Wavelet Convolutions for Large Receptive Fields

Shahaf E. Finder$^{(\boxtimes)}$, Roy Amoyal , Eran Treister , and Oren Freifeld

The Department of Computer Science, Ben-Gurion University of the Negev,
Be'er Sheva, Israel
{finders,amoyalr}@post.bgu.ac.il, {erant,orenfr}@cs.bgu.ac.il

Abstract. In recent years, there have been attempts to increase the kernel size of Convolutional Neural Nets (CNNs) to mimic the global receptive field of Vision Transformers' (ViTs) self-attention blocks. That approach, however, quickly hit an upper bound and saturated way before achieving a global receptive field. In this work, we demonstrate that by leveraging the Wavelet Transform (WT), it is, in fact, possible to obtain very large receptive fields without suffering from over-parameterization, *e.g.*, for a $k \times k$ receptive field, the number of trainable parameters in the proposed method grows only logarithmically with k. The proposed layer, named WTConv, can be used as a drop-in replacement in existing architectures, results in an effective multi-frequency response, and scales gracefully with the size of the receptive field. We demonstrate the effectiveness of the WTConv layer within ConvNeXt and MobileNetV2 architectures for image classification, as well as backbones for downstream tasks, and show it yields additional properties such as robustness to image corruption and an increased response to shapes over textures. Our code is available at https://github.com/BGU-CS-VIL/WTConv.

Keywords: Wavelet Transform · Receptive Field · Multi-frequency

1   Introduction

In the past decade, Convolutional Neural Networks (CNNs) have largely dominated many areas of computer vision. Nonetheless, with the recent emergence of Vision Transformers (ViTs) [11], which are an adaptation of the Transformer architecture [53] used in natural language processing, CNNs have faced stiff competition. Concretely, the edge that ViTs are now believed to have over CNNs is primarily attributed to their multi-head self-attention layer. This layer facilitates the global mixing of features, in contrast to convolutions that are, by construction, limited to local mixing of features. Consequently, several recent works have attempted to bridge the performance gap between CNNs and ViTs.

(a) RepLK   (b) SLaK   (c) WTConv
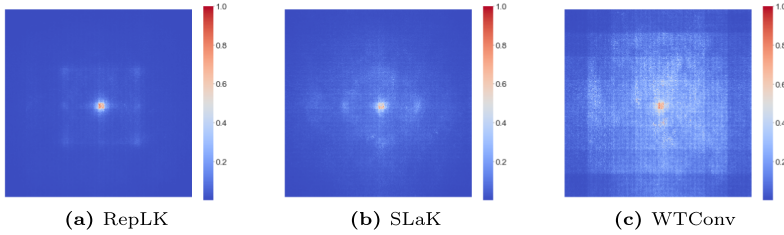
Fig. 1. The Effective Receptive Fields [37] of ConvNeXt-T [36] with different depth-wise convolutions. Evidently, the proposed WTConv achieves the largest field despite using fewer trainable parameters. This improves the convolution's ability to capture low frequencies and thus increases (*i.e.*, improves) its shape bias, among other advantages.

Liu *et al.* [36] reconstructed the ResNet architecture and its training routine to keep up with the Swin Transformer [35]. One of the improvements in [36] was increasing the convolutions' kernel size. Empirically, however, that approach saturated at a kernel size of 7×7, meaning that increasing the kernel further did not help, and at some point even started deteriorating the performance. While naively increasing the size beyond 7×7 was not useful, Ding *et al.* [10] have shown that one can behoove from even larger kernels if those are better constructed. Even then, however, eventually the kernels become over-parameterized and the performance saturates way before reaching a global receptive field.

One intriguing property analyzed in [10] is that using larger kernels renders CNNs more shape-biased, which means that their ability to capture low frequencies in the image is improved. This finding was somewhat surprising because convolutional layers usually tend to respond to high-frequencies in the input [16,18,50,55]. This is unlike attention heads, which are known to be more attuned to low frequencies, as demonstrated in other studies [40,41,50].

The discussion above raises a natural question: Can we utilize signal-processing tools to increase the receptive field of convolutions effectively without suffering from over-parametrization? In other words, can we have very large filters – *e.g.*, with a global receptive field – and still improve the performance? This paper offers a positive answer to this question. Our proposed approach leverages the Wavelet Transform (WT) [8], an established tool from time-frequency analysis, to make the receptive field of convolutions scale up well and, by cascading, also guide the CNNs to better respond to low frequencies. In part, our motivation to base our solution on the WT is that (unlike, *e.g.*, the Fourier Transform) it retains some spatial resolution. This makes spatial operations (*e.g.*, convolutions) in the wavelet domain more meaningful.

More concretely, we propose WTConv, a layer that uses the cascade WT decomposition and performs a set of small-kernel convolutions, each focusing on different frequency bands of the input in an increasingly larger receptive field. This process allows us to put more emphasis on low frequencies in the input while adding only a small number of trainable parameters. In fact, for a $k \times k$

receptive field, our number of trainable parameters grows only logarithmically with k. This fact, which is in contrast to some recent methods whose corresponding growth is quadratic, lets us obtain effective CNNs with an unprecedented effective receptive field (ERF) [37] size (see Fig. 1).

We design WTConv as a drop-in replacement for depth-wise convolutions that can be used as is within any given CNN architecture without additional modifications. We validate the effectiveness of WTConv by incorporating it into ConvNeXt [36] for image classification, demonstrating its utility in a fundamental vision task. Further leveraging ConvNeXt as a backbone, we extend our evaluation to more complex applications: using it within UperNet [59] for semantic segmentation and within Cascade Mask R-CNN [2] for object detection. In addition, we analyze additional benefits WTConv provides to CNNs.

To summarize, our key contributions are:

– A new layer, called WTConv, that uses the WT to increase the receptive field of convolutions effectively.
– WTConv is designed to be a drop-in replacement (for depth-wise convolutions) within given CNNs.
– Extensive empirical evaluation demonstrates that WTConv improves CNNs' results in several key computer-vision tasks.
– Analysis of WTConv's contribution to CNN's scalability, robustness, shape-bias, and ERF.

2 Related Work

2.1 Wavelet Transforms in Deep Learning

The WT [8], a powerful tool for signal processing and analysis, has been widely used since the 1980s. Following its success in classical settings, recently the WT has also been incorporated into neural network architectures for a variety of tasks. Wang *et al.* [57] extract features from the time-frequency components of ECG signals. Huang *et al.* [29] and Guo *et al.* [21] predict wavelet high-frequency coefficients of an input image to reconstruct a higher-resolution output. Duan *et al.* [12] and Williams and Li [58] use WTs as pooling operators within CNNs. Gal *et al.* [15], Guth *et al.* [22], and Phung *et al.* [42] use wavelets in generative models to enhance the visual quality of generated images as well as to improve computational performance. Finder *et al.* [13] utilize wavelets to compress feature maps for more efficient CNNs. Saragadam *et al.* [47] use wavelets as activation functions for implicit neural representations.

More related to our work, Liu *et al.* [33] and Alaba *et al.* [1] utilize the WT in a modified U-Net architecture [45] for down-sampling and the inverse WT for up-sampling. Jeevan *et al.* [30] suggest replacing token mixing with WT together with channel mixing in the wavelet domain. In another work related to ours, Fujieda *et al.* [14] propose a DenseNet-type architecture that uses wavelets to reintroduce lower frequencies in the input to later layers. Although not wavelet-related, Chen *et al.* [3] propose performing convolutions on multi-resolution input

by initially separating the image to high and low resolutions and performing information exchange between the two along the network. These works exemplify the benefits of performing convolutions on the input's low-frequency components separately from the high-frequency components for a more informative feature map. This feature motivates our work as well. However, the methods from [1, 14,30,33] are highly customized architectures and cannot seamlessly be used within other CNN architectures, while [3] focuses on computational efficiency. In contrast, we propose a lighter, easier-to-user, and linear layer that can be used as a drop-in replacement for depth-wise convolutions and that results in an improved receptive field. Importantly, our method can fit within any network that uses depth-wise convolution and, therefore, is not limited to a single task.

2.2   Large-Kernel Convolutions

When regarding convolution configurations, VGG [48] has set the standard for modern CNNs by using 3×3 convolutions, sacrificing the size of single-layer receptive field to increase the network's depth (from under 10 layers to around 20). Since then, with increased computations and improved architectures, CNNs have become much deeper, but the kernel-size parameter was left largely unexplored.

One major change to the traditional convolutions was the introduction of separable convolutions [52,56]. The separable convolutions were popularized by Xception [5] and MobileNet [28] and are adopted in most modern architectures [36,46]. In this approach, spatial convolutions are performed per channel (*i.e.*, depth-wise), and the cross-channel operations are performed using 1×1 kernels (*i.e.*, point-wise). This separation of convolutions also creates a degree of separation between the kernel size and the channel dimension w.r.t.the number of parameters and operations. Each spatial convolution of kernel size k and c channels now has only $k^2 \cdot c$ parameters (instead of $k^2 \cdot c^2$), and this allows it to scale better with k, albeit still quadratically.

Concurrently, the introduction to vision tasks [11,35] of transformers – with their non-local self-attention layers – has typically yielded better results than those of the local-mixing convolutions. This, together with the aforementioned recent use of separable convolutions, has reignited the interest in exploring larger kernels for CNNs. In particular, Liu *et al.* [36] have reexamined the popular ResNet architecture [24], including an empirical comparison of different kernel sizes, concluding that the performance saturated at a kernel size of 7×7. Trockman and Kolter [49] have attempted to mimic the ViT architecture using only convolutions and have shown impressive results by using 9×9 convolutions to replace the attention (or "mixer") component. Ding *et al.* [10] have suggested that simply increasing the size of the kernel hurts the locality property of convolutions. Thus, they proposed using a small kernel in parallel to a large one and then summing their outputs. With that technique, they successfully trained CNNs with kernel sizes up to 31×31. Liu *et al.* [34] have successfully increased their kernels to size 51×51 by factorizing it to a set of parallel 51×5 and 5×51 kernels. In addition, they introduced sparsity together with the expansion of the

width of the network. However, this idea of using more channels (with sparsity) is orthogonal to increasing the kernel size. While our work partly draws inspiration from [10,34], in our case the proposed layer sums outputs from various frequency components of the input, capturing multiple receptive fields.

Another way to achieve a global receptive field is performing the spatial mixing in the frequency domain following a Fourier transform (*e.g.* [4,23,43]). However, the Fourier transform converts the input to be represented entirely in the frequency domain, therefore it fails to learn local interactions between neighboring pixels. In contrast, the WT successfully preserves some local information while decomposing the image to different frequency bands, allowing us to operate on different levels of decomposition. Moreover, Fourier-based methods tend to rely on a specific size of input for the number of weights and, therefore, can be difficult to use for downstream tasks. A concurrent work [19] utilizes neural implicit functions for efficient mixing in the frequency domain.

3   Method

In this section, we first describe how the wavelet transform is performed using convolutions, and then we propose our solution for performing convolution in the wavelet domain, named WTConv. We also describe WTConv's theoretical benefits and analyze its computational cost.

3.1   Preliminaries: The Wavelet Transform as Convolutions

In this work, we employ the Haar WT as it is efficient and straightforward [13,15,29]. However, we note that our approach is not limited to it as other wavelet bases can be used, albeit at an increased computational cost.

Given an image X, the one-level Haar WT over one spatial dimension (width or height) is given by a depth-wise convolution with the kernels $[1, 1]/\sqrt{2}$ and $[1, -1]/\sqrt{2}$ followed by a standard downsampling operator of factor 2. To perform the 2D Haar WT, we compose the operation on both dimensions, resulting in a depth-wise convolution with a stride of 2 using the following set of four filters:

$$f_{LL} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, f_{LH} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, f_{HL} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, f_{HH} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \quad (1)$$

Note that f_{LL} is a low-pass filter, and f_{LH}, f_{HL}, f_{HH} are a set of high-pass filters. For each input channel, the output of the convolution

$$[X_{LL}, X_{LH}, X_{HL}, X_{HH}] = \text{Conv}([f_{LL}, f_{LH}, f_{HL}, f_{HH}], X) \quad (2)$$

has four channels, each of which has (in each spatial dimension) half the resolution of X. X_{LL} is the low-frequency component of X, while X_{LH}, X_{HL}, X_{HH} are its horizontal, vertical, and diagonal high-frequency components.
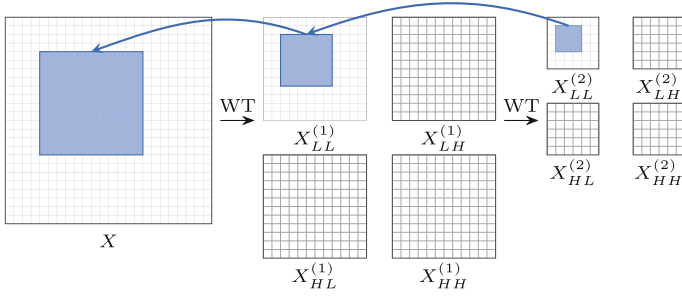
Fig. 2. Performing convolution in the wavelet domain results in a larger receptive field. In this example, a 3×3 convolution is performed on the low-frequency band of the second-level wavelet domain $X_{LL}^{(2)}$, resulting in a 9-parameter convolution that responds to lower frequencies of a 12×12 receptive field in the input X.

Since the kernels in Eq. (1) form an orthonormal basis, applying the inverse wavelet transform (IWT) is obtained by the transposed convolution:

$$X = \text{Conv-transposed}([f_{LL}, f_{LH}, f_{HL}, f_{HH}] , \\ [X_{LL}, X_{LH}, X_{HL}, X_{HH}]). \tag{3}$$

The cascade wavelet decomposition is then given by recursively decomposing the low-frequency component. Each level of the decomposition is given by

$$X_{LL}^{(i)}, X_{LH}^{(i)}, X_{HL}^{(i)}, X_{HH}^{(i)} = \text{WT}(X_{LL}^{(i-1)}) \tag{4}$$

where $X_{LL}^{(0)} = X$ and i is the current level. This results in an increased frequency resolution and a reduced spatial resolution for the lower frequencies.

3.2   Convolution in the Wavelet Domain

As described in Sect. 2.2, increasing the kernel size of a convolutional layer increases the number of parameters (and, therefore, the degrees of freedom) quadratically. To mitigate that, we propose the following. First, use the WT to filter and downscale the lower- and higher-frequency content of the input. Then, perform a small-kernel depth-wise convolution on the different frequency maps before using the IWT to construct the output. In other words, the process is given by

$$Y = \text{IWT}(\text{Conv}(W, \text{WT}(X))), \tag{5}$$

where X is the input tensor, and W is the weight tensor of a $k \times k$ depth-wise kernel with four times as many input channels as X. This operation not only separates the convolution between the frequency components but also allows a
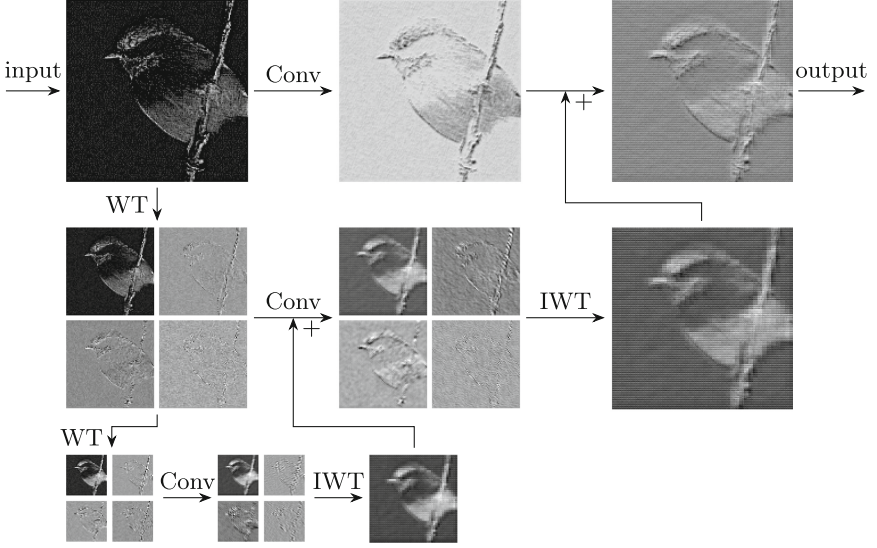
Fig. 3. An example of the WTConv operation on a single channel taken from the third inverted residual block of MobileNetV2 (see Sect. 4.4) using a 2-level wavelet decomposition and 3×3 kernel sizes for the convolutions.

smaller kernel to operate in a larger area of the original input, *i.e.*, increasing its receptive field w.r.t.the input. See Fig. 2 for an illustration.

We take this 1-level combined operation and increase it further by using the same cascade principle from Eq. (4). The process is given by:

$$X_{LL}^{(i)}, X_H^{(i)} = \text{WT}(X_{LL}^{(i-1)}), \tag{6}$$

$$Y_{LL}^{(i)}, Y_H^{(i)} = \text{Conv}(W^{(i)}, (X_{LL}^{(i)}, X_H^{(i)})), \tag{7}$$

where $X_{LL}^{(0)}$ is the input of the layer, and $X_H^{(i)}$ represents all three high-frequency maps of level i described in Sect. 3.1.

To combine the outputs of the different frequencies, we use the fact that the WT and its inverse are linear operations, meaning that $\text{IWT}(X + Y) = \text{IWT}(X) + \text{IWT}(Y)$. Therefore, performing

$$Z^{(i)} = \text{IWT}(Y_{LL}^{(i)} + Z^{(i+1)}, Y_H^{(i)}) \tag{8}$$

results in the summation of the different levels' convolutions, where $Z^{(i)}$ is the aggregated outputs from level i onward. This is in line with [10], where two outputs of different-sized convolutions are summed as the output.

In contrast to [10], we can not normalize each of the $Y_{LL}^{(i)}, Y_H^{(i)}$, as the separate normalization of those does not correspond to normalization in the original domain. Instead, we find it is sufficient to perform only a channel-wise scaling

to weigh each frequency component's contribution. Figure 3 visualizes WTConv for the case of a 2-level WT. The algorithm is provided in **Appendix A**.

3.3   The Benefits of Using WTConv

There are two main technical benefits of incorporating WTConv within a given CNN. First, each level of WT increases the size of the layer's receptive field with only a small increase in the number of trainable parameters. That is, the ℓ-level cascading frequency decomposition of the WT, together with a fixed-size kernel, k, for each level, allows the number of parameters to scale linearly in the number of levels $(\ell \cdot 4 \cdot c \cdot k^2)$ while the receptive field grows exponentially $(2^\ell \cdot k)$.

The second benefit is that the WTConv layer is constructed to capture low frequencies better than a standard convolution. This is because the repeated WT decomposition of the low frequencies of the input emphasizes them and increases the layer's corresponding response. This discussion complements the analysis that convolutional layers are known to respond to high frequencies in the input [18,41]. By leveraging compact kernels on the multifrequency inputs, the WTConv layer places the additional parameters where they are most needed.

In addition to yielding improved results on standard benchmarks, these technical benefits translate to improvement in the network's scalability in comparison to large-kernel methods, robustness w.r.t.corruption and distribution shift, and responding more to shapes over textures. We empirically confirm these assumptions in Sect. 4.4.

3.4   Computational Cost

The computational cost, in terms of floating-point operations (FLOPs), of a depth-wise convolution is

$$C \cdot K_W \cdot K_H \cdot N_W \cdot N_H \cdot \frac{1}{S_W} \cdot \frac{1}{S_H}, \tag{9}$$

where C is the number of input channels, (N_W, N_H) is the spatial dimension of the input, (K_W, K_H) is the kernel size, and (S_W, S_H) is the stride in each dimension. For example, consider a single-channel input of spatial dimensions 512×512. Performing convolution with a kernel of size 7×7 results in $12.8M$ FLOPs, while a 31×31 kernel size results in $252M$ FLOPs. Considering the WTConv set of convolutions, each wavelet-domain convolution is performed over a reduced spatial dimension in a factor of 2, albeit at four times as many channels as the original input. This results in a FLOP count of

$$C \cdot K_W \cdot K_H \cdot \left(N_W \cdot N_H + \sum_{i=1}^{\ell} 4 \cdot \frac{N_W}{2^i} \cdot \frac{N_H}{2^i} \right), \tag{10}$$

where ℓ is the number of WT levels. Continuing the earlier example of input size 512×512, performing the multi-frequency convolutions of a 3-level WTConv with

kernel size of 5×5 (which covers a receptive field of $40 \times 40 = (5 \cdot 2^3) \times (5 \cdot 2^3)$) results in $15.1M$ FLOPs. Of course, one also needs to add the cost of the WT calculation itself. We note that when the Haar basis is used, WTs can be implemented in a highly efficient way [13]. That said, with a naive implementation using the standard convolution operation, the FLOP count of the WTs is

$$4C \cdot \sum_{i=0}^{\ell-1} \frac{N_W}{2^i} \cdot \frac{N_H}{2^i}, \tag{11}$$

since the four kernels are of size 2×2 with a stride of 2 on each spatial dimension and operate on each of the input channels (see Sect. 3.1). Likewise, a similar analysis shows that the IWT has the same FLOP count as the WT. Continuing the example, this results in an addition of $2.8M$ FLOPs for 3 levels of WT and IWT, totaling $17.9M$ FLOPs, which still represent a substantial saving over the standard depth-wise convolutions of similar receptive fields.

4   Results

In this section, we experiment with WTConv in several settings. First, in Sect. 4.1, we train and evaluate ConvNeXt [36] with WTConv for ImageNet-1K [9] classification. Then, in Sect. 4.2 and Sect. 4.3, we use the trained models as backbones for downstream tasks. Finally, in Sect. 4.4, we further analyze the contribution of WTConv to the network.

4.1   ImageNet-1K Classification

For ImageNet-1K [9], we use ConvNeXt [36] as our base architecture and replace the 7×7 depth-wise convolution with WTConv. ConvNeXt, as an extension of ResNet, mainly consists of four stages with downsampling operations between them. We set WTConv's levels to [5,4,3,2] and kernel size to 5×5 for these stages to achieve a global receptive field at each step for an input size of 224×224. We use two training schedules of 120 and 300 epochs (see **Appendix B** for details).

Table 1. Classification accuracy on ImageNet-1k using a 120-epoch training schedule.

Configuration	Receptive field	D-W Param.	Top-1
ConvNeXt-T	[7, 7, 7, 7]	0.32M	81.0
+ VAN[†] [20]	[21,21,21,21]	0.38M	81.1
+ GFNet[‡] [43]	Global	4.29M	81.2
+ RepLK [10]	[31,29,27,13]	3.84M	81.5
+ SLaK[†] [34]	[51,49,47,13]	2.52M	81.5
+ WTConv	[160,80,40,20]	2.04M	**81.7**

† - using only kernel decomposition. ‡ - FFT-based method.

Table 1 shows the results for the 120-epoch schedule. Since all networks use the same base ConvNeXt-T architecture, we report the number of parameters for the depth-wise (marked D-W) convolutions. Note that, for a fair comparison, we report SLaK's and VAN's results using only the depth-wise kernel decomposition, as we compare only the effect of increasing the receptive field. We emphasize that WTConv achieves the best results while being the most parameter-efficient among the top-scoring methods. Moreover, it achieves a global receptive field with less than half of GFNet's number of parameters.

Table 2. Classification accuracy on ImageNet-1k using a 300-epoch training schedule.

Configuration	Param.	FLOPs	Top-1
Swin-T [35]	28M	4.5G	81.3
ConvNeXt-T [36]	29M	4.5G	82.1
WTConvNeXt-T	30M	4.5G	**82.5**
Swin-S [35]	50M	8.7G	83.0
ConvNeXt-S [36]	50M	8.7G	83.1
WTConvNeXt-S	54M	8.8G	**83.6**
Swin-B [35]	88M	15.4G	83.5
ConvNeXt-B [36]	89M	15.4G	83.8
WTConvNeXt-B	93M	15.5G	**84.1**

In Table 2, which shows the results for the 300-epoch schedule, we compare WTConvNeXt to Swin [35] and ConvNeXt [36]. Table 1 and Table 2 both show that introducing WTConv to ConvNeXt results in a substantial improvement in classification accuracy while introducing only a slight increase in parameters and FLOPs. *E.g.*, moving from ConvNeXt-S to ConvNeXt-B adds 39M parameters and 6.7 GFLOPs for a 0.7% accuracy gain, while moving to WTConvNeXt-S adds only 4M parameters and 0.1 GFLOPs for a 0.5% accuracy gain.

4.2   Semantic Segmentation

We evaluate WTConvNeXt as a backbone for UperNet [59] for the ADE20K [60] semantic segmentation task. We use MMSegmentation [7] for UperNet's implementation, training, and evaluation. The training follows the exact configuration for ConvNeXt without any parameter tuning. We use the 80K and 160K iterations training scheme for the 120- and 300-epoch pre-trained models from Sect. 4.1, respectively, and report the mean intersection over union (mIoU) index with single-scale testing. Table 3 presents the results and shows an improvement of 0.3–0.6% in mIoU when using WTConv.

Table 3. ADE20K validation results using UperNet [59]. FLOPs are based on an input size of 2048×512.

Configuration	Param.	FLOPs	mIoU
120-e pre-train + 80K finetune			
ConvNeXt-T	60M	939G	44.6
+ RepLK	64M	975G	45.0
WTConvNeXt-T	62M	939G	**45.4**
300-e pre-train + 160K finetune			
Swin-T	60M	945G	44.5
ConvNeXt-T	60M	939G	46.0
WTConvNeXt-T	62M	939G	**46.6**
Swin-S	81M	1038G	47.6
ConvNeXt-S	82M	1027G	48.7
WTConvNeXt-S	86M	1028G	**49.0**

Table 4. COCO validation results using Cascade Mask-RCNN [2]. FLOPs are based on an input size of 1280×800

Configuration	FLOPs	AP^{box}	AP^{mask}
120-e pre-train + 1x finetune			
ConvNeXt-T	741G	47.3	41.1
+ RepLK	776G	47.8	41.4
WTConvNeXt-T	741G	**47.9**	**41.5**
300-e pre-train + 3x finetune			
Swin-T	745G	50.4	43.7
ConvNeXt-T	741G	50.4	43.7
WTConvNeXt-T	741G	**51.0**	**44.4**
Swin-S	838G	51.9	45.0
ConvNeXt-S	827G	51.9	45.0
WTConvNeXt-S	827G	**52.6**	**45.6**

4.3   Object Detection

We also evaluate WTConvNeXt as a backbone for Cascade Mask R-CNN [2] on the COCO dataset [32]. We use MMDetection [6] for Cascade Mask R-CNN's implementation, training, and evaluation. The training follows the exact configuration for ConvNeXt without any parameter tuning. We use the 1x and 3x fine-tuning schedules for the 120- and 300-epoch pre-trained models, respectively, and report box and mask average precision (AP). The results are presented in Table 4, where we see a considerable improvement, as AP^{box} and AP^{mask} both increase by 0.6–0.7%. Detailed results are available at **Appendix F**.

4.4   WTConv Analysis

Scalability.   We construct a small-scale scalability analysis for the task of classification on ImageNet-50/100/200 [44,51], which are subsets of ImageNet [9] containing 50/100/200 classes, respectively. In this experiment, we use MobilenetV2 [46] where each depth-wise convolution is replaced with RepLK [10], GFNet [43], FFC [4], and the proposed WTConv layer. We set WTConv's kernel sizes to 3×3. For RepLK, we use the closest possible kernel size to WTConv's receptive field, *e.g.*, 13×13 against 2-levels WTConv with a receptive field of 12×12. GFNet and FFC are Fourier-based methods. GFnet's global filter layer requires $h \cdot w$ parameters per channel, where (w, h) is the spatial dimension of the input, and is thus highly over-parameterized, especially on MobileNetV2 where the first few layers have an input of size 112^2. In contrast, FFC uses the same weights across different frequencies, and therefore, it is not directly dependent on (w, h) as GFNet is. The training parameters are specified in **Appendix B**.

Table 5. Classification accuracy on ImageNet-50/100/200, using MobileNetV2 with different depth-wise convolutions. Param./c is the parameter count per channel.

Conv Type	Param./c	IN-50	IN-100	IN-200
Baseline 3×3	9	85.08	84.64	82.38
RepLK 7×7	58	86.16	85.02	83.31
RepLK 13×13	178	85.28	85.26	83.50
RepLK 25×25	634	84.96	84.94	83.38
GFNet	$h \cdot w$	54.68	55.72	56.65
FFC	14	84.92	84.64	82.96
WTConv 1 level	45	**86.24**	85.14	83.29
WTConv 2 levels	81	86.04	85.36	83.45
WTConv 3 levels	117	85.96	**85.44**	**83.84**

The results, summarized in Table 5, show that WTConv scales better than RepLK when increasing the receptive field. We assume this is due to insufficient data to support the large number of trainable parameters of the RepLK layers. This also aligns with the findings of [34] for ImageNet-1K, where simply increasing the filter size in RepLK hurts the result. GFNet severely suffers from its over-parameterization, and its results drop significantly. FFC performs better, albeit the limited frequency mixing hurts its results.

Robustness. We perform robustness evaluation for classification on ImageNet-C/$\overline{\text{C}}$ [26,39], ImageNet-R [25], ImageNet-A [27], and ImageNet-Sketch [54]. We report mean corruption error for ImageNet-C, corruption error for ImageNet-$\overline{\text{C}}$, and top-1 accuracy for all other benchmarks. We also evaluate object detection on COCO under corruption [38], measured in mean and relative performance under corruption (mPC and rPC). We report results for models trained using the 300-epoch schedule from Sect. 4.1 without modifications or fine-tuning.

Table 6. Robustness to corruption in classification over different benchmarks. ImageNet-C/$\overline{\text{C}}$ are measured in corruption error, and ImageNet-A/R/SK in top-1 accuracy.

Model	C ↓	$\overline{\text{C}}$ ↓	A ↑	R ↑	SK ↑
ConvNeXt-T	53.2	40.0	24.2	47.2	35.1
WTConvNeXt-T	**52.0**	**38.0**	**25.3**	**48.0**	**35.8**
ConvNeXt-B	46.8	34.4	36.7	51.3	39.4
WTConvNeXt-B	**45.6**	**32.2**	**36.9**	**52.6**	**39.9**

Table 7. Robustness to corruptions in object detection measured in mean and relative performance under corruption (mPC and rPC).

Model	AP	mPC	rPC
ConvNeXt-T	50.4	31.8	63.2
WTConvNeXt-T	**51.0**	**34.2**	**67.1**
ConvNeXt-S	51.9	35.2	67.8
WTConvNeXt-S	**52.6**	**36.9**	**70.3**
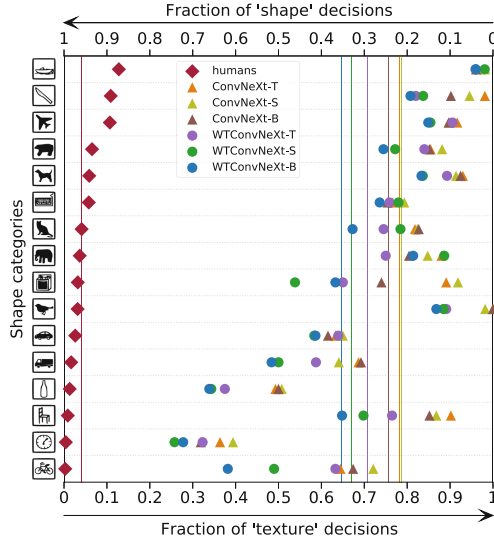
Fig. 4. Shape bias comparison of ConvNeXt-T/S/B and WTConvNeXt-T/S/B over 16 categories. The vertical line is the average across categories.

Table 6 and Table 7 summarizes the results. Note that even though WTConvNeXt accuracy is 0.3–0.4% above ConvNeXt in ImageNet-1K, the accuracy gain in most of the robustness datasets is above 1% and gets as high as 2.2%. A similar trend is evident in corrupted object detection, which can be explained by the improved response to low frequencies [31]. More detailed tables and qualitative examples are provided in **Appendix G**.

Shape-Bias. We use the *modelvshuman* benchmark [17] to quantify the improvement in shape bias (*i.e.*, the fraction of predictions made based on shapes rather than on textures). Increased shape bias is associated with human perception and, therefore, considered more desirable.

The results, presented in Fig. 4, confirm our assumption that WTConv renders the network more shape-biased, increasing the fraction of "shape" decisions by 8–12%. Note that even the smaller WTConvNeXt-T responds better to shapes than the larger ConvNeXt networks, even though the latter score better at ImageNet-1k accuracy. This is most likely due to the increased emphasis on the lower frequencies induced by WTConv, as shapes are generally associated with low frequencies, while textures are associated with high frequencies. The quantitative results are available in **Appendix E**

Effective Receptive Field. We evaluate the contribution of WTConv to ConvNeXt-T's ERF [37], using the code provided by [10]. Theoretically, in CNNs, the ERF is proportional to $\mathcal{O}(K\sqrt{L})$ [37], where K is the kernel size and L is the

depth of the network. However, since we introduce WT for an increased recep-
tive field while using a smaller kernel, we assume it holds when considering K
as the size of the receptive field induced by the layer. The empirical evaluation
of the ERF includes sampling 50 random images from ImageNet's validation set
resized to 1024×1024. Then, for each image, calculate each pixel's contribution,
measured by the gradient, to the central point of the feature map produced by
the last layer. The result is visualized in Fig. 1, where high-contribution pixels
are brighter. We note that WTConv has a nearly-global ERF despite having
fewer parameters than RepLK and SLaK.

Ablation Study. We conduct an ablation study to see how different configu-
rations of the WTConv layer impact the final result. We train WTConvNeXt-T
for 120 epochs on ImageNet-1K as described in Sect. 4.1, with various config-
urations. First, we experiment with different combinations of WT levels and
kernel sizes; note that ConvNeXt's convolutions operate on inputs of resolutions
$56^2, 28^2, 14^2, 7^2$ (for 224^2 input), allowing for maximal WT levels of $5, 4, 3, 2$
respectively. Second, we evaluate the contribution of the high- and low-frequency
components by using only one of the sets of high/low in the convolution each
time. Lastly, we train the model with different wavelet bases.

Table 8 shows the results of all the described configurations. Here, increas-
ing the levels and kernel size is mostly beneficial. We also see that using each
frequency band separately improves the model's performance; however, using
both is superior. The results confirm that the Haar WT is sufficient, although
exploring other bases may improve performance. We leave this for future work.

Table 8. Ablation study with WTConvNeXt-T. Comparing different configurations of
WTConv.

Levels	Kernel size	Wavelet	Receptive field	D-W Param.	Top-1
$[4, 3, 2, 1]$	3×3	Haar	$[48, 24, 12, 6]$	0.50M	81.24
$[4, 3, 2, 1]$	5×5	Haar	$[80, 40, 20, 10]$	1.38M	81.32
$[4, 3, 2, 1]$	7×7	Haar	$[112, 56, 28, 14]$	2.70M	81.56
$[5, 4, 3, 2]$	3×3	Haar	$[96, 48, 24, 12]$	0.73M	81.49
$[5, 4, 3, 2]$	5×5	Haar	$[160, 80, 40, 20]$	2.04M	**81.75**
$[5, 4, 3, 2]$	7×7	Haar	$[224, 112, 28, 14]$	3.99M	81.69
$[5, 4, 3, 2]$	5×5 Lows	Haar	$[160, 80, 40, 20]$	0.63M	81.46
$[5, 4, 3, 2]$	5×5 Highs	Haar	$[160, 80, 40, 20]$	1.57M	81.24
$[5, 4, 3, 2]$	5×5	db2	$[320, 160, 80, 40]$	2.04M	81.68
$[5, 4, 3, 2]$	5×5	db3	$[640, 320, 160, 80]$	2.04M	81.56

5   Limitations

Although the WTConv layer does not require many FLOPs, its running time can be relatively high within existing frameworks. This is due to the overhead for the multiple sequential operations (WT-conv-IWT), which may be more costly than the calculation itself. We note, however, that this can be alleviated by using a specialized implementation, *e.g.*, performing WT in parallel to convolution in each level to reduce memory reads or performing WT and IWT in-place to reduce memory allocations. More implementation details are provided in **Appendix C**.

6   Conclusion

In this work, we utilize wavelet transforms to introduce WTConv, a drop-in replacement for depth-wise convolutions that achieves a larger receptive field and better captures low frequencies in the input. Using WTConv, one can configure a spatial mixing of a global receptive field in a pure convolutional way. We demonstrate empirically that WTConv substantially increases the CNNs' effective receptive fields, improve the CNNs' shape bias, renders the networks more robust to corruptions, and yields better performance for various vision tasks.

References

1. Alaba, S.Y., Ball, J.E.: WCNN3D: wavelet convolutional neural network-based 3D object detection for autonomous driving. Sensors **22**(18), 7010 (2022)
2. Cai, Z., Vasconcelos, N.: Cascade R-CNN: high quality object detection and instance segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **43**(5), 1483–1498 (2019)
3. Chen, Y., et al.: Drop an octave: reducing spatial redundancy in convolutional neural networks with octave convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision,pp. 3435–3444 (2019)
4. Chi, L., Jiang, B., Mu, Y.: Fast Fourier convolution. Adv. Neural. Inf. Process. Syst. **33**, 4479–4488 (2020)
5. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017)
6. Contributors, M.: MMDetection: OpenMMLab detection toolbox and benchmark (2018). https://github.com/open-mmlab/mmdetection
7. Contributors, M.: MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark (2020). https://github.com/open-mmlab/mmsegmentation
8. Daubechies, I.: Ten Lectures on Wavelets. SIAM (1992)

9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)

10. Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31×31: revisiting large kernel design in CNNs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11963–11975 (2022)

11. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. In: International Conference on Learning Representations (2021)

12. Duan, Y., Liu, F., Jiao, L., Zhao, P., Zhang, L.: SAR image segmentation based on convolutional-wavelet neural network and Markov random field. Pattern Recogn. **64**, 255–267 (2017)

13. Finder, S.E., Zohav, Y., Ashkenazi, M., Treister, E.: Wavelet feature maps compression for image-to-image CNNs. In: Advances in Neural Information Processing Systems (2022)

14. Fujieda, S., Takayama, K., Hachisuka, T.: Wavelet convolutional neural networks. arXiv preprint arXiv:1805.08620 (2018)

15. Gal, R., Hochberg, D.C., Bermano, A., Cohen-Or, D.: SWAGAN: a style-based wavelet-driven generative model. ACM Trans. Graph. (TOG) **40**(4), 1–11 (2021)

16. Gavrikov, P., Keuper, J.: Can biases in ImageNet models explain generalization? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22184–22194 (2024)

17. Geirhos, R., et al.: Partial success in closing the gap between human and machine vision. In: Advances in Neural Information Processing Systems, vol. 34 (2021)

18. 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: International Conference on Learning Representations (2019)

19. Grabinski, J., Keuper, J., Keuper, M.: As large as it gets – studying infinitely large convolutions via neural implicit frequency filters. Trans. Mach. Learn. Res. (2024)

20. Guo, M.H., Lu, C.Z., Liu, Z.N., Cheng, M.M., Hu, S.M.: Visual attention network. Comput. Vis. Media **9**(4), 733–752 (2023)

21. Guo, T., Seyed Mousavi, H., Huu Vu, T., Monga, V.: Deep wavelet prediction for image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 104–113 (2017)

22. Guth, F., Coste, S., De Bortoli, V., Mallat, S.: Wavelet score-based generative modeling. In: Advances in Neural Information Processing Systems (2022)

23. Haber, E., Lensink, K., Treister, E., Ruthotto, L.: IMEXnet a forward stable deep neural network. In: Proceedings of the 36th International Conference on Machine Learning (2019)

24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

25. Hendrycks, D., et al.: The many faces of robustness: a critical analysis of out-of-distribution generalization. In: Proceedings of the IEEE International Conference on Computer Vision (2021)

26. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: Proceedings of the International Conference on Learning Representations (2019)

27. Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., Song, D.: Natural adversarial examples. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
28. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
29. Huang, H., He, R., Sun, Z., Tan, T.: Wavelet-SRNet: a wavelet-based CNN for multi-scale face super resolution. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1689–1697 (2017)
30. Jeevan, P., Viswanathan, K., Sethi, A., et al.: WaveMix: a resource-efficient neural network for image analysis. arXiv preprint arXiv:2205.14375 (2022)
31. Li, Z., et al.: Robust deep learning object recognition models rely on low frequency information in natural images. PLOS Comput. Biol. (2023)
32. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
33. Liu, P., Zhang, H., Zhang, K., Lin, L., Zuo, W.: Multi-level wavelet-CNN for image restoration. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 773–782 (2018)
34. Liu, S., et al.: More convnets in the 2020s: Scaling up kernels beyond 51×51 using sparsity. In: International Conference on Learning Representations (2023)
35. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022 (2021)
36. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Conference on Computer Vision and Pattern Recognition (2022)
37. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
38. Michaelis, C., et al.: Benchmarking robustness in object detection: autonomous driving when winter is coming. arXiv preprint arXiv:1907.07484 (2019)
39. Mintun, E., Kirillov, A., Xie, S.: On interaction between augmentations and corruptions in natural corruption robustness. In: Advances in Neural Information Processing Systems (2021)
40. Naseer, M.M., Ranasinghe, K., Khan, S.H., Hayat, M., Shahbaz Khan, F., Yang, M.H.: Intriguing properties of vision transformers. In: Advances in Neural Information Processing Systems, vol. 34, pp. 23296–23308 (2021)
41. Park, N., Kim, S.: How do vision transformers work? arXiv preprint arXiv:2202.06709 (2022)
42. Phung, H., Dao, Q., Tran, A.: Wavelet diffusion models are fast and scalable image generators. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10199–10208 (2023)
43. Rao, Y., Zhao, W., Zhu, Z., Lu, J., Zhou, J.: Global filter networks for image classification. In: Advances in Neural Information Processing Systems, vol. 34, pp. 980–993 (2021)
44. Ronen, M., Finder, S.E., Freifeld, O.: DeepDPM: deep clustering with an unknown number of clusters. In: Conference on Computer Vision and Pattern Recognition (2022)
45. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28

46. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
47. Saragadam, V., LeJeune, D., Tan, J., Balakrishnan, G., Veeraraghavan, A., Baraniuk, R.G.: WIRE: wavelet implicit neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18507–18516 (2023)
48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2015)
49. Trockman, A., Kolter, J.Z.: Patches are all you need? Trans. Mach. Learn. Res. (2023)
50. Tuli, S., Dasgupta, I., Grant, E., Griffiths, T.L.: Are convolutional neural networks or transformers more like human vision? arXiv preprint arXiv:2105.07197 (2021)
51. Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., Van Gool, L.: SCAN: learning to classify images without labels. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12355, pp. 268–285. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58607-2_16
52. Vanhoucke, V.: Learning visual representations at scale. In: ICLR Invited Talk, vol. 1, no. 2 (2014)
53. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
54. Wang, H., Ge, S., Lipton, Z., Xing, E.P.: Learning robust global representations by penalizing local predictive power. In: Advances in Neural Information Processing Systems, pp. 10506–10518 (2019)
55. Wang, H., Wu, X., Huang, Z., Xing, E.P.: High-frequency component helps explain the generalization of convolutional neural networks. In: Conference on Computer Vision and Pattern Recognition (2020)
56. Wang, M., Liu, B., Foroosh, H.: Factorized convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 545–553 (2017)
57. Wang, T., Lu, C., Sun, Y., Yang, M., Liu, C., Ou, C.: Automatic ECG classification using continuous wavelet transform and convolutional neural network. Entropy (2021)
58. Williams, T., Li, R.: Wavelet pooling for convolutional neural networks. In: International Conference on Learning Representations (2018)
59. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 418–434 (2018)
60. Zhou, B., et al.: Semantic understanding of scenes through the ADE20K dataset. Int. J. Comput. Vision **127**(3), 302–321 (2019)