5, 0.75, and 1.0 respectively. The results on ResNets and MobileNets further demonstrate that our kernel representation can be integrated easily with different types of backbone structures to boost their representation power with little extra computation cost.

4.4 Ablation Studies

We conducted ablation studies for various design choices. Models were trained for 300 epochs on ImageNet-1k using the DeiT [23] setting. Ablations were done with ViT-Nano and ViT-Tiny backbones, setting the kernel parameter multiplier to 4 by default.

Kernel representation. Our structure differs from prior dynamic convolutions due to the added guidance from kernel representation during kernel generation. We first assess the performance impact of the proposed kernel representation. Table 6 presents results of ViT networks using dynamic kernels with and without kernel representation. Without KR, accuracy decreases by 2.7% and 1.5% for ViT-Nano and ViT-Tiny models, respectively, highlighting the benefits of incorporating kernel representation.

Low Frequency Components Embedding. As mentioned in previously, this is one key design in kernel representation enhanced dynamic networks. A projection from the low frequency components of the input image can encode the semantic information into the first kernel representation $\boldsymbol{d}_0$ effectively. A baseline setting for $\boldsymbol{d}_0$ is a randomly initialized learnable vector, which completes the formulation of Eqn. (4) but does not provide any information about the input samples. The performance of kernel representation networks without low frequency inputs is shown in Table 6. The drops in accuracy are 1.7% and 1.0% respectively for ViT-Nano and Vit-Tiny, which are non-trivial.

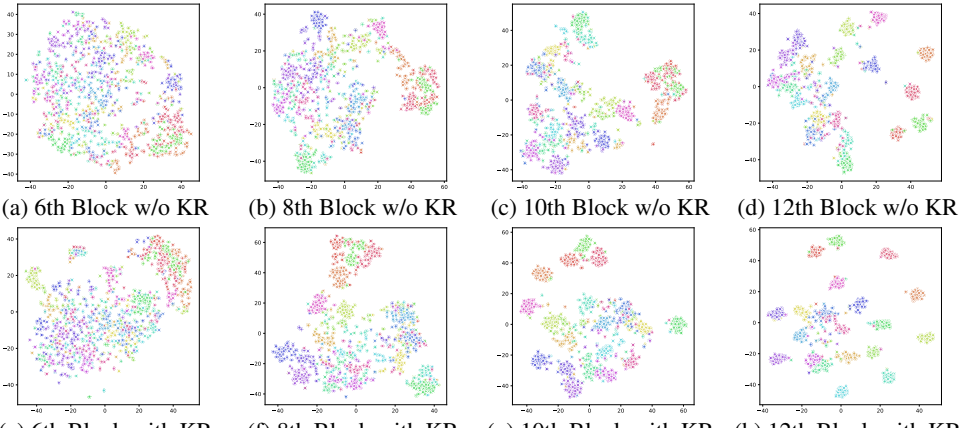
Batch Normalization on kernel representation. We use BatchNorm [9] on each kernel representation output $\tilde{\boldsymbol{d}}_i$ to enhance training optimization. Table 6 indicates a 1.0% and 0.6% accuracy drop without these BatchNorms, emphasizing their role in stabilizing training.

Hyper-parameters. Here we study the effects of several hyper-parameters in the kernel rep-

resentation networks, including 1) parameter multiplier: the number of multiples of kernel parameters [1, 12, 15]; 2) reduction ratio: the ratio between the dimensions of input to output channels for the linear layer before kernel generator; and 3) dimension of KR: the dimensions of kernel representation in the network, with default being 64 for every Transformer block. The results are shown in Table 7. The default setting with a parameter multiplier of 4, a reduction ratio of 8 and an KR dimension of 64 performs best among various settings.

4.5 Visualizations

In addition to the T-SNE [25] visualization of the features as shown in Fig 1b and 1c, we further verify the effectiveness of kernel representation (KR) via visualizing the dynamic weights generated from trained networks. For both the ViT-Nano models trained with the proposed KR and without KR, we collect the dynamic weights (kernels) generated for 1000 different samples from 20 classes, from the 6th, 8th, 10th, and 12th MLP-Blocks of the trained dynamic ViT-Nano networks. We present the T-SNE visualization of our weight distribution in Figure 4. It can be observed that, at each stage of the network, the weights for different samples of the network trained with the proposed kernel representation (in the bottom row) exhibit better clustering for samples from the same class and greater separation for samples from different classes. Moreover, the distributions of the weights follow a similar pattern as the distributions of feature representations presented in Figure 1, indicating that the class-specific information provided by the kernel representation can be utilized effectively during the dynamic kernel generation process.



(a) 6th Block w/o KR (b) 8th Block w/o KR (c) 10th Block w/o KR (d) 12th Block w/o KR

(e) 6th Block with KR (f) 8th Block with KR (g) 10th Block with KR (h) 12th Block with KR

Figure 4: **T-SNE Visualization** of distributions of dynamic weight generated.

5 Conclusion

In this paper, we present a unique approach to sample-adaptive dynamic networks: kernel representation. This method retains semantic information alongside the traditional feature-processing backbone, enhancing kernel generation at each layer with added insight from KR. Consequently, our approach generates kernels that are better suited for extracting features for different samples. Through comprehensive experiments on multiple backbones, our results confirm the considerable performance gains achieved by incorporating kernel representations into dynamic networks, with little additional computation cost.

References

[1] 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.

[2] 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.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] 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.

[5] 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.

[6] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. *Advances in neural information processing systems*, 31, 2018.

[7] 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.

[8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[12] Chao Li, Aojun Zhou, and Anbang Yao. Omni-dimensional dynamic convolution. *arXiv preprint arXiv:2209.07947*, 2022.

[13] 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.

[14] Xudong Lin, Lin Ma, Wei Liu, and Shih-Fu Chang. Context-gated convolution. In *European Conference on Computer Vision*, pages 701–718. Springer, 2020.

[15] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. In *European Conference on Computer Vision*, pages 776–792. Springer, 2020.

[16] Niamul Quader, Md Mafijul Islam Bhuiyan, Juwei Lu, Peng Dai, and Wei Li. Weight excitation: Built-in attention mechanisms in convolutional neural networks. In *European Conference on Computer Vision*, pages 87–103. Springer, 2020.

[17] 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.

[18] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[21] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[22] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

[23] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[24] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[25] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[26] 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.

[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[28] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10819–10829, 2022.

[30] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[31] Yikang Zhang, Jian Zhang, Qiang Wang, and Zhao Zhong. Dynet: Dynamic convolution for accelerating convolutional neural networks. *arXiv preprint arXiv:2004.10694*, 2020.