HorNet: Efficient High-Order Spatial Interactions with Recursive Gated Convolutions

Yongming Rao[1*] **Wenliang Zhao**[1*] **Yansong Tang**[1]
Jie Zhou[1†] **Ser-Nam Lim**[2†] **Jiwen Lu**[1†]
[1]Tsinghua University [2]Meta AI

Abstract

Recent progress in vision Transformers exhibits great success in various tasks driven by the new spatial modeling mechanism based on dot-product self-attention. In this paper, we show that the key ingredients behind the vision Transformers, namely input-adaptive, long-range and high-order spatial interactions, can also be efficiently implemented with a convolution-based framework. We present the Recursive Gated Convolution (g^nConv) that performs high-order spatial interactions with gated convolutions and recursive designs. The new operation is highly flexible and customizable, which is compatible with various variants of convolution and extends the two-order interactions in self-attention to arbitrary orders without introducing significant extra computation. g^nConv can serve as a plug-and-play module to improve various vision Transformers and convolution-based models. Based on the operation, we construct a new family of generic vision backbones named HorNet. Extensive experiments on ImageNet classification, COCO object detection and ADE20K semantic segmentation show HorNet outperform Swin Transformers and ConvNeXt by a significant margin with similar overall architecture and training configurations. HorNet also shows favorable scalability to more training data and larger model sizes. Apart from the effectiveness in visual encoders, we also show g^nConv can be applied to task-specific decoders and consistently improve dense prediction performance with less computation. Our results demonstrate that g^nConv can be a new basic module for visual modeling that effectively combines the merits of both vision Transformers and CNNs. Code is available at https://github.com/raoyongming/HorNet.

1 Introduction

Convolutional neural networks (CNN) have driven remarkable progress in deep learning and computation vision since the introduction of AlexNet [31] in the last decade. There are quite a few nice properties of CNNs making them naturally suitable for a wide range of vision applications. Translation equivariance introduces useful inductive biases to major vision tasks and enables transferability across different input resolutions. The highly optimized implementation makes it efficient on both high-performance GPUs and edge devices. The evolution of architectures [32, 31, 49, 50, 22, 24, 51] further increases its popularity on various vision tasks.

The emergence of Transformer-based architectures [16, 52, 42] greatly challenges the dominance of CNNs. By combining some successful designs in CNN architectures and the new self-attention mechanism, vision Transformers have shown leading performance on various vision tasks such as image classification [12, 42, 48], object detection [70, 41], semantic segmentation [6, 8] and video understanding [64, 18]. *What makes vision Transformers more powerful than CNNs?* Some efforts have been made to improve the CNN architectures by learning from the new designs in vision

[*]Equal contribution. [†]Corresponding authors.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

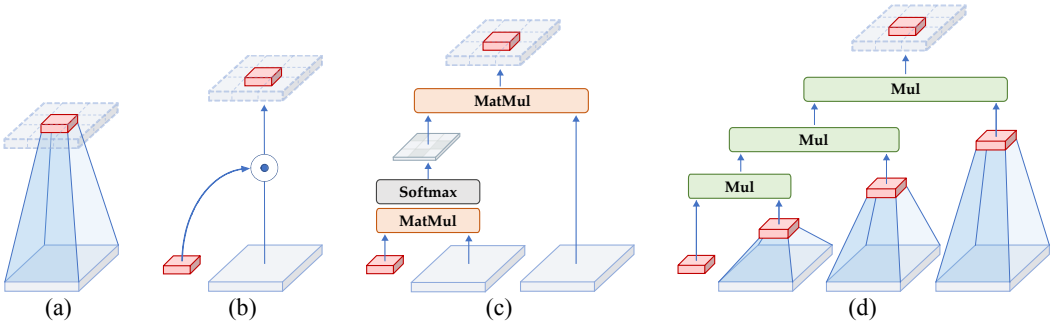

Figure 1: **Illustration of our main idea.** We show representative spatial modeling operations that perform different orders of interactions. In this paper, we focus on studying *explicit* spatial interactions between a feature (red) and its neighboring region (light gray). (a) The standard convolution operation does not explicitly consider the spatial interaction. (b) Dynamic convolution [28, 4] and SE [25] introduce the dynamic weights to improve the modeling power of convolutions with extra spatial interactions. (c) The self-attention operation [56] performs two-order spatial interactions with two successive matrix multiplications. (d) g^nConv realizes arbitrary-order spatial interactions using a highly efficient implementation with gated convolutions and recursive deigns.

Transformers. [43] presents a thorough study to adopt the *meta architecture* of vision Transformer to improve CNNs and proposes to use a *large 7×7 kernel* to construct a modern CNN. [46] and [14] propose to use even *larger kernels* to learn long-range relations with global filters and up to 31×31 convolutions, respectively. [20] shows that the *input-adaptive weights* play a key role in vision Transformers and achieve similar performance with Swin Transformers with dynamic convolutions [4, 28]. However, the effectiveness of dot-product self-attention in vision tasks has not been analyzed from the prospective of *high-order spatial interactions*.

While there exists complex and often high-order interactions between two spatial locations in a deep model due to the non-linearity, the success of self-attention and other dynamic networks suggests that the *explicit* and *high-order* spatial interactions introduced by the architectural designs are beneficial to improving the modeling power of vision models. As illustrated in Figure 1, the plain convolution operation does not explicitly consider the spatial interactions between a spatial location (*i.e.*, the red feature) and its neighboring region (*i.e.*, the light gray region). Enhanced convolution operations like dynamic convolution [4, 28, 20] introduce explicit spatial interaction by generating dynamic weights. The dot-product self-attention operation in Transformers [56] consists of two successive spatial interactions by performing matrix multiplication among queries, keys and values. The trend of the basic operations for visual modeling indicates that the network capacity can be improved by increasing the order of spatial interactions.

In this paper, we summarize that the key ingredient behind the success of vision Transformers is the new way of spatial modeling with *input-adaptive*, *long-range* and *high-order* spatial interactions performed by the self-attention operation. While previous work has successfully migrated the meta architecture [43, 20, 46, 14], input-adaptive weight generation strategy [20] and large-range modeling ability [46, 14] of vision Transformers to CNN models, a higher-order spatial interaction mechanism has not been studied. We show that all the three key ingredients can be efficiently implemented using a convolution-based framework. We propose the Recursive Gated Convolution (g^nConv) that performs high-order spatial interactions with gated convolutions and recursive deigns. Instead of simply imitating the successful designs in self-attention, g^nConv has several extra favorable properties: 1) *Efficient*. The convolution-based implementation avoids the quadratic complexity of self-attention. The design that progressively increases the channel width during performing spatial interactions also enables us to achieve higher-order interactions with bounded complexity; 2) *Extendable*. We extend the two-order interaction in self-attention to arbitrary orders to further improve the modeling power. Since we do not make assumptions on the type of spatial convolution, g^nConv is compatible with various kernel size and spatial mixing strategies like [46, 14]; 3) *Translation-equivariant*. g^nConv fully inherits the translation equivariance of the standard convolution, which introduces beneficial inductive biases to major vision tasks and avoids the asymmetry brought by local attention [42, 34].

Based on g^nConv, we construct a new family of generic vision backbones named HorNet. We conduct extensive experiments on ImageNet classification [13], COCO object detection [38] and ADE20K semantic segmentation [71] to verify the effectiveness of our models. With the same 7×7 kernel/window and similar overall architecture and training configurations, HorNet outperforms Swin

and ConvNeXt by a large margin on all tasks at different levels of complexity. The gap can be further enlarged by using a global kernel size [46]. HorNet also shows favorable scalability to more training data and larger model size, attaining 87.7% top-1 accuracy on ImageNet, 57.9% mIoU on ADE20K val and 59.2% bounding box AP on COCO val with ImageNet-22K pre-training. Apart from applying g^nConv in visual encoders, we further test the generality of our designs on task-specific decoders. By adding gConv to the widely used feature fusion model FPN [36], we develop HorFPN to model the high-order spatial relationships of features from different hierarchical levels. We observe that HorFPN can also consistently improve various dense prediction models with lower computational costs. Our results demonstrate that g^nConv can be a promising alternative to self-attention for visual modeling and effectively combine the merits of both vision Transformers and CNNs.

2 Related Work

Vision Transformers. The Transformer architecture [56] is originally designed for the natural language processing tasks. Since Dosovitskiy *et al*. [16] show that vision models constructed only by the Transformer blocks and a patch embedding layer can also achieve competitive performance to CNNs, many new models have been proposed to modify the Transformer-based architecture and make it more suitable for various vision tasks [42, 58, 60, 9, 66, 55]. Different from the original designs in [16], state-of-the-art vision Transformers usually utilize a CNN-like hierarchical architecture and change the global self-attention among all patches to local self-attention to avoid the quadratic complexity. In this paper, we follow the overall architecture of the previous hierarchical vision Transformers [42] and replace the self-attention sub-layer with our proposed g^nConv to fairly compare with the previous Transformer-based models.

Convolution-based models. Inspired by the recent success of vision Transformers, several papers propose to adopt the Transformer-style architecture and spatial convolutions with a large kernel size to improve the performance of CNNs. Han *et al*. [20] replace the window self-attention in Swin Transformers with large-kernel dynamic convolutions and achieve better performance. GFNet [46] proposes to perform the global spatial interactions like vision Transformers with global filters in the frequency domain, which are equivalent to depth-wise convolutions with a global kernel size and circular padding. ConvNeXt [43] thoroughly analyzes the designs in recent vision Transformers and presents a strong convolutional model with 7×7 depth-wise convolutions. RepLKNet [14] explores CNN models with very large kernels (up to 31×31), showing good scalability as vision Transformers. VAN [19] and FocalNet [65] use gated convolutions to perform input-adaptive attention and adopts large-kernel dilated convolutions and multiple successive 3×3 convolutions respectively to produce the weights. Previous work focuses on the meta architecture [67], large-kernel designs and input-adaptive weights to improve CNNs by learning from vision Transformers. In this paper, we offer a new perspective of high-order spatial attention to analyze the merits of vision Transformers. We show that the proposed HorNet that combines the advantages of both CNNs and vision Transformers is a better architecture for various vision tasks.

Hybrid models. Combining vision Transformers and CNNs to develop hybrid architectures is a new direction in various visual recognition problems. Recently, several efforts have been made to integrate the two types of blocks into a unified model with a sequential [12, 29, 68, 63] or parallel [45, 11] design. Many enhanced vision Transformers also use lightweight convolutions in the basic building block to efficiently capture neighboring patterns [15, 60, 17] or relax the quadratic complexity of self-attention [9, 58, 18]. Different from these hybrid models, we aim to develop a self-attention free model while combining the favorable properties of both vision Transformers and CNNs.

3 Method

3.1 g^nConv: Recursive Gated Convolutions

In this section, we will present g^nConv, an efficient operation to achieve long-term and high-order spatial interactions. The g^nConv is built with standard convolutions, linear projections and element-wise multiplications, but has a similar function of input-adaptive spatial mixing to self-attention.

Input-adaptive interactions with gated convolution. Recent success in vision Transformers mainly depends on the proper modeling of the spatial interactions in visual data. Unlike CNNs

that simply use the static convolution kernel to aggregate neighboring features, vision Transformers apply multi-head self-attention to dynamically generate the weights to mix spatial tokens. However, the quadratic complexity w.r.t. the input size of the self-attention largely hinders the application of vision Transformers, especially on downstream tasks including segmentation and detection where higher-resolution feature maps are required. In this work, instead of reducing the complexity of self-attention like previous methods [42, 9, 57], we seek a more efficient and effective way to perform spatial interactions with simple operations like convolution and fully-connected layers.

The basic operation of our method is the gated convolution (gConv). Let $\mathbf{x} \in \mathbb{R}^{HW \times C}$ be the input feature, the output of the gated convolution $\mathbf{y} = g\text{Conv}(\mathbf{x})$ can be written as:

$$\begin{aligned} [\mathbf{p}_0^{HW\times C}, \mathbf{q}_0^{HW\times C}] &= \phi_{\text{in}}(\mathbf{x}) \in \mathbb{R}^{HW\times 2C}, \\ \mathbf{p}_1 = f(\mathbf{q}_0) \odot \mathbf{p}_0 \in \mathbb{R}^{HW\times C}, &\quad \mathbf{y} = \phi_{\text{out}}(\mathbf{p}_1) \in \mathbb{R}^{HW\times C}, \end{aligned} \tag{3.1}$$

where $\phi_{\text{in}}, \phi_{\text{out}}$ are linear projection layers to perform channel mixing and f is a depth-wise convolution. Note that $p_1^{(i,c)} = \sum_{j\in\Omega_i} w_{i\to j}^c q_0^{(j,c)} p_0^{(i,c)}$, where Ω_i is the local window centered at i and w represents the convolution weight of f. Therefore, the above formulation explicitly introduce interactions among the neighboring features $\mathbf{p}_0^{(i)}$ and $\mathbf{q}_0^{(j)}$ through the element-wise multiplication. We consider the interaction in gConv as *1-order interaction* as each $\mathbf{p}_0^{(i)}$ has interacted with its neighbor feature $\mathbf{q}_0^{(j)}$ only once.

High-order interactions with recursive gating. After achieving an efficient 1-order spatial interactions with the gConv, we then design the g^nConv, a recursive gated convolution to further enhance the model capacity by introducing higher-order interactions. Formally, we first use ϕ_{in} to obtain a set of projected features $\mathbf{p}_0$ and $\{\mathbf{q}_k\}_{k=0}^{n-1}$:

$$\left[\mathbf{p}_0^{HW\times C_0}, \mathbf{q}_0^{HW\times C_0}, \ldots, \mathbf{q}_{n-1}^{HW\times C_{n-1}}\right] = \phi_{\text{in}}(\mathbf{x}) \in \mathbb{R}^{HW\times(C_0+\sum_{0\le k\le n-1} C_k)}. \tag{3.2}$$

We then perform the gated convolution *recursively* by

$$\mathbf{p}_{k+1} = f_k(\mathbf{q}_k) \odot g_k(\mathbf{p}_k)/\alpha, \qquad k = 0, 1, \ldots, n-1, \tag{3.3}$$

where we scale the output by $1/\alpha$ to stabilize the training. $\{f_k\}$ are a set of depth-wise convolution layers and $\{g_k\}$ are used to match the dimension in different orders:

$$g_k = \begin{cases} \text{Identity}, & k = 0, \\ \text{Linear}\left(C_{k-1}, C_k\right), & 1 \le k \le n-1. \end{cases} \tag{3.4}$$

Finally, we feed the output of the last recursion step $\mathbf{q}_n$ to the projection layer ϕ_{out} to obtain the result of the g^nConv. From the recursive formula Equation (3.3), it is easy to show that the interaction-order of $\mathbf{p}_k$ will be increased by 1 after each step. As a result, we can see that the g^nConv achieves n-order spatial interactions. It is also worth noting that we need only a single f to perform depth-wise convolution to the concatenation of the features $\{\mathbf{q}_k\}_{k=0}^{n-1}$ together instead of computing the convolution in each recursive step as in Equation (3.3), which can further simplify the implementation and improve the efficiency on GPUs. To ensure that the high-order interactions do not introduce too much computational overhead, we set the channel dimension in each order as:

$$C_k = \frac{C}{2^{n-k-1}}, \qquad 0 \le k \le n-1. \tag{3.5}$$

This design indicates that we perform the interactions in a coarse-to-fine manner, where lower orders are computed with fewer channels. Besides, the channel dimension of $\phi_{\text{in}}(\mathbf{x})$ is exactly $2C$ and the total FLOPs can be strictly bounded even with n increasing. It can be proved that (see Appendix A):

$$\text{FLOPs}(g^n\text{Conv}) < HWC(2K^2 + 11/3 \times C + 2), \tag{3.6}$$

where K is the kernel size of the depth-wise convolution. Therefore, our g^nConv achieves high-order interactions with a similar computational cost to a convolutional layer.

Long-term interactions with large kernel convolutions. Another difference between vision Transformers and conventional CNNs is the receptive field. Conventional CNNs [49, 22] often use 3×3 convolution through the whole network, while vision Transformers calculate self-attention on the whole feature maps [16, 52] or inside a relatively large local window (*e.g.*, 7×7). The large

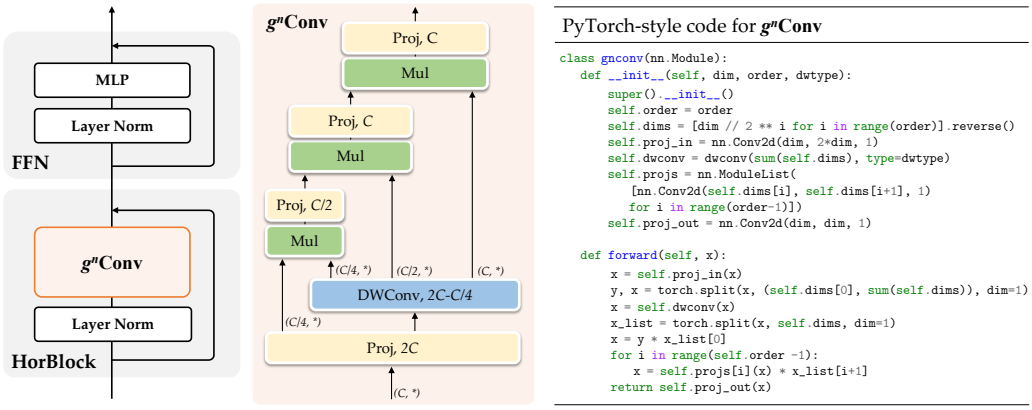

Figure 2: **Overview of the basic building block in HorNet with g^nConv.** We adopt the block design of Transformers [56] and replace the self-attention sub-layer with g^nConv to develop our HorNet (*left*). We also provide the detailed implementation of g^3Conv (*middle*) and the Pytorch-style code for an arbitrary order (*right*).

receptive field in vision Transformers makes it easier to capture long-term dependencies, which is also recognized as one of the key advantages of vision Transformers. Inspired by this design, there are some efforts to introduce large kernel convolutions to CNNs recently [14, 43, 46]. To make our g^nConv capable of capturing long-term interactions, we adopt two implementations for the depth-wise convolution f:

- *7×7 Convolution*. 7×7 is the default window/kernel size of Swin Transformers [42] and ConvNext [43]. Studies in [43] show that the kernel size produces good performance on ImageNet classification and various downstream tasks. We follow this configuration to fairly compare with representative work of vision Transformers and modern CNNs.
- *Global Filter (GF)*. The GF layer [46] multiplies the frequency domain features with learnable global filters, which is equivalent to a convolution in the spatial domain with a global kernel size and circular padding. We use a modified version of the GF layer by processing half of the channels with the global filter and the other half with 3×3 depth-wise convolutions and only use GF layers in late stages to preserve more local details.

Spatial interactions in vision models. We review some representative vision model designs from the perspective of spatial interactions, as shown in Figure 1. Specifically, we are interested in the interactions between a feature $\mathbf{x}_i$ and its neighboring feature $\mathbf{x}_j, j \in \Omega_i$. By using the tool designed for explaining the interaction effect (IE) in [33, 1], we provide an intuitive analysis of the order of explicit spatial interactions in Appendix B. Our analysis reveals a key difference between vision Transformers and previous architectures from a new view, *i.e.*, vision Transformers have higher-order spatial interactions in each basic block. The result inspires us to explore an architecture that can realize more efficient and effective spatial interactions with more than two orders. As discussed above, our proposed g^nConv can achieve arbitrary-order interactions with bounded complexity. It is also worth noting that similar to other scaling factors in deep models like width [69] and depth [22], simply increasing the order of spatial interactions without considering the overall model capacity will not lead to a good trade-off [51]. In this paper, we focus on developing a stronger visual modeling architecture based on the analysis of the spatial interaction orders of well-designed models. We believe a more thorough and formal discussion on the high-order spatial interactions can be an important future direction.

Relation to dot-product self-attention. Although the computation of our g^nConv largely differs from dot-product self-attention, we will show that g^nConv also accomplishes the goal of input-adaptive spatial mixing. Let $\mathbf{M}$ be the attention matrix obtained by multi-head self-attention (MHSA), we write $\mathbf{M}$ as (m^c_{ij}) since the mixing weight may vary across the channels. The spatial mixing result (before the final channel mixing projection) of the c-th channel at location i is

$$x^{(i,c)}_{\text{MHSA}} = \sum_{j\in\Omega_i} m^c_{ij} v^{(i,j)} = \sum_{j\in\Omega_i} \sum_{c'=1}^{C} \underline{m^c_{ij}} w_V^{(c',c)} x^{(j,c')}, \tag{3.7}$$

where w_V is the weight of the V-projection layer. Note that m_{ij} obtained by the dot-product operation contains 1-order interaction. On the other hand, the output of our g^nConv (before the ϕ_{out}) can be written as

$$x^{(i,c)}_{g^n\text{Conv}} = p^{(i,c)}_n = \sum_{j\in\Omega_i}\sum_{c'=1}^{C} \underline{w^c_{n-1,i\to j}\mathbf{g}^{(i,c)}_{n-1}} w^{(c',c)}_{\phi_{\text{in}}} x^{(j,c')} \triangleq \sum_{j\in\Omega_i}\sum_{c'=1}^{C} \underline{h^c_{ij}} w^{(c',c)}_{\phi_{\text{in}}} x^{(j,c')}, \tag{3.8}$$

where w_{n-1} is the convolutional weight for f_{n-1}, $w_{\phi_{\text{in}}}$ is the linear weight of ϕ_{in}, and $\mathbf{g}_{n-1} = g_{n-1}(\mathbf{p}_{n-1})$ is a projection of $\mathbf{p}_{n-1}$. From the formulation in Equation (3.8) we find our g^nConv also achieves input-adaptive spatial mixing with $\{h^c_{ij}\}$ as the weights. Observing that h_{ij} is computed from $\mathbf{p}_{n-1}$ which contains $n-1$ order interactions, we can regard our g^nConv as an extension of the self-attention in terms of the order of the spatial mixing weight. Therefore, our g^nConv can better model more complex spatial interactions.

The details of g^nConv and our implementation are summarized in Figure 2.

3.2 Model Architectures

HorNet. The g^nConv can be a drop-in replacement of the spatial mixing layer in vision Transformers [52, 42] or modern CNNs [43]. We follow the same meta-architecture as [56, 42] to construct HorNet, where the basic block contains a spatial mixing layer and a feed-forward network (FFN). Depending on the model size and the implementation of the depth-wise convolution f_k in our g^nConv, we have two series of model variants named HorNet-T/S/B/L$_{7\times7}$ and HorNet-T/S/B/L$_{\text{GF}}$. We consider the popular Swin Transformer [42] and ConvNeXt [43] as the vision Transformer and CNN baselines since our models are implemented based on a convolution-based framework while having high-order interactions like vision Transformers. To fairly compare with the baselines, we directly follow the number of blocks of Swin Transformers-S/B/L [42] but insert an extra block to the stage 2 to make the overall complexity close, resulting in $[2, 3, 18, 2]$ blocks in each stage in all of the model variants. We simply adjust the base number of channels C to construct models with different sizes and set the number of channels in 4 stages as $[C, 2C, 4C, 8C]$ following common practice. We use $C = 64, 96, 128, 192$ for HorNet-T/S/B/L, respectively. We set the interaction orders (*i.e.*, the n in g^nConv) for each stage as 2,3,4,5 by default, such that the channels of the coarsest order C_0 is the same across different stages.

HorFPN. Apart from using g^nConv in visual encoders, we find our g^nConv can be an enhanced alternative for standard convolution that considers higher-order spatial interactions in a wide range of convolution-based models. Thus, we replace spatial convolutions for feature fusion in the FPN [37] with our g^nConv to improve spatial interactions for downstream tasks. Specifically, we add our g^nConv after the fusion of features from different pyramid levels. For object detection, we replace the 3×3 convolution after the top-down pathway with the g^nConv in each level. For semantic segmentation, we simply replace the 3×3 convolution after the concatenation of the multi-level feature maps with g^nConv since the final results are directly predicted from this concatenated feature. We also have two implementations called HorFPN$_{7\times7}$ and HorFPN$_{\text{GF}}$ decided by the choice of f_k.

4 Experiments

We conduct extensive experiments to verify the effectiveness of our method. We present the main results on ImageNet [13] and compare them with various architectures. We also test our models on the downstream dense prediction tasks on commonly used semantic segmentation benchmark ADE20K [71] and object detection dataset COCO [38]. Lastly, we provide ablation studies of our designs and analyze the effectiveness of g^nConv on a wide range of models.

4.1 ImageNet Classification

Setups. We conduct image classification experiments on the widely used ImageNet [13] dataset. We train our HorNet-T/S/B models using the standard ImageNet-1K dataset following common practice. To fairly compare with previous work, we directly use the training configurations of [43, 42, 52] to train our models. We train the models for 300 epochs with 224×224 input. To evaluate the scaling ability of our designs, we further train the HorNet-L models on the ImageNet-22K dataset that contains over $10\times$ images and more categories. We follow previous practice [42, 43] to train our models for 90 epochs and use a similar data augmentation strategy as ImageNet-1K experiments.

Table 1: **ImageNet classification results.** We compare our models with state-of-the-art vision Transformers and CNNs that have comparable FLOPs and parameters. We report the top-1 accuracy on the validation set of ImageNet as well as the number of parameters and FLOPs. We also show the improvements over Swin Trasnformers that have similar overall architectures and training configurations to our models. "↑384" indicates that the model is fine-tuned on 384×384 images for 30 epochs. Our models are highlighted in gray.

Model	Image Size	Params (M)	FLOPs (G)	Top-1 Acc. (%)
ImageNet-1K trained models				
EfficientNet-B4 [51]	380^2	19	4.2	82.9
EfficientNet-B5 [51]	456^2	30	9.9	83.6
EfficientNet-B6 [51]	528^2	43	19.0	84.0
EfficientNetV2-S [51]	300^2	24	8.8	83.9
RepLKNet-31B [14]	224^2	79	15.3	83.5
VAN-B [19]	224^2	27	5.0	82.8
VAN-L [19]	224^2	45	9.0	83.9
CSWin-T [15]	224^2	23	4.3	82.7
CSWin-S [15]	224^2	35	6.9	83.6
CSWin-B [15]	224^2	78	15.0	84.2
Swin-T [42]	224^2	28	4.5	81.3
ConvNeXt-T [43]	224^2	29	4.5	82.1$_{(+0.7)}$
HorNet-T$_{7\times7}$	224^2	22	4.0	82.8$_{(+1.5)}$
HorNet-T$_{\mathrm{GF}}$	224^2	23	3.9	**83.0**$_{(+1.7)}$
Swin-S [42]	224^2	50	8.7	83.0
ConvNeXt-S [43]	224^2	50	8.7	83.1$_{(+0.1)}$
HorNet-S$_{7\times7}$	224^2	50	8.8	83.8$_{(+0.8)}$
HorNet-S$_{\mathrm{GF}}$	224^2	50	8.7	**84.0**$_{(+1.0)}$
Swin-B [42]	224^2	89	15.4	83.5
ConvNeXt-B [43]	224^2	88	15.4	83.8$_{(+0.3)}$
HorNet-B$_{7\times7}$	224^2	87	15.6	84.2$_{(+0.7)}$
HorNet-B$_{\mathrm{GF}}$	224^2	88	15.5	**84.3**$_{(+0.8)}$

Model	Image Size	Params (M)	FLOPs (G)	Top-1 Acc. (%)
ImageNet-1K trained models (fine-tuned at 384×384)				
Swin-B↑384 [42]	384^2	89	47.1	84.5
ConvNeXt-B↑384 [43]	384^2	88	45.0	85.1$_{(+0.6)}$
HorNet-B$_{7\times7}$↑384	384^2	87	45.8	85.3$_{(+0.8)}$
HorNet-B$_{\mathrm{GF}}$↑384	384^2	92	45.4	**85.6**$_{(+1.1)}$
ImageNet-22K trained models (fine-tuned to ImageNet-1K)				
R-101x3 [30]	384^2	388	204.6	84.4
R-152x4 [30]	480^2	937	840.5	85.4
ViT-B/16 [16]	384^2	87	55.5	84.0
ViT-L/16 [16]	384^2	305	191.1	85.2
EfficientNetV2-L [51]	380^2	121	53.0	86.8
CSWin-L [15]	384^2	173	96.8	87.5
SwinV2-L [41]	384^2	197	115.4	87.6
RepLKNet-31L [14]	384^2	172	96.0	86.6
Swin-L [42]	224^2	197	34.5	86.3
ConvNeXt-L [43]	224^2	198	34.4	86.6$_{(+0.3)}$
HorNet-L$_{7\times7}$	224^2	195	34.8	86.8$_{(+0.5)}$
HorNet-L$_{\mathrm{GF}}$	224^2	196	34.6	**87.0**$_{(+0.7)}$
Swin-L↑384 [42]	384^2	197	103.9	87.3
ConvNeXt-L↑384 [43]	384^2	198	101.0	87.5$_{(+0.2)}$
HorNet-L$_{7\times7}$↑384	384^2	195	102.3	87.6$_{(+0.3)}$
HorNet-L$_{\mathrm{GF}}$↑384	384^2	202	101.8	**87.7**$_{(+0.4)}$

We fine-tune the models pre-trained on ImageNet-22K or at the 224×224 resolution to ImageNet-1K or/and 384×384 resolution for 30 epochs following [43]. When adapting the ImageNet-22K models to ImageNet-1K, we initialize the classifier with the pre-trained class centers to stabilize the training process. More details can be found in Appendix C.

Results. The results of our ImageNet classification experiments are summarized in Table 1. We see that our models achieve very competitive performance with state-of-the-art vision Transformers and CNNs. Notably, HorNet surpasses Swin Transformers and ConvNeXt which have similar overall architectures and training configurations by a healthy margin on various model sizes and settings. Our models also generalize well to a larger image resolution, larger model sizes and more training data. These results clearly demonstrate the effectiveness and generality of our designs.

4.2 Dense Prediction Tasks

HorNet for semantic segmentation. We evaluate our HorNet for semantic segmentation task on ADE20K [71] dataset using the commonly used UperNet [62] framework. All the models are trained for 160k iterations using AdamW [44] optimizer with a global batch size of 16. The image size during training is 512×512 for ImagNet-1k (HorNet-T/S/B) pre-trained models and 640×640 for the ImageNet-22K pre-trained models (HorNet-L). The results are summarized in the left part of Table 2, where we report both the single-scale (SS) and multi-scale (MS) mIoU on the validation set. Both our HorNet$_{7\times7}$ and HorNet$_{\mathrm{GF}}$ models outperform Swin [42] and ConvNeXt [43] models with similar model sizes and FLOPs. Specifically, HorNet$_{\mathrm{GF}}$ models achieve better results than HorNet$_{7\times7}$ and ConvNeXt series by large margins in single-scale mIoU, indicating the global interactions captured by the global filter are helpful for semantic segmentation. Notably, we find both our HorNet-L$_{7\times7}$ and HorNet-L$_{\mathrm{GF}}$ even outperform ConvNeXt-XL with ∼25% fewer FLOPs. These results clearly demonstrate the effectiveness and scalability of our HorNet on semantic segmentation.

HorNet for object detection. We also evaluate our models on the COCO [38] dataset. We adopt the cascade Mask R-CNN framework [21, 2] to perform object detection and instance segmentation using HorNet-T/S/B/L backbones. Following Swin [42] and ConvNeXt [43], we use 3× schedule with multi-scale training. The right part of Table 2 compares the box AP and mask AP of our HorNet models and Swin/ConvNeXt models. Similarly, we show our HorNet models achieve consistently and significantly better performance than the Swin/ConvNeXt counterparts, in both box AP and mask AP. The HorNet$_{\mathrm{GF}}$ series obtain +1.2∼2.0 box AP and +1.0∼1.9 mask AP compared with ConvNeXt.

Table 2: **Object detection and semantic segmentation results with different backbones.** We use UperNet [62] for semantic segmentation and Cascade Mask R-CNN [2] for object detection. ‡ indicates that the model is pre-trained on ImageNet-22K. For semantic segmentation, we report both single-scale (SS) and multi-scale (MS) mIoU. The FLOPs are calculated with image size (2048, 512) for ImageNet-1K pre-trained models and (2560, 640) for ImageNet-22K pre-trained models. For object detection, we report the box AP and the mask AP. FLOPs are measured on input sizes of (1280, 800). Our models are highlighted in gray.

Backbone	Semantic Segmentation with *UperNet 160K*				Object Detection with *Cascade Mask R-CNN 3×*			
	mIoU^{ss}	mIoU^{ms}	Params	FLOPs	AP^{box}	AP^{mask}	Params	FLOPs
Swin-T [42]	44.5	45.8	60M	945G	50.4	43.7	86M	745G
ConvNeXt-T [43]	46.0	46.7	60M	939G	50.4	43.7	86M	741G
HorNet-T$_{7\times7}$	48.1	48.9	52M	926G	51.7	44.8	80M	730G
HorNet-T$_{\text{GF}}$	**49.2**	**49.3**	55M	924G	**52.4**	**45.6**	80M	728G
Swin-S [42]	47.6	49.5	81M	1038G	51.9	45.0	107M	838G
ConvNeXt-S [43]	48.7	49.6	82M	1027G	51.9	45.0	108M	827G
HorNet-S$_{7\times7}$	49.2	49.8	81M	1030G	52.7	45.6	107M	830G
HorNet-S$_{\text{GF}}$	**50.0**	**50.5**	85M	1027G	**53.3**	**46.3**	108M	827G
Swin-B [42]	48.1	49.7	121M	1188G	51.9	45.0	145M	982G
ConvNeXt-B [43]	49.1	49.9	122M	1170G	52.7	45.6	146M	964G
HorNet-B$_{7\times7}$	50.0	50.5	121M	1174G	53.3	46.1	144M	969G
HorNet-B$_{\text{GF}}$	**50.5**	**50.9**	126M	1171G	**54.0**	**46.9**	146M	965G
Swin-L‡ [42]	52.1	53.5	234M	2468G	53.9	46.7	253M	1382G
ConvNeXt-L‡ [43]	53.2	53.7	235M	2458G	54.8	47.6	255M	1354G
ConvNeXt-XL‡ [43]	53.6	54.0	391M	3335G	55.2	47.7	407M	1898G
HorNet-L‡$_{7\times7}$	54.1	54.5	232M	2473G	55.4	48.0	251M	1363G
HorNet-L‡$_{\text{GF}}$	**55.0**	**55.2**	239M	2465G	**56.0**	**48.6**	259M	1358G

Table 3: **Comparisons of HorFPN with standard FPN on different backbones.** We use UperNet 160K and Mask R-CNN 1× schedule for semantic segmentation and object detection, respectively. We find our HorFPN consistently outperforms standard FPN with various of backbones on both the two tasks.

Backbone	Fusion Module	Semantic Segmentation with *UperNet 160K*				Object Detection with *Mask R-CNN 1×*			
		mIoU^{ss}	mIoU^{ms}	Params	FLOPs	AP^{box}	AP^{mask}	Params	FLOPs
ResNet-50 [22]	FPN [37]	40.7	41.8	66M	947G	38.2	34.7	44M	260G
	HorFPN$_{7\times7}$	41.8	44.1	60M	499G	38.7	35.1	43M	226G
	HorFPN$_{\text{GF}}$	**43.2**	**44.5**	60M	497G	**39.1**	**35.5**	43M	224G
ResNet-101 [22]	FPN [37]	42.9	44.0	85M	1025G	40.0	36.1	63M	336G
	HorFPN$_{7\times7}$	44.1	45.5	79M	577G	40.3	36.4	62M	302G
	HorFPN$_{\text{GF}}$	**44.5**	**46.4**	79M	574G	**40.5**	**36.7**	62M	300G
Swin-S [42]	FPN [37]	47.6	49.5	81M	1038G	45.5	40.9	69M	354G
	HorFPN$_{7\times7}$	48.0	49.2	74M	580G	46.3	41.1	68M	325G
	HorFPN$_{\text{GF}}$	**49.0**	**49.9**	75M	578G	**46.8**	**41.9**	69M	323G
HorNet-S	FPN [37]	49.2	49.8	81M	1030G	47.1	42.2	69M	351G
	HorFPN$_{7\times7}$	49.4	50.1	74M	577G	47.4	42.3	68M	322G
	HorFPN$_{\text{GF}}$	**49.7**	**50.3**	75M	575G	**47.7**	**42.4**	68M	321G

Again, our large model HorNet-L$_{7\times7}$ and HorNet$_{\text{GF}}$ can outperform ConvNeXt-XL, which further validates the favorable transferability with a larger model size and larger pre-trained dataset.

HorFPN for dense prediction. We now show another application of the proposed g^nConv, *i.e.*, to serve as a better fusion module that can better capture the higher-order interactions among different levels of features in dense prediction tasks. Specifically, we directly modify the FPN [37] as described in Section 3.2 in UperNet [62] and Mask R-CNN [21] for semantic segmentation and object detection, respectively.We show the results in Table 3, where we compare the performance of our HorFPN and standard FPN on different backbones including ResNet-50/101 [22], Swin-S [42] and HorNet-S$_{7\times7}$. For semantic segmentation, we find our HorFPN can significantly reduce the FLOPs (∼50%) while achieving better validation mIoU. For object detection,

Table 4: **Object detection results with recent state-of-the-art frameworks.** We report the single-scale AP^{box} and AP^{mask} on the validation set of COCO. Our models are highlighted in gray.

Backbone	Framework	AP^{box}	AP^{mask}
Swin-L [42]	HTC++ [3]	57.1	49.5
ViT-Adapter-L [5]	HTC++ [3]	57.9	50.2
HorNet-L$_{\text{GF}}$	HTC++ [3]	**58.1**	**50.5**
Swin-L [42]	DINO [70]	58.5	-
HorNet-L$_{\text{GF}}$	DINO [70]	**59.2**	-

Table 5: **Semantic Segmentation results with recent state-of-the-art frameworks.** We report the single-scale (SS) and multi-scale (MS) mIoU on the validation set of ADE20K. Our models are highlighted in gray.

Backbone	Framework	mIoU^{ss}	mIoU^{ms}
Swin-L [42]	Mask2Former [7]	56.1	57.3
Swin-L-FaPN [27]	Mask2Former [7]	56.4	57.7
HorNet-L$_{\text{GF}}$	Mask2Former [7]	**57.5**	**57.9**

Table 6: **Ablation study and results of applying g^nConv to other models/operations.** We provide the ablation study of our designs in (a). [*] indicates the baseline of our model. The baseline and our final models are highlighted in gray. In (b) and (c), we apply the proposed g^nConv to isotropic models that have a similar level of complexity with ViT/DeiT-S [16, 52] and other spatial mixing operations including the 3×3 depth-wise convolution and 3×3 pooling used in [67].

(a) Ablation study.

Model	Params	FLOPs	Acc. (%)
Swin-T [42]	28M	4.5G	81.3 (+0.1)
- Self-Attention + DWConv$_{7\times7}$ [*]	29M	4.5G	81.2
+ SE [25]	30M	4.5G	81.5 (+0.3)
- SE + $g^{\{1,1,1,1\}}$Conv	28M	4.3G	81.7 (+0.5)
+ $g^{\{2,2,2,2\}}$Conv	28M	4.3G	82.2 (+1.0)
+ $g^{\{3,3,3,3\}}$Conv	28M	4.3G	82.5 (+1.3)
+ $g^{\{4,4,4,4\}}$Conv	28M	4.3G	82.5 (+1.3)
+ $g^{\{1,2,3,4\}}$Conv	28M	4.3G	82.5 (+1.3)
+ $g^{\{2,3,4,5\}}$Conv	28M	4.3G	82.6 (+1.4)
+ Deeper & Narrower (HorNet-T$_{7\times7}$)	22M	4.0G	82.8 (+1.6)
+ Global Filters [46] (HorNet-T$_{\mathrm{GF}}$)	23M	3.9G	83.0 (+1.8)
ConvNeXt [43]	28M	4.5G	82.1 (+0.9)

(b) Results on isotropic models.

Model	FLOPs	Acc. (%)
DeiT-S [52]	4.6G	79.8
ConvNeXt-S (iso.) [43]	4.3G	79.7
HorNet-S$_{7\times7}$ (iso.)	4.5G	80.6
HorNet-S$_{\mathrm{GF}}$ (iso.)	4.5G	81.0

(c) g^nConv for other operations.

Model	FLOPs	Acc. (%)
DWConv$_{3\times3}$	4.0G	80.7
g^nConv$_{3\times3}$	3.9G	82.1
Pool [67]	3.9G	78.1
g^nConv$_{\mathrm{pool}}$	3.8G	79.3

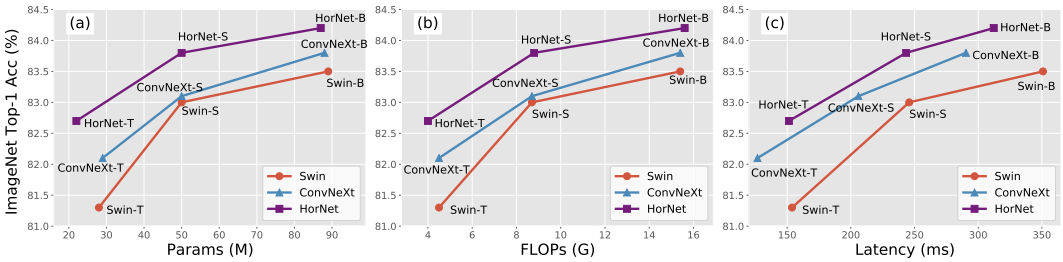

Figure 3: **Comparisons of trade-offs of Swin, ConvNeXt and HorNet.** We compare the trade-offs of the models via the top-1 accuracy on ImageNet w.r.t. **(a)** number of parameters; **(b)** FLOPs; **(c)** latency. The latency is measured with a single NVIDIA RTX 3090 GPU with a batch size of 128.

our HorFPN can also outperform standard FPN in terms of both box AP and mask AP on different backbones with about 30G fewer FLOPs. Besides, we observe that the HorFPN$_{\mathrm{GF}}$ is consistently better than HorFPN$_{7\times7}$, indicating that global interactions are also important when fusing hierarchical features.

Results with state-of-the-art frameworks. To further show the effectiveness our backbone, we conduct experiments to combine our large HorNet model with recent state-of-the-art dense prediction frameworks including HTC++ [3], DINO [70] and Mask2Former [7]. For HTC++ and DINO, we train our models on COCO for 36 epochs (3× schedule) and does not introduce extra pre-training data like Object365 in [70]. We report the single-scale performance on the validation set and compared with several state-of-the-art methods in Table 4. For Mask2Former, we train our models on ADE20K with 640×640. We report the mIoU of both single-scale and multi-scale testing on the validation set in Table 5.

4.3 Analysis

Ablation study. We provide detailed ablation studies of the g^nConv and our HorNet in Table 6. We first study the model designs of our HorNet in Table 6a. Our baseline ([*]) is obtained by simply replacing the self-attention with 7×7 depth-wise convolution in Swin-T [42]. We first show that both SE [25] and our g^nConv with $n = 1$ ($g^{\{1,1,1,1\}}$Conv) can improve over the baseline model [*], and $g^{\{1,1,1,1\}}$Conv is slightly better. We then perform ablations on the interaction order n for each stage and find: (1) if n is shared across the 4 stages, the accuracy will increase with larger n but saturate at 82.5 when $n = 4$; (2) progressively increased order ($g^{\{2,3,4,5\}}$Conv) can further improve the accuracy. Our final models are built on $g^{\{2,3,4,5\}}$Conv by adjusting the depth and width of the networks (HorNet-T$_{7\times7}$) and applying Global Filter [46] for the depth-wise convolution (HorNet-T$_{\mathrm{GF}}$). These results clearly show that our g^nConv is an efficient and extendable operation that can better capture high-order spatial interactions than both self-attention and depth-wise convolution.

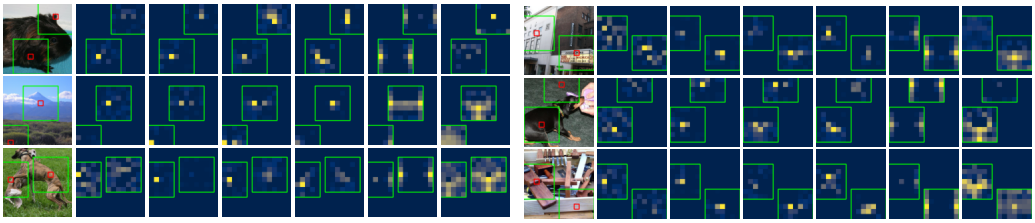

Figure 4: Visualization of the adaptive weights generated by g^nConv. We see that the spatial mixing weights of our g^nConv are adaptive both to input samples and spatial locations, which further indicates that g^nConv shares these two desirable characteristics with the self-attention operation.

g^nConv for isotropic models. We also evaluate g^nConv on isotropic architectures (with constant spatial resolutions). We replace the self-attention in DeiT-S [52] with our g^nConv and adjust the number of blocks to 13 to obtain the isotropic HorNet-S$_{7\times7}$ and HorNet-S$_{\mathrm{GF}}$. We compare DeiT-S, isotropic ConvNeXt-S and isotropic HorNet-S in Table 6b. While isotropic ConvNeXt-S cannot improve DeiT-S, our isotropic HorNet surpasses DeiT-S by a large margin. These results indicate that our g^nConv can better realize the functions of self-attention compared to plain convolutions and have better ability to model the complex spatial interactions.

g^nConv for other operations. To further demonstrate the universality of g^nConv, we use 3×3 depth-wise convolution and 3×3 pooling [67] as the basic operation in the g^nConv. The results in Table 6c show that g^nConv can also improve these two operations by large margins, indicating our g^nConv is potentially more powerful when equipped with some better basic operations.

Accuracy-complexity trade-offs. We visualize accuracy-complexity trade-offs of Swin, ConvNeXt and HorNet series in Figure 3. For fair comparisons, we fix the input image size to 224×224 and use HorNet$_{7\times7}$ such that all the compared models are based on 7×7 local window. We see HorNet can achieve better trade-offs than the representative vision Transformers and modern CNNs with regards to model size, FLOPs and GPU latency.

Visualization. We provide some visualizations of the adaptive weights learned by g^nConv in Figure 4. For each sample, we show the value of $\frac{1}{C}\sum_{c=1}^{C} h_{ij}^c$ (see Equation (3.8) or the definition of h_{ij}^c) for two random spatial locations i from layer {1, 3, 5, 7, 8, 12} of the isotropic HorNet-S model. Figure 4 demonstrates that the spatial mixing weights of our g^nConv are adaptive both to input samples and spatial locations, which further indicates that g^nConv shares these two desirable characteristics with the self-attention operation.

Limitations. While HorNet shows better overall latency-accuracy trade-offs, we notice that HorNet is slower than ConvNeXt with similar FLOPs on GPU, which may be caused by the more complex designs to perform the high-order interactions. We think that developing a more hardware-friendly operation for high-order spatial interactions is an interesting future direction to improve our work.

5 Conclusion

We have presented the Recursive Gated Convolution (g^nConv) that performs efficient, extendable, and translation-equivariant high-order spatial interactions with gated convolutions and recursive deigns. g^nConv can serve as a drop-in replace of the spatial mixing layer in various vision Transformers and convolution-based models. Based on the operation, we have constructed a new family of generic vision backbones HorNet. Extensive experiments demonstrate the effectiveness of g^nConv and HorNet on commonly used visual recognition benchmarks. We hope our attempt can inspire future work to further explore the high-order spatial interactions in vision models.

Acknowledgments

Jiwen Lu was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, the National Natural Science Foundation of China under Grant 62125603 and Grant U1813218, and a grant from the Beijing Academy of Artificial Intelligence (BAAI).

References

[1] Chunrong Ai and Edward C Norton. Interaction terms in logit and probit models. *Economics letters*, 80(1):123–129, 2003. 5, 15

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 7, 8, 17

[3] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, pages 4974–4983, 2019. 8, 9

[4] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *CVPR*, pages 11030–11039, 2020. 2

[5] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 8

[6] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv preprint arXiv:2112.01527*, 2021. 1

[7] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, pages 1290–1299, 2022. 8, 9

[8] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *NeurIPS*, 34, 2021. 1

[9] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *NeurIPS*, 34, 2021. 3, 4

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, pages 702–703, 2020. 16

[11] Yutao Cui, Jiang Cheng, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. *CVPR*, 2022. 3

[12] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *NeurIPS*, 34:3965–3977, 2021. 1, 3

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2, 6, 16, 17

[14] Xiaohan Ding, Xiangyu Zhang, Yizhuang Zhou, Jungong Han, Guiguang Ding, and Jian Sun. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. *CVPR*, 2022. 2, 3, 5, 7

[15] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *CVPR*, 2022. 3, 7

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3, 4, 7, 9

[17] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*, pages 2286–2296, 2021. 3

[18] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, pages 6824–6835, 2021. 1, 3

[19] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022. 3, 7

[20] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. Demystifying local vision transformer: Sparse connectivity, weight sharing, and dynamic weight. *arXiv preprint arXiv:2106.04263*, 2021. 2, 3

[21] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 7, 8

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 4, 5, 8, 16

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 16

[24] 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. 1

[25] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 2, 9, 18

[26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016. 17

[27] Shihua Huang, Zhichao Lu, Ran Cheng, and Cheng He. Fapn: Feature-aligned pyramid network for dense image prediction. In *ICCV*, pages 864–873, 2021. 8

[28] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *NeurIPS*, 29, 2016. 2

[29] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. *arXiv preprint arXiv:2104.10858*, 2021. 3

[30] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, pages 491–507. Springer, 2020. 7

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25:1097–1105, 2012. 1

[32] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1

[33] Samuel Lerman, Charles Venuto, Henry Kautz, and Chenliang Xu. Explaining local, global, and higher-order interactions in deep learning. In *ICCV*, pages 1224–1233, 2021. 5, 15

[34] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022. 2

[35] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *CVPR*, pages 4804–4814, 2022. 17, 18

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 3

[37] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 6, 8

[38] 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 *ECCV*, pages 740–755. Springer, 2014. 2, 6, 7

[39] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *NeurIPS*, 34:9204–9215, 2021. 16

[40] Kai Liu, Tianyi Wu, Cong Liu, and Guodong Guo. Dynamic group transformer: A general vision transformer backbone with dynamic group attention. *IJCAI*, 2022. 18

[41] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021. 1, 7

[42] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 17

[43] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022. 2, 3, 5, 6, 7, 8, 9, 16, 17

[44] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7, 17

[45] Xuran Pan, Chunjiang Ge, Rui Lu, Shiji Song, Guanfu Chen, Zeyi Huang, and Gao Huang. On the integration of self-attention and convolution. *arXiv preprint arXiv:2111.14556*, 2021. 3

[46] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. In *NeurIPS*, 2021. 2, 3, 5, 9

[47] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv:2104.10972*, 2021. 17

[48] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *NeurIPS*, 34, 2021. 1

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 4

[50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1

[51] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114, 2019. 1, 5, 7

[52] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1, 4, 6, 9, 10

[53] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 16

[54] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxim: Multi-axis mlp for image processing. In *CVPR*, pages 5769–5780, 2022. 16

[55] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *ECCV*, 2022. 3

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 2, 3, 5, 6

[57] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 4

[58] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. 3

[59] Yan Wang, Lingxi Xie, Chenxi Liu, Siyuan Qiao, Ya Zhang, Wenjun Zhang, Qi Tian, and Alan L. Yuille. SORT: second-order response transform for visual recognition. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1368–1377. IEEE Computer Society, 2017. 16

[60] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 3

[61] Sitong Wu, Tianyi Wu, Haoru Tan, and Guodong Guo. Pale transformer: A general vision transformer backbone with pale-shaped attention. In *AAAI*, volume 36, pages 2731–2739, 2022. 18

[62] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018. 7, 8, 17

[63] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *NeurIPS*, 34:30392–30400, 2021. 3

[64] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. *arXiv preprint arXiv:2201.04288*, 2022. 1

[65] Jianwei Yang, Chunyuan Li, and Jianfeng Gao. Focal modulation networks. *arXiv preprint arXiv:2203.11926*, 2022. 3

[66] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. *NeurIPS*, 34, 2021. 3

[67] 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. *arXiv preprint arXiv:2111.11418*, 2021. 3, 9, 10

[68] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv:2101.11986*, 2021. 3

[69] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5

[70] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022. 1, 8, 9

[71] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017. 2, 6, 7

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See our analysis in Section 4.3.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We propose a general framework for visual recognition. Our method is not for specific applications, which does not directly involve societal issues.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We follow the common practice in previous papers, where they didn't report the error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See our implementation details provided in the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A FLOPs of g^nConv

We will divide the computation of our g^nConv into 3 parts, and calculate the FLOPs for each part.

- **Projection layers.** The FLOPs of two projection layers ϕ_{in} and ϕ_{out} can be easily derived as:

$$\text{FLOPs}(\phi_{\text{in}}) = 2HWC^2, \quad \text{FLOPs}(\phi_{\text{out}}) = HWC^2 \tag{A.1}$$

- **Depth-wise convolution.** We first consider the standard depth-wise convolution (DWConv) with kernel size K. The DWConv is performed for all $\{\mathbf{q}_k\}_{k=1}^{n-1}$, where $\mathbf{q}_k \in \mathbb{R}^{HW \times C_k}$ and $C_k = \frac{C}{2^{n-k-1}}$. Therefore, the FLOPs for DWConv are

$$\text{FLOPs}(\text{DWConv}) = HWK^2 \sum_{k=0}^{n-1} \frac{C}{2^{n-k-1}} = 2HWCK^2 \left(1 - \frac{1}{2^n}\right). \tag{A.2}$$

- **Recursive Gating.** We consider both the flops of the projection layer g_k and the element-wise multiplication.

$$\begin{aligned} \text{FLOPs}(\text{RecursiveGating}) &= HWC_0 + \sum_{k=1}^{n-1} (HWC_{k-1}C_k + HWC_k) \\ &= HWC \left[\frac{2}{3}C\left(1 - \frac{1}{4^{n-1}}\right) + 2 - \frac{1}{2^{n-1}}\right]. \end{aligned} \tag{A.3}$$

Therefore, the total FLOPs are:

$$\text{FLOPs}(g^n\text{Conv}) = HWC \left[2K^2\left(1 - \frac{1}{2^n}\right) + \left(\frac{11}{3} - \frac{2}{3 \times 4^{n-1}}\right)C + 2 - \frac{1}{2^{n-1}}\right]. \tag{A.4}$$

B Spatial Interactions in Vision Models.

We review some representative vision model designs from the perspective of spatial interactions, as shown in Figure 1. Specifically, we are interested in the interactions between a feature $\mathbf{x}_i$ and its neighbor feature $\mathbf{x}_j, j \in \Omega_i$. Inspired by the interaction effect (IE) [33, 1], we consider that a binary function $F(\mathbf{x}_i, \mathbf{x}_j)$ which directly operates on $\mathbf{x}_i, \mathbf{x}_j$ introduces an effective interaction between $\mathbf{x}_i$ $\mathbf{x}_j$, if

$$\text{IE}(F) = \frac{\partial F}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \neq \mathbf{0}. \tag{B.1}$$

We now analyze the cases in Figure 1 of our main paper using the above rule. **(a): Convolution.** The output $F_i = \sum_{j \in \Omega} w_{i \to j} \mathbf{x}_j$, which leads to $\text{IE}(F) = \mathbf{0}$. Therefore, standard convolution introduce no interaction between $\mathbf{x}_i$ and $\mathbf{x}_j$ and we call it a *0-order interaction*. **(b): SE Block/Gated Convolution.** In this case, we have $F_i = \sum_{j \in \Omega} w_{i \to j} \mathbf{x}_j s_i(\mathbf{x})$, where $s_i(\mathbf{x}) = \frac{1}{HW} \sum_{l=1}^{HW} x_l$ for the SE block and $s_i(\mathbf{x}) = \mathbf{x}_i$ for the gated convolution. It is easy to show $\text{IE}(F) \neq \mathbf{0}$ because $\frac{\partial s_i}{\partial \mathbf{x}_i} \neq \mathbf{0}$. Hence, these two operations both introduce *1-order interaction*. **(c): Self-attention (SA).** We first denote the projected query/key/value features as $\mathbf{q}, \mathbf{k}, \mathbf{v}$. The SA first perform an 1-order interaction by computing the attention with dot-product: $\mathbf{a}_i = \mathbf{q}_i^\top [\mathbf{k}_1, \ldots, \mathbf{k}_{HW}]/\sqrt{C}$. We then view $\mathbf{a}_i$ as the feature at location i in the following computation. The normalized $\hat{\mathbf{a}}_i$ is then obtained by Softmax, which do not contribute to the order since it can be viewed as an implicit interaction that does not explicitly introduce $\mathbf{x}_j$ to the computation. The second interaction is performed by $\mathbf{x}_i = \sum_{j \in \Omega} \hat{\mathbf{a}}_i \mathbf{v}_j$. To sum up, the SA is a *2-order interaction*. **(d)**: g^nConv. According to Section 3.1, we have already known that g^nConv can achieve n-order interaction with bounded computational cost.

From the above discussion, we reveal a key difference between ViTs and previous architectures from a new view, *i.e.*, ViTs have higher-order spatial interactions in each basic block. Then it begs the question that whether we can achieve better accuracy-complexity trade-offs viz interactions with more than 2 orders. Our proposed g^nConv exactly targets this question for the first time. First, we can realize arbitrary n-order interaction as long as $1 \leq n \leq 1 + \log_2 C$ easily. Second, unlike the

Table 7: The detailed architectures of ConvNeXt [43], Swin Transformers [42], and HorNet.

	Output Size	**ConvNeXt-S/B/L** C=96/128/192	**Swin-S/B/L** C=96/128/192	**HorNet-T/S/B/L** C=64/96/128/192
Stem	56×56	$\text{Conv}_{4\times4}, C$, stride 4	$\text{Conv}_{4\times4}, C$, stride 4	$\text{Conv}_{4\times4}, C$, stride 4
Stage1	56×56	$\begin{bmatrix}\text{DWConv}_{7\times7}, C\\ \text{MLP}, 4C, C\end{bmatrix}\times 3$	$\begin{bmatrix}\text{MSA}_{7\times7}^{H=C/32}, C\\ [\text{MLP}, 4C, C]\end{bmatrix}\times 2$	$\begin{bmatrix}g^2\text{Conv}_{7\times7/\text{GF}}, C\\ [\text{MLP}, 4C, C]\end{bmatrix}\times 2$
Stage2	28×28	$\begin{bmatrix}\text{DWConv}_{7\times7}, 2C\\ \text{MLP}, 8C, 2C\end{bmatrix}\times 3$	$\begin{bmatrix}\text{MSA}_{7\times7}^{H=C/32}, 2C\\ [\text{MLP}, 8C, 2C]\end{bmatrix}\times 2$	$\begin{bmatrix}g^3\text{Conv}_{7\times7/\text{GF}}, 2C\\ [\text{MLP}, 8C, 2C]\end{bmatrix}\times 3$
Stage3	14×14	$\begin{bmatrix}\text{DWConv}_{7\times7}, 4C\\ \text{MLP}, 16C, 4C\end{bmatrix}\times 27$	$\begin{bmatrix}\text{MSA}_{7\times7}^{H=C/32}, 4C\\ [\text{MLP}, 16C, 4C]\end{bmatrix}\times 18$	$\begin{bmatrix}g^4\text{Conv}_{7\times7/\text{GF}}, 4C\\ [\text{MLP}, 16C, 4C]\end{bmatrix}\times 18$
Stage4	7×7	$\begin{bmatrix}\text{DWConv}_{7\times7}, 8C\\ \text{MLP}, 32C, 8C\end{bmatrix}\times 3$	$\begin{bmatrix}\text{MSA}_{7\times7}^{H=C/32}, 8C\\ [\text{MLP}, 32C, 8C]\end{bmatrix}\times 2$	$\begin{bmatrix}g^5\text{Conv}_{7\times7/\text{GF}}, 8C\\ [\text{MLP}, 32C, 8C]\end{bmatrix}\times 2$
Classifier		Global Average Pooling, Linear		

quadratic complexity of self-attention, the computational cost of g^nConv has an upper bound w.r.t. the order n.

In our implementation of g^nConv, the higher-order spatial interactions are based on the gating mechanism, which has also been investigated in LSTM [23] and some vision modules [59, 54, 39]. However, these previous methods can only achieve up to 2-order interactions, and did not fully reveal the potential of higher-order interactions. On the contrary, our g^nConv is more extendable to achieve arbitrary higher-order spatial interactions under a controllable computational budget.

C Implementation Details

C.1 Architecture Details.

To better verify the effectiveness of our new designs, we introduce minimal changes in the overall architecture of Swin Transformers [42]. Specifically, we make two changes to the overall architecture of Swin Transformers [42]: 1) We add one block in stage 2 to make the overall computation and parameters close to previous models; 2) We use the LayerScale [53] techniques to make our models more stable during training following the practice of ConvNeXt [43]. Note that the two changes have been applied to the baseline model considered in our ablation study to clearly show the effects of our designs. The detailed architectures of ConvNeXt [43], Swin Transformers [42] and HorNet are summarized in Table 7.

C.2 Experimental Settings for Image Classification.

ImageNet-1K training. ImageNet-1K [13] is a widely used large-scale benchmark for image classification, which contains around 1.2 million images from 1,000 categories. Following common practice [22, 43], we train our models on the training set of ImageNet and report the single-crop top-1 accuracy on 50,000 validation images. To fairly compare with our baseline methods (*i.e.*, Swin Transformers [42] and ConvNeXt [43]), we follow the most training details of ConvNeXt and make several small modifications to make the training configurations suitable for our models. For HorNet with 7×7 convolutions, we find that applying gradient clipping with a maximal norm of 5 will significantly stabilize the training process, which may be due to the large gradients brought by the high-order structures in our models. For HorNet with global filters, we use stronger regularization strategies since we find that larger kernels will improve the model capacity but may also cause more severe overfitting. Specifically, we set the gradient norm to 1 and use more aggressive RandAug [10] data augmentation strategies (*i.e.*, we adjust the magnitudes for tiny, small and base models to 9, 12 and 15, respectively). We set the stochastic depth coefficient of HorNet-T/S/B models to 0.2, 0.4 and 0.5. The other details are identical to ConvNeXt [43]. Our models are trained using 32 NVIDIA A100 GPUs with a global batch size of 4096.

Table 8: **Throughput analysis.** We provide the detailed throughput statistics of our models and several baseline methods. The throughput is measured with a single NVIDIA RTX 3090 GPU with a batch size of 128.

Model	FLOPs (G)	Throughput (img/s)	Top-1 Acc. (%)
ConvNeXt-T [43]	4.5	1010.3	82.1
Swin-T [42]	4.5	832.2	81.3
MViTv2-T [35]	4.7	728.4	82.3
HorNet-T$_{7\times7}$	4.0	845.7	82.8
ConvNeXt-S [43]	8.7	621.5	83.1
Swin-S [42]	8.7	520.7	83.0
MViTv2-S [35]	7.0	531.5	83.6
HorNet-S$_{7\times7}$	8.8	525.8	83.8
ConvNeXt-B [43]	15.4	440.8	83.8
Swin-B [42]	15.4	364.8	83.5
MViTv2-B [35]	10.2	369.1	84.4
HorNet-B$_{7\times7}$	15.6	410.0	84.2

ImageNet-22K training. ImageNet-22K [13] is a larger dataset that contains >21k classes and around 14M images. We use the subset suggested by [47] since the new `winter 2021 release` is the accessible version now. We also follow the [47] to remove categories with few images, resulting in roughly half fewer categories and only 13% fewer images compared to the original dataset. We follow previous practice [42, 43] to train our models for 90 epochs and use a similar data augmentation strategy as ImageNet-1K experiments. We set the stochastic depth coefficient [26] to 0.2. We also set the maximal gradient norm to 5 and 1 for our large models with standard 7×7 convolutions and global filters respectively. We also adjust the weight decay to 0.1. The other details are identical to ConvNeXt [43]. We also fine-tune our best model HorNet-L$_{\mathrm{GF}}$ on 384×384 images on ImageNet-22K for 10 epochs compete with state-of-the-art models on downstream tasks. The model is only used in the experiments in Appendix 4.2.

ImageNet-1K fine-tuning. We fine-tune the models pre-trained on ImageNet-22K or at the 224×224 resolution to ImageNet-1K or/and 384×384 resolution for 30 epochs with a batch size of 512 and a cosine learning rate schedule with an initial learning rate of $5e^{-5}$. We set the weight decay to $1e^{-6}$ and disable MixUp and CutMix following [43]. We initialize the ImageNet-1K classifier with the corresponding classifier weights for ImageNet-22K classes to further stabilize the training process.

C.3 Experimental Settings for Downstream Tasks.

Object detection and instance segmentation on COCO. We adopt the widely used Cascade Mask R-CNN [2] framework to perform object detection and instance segmentation on COCO, following Swin [42] and ConvNeXt [43]. Our backbones are pre-trained on ImageNet-1K for the HorNet-T/S/B and ImageNet-22K for the HorNet-L. We use the 3× schedule where we train all of our model for 36 epochs with AdamW [44] optimizer and a global batch size of 16. We set the learning rate of as {2e-4, 2e-4, 2e-4, 1e-4} and the stochastic depth rate as {0.4, 0.6, 0.7, 0.7}for HorNet-T/S/B/L. We set the weight decay as 0.05 for all the models.

Semantic Segmentation on ADE20K. We use the UperNet 160K [62] framework for semantic segmentation on ADE20K. We use a global batch size of 16 and train all the models for 160 iterations with the AdamW [44] optimizer. We use 512×512 image for ImageNet-1K pre-trained HorNet-T/S/B and 640×640 image for ImagNet-22K pre-trained HorNet-L. We set the learning rate as 1e-4 and the weight decay as 0.05 for all the models. We report the mIoU of both single-scale and multi-scale testing on the validation set.

Table 9: Effects of α.

α	1	2	3	5	10
ImageNet Top-1 Acc. (%)	82.71	82.76	**82.81**	82.74	82.69

Table 10: Effects of activation functions in gated convolutions.

	None	`Sigmoid`	`Tanh`	`GELU`
ImageNet Top-1 Acc. (%)	82.7	82.3 (-0.4)	82.6 (-0.1)	82.2 (-0.5)

D More Analysis and Visualization

Comparisons with state-of-the-art methods on ImageNet. Our experiments are designed to clearly verify the superior of our design over previous basic operations like plain convolution and self-attention. Therefore, we choose to follow the basic architecture and the training configuration of widely used architectures Swin Transformers and ConvNeXt Therefore, there is still substantial room to further improve the performance on ImageNet-1K. We notice that some recent work like Pale Transformers [61] and Dynamic Group Transformer [40] with hybrid architectures or more careful designs achieve better performance than HorNet on ImageNet-1K. We think many techniques that have been used in previous work can be useful to further imporve our models, including further optimized overall architectures (*e.g.*, optimized depth/width for each stage), better patch embedding strategies (*e.g.*, overlapping convolutional layers for input embedding and downsampling), more efficient ways to compute adaptive weights (*e.g.*, using downsmapled features to produce attention weights like [35]), and more advanced training methods and hybrid architectures (*e.g.*, combining g^nConv with self-attention and plain convolutions).

Throughput analysis. We provide the detailed throughput statistics of our models and several baseline methods in Table 8. Apart from ConvNeXt and Swin Transformers, we also compare our model with recent MViTv2 [35] models. The multiple small matrix multiplications introduced by g^nConv will affect the speed of our models on GPU. We observed that our method is slower than ConvNeXt by 7% 15% with similar FLOPs. Meanwhile, thanks to the highly efficient depth-wise convolutions implementation of CuDNN, we also see that our models achieve similar or slightly faster speed than typical vision Transformers with similar FLOPs. Notably, as shown in Figure 3(c), the higher classification accuracy helps our models achieve better speed-accuracy trade-offs than ConvNeXt and Swin Transformers. Therefore, we believe the speed of our method is still competitive with these recent models.

Effects of α. We find that re-scaling the output of gated convolution will avoid the large values produced by the recursive process and stabilize the training process. We analyze the the effects of α on our ImageNet experiments based on HorNet-B$_{7\times7}$. The results are summarized in Table 9. We see $\alpha = 3$ leads the best performance. Therefore, we set α to 3 in all our models.

Effects of activation functions in gated convolutions. The gated convolutions used in our models can be viewed as a type of channel attention that uses different attention weights in different locations and generates weights based on spatial interactions. Previous channel attention methods like SE-Net [25] usually add a `sigmoid` function to the attention weights to generate bounded attention. Therefore, we investigate several possible activation functions in our models. The results are presented in Table 10. We see the version described in our paper (*i.e.*, no activation function) achieves the best performance. The result also suggests that g^nConv exhibits a different behavior from conventional channel attention methods. Since the gating weights are critical components in our models, activation functions that can cause significant information losses like `GELU` and `sigmoid` will severely hurt the performance.