TransNeXt: Robust Foveal Visual Perception for Vision Transformers Appendix

Dai Shi
daishiresearch@gmail.com
Code: https://github.com/DaiShiResearch/TransNeXt

A. Equivalent Form of Pixel-Focused Attention

Mathematically, pixel-focused attention is equivalent to the following form:

$$\begin{aligned} K_{concat} &= Concat(K_{\rho(i,j)}, K_{\sigma(X)}) \\ V_{concat} &= Concat(V_{\rho(i,j)}, V_{\sigma(X)}) \end{aligned} \tag{17}$$

$$\textbf{PFA}(X_{(i,j)}) = \text{softmax}(\frac{Q_{(i,j)}K_{concat}^T}{\sqrt{d}} + B_{(i,j)})V_{concat} \tag{18}$$

This form is more concise mathematically. However, in parallel computation, merging $\rho(i,j)$ and $\sigma(X)$ on key and value to form $K_{concat}, V_{concat}\ in\ \mathbb{R}^{nh\times HW\times(H_pW_p+k^2)\times d}$ creates two large temporary tensors. This results in significant memory usage and memory access pressure, severely slowing down the model speed. Therefore, in practical applications, we use the method of separately calculating the attention of the two paths and only adding the results. Calculating the attention weight from the concatenated similarity result in the same softmax is crucial, as it ensures the mathematical equivalence of these two forms.

B. Comparative Analysis of Human Vision and Attention Visualization

The human visual system is characterized by a dichotomy between foveal and peripheral vision. The foveal vision, covering only 1 to 2 degrees of the central field of view, is optimized for high sensitivity, while the peripheral vision, with a much larger receptive field, is optimized for a broad field of view. This dichotomy suggests that humans primarily utilize peripheral vision for object localization and navigation, but lack precision in detail. To compensate for this, the human eye executes rapid movements, known as saccades, allowing for the processing of information from multiple fields of view and the subsequent integration of this information. Recent research [1] on biological vision suggests that when superior foveal vision information is available, the human brain does not simply discard peripheral vision information, but compares the inputs from both, weighing their relative reliability to complete the integration of information. This process is a highly complex computation known as transsaccadic perception.

Pixel-focused attention, in comparison to human vision, maintains a high degree of similarity with the human visual system in numerous design aspects. Empirical studies D.5 demonstrate that the sliding window path in pixel-focused attention, which simulates foveal vision and continuous eye movements, can achieve superior performance with a minimal 3×3 window size for perception. This experimental result aligns closely with the extremely narrow field of view coverage of human foveal vision. Particularly in the context of high-resolution image input, the coverage range of the 3×3 sliding window is exceedingly small. The pooling path in pixel-focused attention, which simulates human peripheral vision, maintains full-image perception and overlaps with the fine-grained perception area of the sliding window path. The features of these two window paths compete and correlate in the same softmax. This design enables the model to compare the detailed features perceived by foveal vision with the comprehensive rough outline of the object perceived by peripheral vision in terms of similarity, thereby integrating and calibrating information across the field of view. From the perspective of biological vision, the current design of pixel-focused attention can effectively simulate the highly complex transsaccadic perception in human visual perception. In contrast, the Focal Transformer, which employs a window partitioning approach, is unable to simulate the continuous movements of the human eye. Furthermore, its method of pooling only the peripheral windows prevents the model from fully perceiving the comprehensive rough outline of the object, thereby increasing its inconsistency with human transsaccadic perception.

In the terminal attention layer of TransNeXt's third stage, we employ the query token located at the center position $(H \div 2, W \div 2)$ of the feature map (subsequently represented by the $[center]$ subscript), to calculate $\text{softmax}(\tau \log N * \hat{Q}_{[center]}\hat{K}^T)$, and $\text{softmax}(\tau \log N * \hat{Q}_{[center]}\hat{K}^T_{\sigma(X)})$ re-

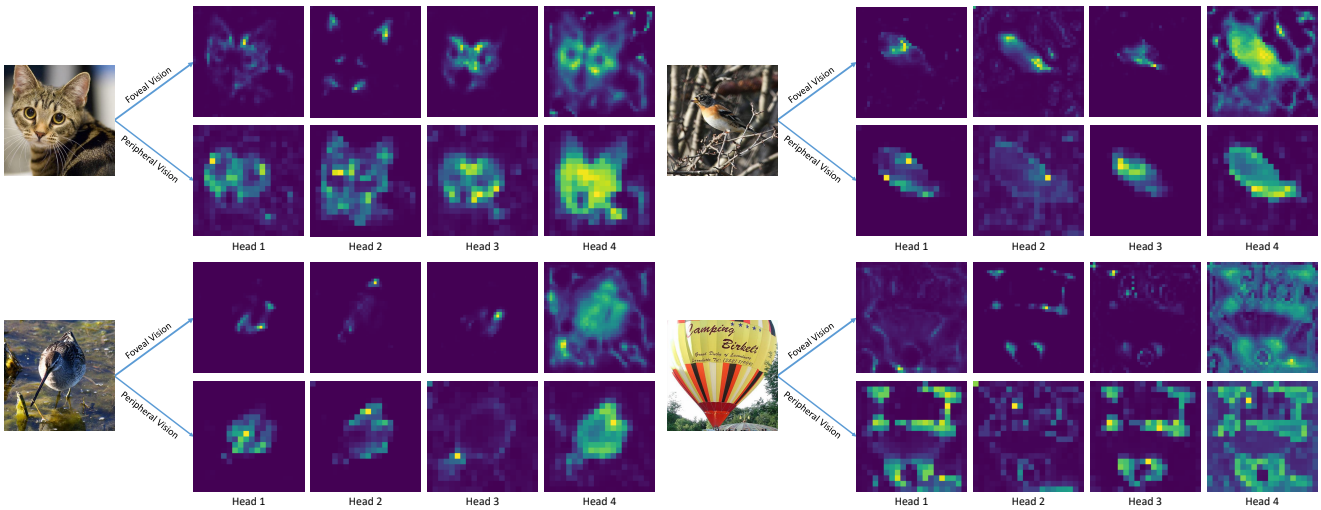

Figure 7. The attention map of foveal and peripheral vision when the visual focus is centered. The central query token of the feature map is utilized to compute softmax$(\tau \log N * \hat{Q}_{[center]}\hat{K}^T)$ and softmax$(\tau \log N * \hat{Q}_{[center]}\hat{K}^T_{\sigma(X)})$. For effective visualization, we employ a high-resolution image input of 640^2 and calculate the attention map using the final attention layer of stage 3. It's important to note that during the model's standard operation, the foveal vision perception only utilizes the features of the $k \times k$ area near the query. However, given that this area is too small to provide sufficient information for observation, we use undownsampled global features for visualization purposes, allowing us to discern the features of interest to the foveal vision perception.

spectively. This allows us to visualize the attention map of the image feature perception in the sliding window path and the pooling feature path at the central position of the picture, with the results presented in Fig 7. It can be observed that in the same head, the sliding window path simulating foveal vision consistently maintains interest in specific textures, fluff, and high sharpness edges, while the pooling path simulating peripheral vision consistently has a reliable grasp of the rough outline of the object. The features of the two paths are further matched and calibrated in the same softmax, resulting in more accurate output. We posit that this working method closely resembles the integration method of foveal and peripheral vision found in human vision research, further substantiating the high consistency of our biomimetic design with human vision.

C. Detailed Settings

C.1. Configurations of TransNeXt Variants

C.2. Training Settings for ImageNet-1K

To ensure reproducibility and consistency with prior work, we adopt the training strategy of PVTv2 [26], which incorporates various data augmentation techniques, including Random Augmentation [4], Mixup [31], CutMix [30], and Random Erasing [32]. To regularize our model, we employ Label Smoothing [23] and DropPath [9]. We optimize our model using AdamW [19] optimizer with a gradient clipping norm of 1.0 and a weight decay of 0.05. The initial learning rate for all models is set to 10^{-3}, with a warm-up period of 5 epochs and an initial warm-up learning rate of 10^{-6}. We utilize the cosine learning rate scheduler [18] to decay the learning rate. During training, we randomly crop images to a size of 224×224. During the evaluation phase, for images with a resolution less than 384×384, we apply a center-crop with a crop ratio of 0.875. However, for images of larger sizes, we do not perform any cropping, following previous work [17]. We do not employ the EMA weights. The stochastic depth drop rates for each model are provided in Table 4.

C.3. Training Settings for Downstream Tasks

For experiments on the ADE20K [33] and COCO [12] datasets, we followed the training settings of Swin [15]. We utilized the MMDetection [2] and MMSegmentation [3] toolboxes for training.

For the COCO 2017 dataset [12], we configured the learning rate to 10^{-4} and the weight decay to 0.05. In the context of the Mask R-CNN and DINO methods, the stochastic depth drop rates for TransNeXt-Tiny, TransNeXt-Small, and TransNeXt-Base were set to 0.3, 0.5, and 0.6, respectively. The model was trained for 12 epochs with a batch size of 16 using the standard $1\times$ schedule.

For the ADE20K dataset [33], in the UperNet method, we set the learning rate to 6×10^{-5} and the weight decay to 0.05. The stochastic depth drop rates for TransNeXt-Tiny, TransNeXt-Small, and TransNeXt-Base were set to 0.4, 0.6, and 0.7, respectively. For the Mask2Former method, we set the learning rate to 10^{-4} and the weight decay to 0.05, with the stochastic depth drop rates for TransNeXt-Tiny, TransNeXt-Small, and TransNeXt-Base set to 0.3, 0.5, and 0.6 respectively. All models were trained for 160K iterations

Model	Channels	Head dims	Blocks	MLP ratio	Token mixer	Window size	Pool size
TransNeXt-Micro	[48, 96, 192, 384]	24	[2, 2, 15, 2]	[8, 8, 4, 4]	**A-A-A-M**	[3, 3, 3, –]	[7, 7, 7, –]
TransNeXt-Tiny	[72, 144, 288, 576]	24	[2, 2, 15, 2]	[8, 8, 4, 4]	**A-A-A-M**	[3, 3, 3, –]	[7, 7, 7, –]
TransNeXt-Small	[72, 144, 288, 576]	24	[5, 5, 22, 5]	[8, 8, 4, 4]	**A-A-A-M**	[3, 3, 3, –]	[7, 7, 7, –]
TransNeXt-Base	[96, 192, 384, 768]	24	[5, 5, 23, 5]	[8, 8, 4, 4]	**A-A-A-M**	[3, 3, 3, –]	[7, 7, 7, –]

Table 3. The configurations of TransNeXt variants. The value of pool size is calculated at 224^2 resolution. **A** = aggregated attention, while **M** = multi-head self-attention.

dataset	**ImageNet-1K**	
configuration task	TransNeXt-Micro/Tiny/Small/Base 224^2 Pre-training	TransNeXt-Small/Base 384^2 Fine-tuning
batch size	1024	1024
base learning rate	1e-3	1e-5
learning rate scheduler	cosine	constant
min learning rate	1e-5	1e-5
training epochs	300	5
warm-up epochs	5	None
warm-up schedule	linear	None
warm-up learning rate	1e-6	None
optimizer	AdamW	AdamW
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$	$\beta_1, \beta_2 = 0.9, 0.999$
color jitter factor	0.4	0.4
auto-aug	rand-m9-mstd0.5-inc1	rand-m9-mstd0.5-inc1
random-erasing prob.	0.25	0.25
random-erasing mode	pixel	pixel
mixup α	0.8	0.8
cutmix α	1.0	None
mixup prob.	1.0	1.0
mixup switch prob.	0.5	0.5
stochastic drop path rate	0.15/0.25/0.45/0.6	0.7/0.8
label smoothing	0.1	0.1
gradient clip	1.0	1.0
weight decay	0.05	0.05
exp. mov. avg. (EMA)	None	None

Table 4. The pre-training and fine-tuning settings of TransNeXt on ImageNet-1K [5].

with a batch size of 16 on the ADE20K dataset.

D. Ablation Study

D.1. More Discussion on roadmap from PVT to TransNeXt

Understanding of query embedding: The query embedding exhibits very unique properties. Incorporating query embedding effectively improved the performance on ImageNet-1K val and ImageNet-V2 test sets but somewhat reduced performance on ImageNet-A, ImageNet-R, ImageNet-Sketch test sets; its impact on ImageNet-C was very weak. Notably, ImageNet-1K val, ImageNet-V2, and ImageNet-C (a distorted test set of ImageNet-1K val) adopted the same sampling strategy as the ImageNet-1K training set, while ImageNet-A, ImagetNet-R, and ImageNet-Sketch did not follow this principle. We believe these experimental results reflect that query embedding restricts the model's response range to enhance current task performance rather than affecting generalization to all types of data. During the learning process, the model optimizes this learnable query token, implicitly learning what the optimal question for the current task is in each attention layer (from a Visual Question Answering (VQA) perspective). This perspective can well explain why in these out-of-distribution test sets, query embedding has a very weak impact on the performance of the ImageNet-C test set which uses the same sampling strategy as the training set. Therefore, we believe there is a potential trade-off here. In the case of TransNeXt, even with query embedding, our model still achieved state-of-the-art model robustness.

Impact of model structure: We adjusted the width and depth of PVTv2 and the number of attention heads to match those of TransNeXt-Micro in steps 1 to 3 to avoid the impact of model structure. During this period, we observed that a deeper and thinner model significantly enhances performance. Reducing the head dimension from 48 to 24 resulted in only a 0.04% performance change, indicating that the performance gain from increasing attention heads is extremely limited.

D.2. Detailed Data of Multi-scale Inference

Model	Method	Inference Size						
		224^2	256^2	320^2	384^2	480^2	512^2	640^2
TransNeXt-Tiny	**Normal Mode**	84.0	84.3	84.3	84.6	83.8	83.2	**81.6**
	No Length-scaling	84.0	84.3	84.4	84.7	83.7	83.2	80.9
	Interpolate RPE	84.0	84.1	84.2	84.3	83.1	82.4	79.5
	Linear Mode	84.0	84.0	83.9	84.1	83.0	82.6	80.7
RepLKNet-31B [6]		83.5	83.6	81.0	70.0	21.4	10.1	0.9
SLaK-S [14]		83.8	83.8	83.2	79.6	65.7	63.7	61.4
ConvNeXt-B [17]		83.8	84.2	84.0	83.6	81.6	80.7	77.3
TransNeXt-Mirco	Normal Mode	82.5	82.8	82.9	83.1	82.1	81.6	79.3
	Linear Mode	82.5	82.5	82.4	82.3	80.9	80.3	77.6
TransNeXt-Small	Normal Mode	84.7	84.9	84.9	85.0	84.1	83.8	82.2
	Linear Mode	84.7	84.7	84.7	84.9	84.0	83.6	81.7
TransNeXt-Base	Normal Mode	84.8	85.1	85.1	85.5	84.7	84.3	82.8
	Linear Mode	84.8	85.0	84.9	85.1	84.1	83.5	81.5

Table 5. The table shows the top-1 accuracy of ImageNet-1K of 224^2-size trained TransNeXt under **normal** and **linear** inference modes on multiple image input sizes. At the same time, the effects of length-scaled cosine attention and log-CPB on multi-scale inference were tested, and the pure convolution model was included for comparison.

Linear complexity mode for inference: We observe that in Equations 15 and 16, if we consistently set H_p and W_p as fixed values independent of the input size, the computational complexity of both pixel-focused attention and aggregated attention grows linearly with the length of the input sequence. In this scenario, both pixel-focused attention and aggregated

attention can operate under a linear complexity mode. This linear mode endows TransNeXt with a computational complexity growth curve close to that of a pure convolutional network when inferring large-size images. We test the performance changes of 224^2-size trained TransNeXt and two prevalent pure convolutional models at multiple resolutions. In the default normal mode, H_p and W_p of aggregated attention are $\frac{1}{32}$ of the input image size, while in the linear mode, H_p and W_p are fixed at $\frac{1}{32}$ of the training image size, *i.e.*, 7×7.

Results and analysis: As shown in Table 5 and Fig 6, our TransNeXt-Tiny achieves better multi-scale extrapolation performance than pure convolutional models in both normal and linear modes. At the maximum resolution of 640^2, the linear mode produces a performance decay of 0.5% to 1.7% relative to the normal mode, but such a trade-off still has advantages over pure convolutional models. As the image size increases, the performance decay of ConvNeXt-B is greater than that of TransNeXt's linear mode. RepLKNet-31B shows a more exaggerated performance decay, with a top-1 accuracy of only 0.9% at a resolution of 640^2, which to some extent reveals the limitations of the super-large convolution kernel scheme. In traditional opinions, pure convolutional models have better multi-scale applicability than ViT models, and such experimental results also imply that this opinion needs to be re-examined.

Impact of length-scaled cosine attention: We compare the performance of length-scaled cosine attention with regular scaled cosine attention during multi-scale inference. According to Fig 6, length-scaling begins to take effect when the resolution reaches 640^2. This implies that when the sequence length variation in softmax exceeds $8\times$, longer sequence lengths begin to significantly reduce the confidence of scaled cosine attention.

Extrapolation vs Interpolation for relative position bias: When a TransNeXt model trained at a resolution of 224^2 infers at other sizes, we default to using log-CPB [16] to extrapolate the $B_{(i,j)\sim\sigma(X)}$ under new resolutions from spatial relative coordinates $\Delta_{(i,j)\sim\sigma(X)}$. However, generating $\Delta_{(i,j)\sim\sigma(X)}$ cannot achieve the same speed as model inference. This is not a major issue in general because when the model needs to continuously infer at one or several new sizes, we only need to pre-calculate these new $\Delta_{(i,j)\sim\sigma(X)}$ and cache them. However, when the new inference resolution of the model is unknown and needs to change instantly according to input size, we need to use traditional interpolation schemes for relative position bias to interpolate $B_{(i,j)\sim\sigma(X)}$. As depicted in Fig 6, the input resolution of 640^2 results in a significant performance degradation due to interpolation for relative position bias, surpassing that of the linear mode. This underscores the efficacy of log-CPB in extrapolating position bias. In our evaluation of UperNet with multi-scale and flip augmentations (Table 14), we present test results under both interpolation and extrapolation for a balanced comparison, highlighting the influence of different schemes on multi-scale performance.

D.3. Ablation on Positional Encoding

Method	Params(M)	FLOPs(G)	Top-1(%)
Remove $B_{(i,j)}$	28.1	5.6	83.2
Calculate $B_{(i,j)\sim\rho(i,j)}$ by **log-CPB**$(\Delta_{(i,j)\sim\rho(i,j)})$	28.2	5.7	83.7
Replace $B_{(i,j)\sim\rho(i,j)}$ by $Q_{(i,j)}T$	28.2	5.7	83.4
Replace **log-CPB**$(\Delta_{(i,j)\sim\sigma(X)})$ by learnable $B_{(i,j)\sim\sigma(X)}$	28.1	5.6	**84.0**
TransNeXt-Tiny	28.2	5.7	**84.0**

Table 6. Ablation experiments on the design of relative position biases.

Results and analysis: We conducted ablation experiments on the design of the relative position bias used in the sliding window path and pooling feature path in aggregated attention, with results shown in Table 6. When we completely removed the relative position bias $B_{(i,j)}$ used in aggregated attention, the model's performance significantly decreased by 0.8%. This indicates that using depthwise convolution to capture positional information from zero-padding is insufficient to represent the positional relationships of global tokens. When we used log-CPB to calculate the relative position bias of the sliding window, it also resulted in a 0.3% performance decline. This suggests that due to different feature scales, the numerical meanings of spatial coordinates $\Delta_{(i,j)\sim\sigma(X)}$ in the pooling feature path and $\Delta_{(i,j)\sim\rho(i,j)}$ in sliding window path are not exactly the same, highlighting the importance of using different methods to learn relative position bias in the two paths. Another consideration is to use dynamic relative position bias $Q_{(i,j)}T$ calculated by positional attention to replace $B_{(i,j)\sim\rho(i,j)}$, but this resulted in a significant performance decline of 0.6%. We believe this is due to inconsistencies in the behavior of the sliding window path and pooling path. The **log-CPB**$(\Delta_{(i,j)\sim\sigma(X)})$ calculated in the pooling path is static, while $Q_{(i,j)}T$ dynamically changes with input, and the two paths are coupled in the same softmax, causing interference with the mechanism of QKV attention. If we also use a learnable relative position bias $B_{(i,j)\sim\sigma(X)}$ instead of calculating by **log-CPB**$(\Delta_{(i,j)\sim\sigma(X)})$ in the pooling path, it does not affect model performance, but it does cause the model to lose its ability to extrapolate position biases for unknown size inputs. This demonstrates the similarity between the relative position biases calculated through log-CPB and those directly learned, also indicating that the log-CPB module is not the source of TransNeXt's high performance.

D.4. Ablation on the Design of Convolutional GLU

We conducted ablation experiments on the design of convolutional GLU on the CIFAR-100 dataset using a 2M-sized model. We designed three optional variants, all using GELU

as the activation function:

$$\textbf{ConvGLU}(X) = (XW_1 + B_1) \odot \text{GELU}(\text{DWConv}(XW_2 + B_2)) \tag{19}$$

$$\textbf{Type-1}(X) = (XW_1 + B_1) \odot \text{DWConv}(\text{GELU}(XW_2 + B_2)) \tag{20}$$

$$\textbf{Type-2}(X) = \text{DWConv}(XW_1 + B_1) \odot \text{GELU}(XW_2 + B_2) \tag{21}$$

$$\textbf{Type-3}(X) = \text{DWConv}((XW_1 + B_1) \odot \text{GELU}(XW_2 + B_2)) \tag{22}$$

The experiments, presented in Table 7, showed that our convolutional GLU, which follows the design philosophy of gated channel attention, is the optimal design. In Type-1, placing DWConv after the gated activation function disrupts the effect of setting value to zero in the gating branch. In Type-2, moving DWConv to the value branch causes a significant performance drop of 0.7% when a gating branch with a smaller receptive field controls a value branch with a larger receptive field, indicating that it is more reasonable to make gating decisions using a branch with a larger receptive field. In Type-3, adding a DWConv after the element-wise dot product result in the GLU module leads to the worst performance, suggesting that merely adding a DWConv to enhance local perceptual ability is not key to improving model performance with convolutional GLU.

Design	Params. (M)	FLOPs (G)	Top-1(%)
ConvGLU	2.3	0.5	**82.9**
Type-1	2.3	0.5	82.6
Type-2	2.3	0.5	82.2
Type-3	2.3	0.5	82.1

Table 7. Ablation study on the design of Convolutional GLU on CIFAR-100 [10] dataset.

D.5. Ablation on Window Size

We conducted fast ablation experiments on CIFAR-100 [10] using a 2M-sized model, results are reported in Table 8. Our observations indicate that an increase in the window size does not necessarily lead to an enhancement in the model's performance. We believe that these experimental results are due to the introduction of pooling features provides coarse-grained global perception abilities, greatly reducing the demand for single queries to perceive the sliding window field. Moreover, the fine-grained tokens overlap with the coarse-grained tokens, leading to additional inductive bias. Since the similarity results between queries and both fine-grained and coarse-grained tokens compete in the same softmax, this approach benefits information aggregation in overlapping regions. However, as the window size increases, this inductive bias may not always be beneficial.

Window Size	Params. (M)	FLOPs (G)	Top-1(%)
3×3	2.3	0.50	82.9
5×5	2.3	0.52	82.0
7×7	2.3	0.54	82.9
9×9	2.3	0.57	82.5

Table 8. The ablation results of window size. We utilized a 2M-sized TransNeXt model to conduct experiments on the CIFAR-100 [10] dataset under various window size settings.

D.6. Ablation on Model Architecture

To further explore the impact of model architecture on performance, we conducted ablation experiments based on TransNeXt-Micro. We attempted to replace aggregated attention with multi-head self-attention in stages 1-3 to observe its impact on model performance. The experimental results are presented in Table 9. We observed that when we replaced aggregated attention with multi-head self-attention in stage 3, where the number of blocks is the highest, the model performance decreased by 0.5%. Further replacement in stage 2 led to an additional 0.1% decline in performance. This suggests that our aggregated attention information aggregation method has advantages over global self-attention. When we tried to replace aggregated attention in stage 1, the model encountered an out-of-memory error on $8\times$ A100s with 80GB of memory, making it impossible to train the model with this configuration.

Under a resolution of 224^2, 7×7 is the smallest size that can be achieved by integer multiple downsampling. For this reason, and to maintain consistency with PVTv2, our model opted for a pooling size of $\frac{1}{32}$ at each stage. However, in stage 4, the input resolution has already been reduced to $\frac{1}{32}$, rendering the downsampling module of aggregated attention ineffective. If aggregated attention is forcibly applied at this stage, features in the sliding window would be input into softmax twice through the pooling path, leading to distortion in importance calculation. Consequently, we selected MHSA in stage 4. At larger resolutions, such as 256^2, we can employ a pooling size of $\frac{1}{64}$ at each stage to implement a model that fully utilizes aggregated attention at all stages. As demonstrated in Table 9, a micro-sized model that fully employs aggregated attention achieved an ImageNet-1K accuracy of 82.6% at a resolution of 256^2.

D.7. CUDA Implementation

In the native PyTorch [21] implementation, feature extraction in the sliding window path is achieved through the unfold operation. The unfold operation involves two stages: 1) ex-

Token mixer	Input size	Window size	Pool size	Params. (M)	FLOPs (G)	Top-1(%)
A-A-A-M	224^2	3×3	7×7	12.8	2.7	**82.5**
A-A-M-M	224^2	3×3	7×7	12.2	2.7	82.0
A-M-M-M	224^2	3×3	7×7	12.2	2.9	81.9
M-M-M-M	224^2	3×3	7×7	12.2	4.7	**OOM**
A-A-A-A	256^2	3×3	4×4	13.1	3.3	**82.6**

Table 9. Ablation study on model architecture on ImageNet-1K dataset. **OOM** means out of memory error.

tracting the tensor within the sliding window through index access, and 2) explicitly creating a large tensor copy for the extracted tensor. This explicit feature extraction operation generates a huge temporary tensor and induces memory access pressure, which significantly reduces the model's speed. To address this, we introduce a CUDA operator implementation for calculating QK similarity and aggregating value by attention weights in the sliding window path. This implementation circumvents the need for explicit tensor extraction from the sliding window, thereby markedly enhancing the model's throughput and training speed. As shown in Table 10, our CUDA implementation provides up to 60.5% acceleration for inference, up to 103.4% acceleration for training and saves up to 16.8% of memory consumption for training.

On the other hand, our current implementation is solely based on native CUDA, without the introduction of substantial optimization. Consequently, it does not match the efficiency of highly optimized dense GPU operators. We evaluated the throughput of our model on a V100 16G with FP32 precision at a batch size of 64, following the methodology of Swin [15]. As shown in Table 11, our biomimetic vision implementation significantly outperforms the Focal Transformer [28] method in terms of both efficiency and accuracy. Moreover, under similar throughput conditions, our model demonstrates competitive top-1 accuracy on ImageNet-1K compared to previous state-of-the-art models such as MaxViT [25], BiFormer [34], and QuadTree [24]. Despite the current throughput of the model still having a certain gap with models [15, 17] benefiting from dense operator implementations, the performance of TransNeXt is anticipated to improve with further engineering efforts. We are committed to providing more efficient operator optimizations in the future to enhance the competitiveness of TransNeXt.

E. Downstream Experimental Results

Model	Throughput of inference			Duration of training (sec/iter)			Memory usage (GB)		
	CUDA	Pytorch	Acceleration	CUDA	Pytorch	Acceleration	CUDA	Pytorch	Saving
TransNeXt-Micro	1117	701	+59.3%	0.218	0.401	+83.9%	14.8	17.8	16.8%
TransNeXt-Tiny	756	471	+60.5%	0.315	0.609	+93.3%	23.2	27.3	15.0%
TransNeXt-Small	394	246	+60.2%	0.595	1.161	+95.1%	41.6	49.3	15.6%
TransNeXt-Base	297	186	+59.6%	0.771	1.568	+103.4%	58.1	68.6	15.3%

Table 10. Performance comparison between CUDA implementation and native PyTorch implementation. We measure throughput using a batch size of 64 on a single V100 with 16GB of memory under FP16, while the iteration time and memory consumption during training are measured on 8× A100s (PCIe) with a total batch size of 1024 under automatic mixed precision.

Model	#Params. (M)	FLOPs (G)	Top-1 (%)	Throughput (img/s)
Swin-T [15]	28.3	4.5	81.2	790
ConvNeXt-T [17]	28.6	4.5	82.3	779
MaxViT-Tiny [25]	30.9	5.6	83.4	459
TransNeXt-Tiny (Ours)	28.2	5.7	**84.0**	413
BiFormer-S [34]	25.5	4.5	83.8	396
QuadTree-B-b2 [24]	24.2	4.5	82.7	361
Focal-T [28]	29.1	4.9	82.2	337

Table 11. Comparison of throughput between TransNeXt and its main competitor models. The throughput results were tested on a single V100 GPU under FP32 precision with a batch size of 64. The results are sorted in descending order of throughput.

Backbone	Encoder size(M)	#Params. (M)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Swin-T [15]	28.3	47.8	43.7	66.6	47.7	39.8	63.3	42.7
PVTv2-B2 [26]	25.4	45.3	45.3	67.1	49.6	41.2	64.2	44.4
FocalNet-T (LRF) [29]	28.6	48.9	46.1	68.2	50.6	41.5	65.1	44.5
Swin-S [15]	49.6	69.1	46.5	68.7	51.3	42.1	65.8	45.2
CSWin-T [7]	23	42	46.7	68.6	51.3	42.2	65.6	45.4
Swin-B [15]	87.8	107.1	46.9	69.2	51.6	42.3	66.0	45.5
PVTv2-B3 [26]	45.2	64.9	47.0	68.1	51.7	42.5	65.7	45.7
InternImage-T [27]	30	49	47.2	69.0	52.1	42.5	66.1	45.8
PVTv2-B5 [26]	82.0	101.6	47.4	68.6	51.9	42.5	65.7	46.0
PVTv2-B4 [26]	62.6	82.2	47.5	68.7	52.0	42.7	66.1	46.1
InternImage-S [27]	50	69	47.8	69.8	52.8	43.3	67.1	46.7
SMT-S [13]	20.5	40.0	47.8	69.5	52.1	43.0	66.6	46.1
BiFormer-S [34]	25.5	45.2	47.8	69.8	52.3	43.2	66.8	46.5
CSWin-S [7]	35	54	47.9	70.1	52.6	43.2	67.1	46.2
FocalNet-S (LRF) [29]	50.3	72.3	48.3	70.5	53.1	43.1	67.4	46.2
BiFormer-B [34]	56.8	76.3	48.6	70.5	53.8	43.7	67.6	47.1
CSWin-B [7]	78	97	48.7	70.4	53.9	43.9	67.8	47.3
InternImage-B [27]	97	115	48.8	70.9	54.0	44.0	67.8	47.4
SMT-B [13]	32	51.7	49.0	70.2	53.7	44.0	67.6	47.4
FocalNet-B (LRF) [29]	88.7	111.4	49.0	70.9	53.9	43.5	67.9	46.7
TransNeXt-Tiny (Ours)	28.2	47.9	**49.9**	70.5	53.7	43.9	67.4	**47.5**
TransNeXt-Small (Ours)	49.7	69.3	**51.1**	**72.6**	**56.2**	**45.5**	**69.8**	**49.1**
TransNeXt-Base (Ours)	89.7	109.2	**51.7**	**73.2**	**56.9**	**45.9**	**70.5**	**49.7**

Table 12. Detailed COCO object detection and instance segmentation results using the Mask R-CNN [8] 1× schedule, sorted in ascending order based on AP^b performance.

Model	Encoder size(M)	#Params. (M)	Epochs	scales	Pre-trained	AP^b
ConvNeXt-B [17]	88.6	110	12	4	IN-1K (384^2)	52.6
ConvNeXt-L [17]	198	221	12	4	IN-1K (384^2)	53.4
TransNeXt-Tiny (Ours)	**28.2**	**47.8**	12	4	IN-1K (**224^2**)	**55.1**
TransNeXt-Tiny (Ours)	28.2	48.1	12	5	IN-1K (224^2)	**55.7**
TransNeXt-Small (Ours)	49.7	69.6	12	5	IN-1K (224^2)	**56.6**
TransNeXt-Base (Ours)	89.7	110	12	5	IN-1K (224^2)	**57.1**
Swin-L [15]	197	218	12	5	IN-22K (384^2)	57.2

Table 13. Comparison of object detection results on the COCO dataset using the DINO method. The data for Swin [15] is sourced from MMDetection [2], while the data for ConvNeXt [17] is referenced from detrex [22] project. The results are sorted in ascending order based on the AP^b scores.

Model	Encoder size(M)	#Params. (M)	Crop -size	Pre-trained	mIoU (%)	+MS (%)
Swin-T [15]	28.3	60	512^2	IN-1K	44.5	45.8
Focal-T [28]	29.1	62	512^2	IN-1K	45.8	47.0
ConvNeXt-T [17]	28.6	60	512^2	IN-1K	46.0	46.7
FocalNet-T(LRF) [29]	28.6	61	512^2	IN-1K	46.8	47.8
Swin-S [15]	49.6	81	512^2	IN-1K	47.6	49.5
UniFormer-S [11]	22	52	512^2	IN-1K	47.6	48.5
Focal-S [28]	51.1	85	512^2	IN-1K	48.0	50.0
Swin-B [15]	87.8	121	512^2	IN-1K	48.1	49.7
ConvNeXt-S [17]	50.2	82	512^2	IN-1K	48.7	49.6
Focal-B [28]	89.8	126	512^2	IN-1K	49.0	50.5
FocalNet-S(LRF) [29]	50.3	84	512^2	IN-1K	49.1	50.1
ConvNeXt-B [17]	88.6	122	512^2	IN-1K	49.1	49.9
SMT-S [13]	20.5	50.1	512^2	IN-1K	49.2	50.2
SMT-B [13]	32	61.8	512^2	IN-1K	49.6	50.6
UniFormer-B [11]	49.8	80	512^2	IN-1K	50.0	50.8
FocalNet-B(LRF) [29]	88.7	126	512^2	IN-1K	50.5	51.4
TransNeXt-Tiny (Ours)	28.2	59	512^2	IN-1K	**51.1**	**51.5/51.7**
TransNeXt-Small (Ours)	49.7	80	512^2	IN-1K	**52.2**	**52.5/52.8**
ConvNeXt-B [17]	88.6	122	640^2	IN-22K	52.6	53.1
TransNeXt-Base (Ours)	89.7	121	**512^2**	**IN-1K**	**53.0**	**53.5/53.7**

Table 14. A comprehensive comparison of semantic segmentation results on the ADE20K dataset using the UperNet method. +MS for evaluation with multi-scale and flip augmentations. In the context of multi-scale evaluation, TransNeXt reports test results under two distinct scenarios: interpolation and extrapolation of relative position bias. The results are sorted in ascending order based on the mIoU scores.

Model	Encoder size(M)	#Params. (M)	Crop -size	Pre-trained	mIoU(%)
Swin-S [15]	49.6	68.8	512^2	IN-1K (224^2)	51.2
Swin-B [15]	87.8	107	640^2	IN-1K (224^2)	52.4
TransNeXt-Tiny (Ours)	**28.2**	**47.5**	**512^2**	IN-1K (224^2)	**53.4**
Swin-B [15]	87.8	107	640^2	IN-22K (384^2)	53.9
TransNeXt-Small (Ours)	**49.7**	**69.0**	**512^2**	**IN-1K (224^2)**	**54.1**
TransNeXt-Base (Ours)	89.7	109	**512^2**	**IN-1K (224^2)**	**54.7**

Table 15. Comparison of semantic segmentation results on the ADE20K dataset using the Mask2Former method. The data for Swin [15] is sourced from MMSegmentation [3]. The results are sorted in ascending order based on the mIoU scores.

F. Visualization Based on Effective Receptive Field

We employ the Effective Receptive Field (ERF) [20] method as a visualization tool to analyze the information aggregation approach of TransNeXt. In Fig 8, we visualize the ERF of four encoder stages for eight models: TransNeXt-Tiny, ConvNeXt-T, Swin-T, CSWin-T, Focal-T, MaxViT-Tiny, BiFormer-S, and SLaK-T. In Fig 9, we further conduct a comprehensive ERF visualization comparison on the fourth stage of the models on ImageNet-A, ImageNet-Sketch, and ImageNet-C datasets.

Our observations are as follows:

1. As depicted in Fig 8, in the comparative visualization of ERF for multi-stage outputs, TransNeXt-Tiny outperforms seven other models in ERF coverage at the third stage, exhibiting a more natural and smoother visual perception. This can partially explain TransNeXt's performance advantage in detection and segmentation tasks, as these tasks rely more heavily on lower-stage outputs.
   In contrast, ConvNeXt, Swin, and CSWin exhibit distinct blocky patterns, which we attribute to artifacts from their token mixer designs. Despite the presence of multiple layers, these token mixers are unable to eliminate artifacts induced by window-based local attention or convolution kernels, resulting in an unnatural information mixing.
   Notably, although Focal Transformer has designed a unique biomimetic attention mechanism, it is not exempt from this issue. The method it employs, which is based on window partitioning, still results in significant blocky artifacts. These blocky artifacts are also markedly evident in MaxViT, which utilizes a hybrid CNN-ViT architecture. Moreover, the grid sampling methodology employed in MaxViT's token mixer introduces additional grid artifacts, further demonstrating the prevalence of these artifact phenomena across different models and designs.
   Cutting-edge ViT and convolutional models have made some improvements in this regard. BiFormer utilizes a data-driven approach that enables window-based ViT models to autonomously select window combinations, thereby circumventing the unnatural traces caused by manual window design. SLaK employs an ultra-large convolution kernel scheme to achieve global perception, thus mitigating the receptive field degradation caused by depth degradation in convolutional models. We observe that these methods have to some extent alleviated the unnatural visual perception caused by manual window design or convolution kernel stacking in previous ViT and convolutional models, but the unnatural blocky artifacts caused by window partitioning and convolution kernel stacking are still not entirely eliminated.
   This observation supports the experimental evidence that deep networks with residual blocks function as ensembles of shallower networks, highlighting the significance of a single token mixer in achieving a local-global modeling approach that is more akin to biological vision. TransNeXt's ERF represents an information perception methodology that aligns more closely with biological vision, achieving a natural visual perception and validating the effectiveness of its biomimetic design.
2. As shown in Fig 9, in a comprehensive visualization evaluation across multiple out-of-distribution test sets, TransNeXt-Tiny demonstrates a more adaptive information perception method. Its effective receptive field's information perception method undergoes significant changes with different datasets. This change can be clearly observed at multiple severity levels on ImageNet-C. Meanwhile, Swin-T's ERF exhibits a similar pattern across all test sets, and ConvNeXt-T's ERF lies somewhere in between. We believe that a more adaptive ERF reflects the model's robustness, and such visualization comparison results are consistent with the robustness evaluation results in Table 1.

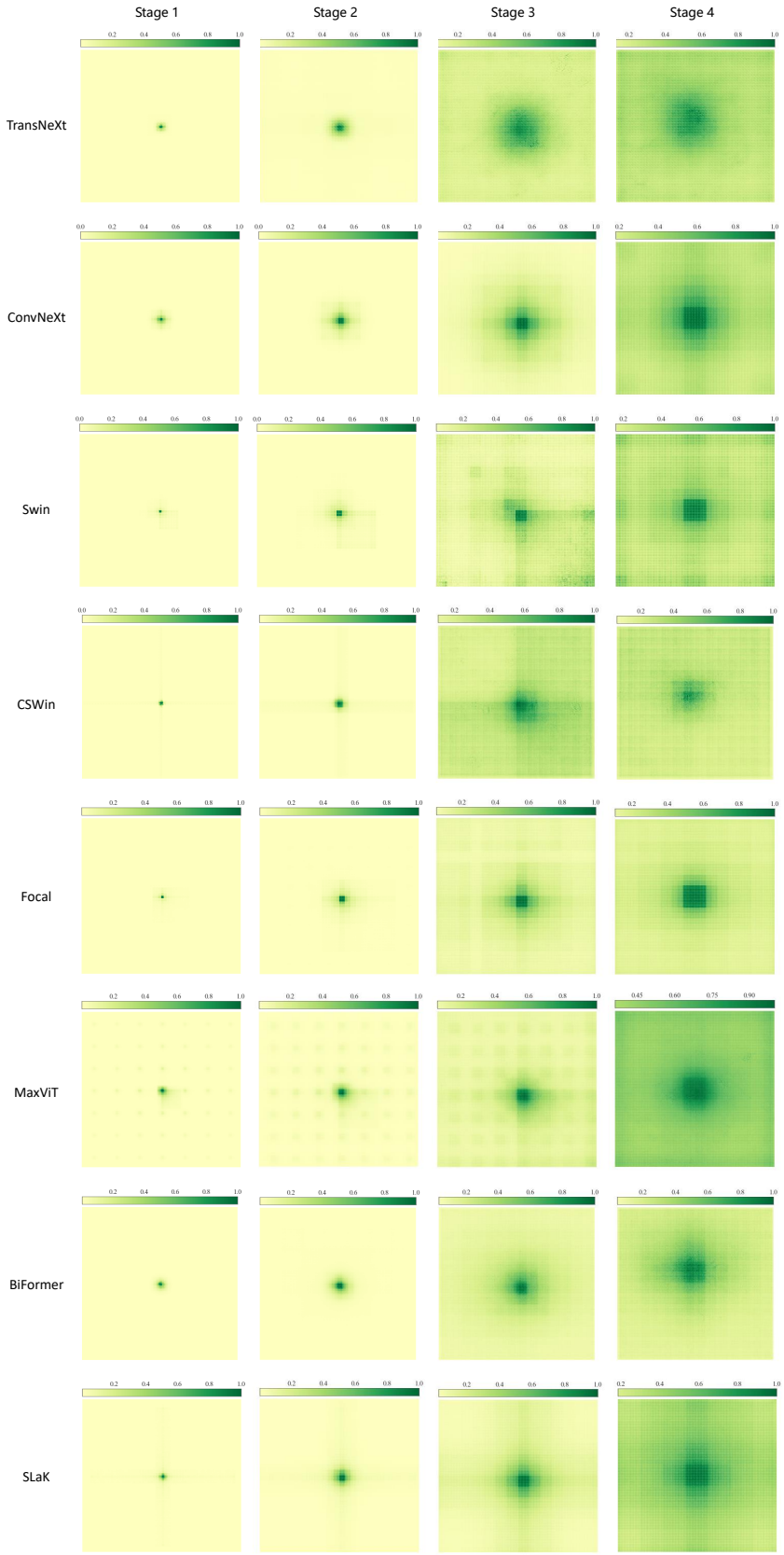

Figure 8. Visualization of the Effective Receptive Field (ERF) on ImageNet-1K validation set. Each visualization is based on an average of 5000 images with a resolution of 224×224. We visualize the ERFs of four stages for eight models: TransNeXt-Tiny, ConvNeXt-T, Swin-T, CSWin-T, Focal-T, MaxViT-Tiny, BiFormer-S, and SLaK-T.

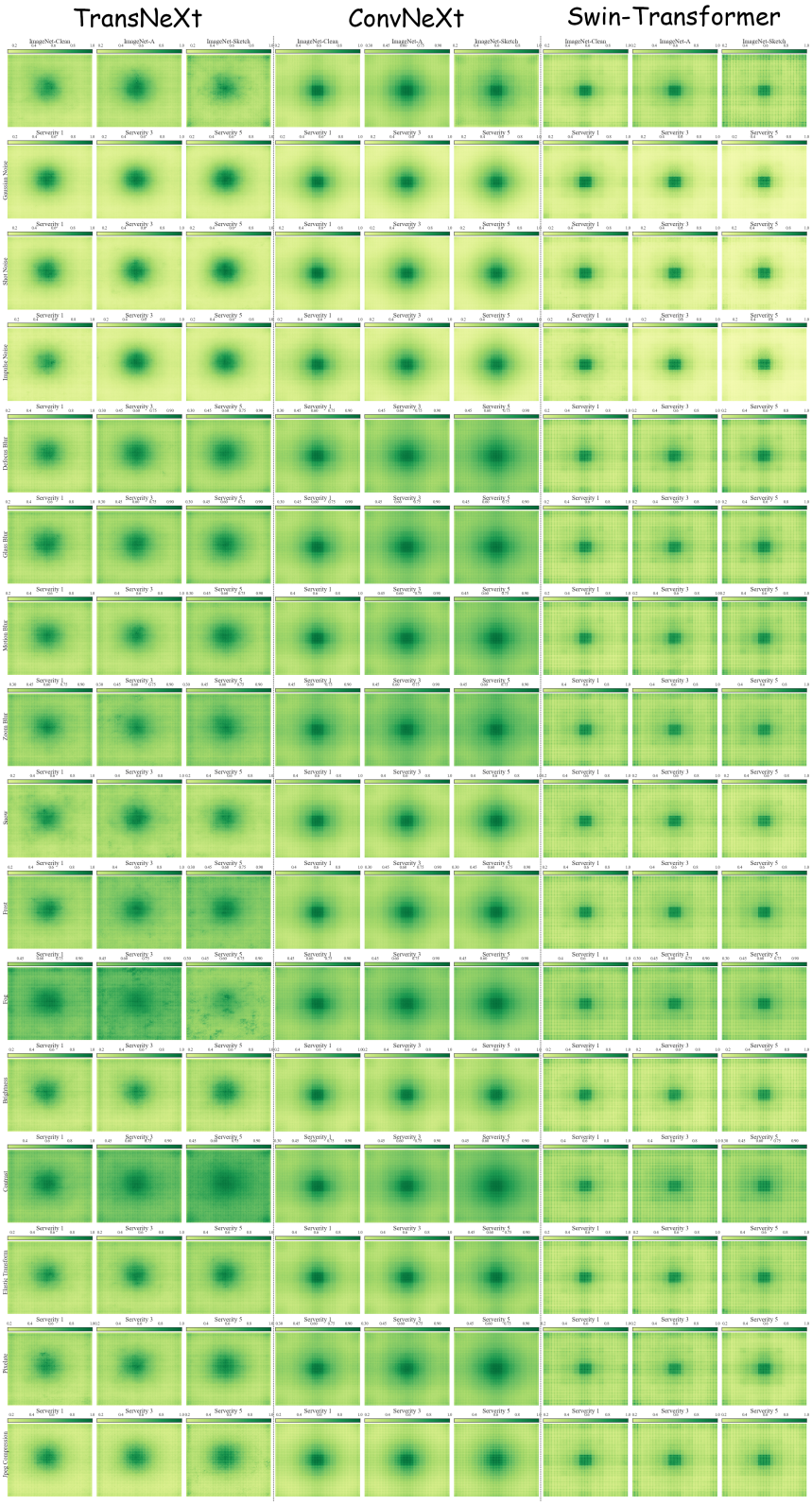

Figure 9. Visualization of the Effective Receptive Field (ERF) for TransNeXt-Tiny, ConvNeXt-T, and Swin-T on various datasets including ImageNet-1K validation set (Clean), ImageNet-Adversarial, ImageNet-Sketch, and ImageNet-C. The visual analysis diagrams for ImageNet-C commence from the second row of the figure. For each corruption mode, we have included visual images with severity levels of 1, 3, and 5. Each ERF image is generated by averaging over 5000 images with a resolution of 224×224 from each dataset.

References

[1] Calibration and integration of peripheral and foveal information in human vision. https://cordis.europa.eu/project/id/676786, 2022. 1

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019. 2, 7

[3] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 2, 7

[4] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 3008–3017. Computer Vision Foundation / IEEE, 2020. 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. 3

[6] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31×31: Revisiting large kernel design in cnns. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11953–11965. IEEE, 2022. 3

[7] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12114–12124. IEEE, 2022. 7

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):386–397, 2020. 7

[9] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 646–661. Springer, 2016. 2

[10] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 5

[11] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 7

[12] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 740–755. Springer, 2014. 2

[13] Weifeng Lin, Ziheng Wu, Jiayu Chen, Jun Huang, and Lianwen Jin. Scale-aware modulation meet transformer. *CoRR*, abs/2307.08579, 2023. 7

[14] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Constantin Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 3

[15] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021. 2, 6, 7

[16] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer V2: scaling up capacity and resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11999–12009. IEEE, 2022. 4

[17] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11966–11976. IEEE, 2022. 2, 3, 6, 7

[18] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 2

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 2

[20] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4898–4906, 2016. 8

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An

imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. 5

[22] Tianhe Ren, Shilong Liu, Feng Li, Hao Zhang, Ailing Zeng, Jie Yang, Xingyu Liao, Ding Jia, Hongyang Li, He Cao, Jianan Wang, Zhaoyang Zeng, Xianbiao Qi, Yuhui Yuan, Jianwei Yang, and Lei Zhang. detrex: Benchmarking detection transformers, 2023. 7

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016. 2

[24] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 6, 7

[25] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan C. Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *Computer Vision - ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXIV*, pages 459–479. Springer, 2022. 6, 7

[26] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *CoRR*, abs/2106.13797, 2021. 2, 7

[27] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 14408–14419. IEEE, 2023. 7

[28] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *CoRR*, abs/2107.00641, 2021. 6, 7

[29] Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks. In *NeurIPS*, 2022. 7

[30] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6022–6031. IEEE, 2019. 2

[31] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 2

[32] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 13001–13008. AAAI Press, 2020. 2

[33] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *Int. J. Comput. Vis.*, 127(3):302–321, 2019. 2

[34] Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson W. H. Lau. Biformer: Vision transformer with bi-level routing attention. *CoRR*, abs/2303.08810, 2023. 6, 7