6-2024

InceptionNeXt: When Inception meets ConvNeXt

Weihao YU

Pan ZHOU
Singapore Management University, panzhou@smu.edu.sg

Shuicheng YAN

Xinchao WANG

InceptionNeXt: When Inception Meets ConvNeXt

Weihao Yu[1] Pan Zhou[2,3] Shuicheng Yan[4] Xinchao Wang[1*]
[1]National University of Singapore [2]Singapore Management University [3]Sea AI Lab [4]Skywork AI
weihaoyu@u.nus.edu panzhou@smu.edu.sg shuicheng.yan@kunlun-inc.com xinchao@nus.edu.sg

Code: https://github.com/sail-sg/inceptionnext

Abstract

Inspired by the long-range modeling ability of ViTs, large-kernel convolutions are widely studied and adopted recently to enlarge the receptive field and improve model performance, like the remarkable work ConvNeXt which employs 7×7 depthwise convolution. Although such depthwise operator only consumes a few FLOPs, it largely harms the model efficiency on powerful computing devices due to the high memory access costs. For example, ConvNeXt-T has similar FLOPs with ResNet-50 but only achieves $\sim 60\%$ throughputs when trained on A100 GPUs with full precision. Although reducing the kernel size of ConvNeXt can improve speed, it results in significant performance degradation, which poses a challenging problem: How to speed up large-kernel-based CNN models while preserving their performance. To tackle this issue, inspired by Inceptions, we propose to decompose large-kernel depthwise convolution into four parallel branches along channel dimension, i.e., small square kernel, two orthogonal band kernels, and an identity mapping. With this new Inception depthwise convolution, we build a series of networks, namely IncepitonNeXt, which not only enjoy high throughputs but also maintain competitive performance. For instance, InceptionNeXt-T achieves $1.6\times$ higher training throughputs than ConvNeX-T, as well as attains 0.2% top-1 accuracy improvement on ImageNet-1K. We anticipate InceptionNeXt can serve as an economical baseline for future architecture design to reduce carbon footprint.

1. Introduction

Reviewing the history of deep learning [35], Convolutional Neural Networks (CNNs) [33, 34] are definitely the most popular models in computer vision. The watershed moment arrived in 2012 when AlexNet [32] claimed victory in the ImageNet contest, ushering in a new era for CNNs in computer vision [11, 32, 54]. Since then, a myriad of influential CNNs has emerged like Network In Network [37], VGG [56], Inception Nets [58], ResNe(X)t [22, 74], DenseNet [27] and other efficient models [25, 55, 61, 62, 84].

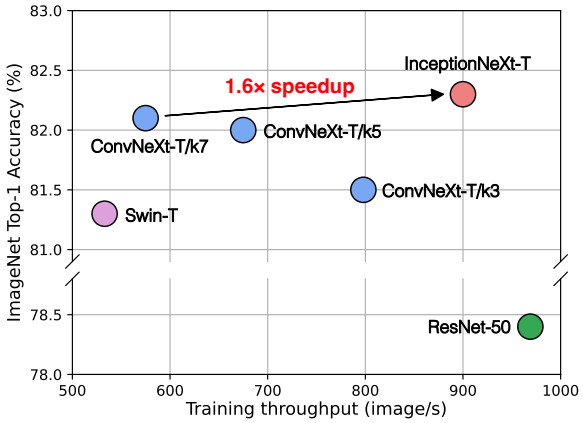

Figure 1. **Trade-off between accuracy and training throughput.** All models are trained under the DeiT training hyperparameters [39, 40, 64, 72]. The training throughput is measured on an A100 GPU with batch size of 128. ConvNeXt-T/kn means variants with depthwise convolution kernel size of $n \times n$. **InceptionNeXt-T enjoys both ResNet-50's speed and ConvNeXt-T's accuracy.**

Motivated by the great achievement of Transformer in NLP, researchers attempt to integrate its modules or blocks into vision CNN models [2, 4, 28, 70], *e.g.*, the representative works like Non-local Neural Networks [70] and DETR [4], or even make self-attention as stand-alone primitive [50, 85]. Moreover, inspired by the language generative pre-training [46], Image GPT (iGPT) [6] treats pixels as tokens and adopts pure Transformer for visual self-supervised learning. However, iGPT faces limitations in handling high-resolution images due to computational costs [6]. The breakthrough came with Vision Transformer (ViT) [16], which treats image patches as tokens, leverages a pure Transformer as the backbone, and has demonstrated remarkable performance in image classification after large-scale supervised image pre-training.

Apparently, the success of ViT [16] further ignites the enthusiasm for Transformer's application in computer vi-

*Corresponding Author.

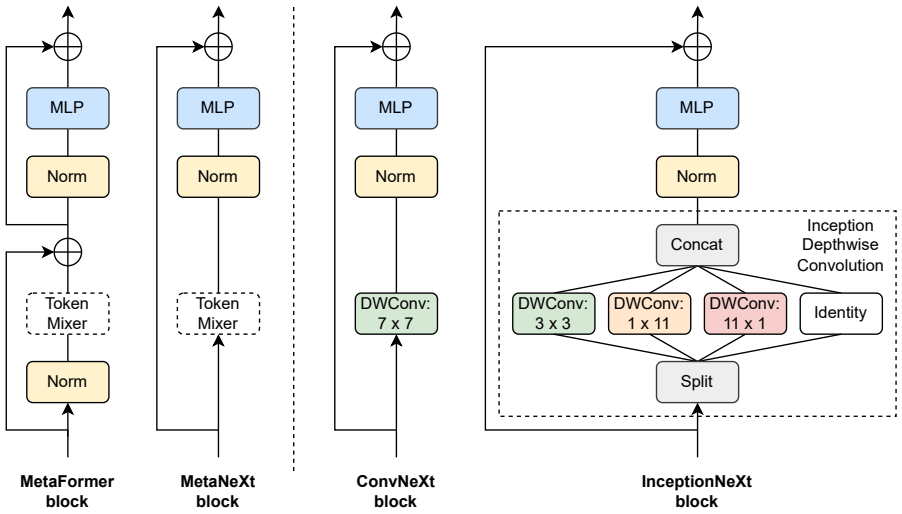

Figure 2. **Block illustration of MetaFormer, MetaNext, ConvNeXt and InceptionNeXt.** Similar to MetaFormer block [77], MetaNeXt is a general block abstracted from ConvNeXt [40]. MetaNeXt can be regarded as a simpler version obtained from MetaFormer by merging two residual sub-blocks into one. It is worth noting that the token mixer used in MetaNeXt cannot be too complex (*e.g.*, self-attention [66]) or it may fail to train to converge. By specifying the token mixer as depthwise convolution or Inception depthwise convolution, the model is instantiated as ConvNeXt or InceptionNeXt block. Compared with ConvNeXt, InceptionNeXt is more efficient because it decomposes expensive large-kernel depthwise convolution into four efficient parallel branches.

sion. Many ViT variants [15, 36, 39, 64, 67, 75, 79], like DeiT [64] and Swin [39], are proposed and have achieved remarkable performance across a wide range of vision tasks. The superior performance of ViT-like models over traditional CNNs (*e.g.*, Swin-T's 81.2% *v.s.* ResNet-50's 76.1% on ImageNet [11, 22, 39, 54]) leads many researchers to believe that Transformers will eventually replace CNNs and dominate the field of computer vision.

It is time for CNN to fight back. With advanced training techniques in DeiT [64] and Swin [39], the work of "ResNet strikes back" [72] shows that the performance of ResNet-50 can rise by 2.3%, up to 78.4%. Further, ConvNeXt [40] demonstrates that with modern modules like GELU [23] activation and large kernel size similar to attention window size [39], CNN models can consistently outperform Swin Transformer [39] in various settings and tasks. ConvNeXt is not alone: More and more works have shown similar observations [14, 18, 24, 38, 51, 68, 76, 78], like RepLKNet [14] and SLaK [38]. Among these modern CNN models, the common key feature is the large receptive field that is usually achieved by depthwise convolution [7, 43] with large kernel size (*e.g.*, 7×7).

However, despite its small FLOPs, depthwise convolution is actually an "expensive" operator because it brings high memory access costs and can be a bottleneck on powerful computing devices, like GPUs [42]. Moreover, as observed in [14], larger kernel sizes lead to significantly lower speeds. As shown in Figure 1, the ConvNeXt-T with a default 7×7 kernel size is $1.4\times$ slower than that with small kernel size of 3×3, and is $1.8\times$ slower than ResNet-50, although they have similar FLOPs. However, using a smaller kernel size limits the receptive field, which can result in performance degradation. For example, ConvNeXt-T/k3 suffers a performance drop of 0.6% top-1 accuracy on the ImageNet-1K dataset when compared to ConvNeXt-T/k7, where kn denotes a kernel size of $n \times n$.

This poses a challenging problem: How to speed up large-kernel CNNs while preserving their performance? In this paper, we aim to address this issue by building upon ConvNeXt as our baseline and improving the depthwise convolution module. Through our preliminary experiments based on ConvNeXt (see Table 1), we find that not all input channels need to undergo the computationally expensive depthwise convolution operation [42]. Accordingly, we propose to leave some channels unaltered and process only a portion of the channels with the depthwise convolution operation. Next, we propose to decompose large kernel of depthwise convolution into several groups of small kernels in Inception style [58–60]. Specifically, for the processing channels, $1/3$ of channels are conducted with kernel of 3×3, another $1/3$ are with $1 \times k$, and the remaining $1/3$ are with $k \times 1$. With this new simple and cheap operator, termed as "*Inception depthwise convolution*", our built model *InceptionNeXt* achieves a much better trade-off between accuracy and speed. For example, as shown in Figure 1, InceptionNeXt-T achieves higher accuracy than ConvNeXt-T while enjoying $1.6\times$ speedup of training throughput similar to ResNet-50.

The contributions of this paper are two-fold. Firstly, we identify the speed bottleneck of ConvNeXt as shown in Fig-

ure 1. To solve this speed bottleneck while keeping accuracy, we propose Inception depthwise convolution which decomposes the expensive depthwise convolution into three convolution branches with small kernel sizes as well as a branch of identity mapping. Secondly, extensive experiments on image classification and semantic segmentation show a better speed-accuracy trade-off of our model InceptionNeXt than ConvNeXt. We hope that InceptionNeXt can serve as a new CNN baseline to speed up the research of neural architecture design.

2. Related work

2.1. Transformer *v.s.* CNN

Transformer [66] was introduced in 2017 for NLP tasks because of its parallel training and also better performance than LSTM. Then many famous NLP models are built on Transformer, including GPT series [3, 44, 46, 47], BERT [12], T5 [49], and OPT [83]. For the application of the Transformer in vision tasks, Vision Transformer (ViT) is definitely the seminal work, showing that Transformer can achieve impressive performance after large-scale supervised training. Follow-up works [20, 52, 53, 64, 67, 69, 79] like Swin [39] continually improve model performance, achieving new state-of-the-art on various vision tasks. These results seem to tell us "Attention is all you need" [66].

But it is not that simple. ViT variants like DeiT usually adopt modern training procedures including various advanced techniques of data augmentation [9, 10, 80, 82, 86], regularization [26, 59] and optimizers [30, 41]. Wightman *et al.* find that with similar training procedures, the performance of ResNet can be largely improved. Besides, Yu *et al.* [77] argue that the general architecture instead of attention plays a key role in model performance. Han *et al.* [21] find by replacing attention in Swin with regular or dynamic depthwise convolution, the model can also obtain comparable performance. ConvNeXt [40], a remarkable work, modernizes ResNet into an advanced version with some designs from ViTs, and the resulting models consistently outperform Swin [39]. Other works like RepLKNet [14], VAN [18], FocalNets [76], HorNet [51], SLKNet [38], ConvFormer [78], Conv2Former [24], and InternImage [68] constantly improve performance of CNNs. Despite the high performance obtained, these models neglect efficiency, exhibiting lower speed than ConvNeXt. Actually, ConvNeXt is also not an efficient model compared with ResNet. We argue that CNN models should keep the original advantage of efficiency. Thus, in this paper, we aim to improve the model efficiency of CNNs while maintaining high performance.

2.2. Convolution with large kernels.

Well-known works, like AlexNet [32] and Inception v1 [58] already utilize large kernels up to 11×11 and 7×7, respectively. To improve the efficiency of large kernels, VGG [56] proposes to heavily stack 3×3 convolutions while Inception v3 [59] factorizes $k \times k$ convolution into $1 \times k$ and $k \times 1$ staking sequentially. For depthwise convolution, MixConv [63] splits kernels into several groups from 3×3 to $k \times k$. Besides, Peng *et al.* find that large kernels are important for semantic segmentation and they decompose large kernels similar to Inception v3 [59]. Witnessing the success of Transformer in vision tasks [16, 39, 67], large-kernel convolution is more emphasized since it can offer a large receptive field to imitate attention [21, 40]. For example, ConvNeXt adopts kernel size of 7×7 for depthwise convolution by default. To employ larger kernels, RepLKNet [14] proposes to utilize structural re-parameterization techniques [13, 81] to scale up kernel size to 31×31; VAN [18] sequentially stacks large-kernel depth-wise convolution (DW-Conv) and depth-wise dilation convolution to obtain 21×21 receptive filed; FocalNets [76] employ a gating mechanism to fuse multi-level features from stacking depthwise convolutions; SegNeXt [17] learns multi-scale features by multiple branches of staking $1 \times k$ and $k \times 1$. Recently, SLaK [38] factorizes large kernel $k \times k$ into two small non-square kernels ($k \times s$ and $s \times k$ with $s < k$). Unlike these works, we do not aim to scale up larger kernels. Instead, we target efficiency and decompose large kernels in a simple and speed-friendly way while keeping comparable performance.

3. Formulation and Method

3.1. MetaNeXt

Formulation of MetaNeXt Block. In ConvNeXt [40], for its each ConvNeXt block, the input X is first processed by a depthwise convolutioin to propagate information along spatial dimensions. We follow MetaFormer [77] to abstract the depthwise convolution as a *token mixer* which is responsible for spatial information interaction. Accordingly, as shown in the second subfigure in Figure 2, the ConvNeXt is abstracted as *MetaNeXt* block. Formally, in a MetaNeXt block, its input X is firstly processed as

$$X' = \text{TokenMixer}(X), \tag{1}$$

where $X, X' \in \mathbb{R}^{B \times C \times H \times W}$ with B, C, H and W respectively denoting batch size, channel number, height and width. Then the output from the token mixer is normalized

$$Y = \text{Norm}(X'). \tag{2}$$

After normalization [1, 29], the features are then fed into an MLP module which consists of two fully-connected layers with an activation function between them, the same as feed-forward network in Transformer [66]. The two fully-connected layers can also be implemented by 1×1 convolutions. Also, shortcut connection [22, 57] is adopted. This

Algorithm 1 Inception Depthwise Convolution (PyTorch-like Code)

```
import torch.nn as nn

class InceptionDWConv2d(nn.Module):
    def __init__(self, in_channels,
                 square_kernel_size=3, band_kernel_size=11,
                 branch_ratio=1/8):
        super().__init__()

        gc = int(in_channels * branch_ratio) # channel
              number of a convolution branch

        self.dwconv_hw = nn.Conv2d(gc, gc,
             square_kernel_size, padding=
             square_kernel_size//2, groups=gc)

        self.dwconv_w = nn.Conv2d(gc, gc, kernel_size
             =(1, band_kernel_size), padding=(0,
             band_kernel_size//2), groups=gc)

        self.dwconv_h = nn.Conv2d(gc, gc, kernel_size
             =(band_kernel_size, 1), padding=(
             band_kernel_size//2, 0), groups=gc)

        self.split_indexes = (gc, gc, gc, in_channels
             - 3 * gc)

    def forward(self, x):
        # B, C, H, W = x.shape
        x_hw, x_w, x_h, x_id = torch.split(x, self.
             split_indexes, dim=1)

        return torch.cat(
            (self.dwconv_hw(x_hw),
            self.dwconv_w(x_w),
            self.dwconv_h(x_h),
            x_id),
            dim=1)
```

process can be expressed by

$$Y = \mathrm{Conv}_{1\times1}^{rC\to C}\{\sigma[\mathrm{Conv}_{1\times1}^{C\to rC}(Y)]\} + X, \quad (3)$$

where $\mathrm{Conv}_{k\times k}^{C_i\to C_o}$ means convolution with kernel size of $k \times k$, input channels of C_i and output channels of C_o; r is the expansion ratio and σ denotes activation function.

Comparison to MetaFormer block. As shown in Figure 2, it can be found that MetaNeXt block shares similar modules with MetaFormer block [77], *e.g.*, token mixer and MLP. Nevertheless, a critical difference between the two models lies in the number of shortcut connections [22, 57]. MetaNeXt block implements a single shortcut connection, whereas the MetaFormer block incorporates two, one for the token mixer and the other for the MLP. From this aspect, MetaNeXt block can be regarded as a result of merging two residual sub-blocks from MetaFormer, thereby simplifying the overall architecture. As a result, the MetaNeXt architecture exhibits a higher speed compared to MetaFormer. However, this simpler design comes with a limitation: the token mixer component in MetaNeXt cannot be complicated (*e.g.*, Attention) as shown in our experiments (Table 5).

Instantiation to ConvNeXt. As shown in Figure 2, in ConvNeXt, the token mixer is simply implemented by a depthwise convolution

$$X' = \mathrm{TokenMixer}(X) = \mathrm{DWConv}_{k\times k}^{C\to C}(X), \quad (4)$$

where $\mathrm{DWConv}_{k\times k}^{C\to C}$ denotes depthwise convolution with kernel size of $k \times k$. In ConvNeXt, k is set as 7 by default.

3.2. Inception depthwise convolution

Kernel size of DWConv	Convolution ratio	Params (M)	MACs (G)	Throughput Train	Throughput Inference	Top-1 (%)
7×7	1.0	28.6	4.5	575	2413	82.1*
5×5	1.0	28.4	4.4	675	2704	82.0
3×3	1.0	28.3	4.4	798	2802	81.5
3×3	1/2	28.3	4.4	818	2740	81.4
3×3	3/8	28.3	4.4	847	2762	81.4
3×3	1/4	28.3	4.4	871	2808	81.3
3×3	1/8	28.3	4.4	901	2833	80.8
3×3	1/16	28.3	4.4	916	2846	80.1

Table 1. **Preliminary experiments based on ConvNeXt-T.** Convolution ratio means the ratio of channels to be processed by depthwise convolution while the other channels keep unchanged. Throughputs are measured on an A100 GPU with batch size of 128 and TF32. * The result is reported in ConvNeXt paper [40].

Preliminary experiments on ConvNeXt-T. We first conducted preliminary experiments based on ConvNeXt-T and report the results in Table 1. Firstly, the kernel size of depthwise convolution is reduced from 7×7 to 3×3. Compared to the model with kernel size of 7×7, the one with kernel size of 3×3 enjoys $1.4\times$ higher training throughput, but suffers a significant performance drop from 82.1% to 81.5%. Next, inspired by ShuffleNet V2 [42], we only feed partial input channels into depthwise convolution while the remaining ones keep unchanged. The number of processed input channels is controlled by a ratio. It is found that when the ratio is reduced from 1 to $1/4$, the training throughput can be further improved while the performance almost maintains. In summary, these preliminary experiments convey two findings on ConvNeXt. Finding 1: Large-kernel depthwise convolution is the speed bottleneck. Finding 2: Processing partial channels is good enough in single depthwise convolution layer [42].

Formulation. Based on the above findings, we propose a new type of convolution to keep both accuracy and effi-

Conv. type	Params	FLOPs
Conventional conv.	k^2C^2	$2k^2C^2HW$
Depthwise conv.	k^2C	$2k^2CHW$
Inception dep. conv.	$(2k+9)C/8$	$(2k+9)CHW/4$

Table 2. **Complexity of different types of convolution.** For simplicity, assume input and output channels are the same, and the bias term is omitted. k, C, H and W denote kernel size, channel number, height and width, respectively. The parameters and FLOPs of vanilla convolution and depthwise convolution are quadratic to kernel size k. In contrast, Inception depthwise convolution is linear to k.

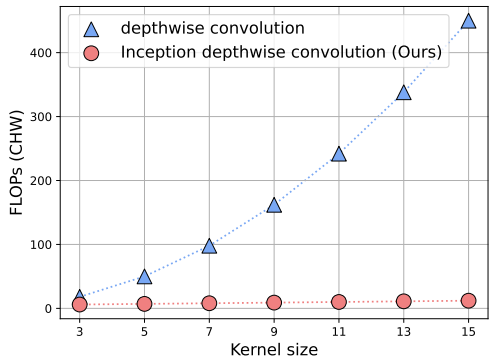

Figure 3. **Comparison of FLOPs between depthwise convolution and Inception depthwise convolution.** Inception depthwise convolution is much more efficient than depthwise convolution as kernel size increases.

Stage	#Tokens	Layer Specification		InceptionNeXt			
				A	T	S	B
1	$\frac{H}{4} \times \frac{W}{4}$	Down-sampling	Kernel Size	4×4, stride 4			
			Embed. Dim.	40	96		128
		InceptionNeXt Block	Band kernel size	9	11		
			Conv. group ratio	1/4	1/8		
			MLP Ratio	4			
			# Block	2	3		
2	$\frac{H}{8} \times \frac{W}{8}$	Down-sampling	Kernel Size	2×2, stride 2			
			Embed. Dim.	90	192		256
		InceptionNeXt Block	Band kernel size	9	11		
			Conv. group ratio	1/4	1/8		
			MLP Ratio	4			
			# Block	2	3		
3	$\frac{H}{16} \times \frac{W}{16}$	Down-sampling	Kernel Size	2×2, stride 2			
			Embed. Dim.	180	384		512
		InceptionNeXt Block	Band kernel size	9	11		
			Conv. group ratio	1/4	1/8		
			MLP Ratio	4			
			# Block	6	9	27	
4	$\frac{H}{32} \times \frac{W}{32}$	Down-sampling	Kernel Size	2×2, stride 2			
			Embed. Dim.	320	768		1024
		InceptionNeXt Block	Band kernel size	9	11		
			Conv. group ratio	1/4	1/8		
			MLP Ratio	3			
			# Block	2	3		
Global average pooling, MLP							
Parameters (M)				4.2	28.1	49.4	86.7
MACs (G)				0.5	4.2	8.4	14.9

Table 3. **Configurations of InceptionNeXt models** which have similar model configurations to ConvNeXt [40]. "A", "T", "S" and "B" represent "Atto", "Tiny", "Small" and "Base", respectively.

ciency. According to Fingding 2, we leave partial channels unchanged and denote them as a branch of identity mapping. Motivated by Fingding 1, for the processing channels, we propose to decompose the depthwise operations in Inception style [58–60]. Inception [58] utilizes several branches of small kernels (*e.g.*, 3×3) and large kernels (*e.g.*, 5×5). Similarly, we adopt 3×3 as one of our branches but get rid of the usage of the large square kernels because of their slow practical speed. Instead, large kernel $k_h \times k_w$ is decomposed as $1 \times k_w$ and $k_h \times 1$ inspired by Inception v3 [59].

Specifically, for input X, we split it into four groups along the channel dimension,

$$\begin{aligned} X_{\text{hw}}, X_{\text{w}}, X_{\text{h}}, X_{\text{id}} &= \text{Split}(X) \\ &= X_{:,:g}, X_{:g:2g}, X_{:2g:3g}, X_{:3g:}, \end{aligned} \tag{5}$$

where g is the channel numbers of convolution branches. We can set a ratio r_g to determine the branch channel numbers by $g = r_g C$. Next, the splitting inputs are fed into different parallel branches,

$$\begin{aligned} X'_{\text{hw}} &= \text{DWConv}^{g \to g}_{k_s \times k_s} g(X_{\text{hw}}), \\ X'_{\text{w}} &= \text{DWConv}^{g \to g}_{1 \times k_b} g(X_{\text{w}}), \\ X'_{\text{h}} &= \text{DWConv}^{g \to g}_{k_b \times 1} g(X_{\text{h}}), \\ X'_{\text{id}} &= X_{\text{id}}, \end{aligned} \tag{6}$$

where k_s denotes the small square kernel size set as 3 by default; k_b represents the band kernel size set as 11 by default. Finally, the outputs from each branch are concatenated,

$$X' = \text{Concat}(X'_{\text{hw}}, X'_{\text{w}}, X'_{\text{h}}, X'_{\text{id}}). \tag{7}$$

The illustration of InceptionNeXt block is shown in Figure 2. Moreover, its PyTorch [45] code is summarized in Algorithm 1.

Complexity. The complexity of three types of convolution, *i.e.*, conventional, depthwise, and Inception depthwise convolution is shown in Table 2. As can be seen, Inception depthwise convolution is much more efficient than the other two types of convolution in terms of parameter numbers of FLOPs. Inception depthwise convolution consumes parameters and FLOPs linear to both channel and kernel size. The comparison of depthwise and Inception depthwise convolutions regarding FLOPs is also clearly shown in Figure 3.

3.3. InceptionNeXt

Based on InceptionNeXt block, we can build a series of models named InceptionNeXt. Since ConvNeXt [40] is the our main comparing baseline, we mainly follow it to build models with several sizes [71]. Specifically, similar to ResNet [22] and ConvNeXt, InceptionNeXt also adopts 4-stage framework. The same as ConvNeXt, the numbers of 4 stages are [2, 2, 6, 2] for atto size, [3, 3, 9, 3] for small size and [3, 3, 27, 3] for base size. We adopt Batch Normalization since this paper emphasizes speed. Another difference with ConvNeXt is that InceptionNeXt uses an MLP ratio of 3 in stage 4 and moves the saved parameters to the classifier, which can help reduce a few FLOPs (*e.g.*, 3% for base size). The detailed model configurations are reported in Table 3.

Model	Mixing Type	Image size	Params (M)	MACs (G)	Throughput (img/s) A100 Train	A100 Infer	2080Ti Train	2080Ti Infer	Top-1 (%)
MobileNetV2 (1.4) [55]	Conv	224^2	6.1	0.60	1001	5190	471	1859	74.7
EfficientNet-B0 [61]	Conv	224^2	5.3	0.40	954	5502	464	1944	77.1
GhostNet 1.3× [19]	Conv	224^2	7.3	0.24	946	7451	589	2757	75.7
ConvNeXt-A [40, 71]	Conv	224^2	3.7	0.55	835	4539	345	1568	75.7
InceptionNeXt-A (Ours)	Conv	224^2	4.2	0.51	$2661_{+219\%}$	$9876_{+118\%}$	$992_{+188\%}$	$3595_{+129\%}$	$75.3_{-0.4}$
DeiT-S [64]	Attn	224^2	22	4.6	1227	3781	276	784	79.8
T2T-ViT-14 [79]	Attn	224^2	22	4.8	–	–	–	–	81.5
TNT-S [20]	Attn	224^2	24	5.2	–	–	–	–	81.5
Swin-T [39]	Attn	224^2	29	4.5	564	1768	184	554	81.3
Focal-T [75]	Attn	224^2	29	4.9	–	–	–	–	82.2
ResNet-50 [22, 72]	Conv	224^2	26	4.1	969	3149	278	977	78.4
RSB-ResNet-50 [22, 72]	Conv	224^2	26	4.1	969	3149	278	977	79.8
RegNetY-4G [48, 72]	Conv	224^2	21	4.0	670	2694	222	859	81.3
FocalNet-T [76]	Conv	224^2	29	4.5	–	–	–	–	82.3
ConvNeXt-T [40]	Conv	224^2	29	4.5	575	2413	177	590	82.1
InceptionNeXt-T (Ours)	Conv	224^2	28	4.2	$901_{+57\%}$	$2900_{+20\%}$	$254_{+44\%}$	$822_{+39\%}$	$82.3_{+0.2}$
T2T-ViT-19 [79]	Attn	224^2	39	8.5	–	–	–	–	81.9
PVT-Medium [67]	Attn	224^2	44	6.7	–	–	–	–	81.2
Swin-S [39]	Attn	224^2	50	8.7	359	1131	109	328	83.0
Focal-S [75]	Attn	224^2	51	9.1	–	–	–	–	83.5
RSB-ResNet-101 [22, 72]	Conv	224^2	45	7.9	620	2057	168	592	81.3
RegNetY-8G [48, 72]	Conv	224^2	39	8.0	689	1326	124	480	82.1
FocalNet-S [76]	Conv	224^2	50	8.7	–	–	–	–	83.5
ConvNeXt-S [40]	Conv	224^2	50	8.7	361	1535	105	353	83.1
InceptionNeXt-S (Ours)	Conv	224^2	49	8.4	$521_{+44\%}$	$1750_{+14\%}$	$130_{+24\%}$	$447_{+27\%}$	$83.5_{+0.4}$
DeiT-B [64]	Attn	224^2	86	17.5	541	1608	86	259	81.8
T2T-ViT-24 [79]	Attn	224^2	64	13.8	–	–	–	–	82.3
TNT-B [20]	Attn	224^2	66	14.1	–	–	–	–	82.9
PVT-Large [67]	Attn	224^2	62	9.8	–	–	–	–	81.7
Swin-B [39]	Attn	224^2	88	15.4	271	843	72	223	83.5
Focal-B [75]	Attn	224^2	90	16.0	–	–	–	–	83.8
RSB-ResNet-152 [22, 72]	Conv	224^2	60	11.6	437	1457	115	415	81.8
RegNetY-16G [48, 72]	Conv	224^2	84	15.9	322	1100	76	295	82.2
RepLKNet-31B [14]	Conv	224^2	79	15.3	–	–	–	–	83.5
FocalNet-B [76]	Conv	224^2	89	15.4	–	–	–	–	83.9
ConvNeXt-B [40]	Conv	224^2	89	15.4	267	1122	68	236	83.8
InceptionNeXt-B (Ours)	Conv	224^2	87	14.9	$375_{+40\%}$	$1244_{+11\%}$	$80_{+18\%}$	$287_{+22\%}$	$84.0_{+0.2}$
DeiT-B [64]	Attn	384^2	86	55.4	131	361	25	73	83.1
Swin-B [39]	Attn	384^2	88	47.1	104	296	21	65	84.5
RepLKNet-31B [14]	Conv	384^2	79	45.1	–	–	–	–	84.8
ConvNeXt-B [40]	Conv	384^2	89	45.0	95	393	23	79	85.1
InceptionNeXt-B (Ours)	Conv	384^2	87	43.6	$139_{+46\%}$	$428_{+9\%}$	$27_{+17\%}$	$97_{+23\%}$	$85.2_{+0.1}$

Table 4. **Performance of models trained on ImageNet-1K.** The throughputs are measured on an A100 GPU (PyTorch 1.13.0 and CUDA 11.7.1) with TF32 (TensorFloat-32), and on a 2080Ti (PyTorch 1.8.1 and CUDA 10.2) with FP32. The batch size for throughput benchmarking is initially set as 128 and is reduced until the GPU can host. The better results of "Channel First" and "Channel Last" memory layouts are reported.

4. Experiment

4.1. Image classification

Setup. For the image classification task, ImageNet-1K [11, 54] is one of the most commonly-used benchmarks, which contains around 1.3 million images in the training set and 50 thousand images in the validation set. To fairly compare with the widely-used baselines, *e.g.*, Swin [39] and ConvNeXt [40], we mainly follow the training hyper-parameters from DeiT [64] without distillation. Specifically, the models are trained by AdamW [41] optimizer with a learning rate $lr = 0.001 \times \text{batchsize}/1024$ ($lr = 4e-3$ and $\text{batchsize} = 4096$ are used in this paper the same as ConvNeXt). Following DeiT, data augmentation includes standard random resized crop, horizontal flip, RandAugment [10], Mixup [82], CutMix [80], Random Erasing [86] and color jitter. For regularization, label smoothing [59], stochastic depth [26], and weight decay are adopted. Like ConvNeXt, we also use LayerScale [65], a technique to help train deep models. Our code is based on PyTorch [45] and timm [71].

Results. We compare InceptionNeXt with various state-of-the-art models, including attention-based and convolution-based models. As can be seen in Table 4, InceptionNeXt achieves highly competitive performance as well as

Model	Params (M)	MACs (G)	Throughput (img/s) Train	Infer	Top-1 (%)
DeiT-S [64]	22	4.6	276	784	79.8
MetaNeXt-Attn	22	4.6	288	816	3.9
ConvNeXt-S (*iso.*) [40]	22	4.3	270	879	79.7
InceptionNeXt-S (*iso.*)	22	4.2	310	998	79.7

Table 5. **Comparison among ViT, isotropic ConvNeXt and InceptionNeXt.** MetaNeXt-Attn is instantiated from MetaNeXt with token mixer of self-attention [66]. The throughputs are measured on 2080Ti (PyTorch 1.8.1 and CUDA 10.2) with FP32. The batch size for throughput benchmarking is initially set as 128 and is reduced until the GPU can host. The better results of "Channel First" and "Channel Last" memory layouts are reported.

enjoys higher speed. InceptionNeXt consistently enjoys better accuracy-speed trade-off than ConvNeXt [40]. For example, InceptionNeXt-T not only surpasses ConvNeXt-T by 0.2%, but also enjoys 1.6×/1.2× training/inference throughputs on A100 than ConvNeXts, similar to those of ResNet-50. That is to say, InceptionNeXt-T enjoys both ResNet-50's speed and ConvNeXt-T's accuracy. Moreover, following Swin and ConvNeXt, we also finetuned the InceptionNeXt-B trained at the resolution of 224×224 to 384×384 for 30 epochs. We can see that InceptionNeXt-B obtains higher train and inference throughputs than ConvNeXt-B while keeping competitive accuracy.

It is observed that the speed improvement is much more significant for the lightweight model size, and the improvement gradually becomes smaller when the model size scales up. The reason is that computation complexity of depthwise and Inception depthwise convolutions are linear to channel number, *i.e.*, $\mathcal{O}(C)$ where C is channel number. For MLPs, their computation complexity is $\mathcal{O}(C^2)$. For larger models (larger C), its computation is further dominated by MLPs. By only improving depthwise convolution, the speed improvement becomes smaller when the model is larger.

Besides the 4-stage framework [22, 39, 56], another notable one is ViT-style [16] isotropic architecture which has only one stage. To match the parameters and MACs of DeiT-S, we construct InceptionNeXt-S (*iso.*) following ConvNeXt-S (*iso.*) [40]. Specifically, we set the embedding dimension as 384 and the block number as 18. Besides, we build a model called MetaNeXt-Attn which is instantiated from MetaNeXt block by specifying self-attention as token mixer. The aim of this model is to investigate whether it is possible to merge two residual sub-blocks of the Transformer block into a single one. The experiment results are shown in Table 5. It can be seen that InceptionNeXt can also perform well with the isotropic architecture, demonstrating InceptionNeXt exhibits good generalization across different frameworks. It is worth noting that MetaNeXt-Attn could not be trained to converge and only achieved an accuracy of 3.9%. This result suggests that, unlike the token mixer in MetaFormer, the token mixer in MetaNeXt cannot be too complex. If it is, the model may not be trainable.

Backbone	UperNet			
	Params (M)	MACs (G)	FPS	mIoU (%)
Swin-T [39]	60	945	20.6	45.8
ConvNeXt-T [40]	60	939	20.6	46.7
InceptionNeXt-T	56	933	22.7	**47.9**
Swin-S [39]	81	1038	16.2	49.5
ConvNeXt-S [40]	82	1027	16.8	49.6
InceptionNeXt-S	78	1020	17.6	**50.0**
Swin-B [39]	121	1188	16.2	49.7
ConvNeXt-B [40]	122	1170	15.7	49.9
InceptionNeXt-B	115	1159	17.5	**50.6**

Table 6. **Performance of semantic segmentation with UperNet [73] on ADE20K [87] validation set.** Images are cropped to 512×512 for training. The MACs are measured with input size of 512×2048. The FPS are benchamrked on 2080Ti.

Backbone	Semantic FPN			
	Params (M)	MACs (G)	FPS	mIoU (%)
ResNet-50 [22]	29	46	30.2	36.7
PVT-Small [67]	28	45	27.2	39.8
PoolFormer-S24 [77]	23	39	28.8	40.3
InceptionNeXt-T	28	44	31.4	**43.1**
ResNet-101 [22]	48	65	22.2	38.8
ResNeXt-101-32x4d [74]	47	65	–	39.7
PVT-Medium [67]	48	61	20.0	41.6
PoolFormer-S36 [77]	35	48	21.6	42.0
PoolFormer-M36 [77]	60	68	15.4	42.4
InceptionNeXt-S	50	65	20.7	**45.6**
PVT-Large [67]	65	80	16.0	42.1
ResNeXt-101-64x4d [74]	86	104	–	40.2
PoolFormer-M48 [77]	77	82	12.1	42.7
InceptionNeXt-B	85	100	20.2	**46.4**

Table 7. **Performance of semantic segmentation with Semantic FPN [31] on ADE20K [87] validation set.** Images are cropped to 512×512 for training. The MACs are measured with input size of 512×512. The FPS are benchamrked on 2080Ti.

4.2. Semantic segmentation

Setup. ADE20K [87], a commonly used scene parsing benchmark, is used to evaluate our models on semantic segmentation task. ADE20K includes 150 fine-grained semantic categories, containing twenty thousand and two thousand images in the training set and validation set, respectively. The checkpoints trained on ImageNet-1K [11] at the resolution of 224^2 are utilized to initialize the backbones. Following Swin [39] and ConvNeXt [40], we firstly evaluate InceptionNeXt with UperNet [73]. The models are trained with AdamW [41] optimizer with learning rate of 6e-5 and batch size of 16 for 160K iterations. Following PVT [67] and PoolFormer [77], InceptionNeXt is also evaluated with Semantic FPN [31]. In common practices [5, 31], the batch size is 16 for the setting of 80K iterations. Following PoolFormer [77], we increase the batch size to 32 and decrease the iterations to 40K to speed up training. AdamW [30, 41] is adopted with a learning rate of 2e-4 and a polynomial decay schedule of 0.9 power. Our code is based on PyTorch [45] and mmsegmentation [8].

Results. For segmentation with UpNet [73], the results are shown in Table 6. As can be seen, InceptionNeXt con-

Ablation	Variant	Params (M)	MACs (G)	Throughput Train	Throughput Inference	Top-1 (%)
Baseline	None (InceptionNeXt-T)	28.1	4.2	901	2900	82.3
Branch	Remove horizontal band kernel	28.0	4.2	947	3093	81.9
	Remove vertical band kernel	28.0	4.2	954	3173	81.9
	Remove small band kernel	28.0	4.2	940	3004	82.0
	horizontal and vertical band kernel in parallel $\rightarrow$ in sequence	28.1	4.2	903	2971	82.1
Band kernel size	Band kernel size 11 $\rightarrow$ 7	28.0	4.2	905	2946	82.1
	Band kernel size 11 $\rightarrow$ 9	28.1	4.2	904	2916	82.1
	Band kernel size 11 $\rightarrow$ 13	28.1	4.2	896	2895	82.0
Convolution branch ratio	Conv. branch ratio $1/8 \rightarrow 1/4$	28.1	4.2	834	2499	82.2
	Conv. branch ratio $1/8 \rightarrow 1/16$	28.0	4.2	936	3097	81.8
Normalization	Batch Norm [29] $\rightarrow$ Layer Norm [1]	28.1	4.2	721	2646	82.4

Table 8. **Ablation for InceptionNeXt on ImageNet-1K classification benchmark.** InceptionNeXt-T is utilized as the baseline for the ablation study. Top-1 accuracy on the validation set is reported. The throughputs are measured on an A100 GPU (PyTorch 1.13.0 and CUDA 11.7.1) with TF32 and batch size of 128.

sistently outperforms Swin [39] and ConvNeXt [40] for different model sizes. In the method of Semantic FPN [31] as shown in Table 7, InceptionNeXt significantly surpasses other backbones, like PVT [67] and PoolFormer [77]. These results show that InceptionNeXt also has a high potential for dense prediction tasks.

4.3. Ablation studies

We conduct ablation studies on ImageNet-1K [11, 54] using InceptionNeXt-T as baseline from the following aspects.

Branch. Inception depthwise convolution includes four branches, three convolutional ones, and identity mapping. When removing any branch of horizontal or vertical band kernel, performance significantly drops from 82.3% to 81.9%, demonstrating the importance of these two branches. This is because these two branches with band kernels can enlarge the receptive field of the model. For the branch of small square kernel size of 3×3, removing it can still achieve up to 82.0% top-1 accuracy and bring higher throughput. This inspires us that if we attach more importance to the model speed, the simple version of InceptionNeXt without the square kernel of 3×3 can be adopted. For the band kernel, Inception v3 mostly equips them in a sequential way. We find that this assembling method can also obtain similar performance and even a little speed up the model. A possible reason is that PyTorch/CUDA may have optimized sequential convolutions well, and we only implement the parallel branches at a high level (see Algorithm 1). We believe the parallel method will be faster when it is optimized better. Thus, parallel method for the band kernels is adopted by default.

Band kernel size. It is found the performance can be improved from kernel size 7 to 11, but it drops when the band kernel size increases to 13. This phenomenon may result from the optimization and can be solved by methods like structural re-parameterization [13, 14]. For simplicity, we set the kernel size as 11 by default except for atto size.

Convolution branch ratio. When the ratio increases from $1/8$ to $1/4$, performance improvement can not be observed. Ma *et al.* [42] also point out that it is not necessary for all channels to conduct convolution. But when the ratio decreases to $1/16$, it brings a serious performance drop. This is because a smaller ratio would limit the degree of token mixing, resulting in performance drop. We thus set the convolution branch ratio as $1/8$ by default except for atto size.

Normalization. When replacing the Batch Normalization [29] with Layer Normalization [1], the performance improvement improve by 0.1% but suffer throughput drop in both training and inference. Since this paper focuses on efficiency, we adopt Batch Normalization for InceptionNeXt.

5. Conclusion

In this work, we propose an effective and efficient CNN architecture, InceptionNeXt, which enjoys a better trade-off between the practical speed and the performance than previous network architectures. InceptionNeXt decomposes large-kernel depthwise convolution along channel dimension into four parallel branches, including identity mapping, a small square kernel, and two orthogonal band kernels. All these four branches are much more computationally efficient than a large-kernel depthwise convolution in practice, and can also work together to have a large spatial receptive field for good performance. Extensive experimental results demonstrate the superior performance and the high practical efficiency of InceptionNeXt.

Acknowledgement. This project is supported by the National Research Foundation Singapore under its Medium Sized Center for Advanced Robotics Technology Innovation, and the Advanced Research and Technology Innovation Centre (ARTIC), the National University of Singapore under Grant (project number: A0005947-21-00, project reference: ECT-RP2). Weihao Yu was partly supported by Snap Research Fellowship, Google's TPU Research Cloud (TRC), and Google Cloud Research Credits program. Pan Zhou was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant. We would like to thank Ross Wightman for integrating the model and code into Hugging Face's pytorch-image-models repository.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3, 8

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019. 1

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. 1

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 7

[6] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020. 1

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 2

[8] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 7

[9] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 3

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 3, 6

[11] 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. 1, 2, 6, 7, 8

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 3

[13] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021. 3, 8

[14] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11963–11975, 2022. 2, 3, 6, 8

[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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12124–12134, 2022. 2

[16] 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. 1, 3, 7

[17] Meng-Hao Guo, Cheng-Ze Lu, Qibin Hou, Zhengning Liu, Ming-Ming Cheng, and Shi-Min Hu. Segnext: Rethinking convolutional attention design for semantic segmentation. *Advances in Neural Information Processing Systems*, 35:1140–1156, 2022. 3

[18] 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. 2, 3

[19] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589, 2020. 6

[20] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021. 3, 6

[21] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. On the connection between local attention and dynamic depth-wise convolution. *arXiv preprint arXiv:2106.04263*, 2021. 3

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 3, 4, 5, 6, 7

[23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2

[24] Qibin Hou, Cheng-Ze Lu, Ming-Ming Cheng, and Jiashi Feng. Conv2former: A simple transformer-style convnet for visual recognition. *arXiv preprint arXiv:2211.11943*, 2022. 2, 3

[25] 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

[26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Com-*

puter Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, pages 646–661. Springer, 2016. 3, 6

[27] 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. 1

[28] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 603–612, 2019. 1

[29] 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. 3, 8

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 3, 7

[31] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 7, 8

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1, 3

[33] 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

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1

[36] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022. 2

[37] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014. 1

[38] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022. 2, 3

[39] 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. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1, 2, 3, 6, 7, 8

[40] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 3, 6, 7

[42] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 2, 4, 8

[43] Franck Mamalet and Christophe Garcia. Simplifying convnets for fast learning. In *Artificial Neural Networks and Machine Learning–ICANN 2012: 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part II 22*, pages 58–65. Springer, 2012. 2

[44] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. 3

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5, 6, 7

[46] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. . 1, 3

[47] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. . 3

[48] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020. 6

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020. 3

[50] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, 32, 2019. 1

[51] Yongming Rao, Wenliang Zhao, Yansong Tang, Jie Zhou, Ser-Nam Lim, and Jiwen Lu. Hornet: Efficient high-order spatial interactions with recursive gated convolutions. *arXiv preprint arXiv:2207.14284*, 2022. 2, 3

[52] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng, and Xinchao Wang. Shunted self-attention via multi-scale token aggregation. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, pages 10853–10862, 2022. 3

[53] Sucheng Ren, Xingyi Yang, Songhua Liu, and Xinchao Wang. Sg-former: Self-guided transformer with evolving token reallocation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6003–6014, 2023. 3

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 1, 2, 6, 8

[55] 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. 1, 6

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1, 3, 7

[57] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 3, 4

[58] 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. 1, 2, 3, 5

[59] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 3, 5, 6

[60] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 2, 5

[61] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1, 6

[62] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 1

[63] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *arXiv preprint arXiv:1907.09595*, 2019. 3

[64] 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. 1, 2, 3, 6, 7

[65] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021. 6

[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3, 7

[67] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 2, 3, 6, 7, 8

[68] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. *arXiv preprint arXiv:2211.05778*, 2022. 2, 3

[69] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022. 3

[70] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 1

[71] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 5, 6

[72] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 1, 2, 6

[73] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 7

[74] 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. 1, 7

[75] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021. 2, 6

[76] Jianwei Yang, Chunyuan Li, and Jianfeng Gao. Focal modulation networks. *arXiv preprint arXiv:2203.11926*, 2022. 2, 3, 6

[77] 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. 2, 3, 4, 7, 8

[78] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2, 3

[79] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567, 2021. 2, 3, 6

[80] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 3, 6

[81] Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388*, 2017. 3

[82] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 3, 6

[83] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 3

[84] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 1

[85] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10076–10085, 2020. 1

[86] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008, 2020. 3, 6

[87] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 7