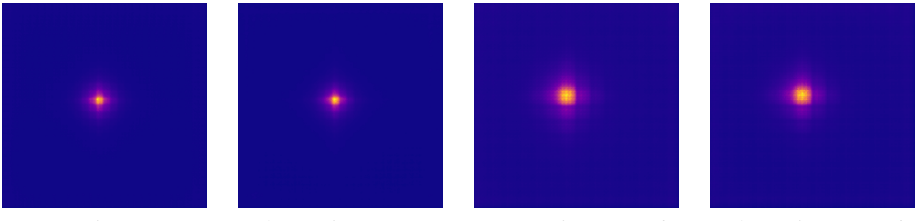
Understanding Gaussian Attention Bias of Vision Transformers Using Effective Receptive Fields (Supplementary Material)

Bum Jun Kim
kmbmjn@postech.ac.kr
Hyeyeon Choi
hyeyeon@postech.ac.kr
Hyeonah Jang
hajang@postech.ac.kr
Sang Woo Kim
swkim@postech.ac.kr

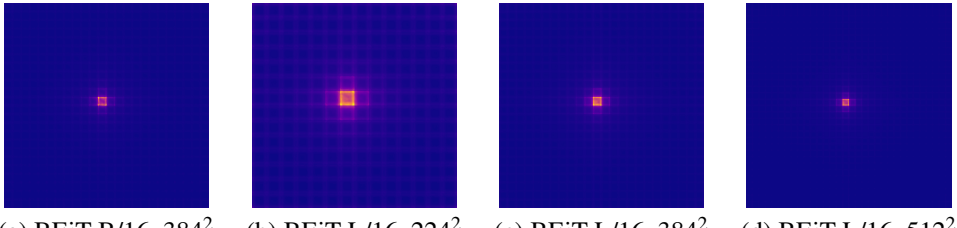
Department of Electrical Engineering
POSTECH
Pohang, South Korea

A More Experimental Results



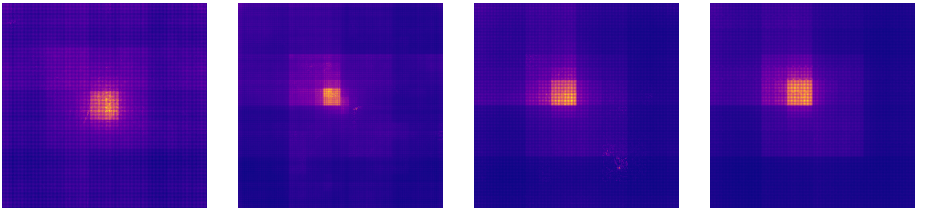
(a) XCiT-S12/8 (b) XCiT-L24/8 (c) XCiT-S12/16 (d) XCiT-L24/16

Figure 1: ERFs of XCiT with different models and patch sizes.



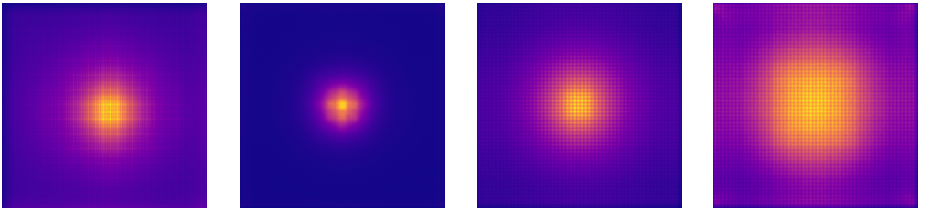
(a) BEiT-B/16, 384^2 (b) BEiT-L/16, 224^2 (c) BEiT-L/16, 384^2 (d) BEiT-L/16, 512^2

Figure 2: ERFs of BEiT with different model sizes and resolution.



(a) Swin-L, 224^2 (b) Swin-L, 384^2 (c) SwinV2-S, W8 (d) SwinV2-B, W8

Figure 3: ERFs of Swin and SwinV2. Exceptionally, SwinV2 uses 256^2 resolution.



(a) EfficientNet-B0 (b) EfficientNet-B7 (c) DenseNet-121 (d) DenseNet-201

Figure 4: ERFs of EfficientNet and DenseNet. Exceptionally, EfficientNet-B7 uses 600^2 resolution.

Further Analysis of ERFs Figure 1 shows the ERFs of XCiTs. Compared with the ERFs of ViT and DeiT, the ERF of XCiT exhibited a smooth shape resembling a 2D Gaussian. Although BEiT exhibited an almost identical architecture to ViT and DeiT, their ERFs highlighted the edge of the targeted patch (Figure 2).[1] The ERFs of Swin and SwinV2 were consistently wider and not restricted to the targeted patch (Figure 3). We further obtained the ERFs for other CNNs, including EfficientNets and DenseNets (Figure 4). Note that a larger resolution may mean a relatively smaller portion of ERF with respect to the image. This phenomenon occurs owing to scale mismatch: ViTs and CNNs perceive images as data in pixel units, resulting in ERFs in pixel units, whereas objects in an image can be represented at arbitrary resolutions, and thus their proportion with respect to the image matters more. Thus, a pixel-portion mismatch may exist.

Size of ERF Table 1 summarizes the size of the ERFs for ViTs and ResNets. Compared with the ResNets' ERFs that fitted suitably the Gaussian with R^2 close to 0.9, the ERFs of ViTs did not. XCiTs demonstrated $R^2 > 0.9$, nonetheless yielding small $\hat{\sigma}_X$ and $\hat{\sigma}_Y$ values around 11, which were smaller than those of ResNets. Note that a direct comparison of the ERF size between ViTs and ResNets would be unfair owing to their different effective strides. For example, XCiT with a patch size of 16 produces a 14×14 target feature from a 224×224 image and has an effective stride of 16, whereas ResNet-50 has an effective stride of 32 up to the target feature, resulting in a larger ERF. Although ERF represents the behavior of ViTs, its size rarely relates to its classification accuracy. For example, compared with DeiT-B/16, DeiT-S/16 showed a wider ERF but lower top-1 and top-5 accuracies. A larger model tended to yield a wider ERF and higher classification accuracy; thus, there was a correlation but not a causal relationship. This observation is contrary to the common belief that a larger receptive field is advantageous to performance gains. Although the size of

[1] For BEiT, we utilized a modified version using LayerScale and RPE, as described in their Appendix.

the receptive field has been widely emphasized in previous literature, we do not encourage comparing the size of the ERFs for ViTs to understand the superiority of their performance. This limitation also arises from the fact that the shape of the ERF is a pseudo-2D Gaussian that does not fit well with a 2D Gaussian.

Model	R^2	$\hat{\sigma_X}$	$\hat{\sigma_Y}$	Top-1 (%)	Top-5 (%)
ViT-S/16	0.819	6.925	6.459	81.388	96.136
ViT-B/16	0.032	159.531	2068.001	84.528	97.294
ViT-L/16	0.039	136.167	2191.316	85.840	97.818
ViT-S/16 (R)	0.784	5.381	5.403	81.462	95.822
ViT-M/16 (R)	0.789	6.980	6.190	82.460	96.084
ViT-B/16 (R)	0.782	6.423	7.016	82.496	96.140
DeiT-S/16	0.813	5.402	5.608	79.852	95.044
DeiT-B/16	0.820	5.183	5.186	81.990	95.736
DeiT III-S/16	0.802	6.212	6.000	81.368	95.460
DeiT III-B/16	0.858	6.103	6.179	83.784	96.588
DeiT III-L/16	0.849	6.519	6.508	84.774	97.038
BEiT-B/16	0.694	12.806	12.166	85.210	97.658
BEiT-L-16	0.721	15.189	14.137	87.476	98.304
CaiT-XXS-24	0.759	5.363	5.369	78.380	94.316
CaiT-S-24	0.780	7.124	6.676	83.452	96.572
XCiT-S24/16	0.908	11.755	11.607	82.580	96.010
XCiT-M24/16	0.915	11.228	11.213	82.640	95.982
XCiT-L24/16	0.921	11.252	11.178	82.896	95.886
Swin-S	0.203	97.528	96.528	83.208	96.320
Swin-B	0.285	128.219	119.284	85.266	97.564
Swin-L	0.333	102.800	97.167	86.316	97.902
SwinV2-S, W8, 256^2	0.423	62.451	62.287	83.852	96.644
SwinV2-S, W16, 256^2	0.224	81.573	77.192	84.220	96.864
SwinV2-B, W8, 256^2	0.531	50.058	47.541	84.252	96.926
SwinV2-B, W16, 256^2	0.170	88.263	84.476	84.604	97.088
ResNet-18	0.905	76.623	78.478	69.760	89.070
ResNet-50	0.948	59.849	60.882	80.374	94.602
ResNet-101	0.924	70.993	71.728	81.928	95.770
ResNet-152	0.893	70.048	71.815	82.824	96.124
WideResNet-101-2	0.949	56.633	63.792	78.842	94.284
ResNeXt-101-32x8d	0.922	65.591	70.820	79.316	94.520

Table 1: Results of fitting ERF to 2D Gaussian. All models use 224^2 resolution unless specified otherwise.

B Hyperparameters and Technical Details on Experiments

ImageNet-1K The ImageNet-1K dataset contains 1.28M images for 1,000 classes. For the image classification experiments with ImageNet-1K, we used the pytorch-image-models library also known as timm. We referred to the hyperparameter recipe described in the official documentation and the recipe of DeiT. For training, AdamW optimizer with learning rate 5×10^{-4}, epochs 400, warm-up learning rate 10^{-6}, cosine annealing schedule, weight decay 0.05, label smoothing 0.1, random erasing with probability 0.25, RandAugment of magnitude 9 and noise-std 0.5 with increased severity (rand-m9-mstd0.5-inc1), Mixup 0.2, Cutmix 1.0, stochastic depth 0.1, mini-batch size 288 per GPU, Exponential Moving Average of model weights with decay factor 0.99996, and image resolution 224^2 were used. An average of three runs was reported for each result. The training was conducted on an 8×A100 GPU machine.

Other Datasets The Oxford-IIIT Pet dataset contains 7K pet images from 37 classes; the Caltech-101 dataset includes 9K object images from 101 classes with a background category; the Stanford Cars dataset includes 16K car images from 196 classes; the Stanford Dogs dataset includes 20K dog images from 120 classes. These datasets are available on their official websites. Each dataset was split into training, validation, and test sets at a ratio of 70:15:15. All the experiments were conducted at a resolution of 224^2 using standard data augmentation, including random resized cropping to 256 pixels, random rotations within 15 degrees, color jitter with a factor of 0.4, random horizontal flip with a probability of 0.5, center cropping with 224-pixel windows, and mean-std normalization based on ImageNet statistics. We used ImageNet-1K pretrained weights. As our goal was to investigate whether Gaussian attention bias helps RPE to obtain a spatial understanding of images, we re-initialized the RPE to random parameters and injected Gaussian attention bias. For training, stochastic gradient descent with a momentum of 0.9, learning rate of 0.01, cosine annealing schedule with 200 iterations, weight decay of 0.0005, and mini-batch size of 128 were used. The model with the highest validation accuracy was obtained for 200 training epochs. These hyperparameters were obtained based on the accuracy of the validation set. An average of three runs was reported for each result. The training was conducted on a single RTX 3090 GPU machine.

Object Detection on the COCO Dataset For training and testing, we used the COCO 2017 dataset comprising 118K training images, 5K validation images, and 41K test images. We used Swin-S 224^2 pretrained on ImageNet-1K as the backbone of Mask R-CNN to conduct object detection and instance segmentation. For training, the 3× learning rate schedule, multi-scale crop, FP16 training, data augmentation rules of DETR and Sparse R-CNN, AdamW with learning rate 10^{-4}, and weight decay 0.05 were used. We used the MMDetection library for this experiment. The training was conducted on an 8×A100 GPU machine.

Semantic Segmentation on the ADE20K Dataset A pixel-wise segmentation model was trained using the ADE20K dataset containing more than 20K images and their corresponding semantic labels. We used Swin-S 224^2 pretrained on ImageNet-1K as the backbone of UPerNet. For training, the 160k learning rate schedule, 512^2 pixel crop for images and labels, mini-batch size of 2 per GPU, AdamW with learning rate 6×10^{-5} with polynomial decay, and weight decay 0.01 were used. We used the MMSegmentation library for this experiment. The training was conducted on an 8×A100 GPU machine.

Technical Details on ERF For the implementation of ERF, we should be aware that the class token corresponds to the zeroth index that shifts the targeted patch index. Furthermore, to correctly apply ReLU to each image, we recommend using a mini-batch size of 1. Note that ERF is affected by the weight of the model; for all experiments, ImageNet-1K pretrained ViTs were used. Furthermore, to obtain valid ERFs, we need to ensure that the correct options are selected for each pretrained model. For example, certain models use bilinear interpolation to resize the input image, whereas other models use bicubic interpolation. Certain models do not use ImageNet statistics but instead use other values such as zeros, ones, or halves for mean-std normalization. These options affect the performance of each model. We followed the correct options described in the official documentation and source code.

Details on ViTs Table 2 summarizes each ViT with their resolution $H \times W$, patch size P, number of blocks L, hidden size D, and number of heads. Note that the base model is larger than the medium model.

Model	Model Size	$H \times W$	P	L	D	# Heads
ViT-S/16, 224^2	Small	224^2	16	12	384	6
ViT-M/16, 224^2	Medium	224^2	16	12	512	8
ViT-B/16, 224^2	Base	224^2	16	12	768	12
ViT-L/16, 224^2	Large	224^2	16	24	1024	16

Table 2: Details on ViTs.

Library Version We used the following libraries in our experiments. However, our results are not limited to the specific version of the software:
- Ubuntu 18.04, CUDA 11.3, cuDNN 8.3.2, Python 3.9.12, PyTorch 1.12.0, timm 0.6.7, LMfit 1.1.0, NumPy 1.23.5, and MMCV 1.7.1.

C More Details on Gaussian Attention Bias

Initialization of Gaussian Attention Bias Although A_l and σ_l were set as learnable parameters like weights in a neural network, we should initialize them as well. The default initializations we recommend are $A_l = 2$ and $\sigma_l = 5$, which were used for all experiments. Furthermore, we verified the learned values of A_l and σ_l for each experiment. Table 3 summarizes their average across layers. Note that A_l and σ_l remained close to the default values, implying that the default initialization functioned as valid values. In addition, depending on the properties of the dataset or the required size of ERF, A_l and σ_l can become slightly higher or lower, as observed in experiments on the Stanford Cars dataset.

On Weight Decay We set the two parameters A_l and σ_l as learnable parameters; nonetheless, they should not be subjected to weight decay because they need to be trained to specific values. Note that for certain libraries such as PyTorch, weight decay is applied to all learnable parameters by default. Thus, we should explicitly apply weight decay to the two parameters with a coefficient of zero. This practice has been commonly deployed in PyTorch when modules such as PReLU employ learnable parameters that should not be subjected to weight decay. This trick was adopted in all experiments. A simple implementation example is provided below.

Dataset	Model	$E[A_l^*]$	$E[\sigma_l^*]$
Oxford-IIIT Pet	ViT-S/16 (R)	1.987	4.996
	ViT-M/16 (R)	1.991	4.997
	ViT-B/16 (R)	1.940	5.007
Caltech-101	ViT-S/16 (R)	1.946	5.012
	ViT-M/16 (R)	1.956	5.011
	ViT-B/16 (R)	1.918	5.014
Stanford Cars	ViT-S/16 (R)	2.334	4.839
	ViT-M/16 (R)	2.316	4.845
	ViT-B/16 (R)	2.262	4.846
Stanford Dogs	ViT-S/16 (R)	1.949	4.979
	ViT-M/16 (R)	1.937	4.988
	ViT-B/16 (R)	1.895	5.006

Table 3: Average values of learned A_l^* and σ_l^* over layers.

```
# ...
self.sigma_param = nn.Parameter(torch.Tensor([hp_sigma_param]))
self.amplitude_param = nn.Parameter(torch.Tensor([hp_amplitude_param]))

# ...
param_original = [param for name, param in model_ft.named_parameters() if "
    param" not in name]
param_GAB = [param for name, param in model_ft.named_parameters() if "param"
     in name]

# ...
optimizer_ft = optim.SGD([
        {"params": param_original, "weight_decay": args.hp_wd},
        {"params": param_GAB, "weight_decay": 0.0},
    ],lr=args.hp_lr, momentum=0.9)

# ...
```

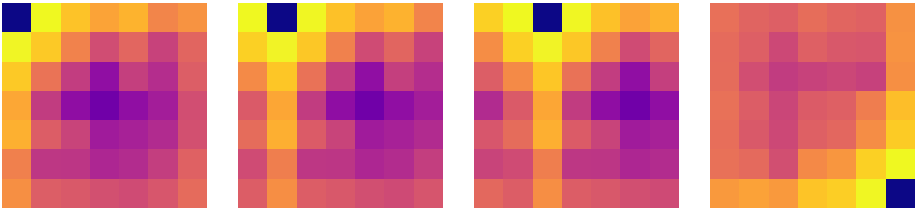
Listing 1: PyTorch example of applying weight decay on the model with Gaussian attention bias.

Ablation Study In the main text, we mentioned that we chose the head-shared Gaussian attention bias rather than the head-wise version. This choice arises because SA is implemented in a multi-head fashion. Table 4 summarizes the experimental results comparing head-shared and head-wise Gaussian attention bias. The performance difference was negligible; therefore, the choice of either is inconsequential. However, in general, head-shared implementation tends to yield slightly higher accuracy. Nevertheless, we observed that the head-wise Gaussian attention bias was at least better than the baseline accuracy obtained from RPE without Gaussian attention bias. This observation highlights that, although head-shared implementation provides more performance gain, head-wise Gaussian attention bias also provides a performance gain as it provides spatial understanding of images.

Dataset	RPE w/o GAB Baseline Val	RPE w/o GAB Baseline Test	RPE w/ GAB Head-shared Val	RPE w/ GAB Head-shared Test	RPE w/ GAB Head-wise Val	RPE w/ GAB Head-wise Test
Oxford-IIIT Pet	93.682	91.486	93.923	92.780	93.773	92.509
Caltech-101	89.959	88.403	91.296	90.202	91.126	90.591
Stanford Cars	81.294	80.126	84.205	83.079	84.411	82.928
Stanford Dogs	82.777	81.535	83.188	82.507	83.501	81.438

Table 4: Comparison of head-shared and head-wise Gaussian attention bias. ViT-S/16 (R) was used for these experiments.

Computation Analysis In terms of computation, because learned bias terms $\mathbf{B}_{\text{rel},l} + \mathbf{B}_{\text{Gaussian},l}$ can be precomputed, using our Gaussian attention bias does not require additional computation time during inference compared to using $\mathbf{B}_{\text{rel},l}$. In terms of the number of parameters, the use of Gaussian attention bias requires two additional parameters per layer, whose total amounts to only 24 or 48, which is minor for ViTs.



(a) $n = 0$ (b) $n = 1$ (c) $n = 2$ (d) $n = 48$

Figure 5: Learned RelPosBias of Swin-S for each patch index.

On RelPosBias The original Swin transformer uses RelPosBias that extracts RPE from a learnable table, whereas SwinV2 uses RelPosMlp, whose MLP yields RPE for each relative coordinate. Note that when RelPosBias is used, there is no guarantee that each value will be continuous with respect to the relative coordinate. However, owing to the piecewise linearity of MLP, RelPosMlp is expected to yield an RPE that is somewhat continuous with the relative coordinates. Because of this difference, the 2D Gaussian pattern is more likely to be observed in RelPosMlp. Figure 5 represents the learned RelPosBias of Swin-S. Note that the learned RelPosBias resembles a 2D Gaussian but exhibits a hole in the targeted patch. We conjecture that the hole appears because its value is irrelevant in discriminating between near and far patches, and the RelPosBias is not continuous with respect to the relative coordinate. In addition, the skip connection renders it easier to propagate the value of the targeted patch. Note that although RelPosBias and RelPosMlp produce RPE differently, the resulting RPE is added as attention bias in the same manner. Furthermore, the RPE in the targeted patch shows a lower value in RelPosBias and a higher value in RelPosMlp, and both work effectively with ViTs. Thus, we conjecture that the hole in RelPosBias is not necessary and that it can be replaced with another higher value. Indeed, in our main text, we confirm that Gaussian attention bias is effective when incorporated with Swin-S that uses RelPosBias.