Can CNNs Be More Robust Than Transformers?

Zeyu Wang[1], Yutong Bai[2], Yuyin Zhou[1], and Cihang Xie[1]

[1] University of California, Santa Cruz
[2] Johns Hopkins University

Abstract. The recent success of Vision Transformers is shaking the long dominance of Convolutional Neural Networks (CNNs) in image recognition for a decade. Specifically, in terms of robustness on out-of-distribution samples, recent research finds that Transformers are inherently more robust than CNNs, regardless of different training setups. Moreover, it is believed that such superiority of Transformers should largely be credited to their *self-attention-like architectures per se*. In this paper, we question that belief by closely examining the design of Transformers. Our findings lead to three highly effective architecture designs for boosting robustness, yet simple enough to be implemented in several lines of code, namely a) patchifying input images, b) enlarging kernel size, and c) reducing activation layers and normalization layers. Bringing these components together, we are able to build pure CNN architectures without any attention-like operations that is as robust as, or even more robust than, Transformers. We hope this work can help the community better understand the design of robust neural architectures. The code is publicly available at https://github.com/UCSC-VLAA/RobustCNN.

Keywords: CNNs, Transformers, Out-of-Distribution Robustness

1 Introduction

The success of deep learning in computer vision is largely driven by Convolutional Neural Networks (CNNs). Starting from the milestone work AlexNet [23], CNNs keep pushing the frontier of computer vision and laying the foundation of modern recognition systems [32,15,33]. Interestingly, the recently emerged Vision Transformer (ViT) [11] challenges the leading position of CNNs in computer vision. ViT offers a completely different roadmap on building recognition models—by applying the pure self-attention-based architecture to sequences of images patches, ViTs are able to attain competitive performance on a wide range of visual benchmarks compared to CNNs.

The recent studies on out-of-distribution robustness [1,51,27] further heats up the debate between CNNs and Transformers. Unlike the situation on standard visual benchmarks where both models are closely matched, Transformers are much more robust than CNNs when testing out of the box. Moreover, Bai *et al.* [1] argue that, rather than being benefited by the advanced training recipe provided in [35], such strong out-of-distribution robustness comes inherently with

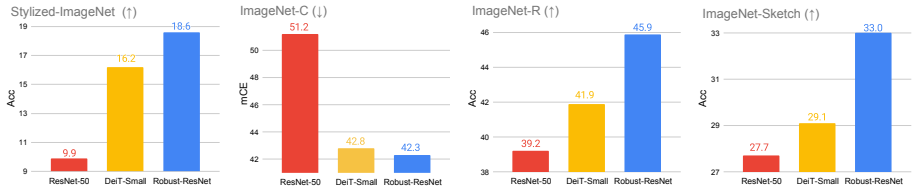

Fig. 1. Comparison of out-of-distribution robustness among ResNet, DeiT, and our enhanced ResNet (dubbed *Robust-ResNet*). Though DeiT-S largely outperforms the vanilla ResNet, it performs worse than Robust-ResNet on these robustness benchmarks.

the Transformer's self-attention-like architecture. For example, simply "upgrading" a pure CNN to a hybrid architecture (*i.e.*, with both CNN blocks and Transformer blocks) can effectively improve out-of-distribution robustness [1].

Though it is generally believed that the architecture difference is the key factor that leads to the robustness gap between Transformers and CNNs, existing works all fail to answer which *architectural elements* in Transformer should be attributed to such stronger robustness. The most relevant analysis is provided in [1,31]—both works point out that Transformer blocks, where self-attention operation is the pivot unit, are critical for robustness. Nonetheless, given a) Transformer block itself is already a compound design with multiple architectural elements, and b) Transformer also contains many other layers (*e.g.*, patch embedding layer) in addition to Transformer blocks, the relationship between robustness and Transformer's architectural elements remains confounding. In this work, we take a closer look at the architecture design of Transformers, focusing on out-of-distribution robustness. More importantly, we aim to explore, with the help of the architectural elements from Transformers, *whether CNNs can be robust learners as well*.

How to improve the robustness?
1. non-overlapped patch-embedding;
2. larger kernel size;
3.fewer Normalization and activation;

Our diagnose delivers three key messages for improving out-of-distribution robustness, from the perspective of neural architecture design. Firstly, patchifying images into non-overlapped patches can substantially contribute to out-of-distribution robustness; more interestingly, regarding the choice of patch size, we find the larger the better. Secondly, despite applying small convolutional kernels is a popular design recipe, we observe adopting a much larger convolutional kernel size (*e.g.*, from 3×3 to 7×7, or even to 11×11) is necessary for securing model robustness on out-of-distribution samples. Lastly, as inspired by the recent work [25], we note aggressively reducing the number of normalization layers and activation functions is beneficial for out-of-distribution robustness; meanwhile, as a byproduct, the training speed could be accelerated by up to ~23%, due to less normalization layers are used [4].

Our experiments verify that all these three architectural elements consistently and effectively improve out-of-distribution robustness on a set of CNN architectures. The largest improvement is reported by integrating all of them into CNNs' architecture design—as shown in Figure 1, without applying any self-attention-like components, our enhanced ResNet (dubbed Robust-ResNet) is able to outperform a similar-scale Transformer, DeiT-S, by 2.4% on Stylized-ImageNet (16.2% *vs*. 18.6%), 0.5% on ImageNet-C (42.8% *vs*. 42.3%), 4.0% on

ImageNet-R (41.9% *vs*. 45.9%) and 3.9% on ImageNet-Sketch (29.1% *vs*. 33.0%). We hope this work can help the community better understand the underlying principle of designing robust neural architectures.

2 Related Works

Vision Transformers. Transformers [38], which apply self-attention to enable global interactions between input elements, underpins the success of building foundation models in natural language processing [10,47,9,28,29,5,3]. Recently, Dosovitskiy *et al*. [11] show Transformer is also applicable to computer vision, attaining competitive performance compared to CNNs on the challenging ImageNet classification task. Later works keep pushing the potential of Transformer on a variety of visual tasks, in both supervised learning [35,48,24,40,48,50,36,46] and self-supervised learning [6,7,2,54,45,14], showing a seemly inevitable trend on replacing CNNs in computer vision.

CNNs striking back. There is a recent surge in work that aims to retake the position of CNNs as the favored neural architecture for building recognition models. Wightman *et al*. [42] demonstrate that, with the advanced training setup, the canonical ResNet-50 is able to achieve 80.4% top-1 accuracy on ImageNet, largely outperforming its vanilla version (76.1% top-1 accuracy). In addition, the prior works show modernizing CNNs' architecture design is essential—by either simply adopting Transformer's patchify layer [37] or aggressively bringing in every possible components from Transformer [25], CNNs are able to attain competitive performance w.r.t. Transformers across a range of visual benchmarks.

Our work is closely related to ConvNeXt [25], while shifting the study focus from standard accuracy to robustness. Moreover, rather than specifically offering a unique neural architecture as in [25], this paper aims to provide a set of useful architectural elements that allows CNNs to be able to match, or even outperform, Transformers when measuring robustness.

Out-of-distribution robustness. Dealing with data of shifted distributions is a commonly encountered problem when deploying models in the real world. To simulate such challenges, several out-of-distribution benchmarks have been established, including measuring model performance on images with common corruptions [17] or with various renditions [16]. Recently, a set of works [1,51,27] find that Transformers are inherently much more robustness than CNNs on out-of-distribution samples. Our work is a direct follow-up of Bai *et al*. [1], whereas stands on the opposite side—*we show CNNs can in turn outperform Transformers in out-of-distribution robustness*.

3 Settings

Following the previous work [1], in this paper the robustness comparison is fairly conducted between CNNs and Transformers based on the architecture of ResNet [15] and ViT [11], two of the most classic milestone architectures in the deep learning history.

CNN Block Instantiations. We consider four different block instantiations, as shown in Fig. 2. The first block is Depth-wise ResNet Bottleneck block [44], in which the 3×3 vanilla convolution layer in each block is replaced with a 3×3 depth-wise convolution layer. The second block is Inverted Depth-wise ResNet Bottleneck block, a design first proposed in [30], which is the same as Depth-wise ResNet Bottleneck block except that the hidden dimension is four times the input dimension. The third block is instantiated based on the second block by moving up the position of depthwise convolution layer as in ConvNeXT[25], which reduces the computation of the more complex 3×3 depthwise convolution layer and puts more weight on the 1×1 convolution layer. Note that this block design is similar to a ConvNeXT block [25]. But a ConvNeXT block comes with other modifications such as changes of activation and normalization layers. Similarly, based on the second block, the forth block moves down the position of depthwise convolution layer.

Note that in all four blocks, depth-wise convolution are adopted, since numerous studies show that it achieves a better computation/accuracy trade-off than a vanilla convolution [21,30,20,34]. We replace the bottleneck building block in the original ResNet architecture with these four blocks, and term the resulting models as ResNet-DW, ResNet-Inverted-DW, ResNet-Up-Inverted-DW, and ResNet-Down-Inverted-DW, respectively.

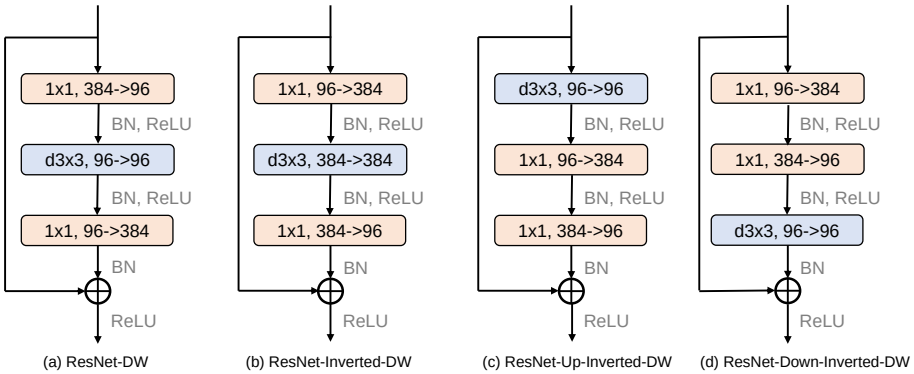

Fig. 2. Architectures of four different instantiations of CNN block.

Computational Cost. In this work, we consider measuring the model size in terms of FLOPs, i.e., FLOating Point operations per second. Here we note that directly replacing the ResNet bottleneck block with the above four instantiated blocks will significantly reduce the total FLOPs of the model, due to the usage of the depth-wise convolution. To mitigate the computational cost loss, and to increase model performance following the spirit of ResNeXT [44], we change the channel setting of each stage from (64, 128, 256, 512) to (96, 192, 384, 768). We then tune the block number in stage 3 to keep the total FLOPs of baseline models roughly the same as DeiT-S. The final FLOPs of our baseline models are shown in Tab. 1. Unless otherwise stated, all models considered in this work are of the similar scale as DeiT-S.

Table 1. Comparison of various baseline models. We replace the ResNet Bottleneck block with four different stronger building blocks, and keep the computational cost of baseline models on par with DeiT counterparts. We observe significant gain in terms of clean image accuracy, but our CNN baseline models are still less robust than DeiT.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5

Robustness Benchmarks. In this work, we extensively evaluate the model performance on out-of-distribution robustness using following benchmarks: 1) Stylized-ImageNet[12], which contains synthesized images with shape-texture conflicting cues; 2) ImageNet-C[17], with various common image corruptions; 3) ImageNet-R [16], which contains natural renditions of ImageNet object classes with different textures and local image statistics; 4) ImageNet-Sketch [39], which includes sketch images of the same ImageNet classes collected online.

Training Recipe. CNNs may achieve better robustness by simply tuning the training recipe [1,42]. Therefore in this work we deliberately apply the standard 300-epoch DeiT training recipe [35] for all models unless otherwise mentioned, so that the performance difference between different models can only be attributed to the architecture difference. Specifically, we train all models using AdamW optimizer [26]; We set the initial learning rate to 5e-4, and apply the cosine learning rate scheduler to decrease it; Besides weight decay, we additionally adopt six data augmentation & regularization strategies (i.e., RandAug [8], MixUp [52] ,CutMix [49], Random Erasing [53], Stochastic Depth [22], and Repeated Augmentation [19]) to regularize training.

Baseline Results. We train six models with the recipe specified above. The results are shown in Tab. 1. For simplicity, we use "IN", "S-IN", "IN-C", "IN-R", and "IN-SK" as abbreviations for "ImageNet", "Stylized-ImageNet", "ImageNet-C", "ImageNet-R", and "ImageNet-Sketch". We already know that DeiT-S shows better robustness generalization than ResNet50 from [1]. Interestingly, even when equipped with stronger depth-wise convolution layers, the ResNet architecture can only attain similar clean image accuracy while still being non-trivially less robust compared with the DeiT counterpart, suggesting the hidden magic lies in the architecture design of Vision Transformer.

4 Component Diagnosis

In this section, we introduce three effective architectural design inspired from Transformers which work surprisingly well on CNNs as well. Concretely, they

Table 2. The performance of different baseline models equipped with ViT-style patchify stem. "PX" refers to that the model has a ViT-style patchify stem with patch size set to X. The block number in stage 3 is increased when patch size is larger. With larger patch size, the models attain better robustness but worse clean accuracy. The results of the original ResNet50 and DeiT-S are included for reference.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
+ P4	4.6G	79.6	11.4	51.2	40.5	28.3
+ P8	4.6G	79.9	12.7	47.6	42.5	30.1
+ P16	4.6G	78.5	**17.0**	**44.3**	**44.8**	**32.0**
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
+ P4	4.5G	80.2	11.2	48.8	41.5	29.8
+ P8	4.6G	79.8	14.8	45.4	44.0	31.8
+ P16	4.6G	77.9	**17.2**	**42.5**	**44.9**	**32.4**
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
+ P4	4.5G	80.0	13.5	48.9	42.3	30.7
+ P8	4.7G	79.8	15.7	46.3	44.0	31.4
+ P16	4.5G	77.9	**17.0**	**43.9**	**44.6**	**32.0**
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5
+ P4	4.4G	79.8	12.4	48.6	41.8	29.4
+ P8	4.5G	79.6	15.2	46.2	43.5	**30.7**
+ P16	4.6G	78.0	**16.2**	**43.8**	**43.8**	30.6

are 1) patchifying input images (Section 4.1), b) enlarging the kernel size (Section 4.2), and finally 3) reducing the number of activation layers and normalization layers (Section 4.3), all of which are conducted under a fair standard. Especially, the first design has a non-negligible effect on network FLOPs, which we mitigate by adding back in the third stage. Next, we will reveal how combining all of them delivers a pure CNN architecture that is as robust as, or even more robust than, Transformers.

4.1 Patchify Stem

A CNN or a Transformer usually down-samples the input image to a proper feature map size at the beginning of the network. A standard ResNet employs a 7×7 convolution layer with stride 2, followed by a 3×3 max pooling with stride 2, resulting in a 4× resolution reduction. ViT, however, adopts a much more aggressive downsampling strategy. It partitions an input image into p×p non-overlapping patches and map each patch to a d-dimension feature vector with a trainable linear projection. In practice, this step is implemented by a convolution layer with kernel size p and stride p, where p is typically set to 16. Here we refer to the layers before typical ResNet block in ResNet, or the layer

Table 3. The performance of different baseline models equipped with ViT-style patchify stem and convolution stem. "PX" refers to that the model has a ViT-style patchify stem with patch size set to X; "C" refers to that it has a convolutional stem.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
+ P16	4.6G	78.5	**17.0**	**44.3**	**44.8**	**32.0**
+ C	4.5G	79.6	16.2	48.4	43.0	30.2
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
+ P16	4.6G	78.1	**17.4**	**42.9**	**44.7**	**32.4**
+ C	4.6G	79.9	16.9	46.2	43.2	31.2
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
+ P16	4.5G	77.9	**17.0**	**43.9**	**44.6**	**32.0**
+ C	4.5G	79.1	16.6	48.4	43.2	30.3
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5
+ P16	4.6G	78.0	16.2	**43.8**	**43.8**	**30.6**
+ C	4.6G	79.3	**16.4**	49.0	42.4	29.2

before self-attention block in ViT, as *stem*. Some works argue for the importance of stem setup in CNN [37] or Transformers [43], but none of them investigate this module in the prospective of robustness. We replace the ResNet-style stem in the baseline models with the ViT-style patchify stem, a convolution layer with kernel size p and stride p, where p varies from 4 to 16. We keep the total stride of the model fixed (so a 224×224 input image will always lead to a 7×7 feature map before the final global pooling layer), which is achieved by setting stride=1 in the first block of some stages. Specifically, the original ResNet sets stride=2 for the first block in stage 2, 3, 4. We set stride=1 for the first block in stage 2 when employing 8×8 patchify stem, and set stride=1 for the first block in stage 2 and 3 when employing 16×16 patchify stem. To ensure a fair comparison, we add some blocks in stage 3 to keep the total FLOPs roughly the same as before.

We can observe in Tab. 1 that increasing patch size of the ViT-style patchify stem leads to increased performance on robustness benchmarks, though at the potential cost of clean accuracy. Specifically, for ResNet-DW, when patch size=8, the performance on all robustness benchmarks are boosted by at least 0.6%; when patch size=16, the performance on all robustness benchmarks are boosted by at least 1.2%, and the biggest improvement is 6.6% on Stylized-ImageNet. Surprisingly, this simple operation demonstrates significant potential for closing the robustness gap between CNNs and Transformers, suggesting that a large portion of the robustness of ViT comes from the patchify stem.

Recent findings also suggest replacing the ViT-style patchify stem with a small number of stacked 2-stride 3×3 convolution layers can greatly ease optimization and thus boost the clean accuracy [13,43]. To verify its effectiveness

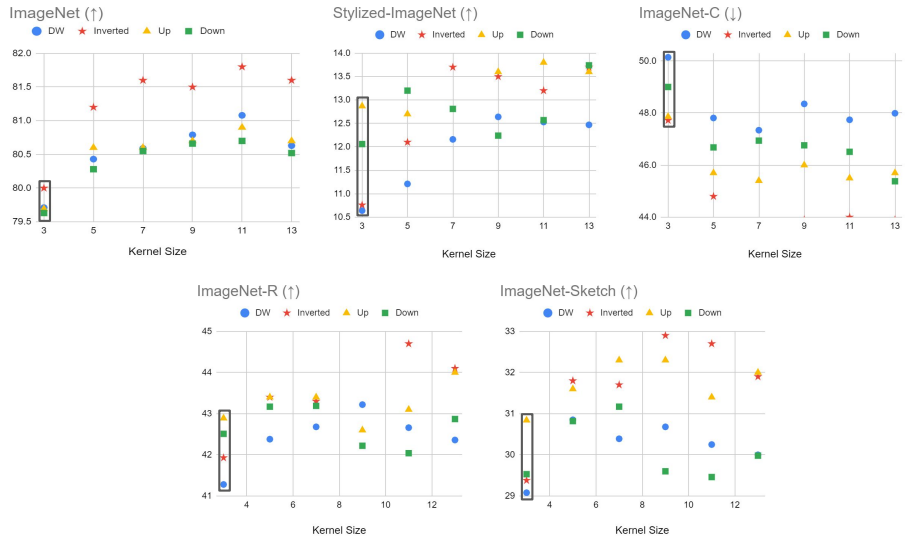

Fig. 3. Robustness evaluation of model with varying kernel size. Different colors and shapes denote different models.

on robustness benchmarks, we also implemented the convolution stem for ViT-S in [13], with a stack of 4 3×3 convolution layers with stride 2. The results are shown as follows in Tab. 3. Surprisingly, though the use of convolutional stem does attain higher clean accuracy, it appears to be not as helpful as ViT-style patchify stem on out-of-distribution robustness. It suggests that the clean accuracy and out-of-distribution robustness are not always positively correlated. And useful design for improving clean accuracy may not necessary attribute to better robustness, which again, shows the importance of searching for methods that can boost robustness in addition to solely clean accuracy.

4.2 Large Kernel Size

One of the most critical properties of self-attention operation that distinguish it from the classic convolution operation, is that it operates on the whole input image or feature map and thus has a global receptive field. Even before the rise of Vision Transformer, the importance of capturing long-range dependency has been demonstrated for CNNs as well. One typical example is the Non-local Neural Network which has been show to be highly effective for both static and sequential image recognition even when equippped with only one non-local block [41]. Still, the most popular design is to stack many 3×3 convolution layers to gradually increase the network receptive field as it goes deeper.

In this section, we try to mimic the behaviour of the self-attention block, by increasing the kernel size of the depth-wise convolution layer. We experiment with various kernel sizes: 5, 7, 9, 11, and 13. The results on different robustness benchmarks are shown as in Fig. 3.

Table 4. The performance of models with differently removed activation layers. We use ✓ to denote keeping a activation layer and ✗ to denote removing a activation layer. The marks here are in order. For example, a ✓ in the first column of the "ReLU" column refers to that the first relu activation in the block is keepped; a ✗ in the second column of "ReLU" column refers to that the first relu activation in the block is removed.

Architecture	FLOPs	ReLU			IN ↑	S-IN ↑	IN-C ↓	IN-R ↑	IN-SK ↑
		1^{st}	2^{nd}	3^{rd}					
ResNet50	4.1G	✓	✓	✓	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	✓	✓	✓	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	✓	✓	✓	79.7	10.6	50.1	41.3	29.1
		✓	✗	✗	74.8	8.3	58.6	37.1	24.5
		✗	✓	✗	73.2	8.6	60.5	36.2	23.3
		✗	✗	✓	79.4	11.7	50.5	42.8	29.8
ResNet-Inverted-DW	4.6G	✓	✓	✓	80.0	10.8	47.7	41.9	29.4
		✓	✗	✗	80.7	12.3	46.4	45.3	31.9
		✗	✓	✗	80.3	11.2	48.5	44.0	30.9
		✗	✗	✓	72.5	9.7	59.5	37.5	24.7
ResNet-Up-Inverted-DW	4.7G	✓	✓	✓	79.7	12.9	47.9	42.9	30.8
		✓	✗	✗	67.8	9.4	65.0	33.0	19.7
		✗	✓	✗	80.1	13.0	47.3	44.2	30.0
		✗	✗	✓	68.7	8.9	64.4	33.4	20.2
ResNet-Down-Inverted-DW	4.5G	✓	✓	✓	79.6	12.1	49.0	42.5	29.5
		✓	✗	✗	80.7	13.0	46.6	45.9	32.8
		✗	✓	✗	70.1	9.9	63.3	35.2	21.8
		✗	✗	✓	68.8	8.3	65.1	33.7	21.9

From Fig. 3, we observe that larger kernel sizes generally come with both better clean accuracy and stronger robustness. We can also see that the performance gain gradually saturates when the kernel size becomes too large.

Since using convolutions of larger kernels will incur notable extra computation, here we would like to highlight that this particular component design would be impossible without the help of depth-wise convolution layer, which significantly reduces the computational cost of large kernel convolution layers in all four blocks. For example, if directly changing the kernel size in ResNet50 from 3 to 5, the total FLOPs of the resulted model would be 7.4G, which is way larger than its Transformer counterpart. Thanks to depth-wise convolution layer, increasing kernel size from 3 to 13 typically only increase the FLOPs by 0.3G, which is relatively small compared to the FLOPs of DeiT-S (4.6G), except for ResNet-Inverted-DW. Because of the large channel dimension in the Inverted Bottleneck design, increasing kernel size from 3 to 13 brings a total of 1.4G FLOPs increase, rendering unfair comparison to some extent. Nevertheless, the trend for all four architectures is the same. And the extra computational cost brought by large kernel size can be mitigated by employing a patchify stem with large patch size so that our final models will still be on the same scale as DeiT-S (see Sec. 5 for the FLOPs of models with multiple proposed designs).

Table 5. The performance of models with differently removed activation and normalization layer. Note that the results are obtained with the best choice of only one activation layer in a block based on the results of Tab. 4. For simplicity, we only draw marks to denote if a normalization layer is kept in this table. We use ✓ to denote keeping a normalization layer and ✗ to denote removing a normalization layer. For example, a ✓ in the first column of the "BN" column refers to that the first batch normalization in the block is kept; a ✗ in the second column of "BN" column refers to that the first batch normalization in the block is removed. It is observed that the optimal choice always lead to best robustness.

Architecture	FLOPs	BN			IN ↑	S-IN ↑	IN-C ↓	IN-R ↑	IN-SK ↑
		1^{st}	2^{nd}	3^{rd}					
ResNet50	4.1G	✓	✓	✓	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	✓	✓	✓	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	✓	✓	✓	79.7	10.6	50.1	41.3	29.1
		✓	✗	✗	79.4	11.6	49.3	43.3	29.9
		✗	✓	✗	79.5	10.6	49.5	43.1	29.3
		✗	✗	✓	79.3	11.1	50.7	42.5	30.2
ResNet-Inverted-DW	4.6G	✓	✓	✓	80.0	10.8	47.7	41.9	29.4
		✓	✗	✗	80.4	12.9	47.0	45.8	32.4
		✗	✓	✗	80.0	11.2	49.3	44.1	31.2
		✗	✗	✓	80.1	13.0	46.8	44.3	30.8
ResNet-Up-Inverted-DW	4.7G	✓	✓	✓	79.7	12.9	47.9	42.9	30.8
		✓	✗	✗	80.5	14.3	45.0	47.2	33.1
		✗	✓	✗	80.4	12.9	47.0	45.8	32.4
		✗	✗	✓	80.1	13.0	46.8	44.3	30.8
ResNet-Down-Inverted-DW	4.5G	✓	✓	✓	79.6	12.1	49.0	42.5	29.5
		✓	✗	✗	80.5	13.6	46.4	45.9	33.3
		✗	✓	✗	80.7	13.1	46.1	45.5	31.7
		✗	✗	✓	80.8	12.7	46.5	46.1	33.1

4.3 Reducing Activation and Normalization Layers

A typical Vision Transformer block usually has fewer activation and normalization layers than a ResNet block [11,24], an architecture design found to be very effective in ConvNeXT [25]. In this work, we directly borrow the idea of reducing activation layers and normalization layers in ConvNeXT for all four block instantiations, and find it to be beneficial to clean accuracy as well as out-of-distribution robustness. Specifically, in a typical ResNet block, there is one normalization layer and one activation layer following each convolution layer, resulting in a total of three normalization layers and activation layers in one block. In our implementation, for each block, we remove two normalization layers and activation layers, and leave only one normalization layer and activation layer. The results of the models with differently removed activation layers are shown in Tab. 4. And the results of the models with differently removed normalization and activation layers are shown in Tab. 5. Note that we only ablate the position of removing normalization layers with the optimal position of activation layer

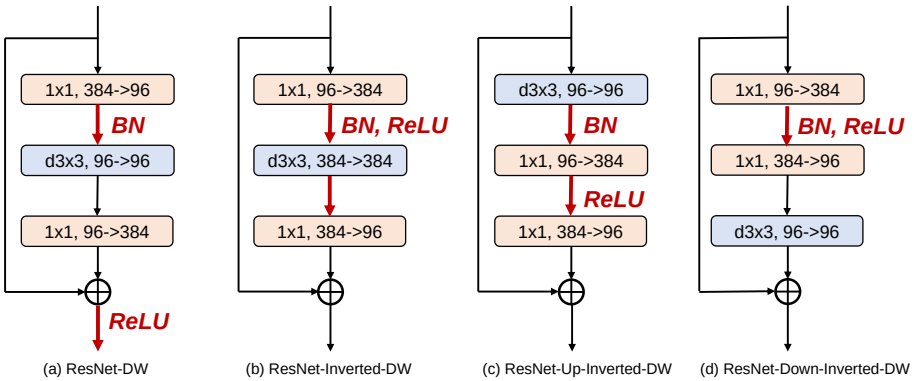

Fig. 4. The optimal position of normalization and activation layers for different block architectures.

shown in Tab. 4, since we find that only removing normalization layers does not result in performance improvement in the preliminary experiments.

We empirically find that among all experimental settings, leaving only one activation layer after the convolution layer where the channel dimension expands (*i.e.*, the number of output channel is larger than that of the input channels), and leaving one normalization layer after the first convolution layer yields the best result. We also visualize the best choice in Fig. 4. For instance, we observe 1.4% improvement on Stylized-ImageNet (12.9% *vs*. 14.3%), 2.9% improvement on ImageNet-C (47.9% *vs*. 45.0%), 4.3% improvement on ImageNet-R (42.9% *vs*. 47.2%), and 2.3% improvement on ImageNet-Sketch (30.8% *vs*. 33.1%) for ResNet-Up-Inverted-DW. Additionally, fewer activation layers and normalizations lead to less computation, which helps speep up training (e.g., up to 23% speed up simply by removing a few normalization layers).

5 Components Combination

In this section, we investigate whether all components combined will lead to better results. We choose a 16×16 patchify stem with patch size, 11×11 kernel size, and the corresponding optimal position for placing normalization and activation layer for all architectures (except for ResNet-Inverted-DW, we use 7×7 kernel size as we empirically find using too large kernel size (e.g., 11×11) can lead to unstable training). As shown in Tab. 6, Tab. 7, Tab. 8, and Tab. 9, *we can see these simple designs not only work well when they are individually employed with ResNet, but even work better when they are simultaneously employed.* Also, when all three designs are used, we see now ResNet beats DeiT on all four out-of-distribution benchmarks . These results again verify the effectiveness of our proposed architecture design, and also implies that a pure CNN without any self-attention-like block achieves just as good robustness as ViT. From now on, for simplicity, we will term ResNet-DW with our choice of ViT-style patchify

Table 6. The performance of different baseline models equipped with ViT-style patchify stem and large kernel size. The models with the suffix "PX" refers to that it has a ViT-style patchify stem with patch size set to X. "KX" refers to that the model uses blocks with a depth-wise convolution layer with kernel size X.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
+ P16 + K11	4.5G	79.0	18.8	42.5	45.2	32.4
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
+ P16 + K7	4.6G	79.2	19.8	41.1	46.4	33.7
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
+ P16 + K11	4.4G	78.0	17.2	43.7	47.1	31.2
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5
+ P16 + K11	4.6G	78.1	18.9	43.7	42.9	29.6

Table 7. The performance of different baseline models equipped with ViT-style patchify stem and less activation and normalization layers. The models with the suffix "PX" refers to that it has a ViT-style patchify stem with patch size set to X. "NormXActY" refers to that the model has only one normalization layer after the Xth convolution layer and one activation layer after the Yth convolution layer in the block.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
+ P16 + Norm1Act3	4.6G	78.7	16.2	43.0	46.3	33.1
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
+ P16 + Norm1Act1	4.6G	79.3	17.5	41.7	46.1	32.9
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
+ P16 + Norm1Act2	4.5G	79.1	18.6	41.5	48.0	33.8
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5
+ P16 + Norm1Act1	4.6G	79.7	18.0	40.7	47.1	33.4

stem, large kernel size, and fewer activation and normalization layers as Robust-ResNet-DW (and similar for others).

6 Knowledge Distillation

Knowledge Distillation [18] is a training heuristic aiming at transferring the knowledge of a stronger teacher model to a weaker student model with lower

Table 8. The performance of different baseline models equipped with large kernel size and less activation and normalization layers. "KX" refers to that the model uses blocks with a depth-wise convolution layer with kernel size X. "NormXActY" refers to that the model has only one normalization layer after the Xth convolution layer and one activation layer after the Yth convolution layer in the block.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
+ K11 + Norm1Act3	5.0G	80.4	13.3	45.2	43.9	30.8
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
+ K7 + Norm1Act1	5.0G	80.8	13.1	46.7	45.7	32.1
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
+ K11 + Norm1Act2	4.9G	81.4	15.8	42.9	48.3	33.8
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5
+ K11 + Norm1Act1	4.6G	81.3	14.2	43.6	46.1	32.8

Table 9. The performance of different baseline models equipped with ViT-style patchify stem, large kernel size, and less activation and normalization layers. The models with the suffix "PX" refers to that it has a ViT-style patchify stem with patch size set to X. "KX" refers to that the model uses blocks with a depth-wise convolution layer with kernel size X. "NormXActY" refers to that the model has only one normalization layer after the Xth convolution layer and one activation layer after the Yth convolution layer in the block.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
ResNet50	4.1G	78.4	9.9	51.2	39.2	27.7
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.7	10.6	50.1	41.3	29.1
+ P16 + K11 + Norm1Act3	4.5G	79.4	18.6	42.3	45.9	33.0
ResNet-Inverted-DW	4.6G	80.0	10.8	47.7	41.9	29.4
+ P16 + K7 + Norm1Act1	4.6G	79.0	19.5	42.1	45.9	32.8
ResNet-Up-Inverted-DW	4.7G	79.7	12.9	47.9	42.9	30.8
+ P16 + K11 + Norm1Act2	4.4G	79.2	20.2	40.9	48.7	35.2
ResNet-Down-Inverted-DW	4.5G	79.6	12.1	49.0	42.5	29.5
+ P16 + K11 + Norm1Act1	4.6G	79.9	19.3	41.6	46.0	32.8

capacity. Usually, the student model can achieve better or at least similar performance than the teacher model in knowledge distillation. Yet, directly employing knowledge distillation to let ResNet-50 (student model) learn from DeiT-S (teacher model) has been found to be less effective: ResNet50 still underperforms DeiT-S by a significant margin; While surprisingly when the model roles are switched, the student model DeiT-S is still able to remarkably outperform

Table 10. The robustness generalization of different models when DeiT-S serves as the teacher model. The results of DeiT-S here are included for reference; it is trained by the default recipe without knowledge distillation.

Student Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
DeiT-S	4.6G	79.8	16.2	42.8	41.9	29.1
ResNet-DW	4.8G	79.5	10.2	50.8	40.9	28.9
Robust-ResNet-DW	4.5G	79.3	20.1	41.2	46.5	33.2
ResNet-Inverted-DW	4.6G	80.1	11.0	47.9	42.4	30.4
Robust-ResNet-Inverted-DW	4.6G	79.1	19.4	42.4	45.9	33.0
ResNet-UpInvertedDW	4.7G	79.9	14.1	48.4	43.6	31.0
Robust-ResNet-Up-Inverted-DW	4.4G	79.3	19.6	41.1	49.3	35.4
ResNet-DownInvertedDW	4.5G	79.7	12.3	48.5	42.3	30.4
Robust-ResNet-Down-Inverted-DW	4.6G	79.9	18.9	41.0	46.4	33.5

the teacher model ResNet-50 [1]. It is therefore concluded that the key to achieving good robustness of DeiT lies in its architecture that cannot be transferred to ResNet via knowledge distillation. We run these experiments again with our training recipe and models with all three architectural designs combined. We can clearly conclude from Tab. 10 that with the help of our proposed architectural elements from ViT, a ResNet does perform better than DeiT on out-of-distribution samples. In contrast, the baseline models, though attaining good performance on clean ImageNet, are not as robust as the teacher model DeiT-S.

7 Larger Models

So far we have only demonstrated the effectiveness of our proposed models on a relative small model scale. To show that they are also beneficial to larger models, we change our model to match the total FLOPs of DeiT-Base. We change the base channel dimensions to (128, 256, 512, 1024), and over 20 blocks are added in stage 3 of the network. We then train it with the ConvNeXT-B recipe [25] and compare its performance with DeiT-Base, as shown in Tab. 11. We have also tried directly applying DeiT recipe and find that brings little improvement. As can be seen from Tab. 11, our proposed Robust-ResNet-Base models still performs favorably against DeiT-B, indicating the great potential of our method when scaling up models.

Table 11. Comparing the robustness of Robust-ResNet and DeiT at a larger scale.

Architecture	FLOPs	IN (↑)	S-IN (↑)	IN-C (↓)	IN-R (↑)	IN-SK (↑)
DeiT-B	17.6G	81.8	19.6	37.9	44.7	32.0
Robust-ResNet-Base-DW	17.2G	80.5	19.2	38.8	46.6	34.2
Robust-ResNet-Base-InvertedDW	17.1G	81.6	22.1	35.5	50.2	36.7
Robust-ResNet-Base-UpInvertedDW	17.1G	81.9	23.1	35.1	50.5	37.1
Robust-ResNet-Base-DownInvertedDW	17.3G	81.3	22.2	35.6	49.7	37.8

8 Conclusion

While the recent work [1] concludes that Transformers are more robust than CNNs on out-of-distribution samples, it also suggests that Transformer's self-attention-like architectures is the key. In this paper, we take a closer look at the Transformer architecture, and find several useful designs other than the self-attention block. We find that combining these designs and introducing them into ResNet can lead to a pure CNN architecture that can be as robust as, sometimes even more robust than, a Vision Transformer model of the same scale. We hope our results can encourage the community to rethink the robustness comparison between Transformers and CNNs, and meanwhile, help the community re-examine and better understand robust architecture design.

Acknowledgement

This work is supported by a gift from Open Philanthropy, TPU Research Cloud (TRC) program, and Google Cloud Research Credits program.

References

1. Bai, Y., Mei, J., Yuille, A., Xie, C.: Are transformers more robust than cnns? In: NeurIPS (2021) 1, 2, 3, 5, 14, 15
2. Bao, H., Dong, L., Wei, F.: BEiT: BERT pre-training of image transformers. In: ICLR (2022) 3
3. Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., et al.: On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258 (2021) 3
4. Brock, A., De, S., Smith, S.L., Simonyan, K.: High-performance large-scale image recognition without normalization. In: ICML (2021) 2
5. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. In: NeurIPS (2020) 3
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV (2021) 3
7. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV (2021) 3
8. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical data augmentation with no separate search. In: NeurIPS (2020) 5
9. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. In: ACL (2019) 3
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019) 3
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020) 1, 3, 10

12. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In: ICLR (2018) 5
13. Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., Douze, M.: Levit: a vision transformer in convnet's clothing for faster inference. In: ICCV (2021) 7, 8
14. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022) 3
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 1, 3
16. Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al.: The many faces of robustness: A critical analysis of out-of-distribution generalization. In: ICCV (2021) 3, 5
17. Hendrycks, D., Dietterich, T.G.: Benchmarking neural network robustness to common corruptions and surface variations. In: ICLR (2018) 3, 5
18. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NeurIPS Workshop (2015) 12
19. Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., Soudry, D.: Augment your batch: Improving generalization through instance repetition. In: CVPR (2020) 5
20. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: ICCV (2019) 4
21. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861 (2017) 4
22. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV (2016) 5
23. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NeurIPS (2012) 1
24. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021) 3, 10
25. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: CVPR (2022) 2, 3, 4, 10, 14
26. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019) 5
27. Paul, S., Chen, P.Y.: Vision transformers are robust learners. In: AAAI (2022) 1, 3
28. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (2018) 3
29. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog (2019) 3
30. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR (2018) 4
31. Shao, R., Shi, Z., Yi, J., Chen, P.Y., Hsieh, C.J.: On the adversarial robustness of visual transformers. arXiv preprint arXiv:2103.15670 (2021) 2
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015) 1

33. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019) 1
34. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019) 4
35. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021) 1, 3, 5
36. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: ICCV (2021) 3
37. Trockman, A., Kolter, J.Z.: Patches are all you need? arXiv preprint arXiv:2201.09792 (2022) 3, 7
38. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017) 3
39. Wang, H., Ge, S., Lipton, Z., Xing, E.P.: Learning robust global representations by penalizing local predictive power. NeurIPS (2019) 5
40. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV (2021) 3
41. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018) 8
42. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training procedure in timm. arXiv preprint arXiv:2110.00476 (2021) 3, 5
43. Xiao, T., Dollar, P., Singh, M., Mintun, E., Darrell, T., Girshick, R.: Early convolutions help transformers see better. NeurIPS (2021) 7
44. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017) 4
45. Xie, Z., Lin, Y., Yao, Z., Zhang, Z., Dai, Q., Cao, Y., Hu, H.: Self-supervised learning with swin transformers. arXiv preprint arXiv:2105.04553 (2021) 3
46. Xue, F., Shi, Z., Lou, Y., Liu, Y., You, Y.: Go wider instead of deeper. arXiv preprint arXiv:2107.11817 (2021) 3
47. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. In: NeurIPS (2019) 3
48. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. In: ICCV (2021) 3
49. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV (2019) 5
50. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. arXiv preprint arXiv:2106.04560 (2021) 3
51. Zhang, C., Zhang, M., Zhang, S., Jin, D., Zhou, Q., Cai, Z., Zhao, H., Yi, S., Liu, X., Liu, Z.: Delving deep into the generalization of vision transformers under distribution shifts. In: CVPR (2022) 1, 3
52. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018) 5
53. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI (2020) 5
54. Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., Kong, T.: ibot: Image bert pre-training with online tokenizer. In: ICLR (2022) 3