Wavelet Convolutions for Large Receptive Fields Appendices

Shahaf E. Finder, 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

A The Algorithm

Algorithm 1 WTConv

Input: $X \in \mathbb{R}^{C \times N_w \times N_h}$
$X_{LL}^{(0)} = X$
$Y_{LL}^{(0)} = \text{Conv}(W^{(0)}, X_{LL}^{(0)})$
for $i = 1, \ldots, \ell$ **do**
  $X_{LL}^{(i)}, X_{LH}^{(i)}, X_{HL}^{(i)}, X_{HH}^{(i)} = \text{WT}(X_{LL}^{(i)})$
  $Y_{LL}^{(i)}, Y_{LH}^{(i)}, Y_{HL}^{(i)}, Y_{HH}^{(i)} = \text{Conv}(W^{(i)}, (X_{LL}^{(i)}, X_{LH}^{(i)}, X_{HL}^{(i)}, X_{HH}^{(i)}))$
end for
$Z^{(\ell+1)} = 0$
for $i = \ell, \ldots, 1$ **do**
  $Z^{(i)} = \text{IWT}(Y_{LL}^{(i)} + Z^{(i+1)}, Y_{LH}^{(i)}, Y_{HL}^{(i)}, Y_{HH}^{(i)})$
end for
$Z^{(0)} = Y_{LL}^{(0)} + Z^{(1)}$
return $Z^{(0)}$

B Training Parameters

B.1 Imagenet-50/100/200

The training process is similar to [3], using a 2-GPU setup, an SGD optimizer with a momentum of 0.9, a batch size of 32 per GPU, input resolution 224×224, weight decay of $4{\cdot}10^{-5}$, and a 5-epoch linear warmup followed by cosine annealing for 100 epochs. The initial learning rate was adjusted to 0.025, taking the number of GPUs into consideration. The implementation is based on [8].

B.2 ImageNet-1K

We follow [3, 6] for the 300-epochs training schedule with AdamW [7] with momentum 0.9 and weight decay 0.05, batch size 4096, learning rate $4 \cdot 10^{-3}$, a

20-epoch linear warm-up followed by cosine annealing, RandAugment [2], label smoothing [9] coefficient 0.1, mixup [12] with $\alpha = 0.8$, CutMix [11] with $\alpha = 1.0$, Random Erasing [13] with probability 25%, Stochastic Depth [5] with a drop-path rate of 10%/40%/50% for T/S/B variations respectively, and an exponential moving average (EMA) with a decay factor of 0.9999. We also provide comparisons for a shorter training schedule (which follows [3]) whose configuration is similar to the one above, except that it uses 120 epochs, batch size 2048, 10-epoch warm-up, and no EMA.

C Naive implementation running times

The implementation used for all our experiments is fairly naive, and many improvements can be made with it. For example, in addition to what we described in Section 5, the Haar wavelet is currently implemented in our model as a regular convolution, which includes multiplications by 1 and -1 as FP32. These operations, however, can be changed to summation and substractions and can be done for all the levels at the same time for more efficient memory reads.

Using the naive implementation, we timed the throughput of ConvNeXt and WTConvNeXt using a single RTX3090 for 300 batches of size 64 after a GPU warmup of 50 batches. Table 9 presents the throughput, measured in images per second. As can be seen, even with the most naive and unoptimized version of the layer, WTConvNeXt has a throughput of 66-70% of the original network.

Table 9: Naive implementation throughput measured by images per second.

Model	Images/sec
ConvNeXt-T	976.65
WTConvNeXt-T	652.76
ConvNeXt-S	565.52
WTConvNeXt-S	384.91
ConvNeXt-B	371.99
WTConvNeXt-B	260.12

D ImageNet-1K - additional results

To demonstrate WTConv compatibility, we incorporate it within two additional networks, GhostNet [4] and EfficientNet [10]. The number of WTConv levels is set to have a global receptive field at each stage w.r.t. input size of 224×224. The training procedure is as described in § B.2 for the 120-epochs training schedule. The results are presented in Table 10.

E quantitative shape-bias results

The quantitative shape-bias results are reported in Table 11.

Table 10: Additional ImageNet-1K results with 120-epochs schedule.

Configuration	Params.	Top-1
GhostNet	5.2M	67.41
+WTConv	5.7M	**68.22**
EfficientNet-B0	5.3M	73.25
+WTConv	6.4M	**74.34**

Table 11: The average fraction of "shape" decisions for every network.

	T	S	B
ConvNeXt	21.7%	21.2%	24.5%
WTConvNeXt	**29.2%**	**32.9%**	**35.3%**

F Object detection - additional results

Table 12 provides the detailed results for COCO object detection experiment, as described in **§4.3**.

Table 12: COCO object detection and segmentation results using Cascade Mask-RCNN [1] with different backbones pre-trained on ImageNet-1k using 120/300-epochs training schedule, and fine-tuned with 1x/3x schedule.

Configuration	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
120-epoch pre-train + 1x finetune						
ConvNeXt-T	47.3	65.9	51.5	41.1	63.2	44.4
ConvNeXt-T + RepLK	47.8	66.7	52.0	41.4	63.9	44.7
WTConvNeXt-T	**47.9**	66.7	52.0	**41.5**	64.0	44.9
300-epoch pre-train + 3x finetune						
Swin-T	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-T	50.4	69.1	54.8	43.7	66.5	47.3
WTConvNeXt-T	**51.0**	69.9	55.3	**44.4**	67.2	48.1
Swin-S	51.9	70.7	56.5	45.0	68.2	48.8
ConvNeXt-S	51.9	70.8	56.4	45.0	68.4	49.1
WTConvNeXt-S	**52.6**	71.5	57.1	**45.6**	69.0	49.5

G Robustness - additional results

Table 13 and Table 14 provide the detailed results for ImageNet-C and ImageNet-$\overline{\text{C}}$, respectively. Figure 5, Figure 6, Figure 7, and Figure 8 are qualitative examples of object detection under different types of corruption. These examples show that as the corruption is more severe, WTConvNeXt loses fewer details.

Table 13: ImageNet-C detailed results.

			Noise			Blur			
Model	Err	mCE	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom
ConvNeXt-T	41.6	53.2	37.1	38.7	39.6	**50.4**	63.0	43.6	54.2
WTConvNeXt-T	**40.7**	**52.0**	**36.6**	**37.6**	**37.6**	50.9	**60.5**	**43.4**	**52.0**
ConvNeXt-B	36.6	46.8	32.0	33.1	31.7	46.0	56.3	38.4	47.2
WTConvNeXt-B	**35.6**	**45.6**	**30.0**	**31.3**	**30.9**	**45.7**	**55.8**	**36.3**	**44.6**

	Weather				Digital			
Model	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
ConvNeXt-T	42.1	38.5	**38.4**	**23.3**	**32.1**	46.0	44.5	**32.6**
WTConvNeXt-T	**41.4**	**37.9**	39.1	**23.3**	32.4	**45.8**	**40.0**	**32.6**
ConvNeXt-B	38.0	**33.3**	35.0	21.1	**28.1**	41.0	37.0	30.2
WTConvNeXt-B	**35.4**	33.8	**32.6**	**21.0**	31.8	**39.4**	**36.3**	**29.4**

Table 14: ImageNet-$\overline{\text{C}}$ detailed results.

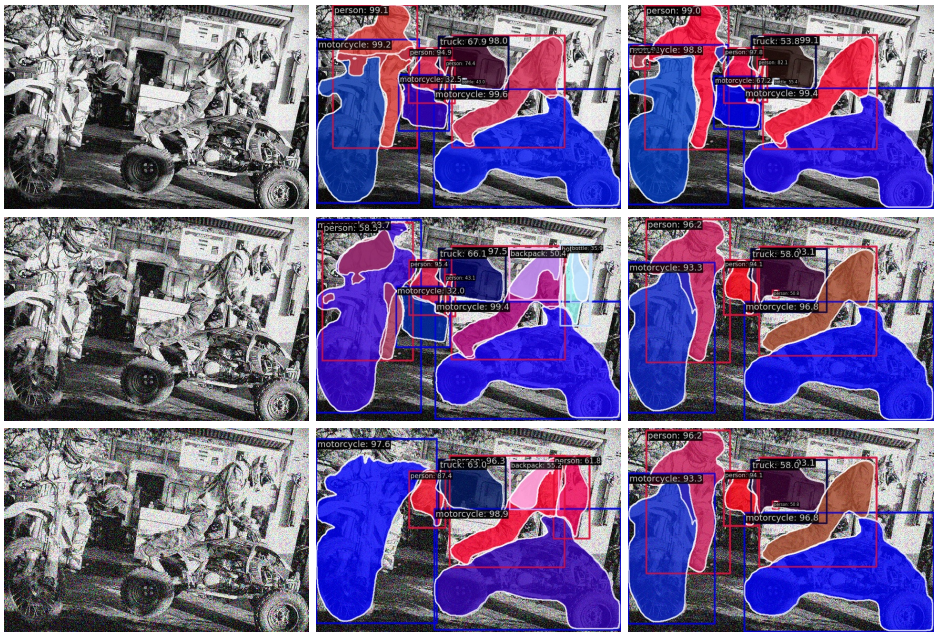
Model	Err	BSmpl	Plsm	Ckbd	CSin	SFrq	Brwn	Prln	Sprk	ISprk	Rfrac
ConvNeXt-T	40.0	30.8	43.8	33.0	69.0	53.5	24.9	**25.3**	26.8	59.4	**33.9**
WTConvNeXt-T	**38.0**	**29.8**	**40.5**	**31.3**	**60.0**	**49.3**	**24.8**	25.5	**26.3**	**59.0**	34.0
ConvNeXt-B	34.4	25.6	40.6	29.2	56.6	38.6	22.5	22.7	24.5	53.3	30.7
WTConvNeXt-B	**32.2**	**25.0**	**36.0**	**27.3**	**46.3**	**38.4**	**22.2**	**22.6**	**24.0**	**50.1**	**29.5**

Input image ConvNeXt-S backbone WTConvNeXt-S backbone

Fig. 5: Motion blur corruption example. Each row represents an increased corruption severity. Left to right: the corrupted input, ConvNeXt-S backbone detection, WTConvNeXt-S backbone detection. Note that WTConvNeXt detects the traffic light even at the worst corruption level.

Input image ConvNeXt-S backbone WTConvNeXt-S backbone

Fig. 6: Pixelate corruption example. Each row represents an increased corruption severity. Left to right: the corrupted input, ConvNeXt-S backbone detection, WTConvNeXt-S backbone detection. Note that WTConvNeXt detects more zebras than ConvNeXt and classifies them more accurately.



Input image ConvNeXt-S backbone WTConvNeXt-S backbone

Fig. 7: Gaussian noise corruption example. Each row represents an increased corruption severity. Left to right: the corrupted input, ConvNeXt-S backbone detection, WTConvNeXt-S backbone detection. WTConvNeXt clearly captures the object better as the corruption level increases.

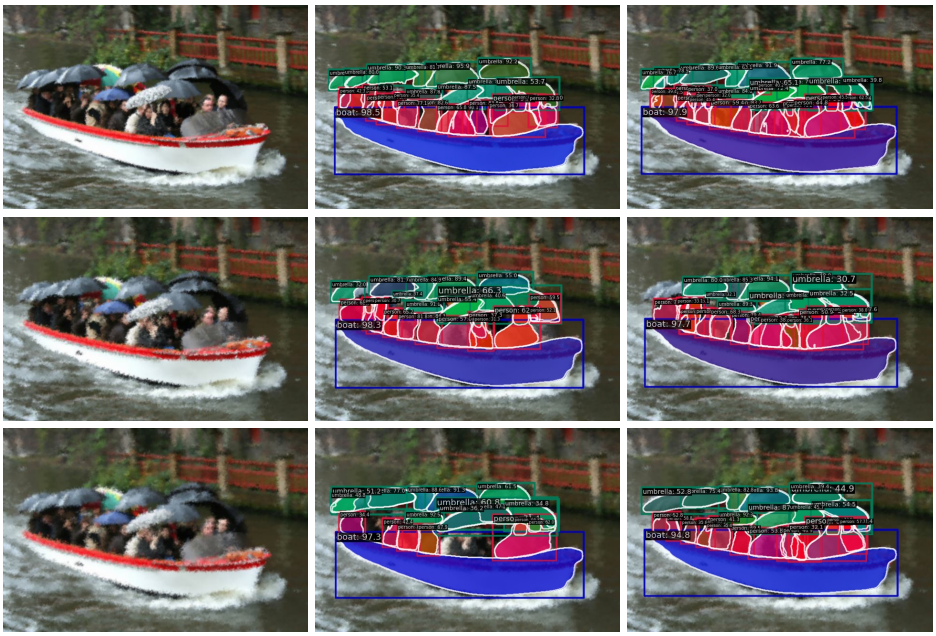


Fig. 8: Glass blur corruption example. Each row represents an increased corruption severity. Left to right: the corrupted input, ConvNeXt-S backbone detection, WTConvNeXt-S backbone detection. Note that ConvNeXt misses some of the persons on the boat, while WTConvNeXt classifies them correctly.

References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: high quality object detection and instance segmentation. IEEE transactions on pattern analysis and machine intelligence **43**(5), 1483–1498 (2019)
2. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 702–703 (2020)
3. Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11963–11975 (2022)
4. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: More features from cheap operations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1580–1589 (2020)
5. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 646–661. Springer (2016)
6. 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)
7. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
8. maintainers, T., contributors: Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision (2016)
9. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
10. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
11. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6023–6032 (2019)
12. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018)
13. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 13001–13008 (2020)