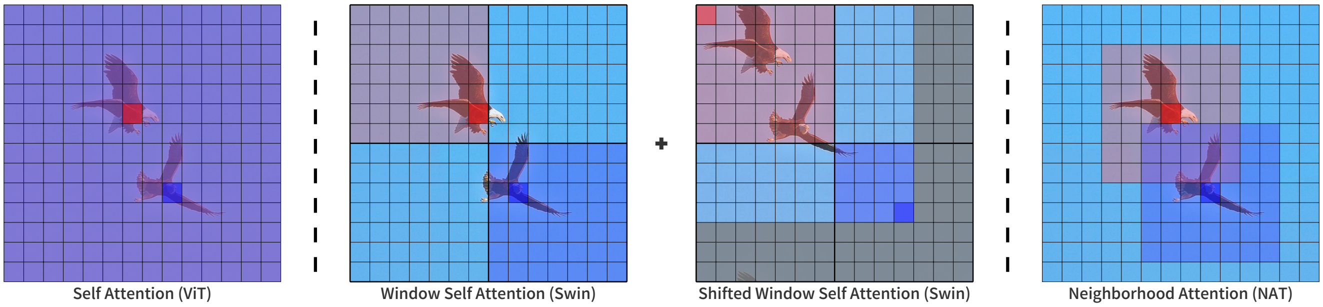
Neighborhood Attention Transformer

Ali Hassani[1,2], Steven Walton[1,2], Jiachen Li[1,2], Shen Li[3], Humphrey Shi[1,2]
[1]SHI Lab @ U of Oregon & UIUC, [2]Picsart AI Research (PAIR), [3]Meta/Facebook AI



Self Attention (ViT) | Window Self Attention (Swin) | Shifted Window Self Attention (Swin) | Neighborhood Attention (NAT)

Figure 1: **An illustration of receptive fields in Self Attention, Shifted Window Self Attention, and our Neighborhood Attention.** Self Attention has the same maximum receptive field for each query token. Shifted Window Self Attention divides inputs into sub-windows and performs self attention, and is followed by another layer with a shift to the input with attention masked for the shifted-out-of-place pixels (gray area). Neighborhood Attention adaptively localizes the receptive field to a neighborhood around each token, introducing local inductive biases without a need for extra operations.

Abstract

*We present **Neighborhood Attention Transformer (NAT)**, an efficient, accurate and scalable hierarchical transformer that works well on both image classification and downstream vision tasks. It is built upon Neighborhood Attention (NA), a simple and flexible attention mechanism that localizes the receptive field for each query to its nearest neighboring pixels. NA is a localization of self-attention, and approaches it as the receptive field size increases. It is also equivalent in FLOPs and memory usage to Swin Transformer's shifted window attention given the same receptive field size, while being less constrained. Furthermore, NA includes local inductive biases, which eliminate the need for extra operations such as pixel shifts. Experimental results on NAT are competitive; NAT-Tiny reaches 83.2% top-1 accuracy on ImageNet with only 4.3 GFLOPs and 28M parameters, 51.4% mAP on MS-COCO and 48.4% mIoU on ADE20k. We will open-source our checkpoints, training script, configurations, and our CUDA kernel at: https://github.com/SHI-Labs/Neighborhood-Attention-Transformer.*

1. Introduction

Convolutional neural networks (CNNs) [17] have been the *de facto* architecture for computer vision models across different applications for years. AlexNet [16] showed their usefulness on ImageNet [7] and many others followed suit, with architectures such as VGG [25], ResNet [13], and more recently, EfficientNet [26]. Transformers [30] on the other hand, were originally proposed as attention-based models for natural language processing (NLP), trying to exploit the sequential structure of language. They were the basis upon which BERT [8] and GPT [23, 24, 1] were built, and they continue to be the state of the art architecture in NLP.

In late 2020, Vision Transformer (ViT) [9] was proposed as an image classifier using only a Transformer Encoder operating on an embedded space of image patches, mostly for large-scale training. A number of other methods followed, attempting to increase data efficiency [27, 35, 10, 11], eventually making such Transformer-like models the state of the art in ImageNet-1k classification (without pre-training on additional large-scale dataset such as JFT-300M). These high-performing attention-based methods are all based on Transformers, which were originally intended for language processing. Self-attention has a linear complexity with respect to the embedding dimension (excluding linear projections), but a quadratic complexity with respect to the number of tokens. In the scope of vision, the number of tokens is typically in linear correlation with image resolution. As a result, higher image resolution results in a quadratic increase in complexity and memory usage in models strictly using self-attention, such as ViT. This has been one of the problems that prevented such models from being easily applicable to downstream vision tasks, such as object detection and semantic segmentation, in which image resolutions are usually much larger than classification. Another problem is that convolutions benefit from inductive biases such as locality, translational equivariance, and 2-dimensional neighborhood structure, while dot-product self attention is a global 1-dimensional operation by definition. While MLP layers in vision transformers are local and translationally equivariant, the rest of those inductive biases have to be learned with large sums of data [9] or advanced training techniques and augmentations [27, 35, 11].

Local attention modules were therefore applied to alleviate this issue. Swin [20] was one of the first hierarchical vision transformers based on local self attention. Its design and the shifted-window self attention allowed it to be easily applicable to downstream tasks, as they made it computationally feasible, while also boosting performance through the additional biases injected. HaloNet [29] also explored a local attention block, and found that a combination of local attention blocks and convolutions resulted in the best performance, due to the best trade-off between memory requirements and translational equivariance.

In this paper, we propose Neighborhood Attention (NA) and build Neighborhood Attention Transformer (NAT) on top of it, which achieves competitive results across vision tasks. NA is a localization of dot-product self attention, limiting each query token's receptive field to a fixed-size neighborhood around its corresponding tokens in the key-value pair. Smaller neighborhoods result in more local attention, while larger ones yield more global attention. This design enables control of receptive fields in a way to balance between translational invariance and equivariance. Our efforts are inspired by the localized nature of convolutions and how they created more local inductive biases that are beneficial to vision tasks. It is different from self attention being applied to local windows (Swin), and can be thought of as a convolution with a content-dependant kernel.

To summarize, our main contributions are:

1. Proposing **Neighborhood Attention (NA)**: A simple and flexible attention mechanism for vision, which localizes the receptive field for each token to its neighborhood. We compare this module in terms of complexity and memory usage to self attention, window self attention, and convolutions.

2. Introducing **Neighborhood Attention Transformer (NAT)**, a new efficient, accurate and scalable hierarchical transformer made of levels of neighborhood attention layers. Each level is followed by a downsampling operation, which reduces spatial size by half. A similar design can be seen in many recent attention-based models, such as Swin [20] and Focal Transformer [20]. Unlike those models, NAT utilizes small-kernel overlapping convolutions for embedding and downsampling, as opposed to non-overlapping ones. NAT also introduces a more efficient set of architecture configurations, compared to prior arts such as Swin.

3. Demonstrating NAT's effectiveness on both image classification and downstream vision tasks including object detection and semantic segmentation. We observe that NAT can outperform not only Swin Transformer, but also new convolutional contenders [21]. Our NAT-Tiny model can reach 83.2% top-1 accuracy on ImageNet with only 4.3 GFLOPs and 28M parameters, and 51.4% bounding box mAP on MS-COCO and 48.4% multi-scale mIoU on ADE20k, which sets a new state-of-the-art for such simple and small-scale transformer models.

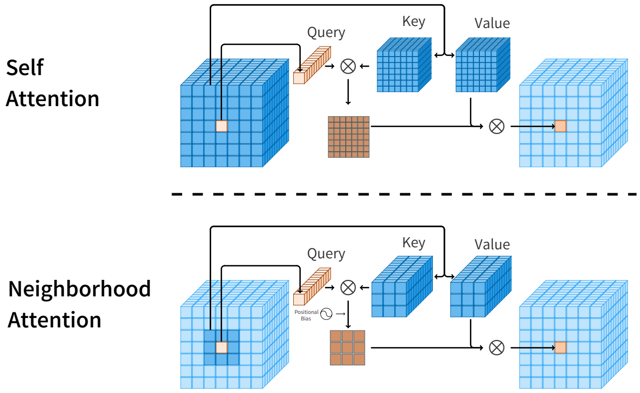


Figure 2: An illustration of the query-key-value structure of Neighborhood Attention vs self attention (for a single pixel). Self attention allows each token to attend all other tokens, while neighborhood attention localizes each token's receptive field to a neighborhood around itself. This is different from the existing window attention mechanism in Swin [20], which divides self attention into subwindows. Both window self attention and neighborhood attention have a linear computational cost and memory usage with respect to resolution, as opposed to self attention's quadratic cost and memory. However, neighborhood attention has the added advantage of limiting each pixel to its neighborhood directly at no extra computational cost, thus not requiring additional shift operations to introduce cross-window interactions. Neighborhood attention is also not limited to working on inputs divisible by the window size like window attention.

2. Related works

In this section, we briefly review the multi-headed self attention (MHSA) mechanism [30], and review some of the notable vision transformers and transformer-like architec-

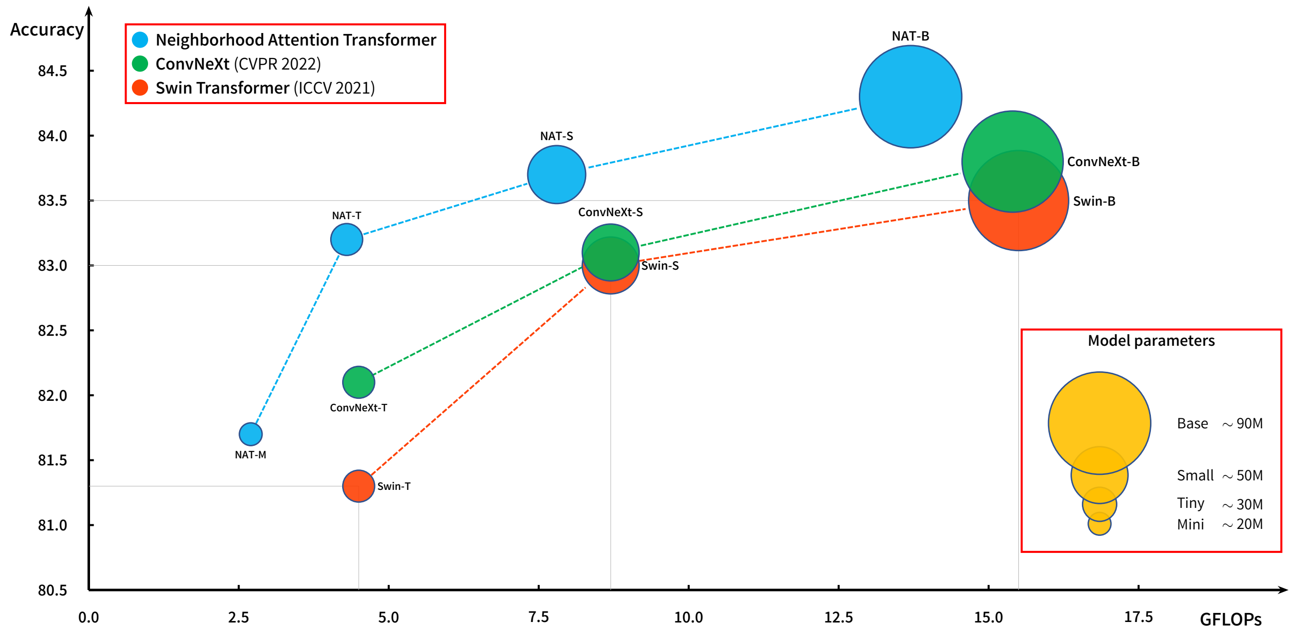


Figure 3: ImageNet-1k classification performance versus compute, with bubble size representing the number of parameters. NAT outperfoms both Swin and ConvNeXt in classification with fewer FLOPs, and a similar number of parameters.

tures [9, 27], as well as some of the notable local attention-based vision transformers [20, 29, 18], and a recent CNN which provides an up-to-date baseline for attention-based models.

2.1. Multi-Headed self attention

Scaled Dot-Product Attention was defined by Vaswani et al. [30] as an operation on a query and a set of key-value pairs. The dot product of query and key is computed and scaled. Softmax is applied to the output in order to normalize attention weights, and is then applied to the values. It can be expressed as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (1)$$

where d_k is the key dimension.

Multi-headed self attention was also proposed by Vaswani et al. [30], which is equivalent to applying the dot-product attention function multiple times with different embeddings, hence forming "attention heads". Self attention by definition takes in an input sequence and uses it as both the query and key-value pairs.

Given an input $X \in \mathbb{R}^{M \times D}$, where M is the number of tokens and D is the embedding dimension, this operation has a complexity of $\mathcal{O}(M^2 D)$ and a space complexity of $\mathcal{O}(M^2)$ for the attention weights.

2.2. Vision Transformer

Dosovitskiy et al. [9] proposed a Transformer-based image classifier that merely consists of a Transformer encoder [30] and an image tokenizer, named **V**ision **T**ransformer. Previous works, such as DETR [3], explored a hybrid of CNNs and Transformer models for detection. ViT on the other hand proposed a model that would only rely on a single non-overlapping convolutional layer (patching and embedding). ViT was pre-trained primarily on the private JFT-300M dataset, and was shown to outperform state-of-the-art CNNs on many benchmarks. However, it was also added that when ViT is pre-trained on medium-scale datasets, such as ImageNet-1k and ImageNet-21k, it no longer achieves competitive results. This was attributed to the lack of inductive biases that are inherent to CNNs, which the authors argued is trumped by large-scale training. While this effectively proved ViT inferior in medium-scale training, it provided empirical evidence that Transformer-based models outperform CNNs in larger scales. ViT paved the way for many more vision transformers, and attention-based models in general, that followed and transferred it to medium-scale learning [27], and even small-scale learning on much smaller datasets [11].

Touvron et al. [27] extended the study of Vision Transformers by exploring data efficiency. Their **D**ata-**e**fficient **i**mage **T**ransformer model performed significantly better than ViT with very few architectural changes, and through the use of advanced augmentations and training techniques.

They explored knowledge transfer [14] through attention by introducing a distillation token, and a hard distillation loss. As for the choice of teacher model, they explored both Transformer-based models, as well as CNNs. It was shown that a convolutional teacher model can improve performance more significantly, as it is arguably transferring inductive biases DeiT lacks. Their experiments highlighted the true potential of a Transformer-based image classifier in the medium-sized data regime, while also inspiring many more to utilize their training techniques.

2.3. Models using local attention

Shifted **Win**dow (**Swin**) Attention [20] was introduced by Liu et al. as one of the first window-based self attention mechanisms. This mechanism partitions input feature maps and applies self attention to each partition separately. This operation has a more favorable complexity, and can be parallelized. These so-called Window Attentions are followed by Shifted Window Attentions, which apply the same operation but with a shift in pixels prior to the window partitioning stage, in order to introduce connections across the extracted windows, while still maintaining the efficiency. Moreover, a relative positional bias is added to the attention weights, as an alternative to the one-time positional embeddings that previous models used. Additionally, the proposed model, **Swin Transformer**, produces pyramid-like feature maps, reducing spatial dimensionality while increasing depth. This structure has been commonly used in CNNs over the years, and is why Swin can be easily integrated with other networks for application to downstream tasks, such as detection and segmentation. Swin outperformed DeiT with a convolutional teacher, at ImageNet-1k classification. Moreover, Swin Transformer is the state of the art method in object detection on the MSCOCO test set, and was until recently the state of the art in semantic segmentation on ADE20K.

Vaswani et al. [29] proposed blocked local attention, which consists of 3 stages: blocking, haloing, and attention. Input feature maps are blocked into non-overlapping subsets, which will serve as queries. Followed by that, neighboring blocks of equal size are extracted (haloing), which will serve as keys and values. The extracted queries and key-value pairs are then sent through self attention. This attention mechanism is built into HaloNet, and is shown to be effective at both reducing cost and improving performance, especially when used in conjunction with more convolutional layers in the network. The authors argued that self attention preserves translational equivariance by definition, and that their local attention improves speed-memory tradeoff by relaxing this equivariance, while not greatly reducing the receptive field.

Yang et al. [34] proposed Focal Attention, a window-based mechanism that limits fine-grained attention to locally surrounding tokens, while applying coarse-grained attention to tokens that are further away. Like Swin, Focal Transformer also utilizes a hierarchical structure that can be seamlessly connected to detection and segmentation heads.

2.4. Recent Convolutional models

Liu et al. [21] recently proposed a new CNN architecture influenced by models such as Swin, dubbed ConvNeXt. These models are not attention-based, and manage to outperform Swin across different vision tasks. This work has since served as a new CNN baseline for fair comparison of convolutional models and attention-based models.

We propose Neighborhood Attention, which by design localizes the receptive field to a window around each query, and therefore would not require additional techniques such as the cyclic shift used by Swin. We introduce a hierarchical transformer-like model with this attention mechanism, dubbed Neighborhood Attention Transformer, and demonstrate its performance compared to Swin on image classification, object detection, and semantic segmentation.

3. Method

In this section, we introduce Neighborhood Attention, a localization of self attention (see Eq. 1) considering the structure of visual data. This not only reduces computational cost compared to self attention, but also introduces local inductive biases, similar to that of convolutions. This operation works with a neighborhood size L, which when at a minimum sets each pixel to attend to only a 1-pixel neighborhood around itself (creating a 3×3 square window). We also show that this definition is equal to self attention when the neighborhood size is at its maximum (i.e. the size of the input). Therefore, if the neighborhood size exceeds or matches the feature map size, neighborhood attention and self attention will have equivalent outputs given the same input. We then introduce our model, **N**eighborhood **A**ttention **T**ransformer (**NAT**), which uses this new mechanism instead of self attention. In addition, NAT utilizes a multi-level hierarchical design, similar to Swin [20], meaning that feature maps are downsampled between levels, as opposed to all at once. Unlike Swin however, NAT uses overlapping convolutions to downsample feature maps, as opposed to non-overlapping (patched) ones. This creates a slight increase in computation and parameters, which we remedy by proposing new configurations that are less computationally expensive.

3.1. Neighborhood Attention

Inspired by how convolutions introduce neighborhood biases and locality, we introduce Neighborhood Attention. This mechanism is designed to allow each pixel in feature maps to only attend to its neighboring pixels. As neighborhood is dependant on a size, so will our new mechanism.

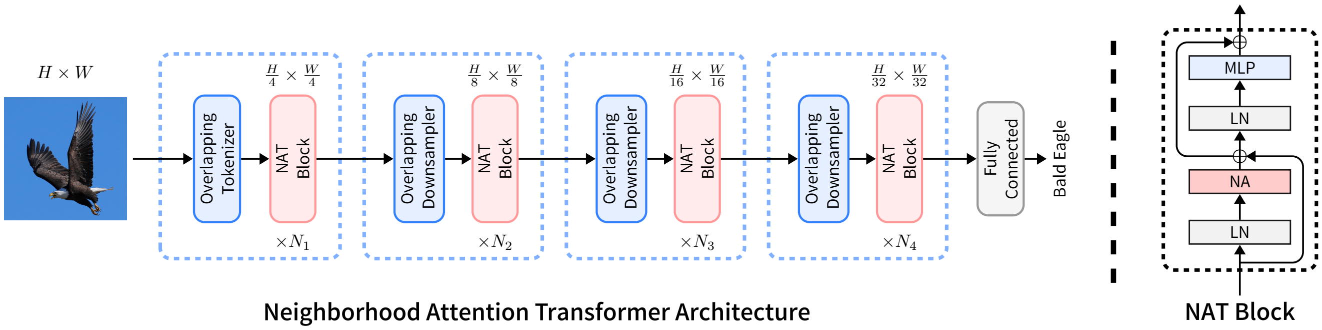


Figure 4: An overview of our model, NAT, with its hierarchical design. The model starts off with a convolutional downsampler, then moves on to 4 sequential levels, each consisting of multiple NAT Blocks, which are transformer-like encoder layers. Each layer is comprised of a multi-headed neighborhood attention (NA), a multi-layered perceptron (MLP), Layer Norm (LN) before each module, and skip connections. Between the levels, feature maps are downsampled to half their spatial size, while their depth is doubled. This eases computation in later layers, and allows for easier transfer to downstream tasks through feature pyramids. As for classification, we simply apply average pooling in the final layer over the height and width.

We denote the neighborhood of a pixel at (i, j) with $\rho(i, j)$, which is a fixed-length set of indices of pixels nearest to (i, j). For a neighborhood of size $L \times L$, $\|\rho(i, j)\| = L^2$. Therefore, neighborhood attention on a single pixel can be defined as follows:

$$\text{NA}(X_{i,j}) = \text{softmax}\left(\frac{Q_{i,j}K^T_{\rho(i,j)} + B_{i,j}}{scale}\right)V_{\rho(i,j)} \quad (2)$$

where Q, K, V are linear projections of X, similar to Eq. 1. $B_{i,j}$ denotes the relative positional bias, which is added to each attention weight based on its relative position. This operation can further be extended to all pixels (i, j), which results in a form of localized attention. However, if the function ρ maps each pixel to **all pixels** (L^2 is equal to feature map size), this will be equivalent to self attention (with the additional positional bias). That is because $\rho(i, j)$ will include all possible pixels when the neighborhood bounds go beyond input size. As a result, $K_{\rho(i,j)} = K$ and $V_{\rho(i,j)} = V$, and by removing the bias term, Eq. 1 is derived. Neighborhood attention is computationally cheap. Its complexity is linear with respect to resolution, unlike self attention's, which is quadratic. Additionally, its complexity is also linear with respect to neighborhood size. ρ, which maps a pixel to a set of neighboring pixels, can be easily produced with a raster-scan sliding window operation, similar to convolutions. Each pixel is simply mapped to a set of neighboring pixels and itself. An illustration of this operation is presented in Figure 2. This operation is repeated for every pixel in the feature map. As for corner pixels that cannot be centered, the neighborhood is expanded to maintain receptive field size. This is a key design choice, which allows NA to generalize to self attention as the neighborhood size grows towards the feature map resolution. Expanded neighborhood is achieved by simply continuing to pick the L^2 nearest neighboring pixels to the original. For example, for $L = 3$, each query will end up with 9 key-value pixels surrounding it (a 3×3 grid with the query positioned in the center). For a corner pixel, the neighborhood is another 3×3 grid, but with the query **not positioned in the center**. An illustration of this idea is presented in Figure 6.

3.2. Complexity analysis

Neighborhood Attention has the same number of FLOPs as window-based self attention mechanisms, such as Swin [20]. We present a complexity analysis in this subsection and discuss their memory usages. We also present a complexity analysis comparing neighborhood attention to convolutions, showing that it involves fewer operations, but uses more memory. For simplicity, we exclude attention heads, and work with single-headed attention. Note that we denote neighborhood and window sizes as L in order to avoid confusion with other notations.

Given input feature maps of shape $H \times W \times C$, where C is the number of channels, H and W are feature map height and width respectively, the QKV linear projections will have a cost of $\mathcal{O}\left(3HWC^2\right)$, which is the same for self-attention, Swin's window self attention, and neighborhood attention.

Swin divides the queries, keys, and values into $\frac{H}{L} \times \frac{W}{L}$ windows of shape $L \times L \times C$, then applies self attention on each window, which would cost $\mathcal{O}(\frac{H}{L}\frac{W}{L}CL^4)$, which is simplified into $\mathcal{O}\left(HWCL^2\right)$. Memory consumption of attention weights, which are of shape $\frac{H}{L} \times \frac{W}{L} \times L^2 \times L^2$, is therefore $\mathcal{O}\left(HWCL^2\right)$.

In NA, each query token $Q_{i,j}$ will have keys $K_{\rho(i,j)}$, and values $V_{\rho(i,j)}$, both of size $L \times L \times C$, which means the cost to compute attention weights is $\mathcal{O}\left(HWCL^2\right)$, similar to Swin. The attention weights will also be in the shape of

Module	Computation	Memory
Self attention	$\mathcal{O}\left(3HWC^2 + 2H^2W^2C\right)$	$\mathcal{O}\left(3HWC + H^2W^2\right)$
2D Window attention (Swin)	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Neighborhood attention	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Convolution	$\mathcal{O}\left(HWC^2L^2\right)$	$\mathcal{O}\left(HWC\right)$

Table 1: Computational cost and memory consumption comparison between self attention, patched self attention, and neighborhood attention, and convolution, with respect to input sizes. We compare to convolutions, as neighborhood attention can be thought of as a convolution with a dynamic kernel.

$H \times W \times L^2$, as each of the $H \times W$ queries attends to L^2 keys. Therefore, the memory required to store attention weights would also be the same as Swin, $\mathcal{O}\left(HWCL^2\right)$.

As for convolutions, computational cost is $\mathcal{O}\left(HWC^2L^2\right)$, and the memory usage would be only $\mathcal{O}\left(HWC\right)$ to store the output. While convolutions are more memory-efficient, their complexity is quadratic with respect to channels and linear with respect to window size $(L \times L)$. By solving the inequality between the complexities of a 2D convolution and a 2D neighborhood attention for $L > 1$, it can be concluded that the latter grows less quickly than the former as the number of channels is increased. Specifically for $L = 3$, NA is more efficient for all $C > 3$, which in practice is usually the case. For $L \geq 5$, NA is more efficient for all $C > 1$. Therefore, it can be concluded that 2D NA is less computationally complex than a 2D convolution in practical scenarios, while only suffering from the additional memory usage due to the QKV projections.

3.3. Implementation

No operations exist in major deep learning libraries that exactly replicate neighborhood attention, or even extract the neighboring pixels. There is a combination of operations that can create the neighborhoods, but they would be highly inefficient and memory-consuming. We therefore wrote custom CUDA kernels for different components of neighborhood attention. We present details on different implementations in Appendix A.

3.4. Neighborhood Attention Transformer

NAT embeds inputs using 2 consecutive 3×3 convolutions with 2×2 strides, resulting in a spatial size $1/4$th the size of the input. This is similar to using a patch and embedding layer with 4×4 patches, but it utilizes overlapping convolutions instead of non-overlapping ones. On the other hand, using overlapping convolutions would increase cost, and two convolutions incurs more parameters. However, we handle that by reconfiguring the model, which results in a better trade-off.

NAT consists of 4 levels, each followed by a **downsampler** (except the last). Downsamplers decrease cut spatial size in half, while doubling the number of channels. We use 3×3 convolutions with 2×2 strides, instead of 2×2 non-overlapping convolutions that Swin uses (patch merge). Since the tokenizer downsamples by a factor of 4, our model produces feature maps of sizes $\frac{H}{4} \times \frac{W}{4}$, $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{16} \times \frac{W}{16}$, and $\frac{H}{32} \times \frac{W}{32}$. This change is motivated by previous successful CNN structures, and it allows for easier transfer of pre-trained models to downstream tasks. It was also motivated by the recent success of other hierarchical attention-based methods, such as PVT [31], ViL [39], Swin Transformer [20] and Focal Transformer [34]. Additionally, we utilize LayerScale [28] for stability when training larger models. An illustration of the overall network architecture is presented in Figure 4. We present a summary of different NAT variants and their key differences in Table 2.

4. Experiments

We demonstrate NAT's applicability and effectiveness by conducting experiments across different vision tasks, such as image classification, object detection, and semantic segmentation. We also conducted ablations of Neighborhood Attention and NAT with Swin as the baseline.

4.1. Classification

We trained our variants on ImageNet-1k [7] in order to compare to other transformer-based and convolutional image classifiers. This dataset continues to be one of the few benchmarks for medium-scale image classification, containing roughly 1.28M training, 50K validation, and 100K test images, categorized into 1000 classes. We train our model using the commonly used `timm` package [32], and use the common augmentations and training techniques used in similar works[27, 28, 15, 35, 20, 11, 10], such as CutMix [37], Mixup [38], RandAugment [6], and Random Erasing [40]. We follow Swin's [20] exact training configuration (learning rate, iteration-wise cosine schedule, and other hyperparameters). Following convention, we train the model for 300 epochs, 20 of which warm up the learning rate, while the rest decay according to the scheduler, and finally do 10 additional cooldown epochs [27].

Variant	Layers	Dim × Heads	MLP ratio	# Params	FLOPs
NAT-Mini	3, 4, 6, 5	32 × 2	3	20 M	2.7 G
NAT-Tiny	3, 4, 18, 5	32 × 2	3	28 M	4.3 G
NAT-Small	3, 4, 18, 5	32 × 3	2	51 M	7.8 G
NAT-Base	3, 4, 18, 5	32 × 4	2	90 M	13.7 G

Table 2: A summary of NAT Configurations. All models use 7 × 7 attention neighborhoods. Dimensions and heads double after every level until the final level.

Model	Top-1	# Params	FLOPs
DeiT-S [27]	79.9%	22 M	4.6 G
NAT-Mini	81.8%	20 M	2.7 G
Swin-Tiny [20]	81.3%	28 M	4.5 G
ConvNeXt-Tiny [21]	82.1%	28 M	4.5 G
NAT-Tiny	83.2%	28 M	4.3 G
Swin-Small [20]	83.0%	50 M	8.7 G
ConvNeXt-Small [21]	83.1%	50 M	8.7 G
NAT-Small	83.7%	51 M	7.8 G
Swin-Base [20]	83.5%	88 M	15.4 G
ConvNeXt-Base [21]	83.8%	89 M	15.4 G
NAT-Base	84.3%	90 M	13.7 G

Table 3: ImageNet Top-1 validation accuracy comparison at 224×224 resolution (no extra data or pretraining).

4.2. Object Detection

We trained Mask R-CNN [12] and Cascade Mask R-CNN [2] on MS-COCO [19], with NAT backbones, which were pre-trained on ImageNet. We followed Swin [20]'s training settings closely, using `mmdetection` [4], and trained with the same accelerated 3× LR schedule. The results are presented in tables 4 and 5.

NAT-Mini outperforms Swin-Tiny with Mask R-CNN, while falling slightly short to it with Cascade Mask R-CNN, all while having significantly fewer FLOPs. NAT-Tiny outperforms both its Swin and ConvNeXt counterparts, again with slightly fewer FLOPs, with both Mask and Cascade Mask R-CNN. NAT-Small and NAT-Base can reach similar-level performance with both detectors compared to their Swin and ConvNeXt counterparts, while being noticeably more efficient.

4.3. Semantic Segmentation

We also trained UPerNet [33] on ADE20K [41], with ImageNet-pretrained NAT backbones. Similar to detection, we followed Swin's configuration for training ADE20k, and used `mmsegmentation` [5]. Additionally, and following standard practice, input images are randomly resized and cropped at 512 × 512 when training. Segmentation results on ADE20K are presented in Table 6. It is noticeable that NAT-Mini outperforms Swin-Tiny, and also comes very close to ConvNeXt-Tiny. NAT-Tiny outperforms ConvNeXt-Tiny significantly, while also slightly more efficient. NAT-Small outperforms Swin-Small on single-scale performance, while matching the multi-scale mIoU. NAT-Base similarly performs on-par with Swin-Base, while falling slightly short of ConvNeXt-Base. It should be noted that both NAT-Small and NAT-Base bear fewer FLOPs with them compared to their Swin and ConvNeXt counterparts, while their performance is within the same region. It is also noteworthy that Swin especially suffers from more FLOPs even beyond the original difference due to the fact that the image resolution input in this task specifically (512 × 512) will not result in feature maps that are divisible by 7 × 7, Swin's window size, which forces the model to pad input feature maps with zeros to resolve that issue, prior to every attention operation. NAT does not require this, as feature maps of any size are compatible.

4.4. Ablation study

In order to evaluate the effects of each different component compared to Swin Transformer [20], we conducted ablations on the attention module, the downsamplers, and the configurations. Attention ablations are presented in Table 7. Swin's window self attention is accompanied by a cyclic shift, which introduces out-of-window interactions. Because neighborhood attention computes dynamic key-value pairs for every query, it would not need any shift in pixels to operate. The model configurations used in this study follow Swin-Tiny, with 96 channels spread across 3 heads in the first level, doubling after every level, and 2 layers per level, except the third level, which has 6 layers. Our second ablation is on the model itself, where we start with Swin-Tiny, and replace the patched tokenizer and downsamplers with overlapping convolutions. This change only involves using two 3 × 3 convolutions with 2 × 2 strides, instead of one 4 × 4 convolution with 4 × 4 strides, along with changing the downsamplers from 2×2 convolutions with 2×2 strides to 3 × 3 convolutions with the same stride. As it can be noticed, model performance jumps up by just under 0.5% in accuracy, while the number of parameters and FLOPs also see a noticeable increase. We then switch to our Tiny con-

Backbone	**# Params**	**FLOPs**	AP^{b}	$\text{AP}^{\text{b}}_{50}$	$\text{AP}^{\text{b}}_{75}$	AP^{m}	$\text{AP}^{\text{m}}_{50}$	$\text{AP}^{\text{m}}_{75}$
NAT-Mini	40 M	225 G	46.5	68.1	51.3	41.7	65.2	44.7
Swin-Tiny [20]	48 M	267 G	46.0	68.1	50.3	41.6	65.1	44.9
ConvNeXt-Tiny [21]	48 M	262 G	46.2	67.0	50.8	41.7	65.0	44.9
NAT-Tiny	48 M	258 G	47.7	69.0	52.6	42.6	66.1	45.9
Swin-Small [20]	69 M	359 G	48.5	70.2	53.5	43.3	67.3	46.6
NAT-Small	70 M	330 G	48.4	69.8	53.2	43.2	66.9	46.5

Table 4: Object detection and instance segmentation performance with Mask R-CNN on MS-COCO.

Backbone	**# Params**	**FLOPs**	AP^{b}	$\text{AP}^{\text{b}}_{50}$	$\text{AP}^{\text{b}}_{75}$	AP^{m}	$\text{AP}^{\text{m}}_{50}$	$\text{AP}^{\text{m}}_{75}$
NAT-Mini	77 M	704 G	50.3	68.9	54.9	43.6	66.4	47.2
Swin-Tiny [20]	86 M	745 G	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-Tiny [21]	86 M	741 G	50.4	69.1	54.8	43.7	66.5	47.3
NAT-Tiny	85 M	737 G	51.4	70.0	55.9	44.5	67.6	47.9
Swin-Small [20]	107 M	838 G	51.9	70.7	56.3	45.0	68.2	48.8
ConvNeXt-Small [21]	108 M	827 G	51.9	70.8	56.5	45.0	68.4	49.1
NAT-Small	108 M	809 G	52.0	70.4	56.3	44.9	68.1	48.6
Swin-Base [20]	145 M	982 G	51.9	70.5	56.4	45.0	68.1	48.9
ConvNeXt-Base [21]	146 M	964 G	52.7	71.3	57.2	45.6	68.9	49.5
NAT-Base	147 M	931 G	52.3	70.9	56.9	45.1	68.3	49.1

Table 5: Object detection and instance segmentation performance with Cascade Mask R-CNN on MS-COCO.

Backbone	**# Params**	**FLOPs**	**mIoU**	**mIoU(ms)**
ResNet101 [13]	47 M	-	38.8	-
DeiT-S [27, 20]	52 M	1094 G	44.0	-
NAT-Mini	50 M	900 G	45.1	46.4
Swin-Tiny [20]	60 M	946 G	44.5	45.8
ConvNeXt-T [21]	60 M	939 G	46.0	46.7
NAT-Tiny	58 M	934 G	47.1	48.4
Swin-Small [20]	81 M	1040 G	47.6	49.5
ConvNeXt-Small [21]	82 M	1027 G	48.7	49.6
NAT-Small	82 M	1010 G	48.0	49.5
Swin-Base [20]	121 M	1188 G	48.1	49.7
ConvNeXt-Base [21]	122 M	1170 G	49.1	49.9
NAT-Base	123 M	1137 G	48.5	49.7

Table 6: Semantic segmentation performance on ADE20k.

figuration, with fewer heads, smaller inverted bottlenecks, and more layers. This results in an improvement of over 0.9% in accuracy, while reducing the number of parameters and FLOPs to below the baseline. We finally switch the attention mechanism with NA, and see an improvement of just under 0.5% in accuracy. This, along with the study conducted just on attention mechanisms, suggests that not only is NA more powerful under similar conditions, it also is affected more significantly by introducing more inductive biases into the model, through overlapping convolutions.

4.5. Saliency analysis

In an effort to further illustrate the differences between attention mechanisms and models, we present salient maps from ViT-Base, Swin-Base, and NAT-Base. We selected a few images from the ImageNet validation set, sent them through the three models, and created the salient maps based on the outputs, which are presented in Table 5. All images are correctly predicted (Bald Eagle, Acoustic Guitar, Hummingbird, Steam Locomotive) except ViT's Acoustic Guitar which predicts Stage. From these salient maps we can see that all models have relatively good interpretability, though they focus on slightly different areas. NAT appears to be slightly better at edge detection, which we believe is due to the localized attention mechanism, that we have presented in this work, as well as the convolutional tokenizer and downsamplers.

Attention	Positional info.	Top-1	# Params	FLOPs
Shifted Window Self Attention	None	80.1%	28.26 M	4.51 G
Neighborhood Attention	None	80.6%	28.26 M	4.51 G
Window Self Attention	Relative Pos. Bias.	80.2%	28.28 M	4.51 G
Shifted Window Self Attention	Relative Pos. Bias.	81.3%	28.28 M	4.51 G
Neighborhood Attention	Relative Pos. Bias.	81.4%	28.28 M	4.51 G

Table 7: Ablation study on Neighborhood Attention vs Window Self-Attention from Swin. Both methods can work without positional information, but do not perform as well. Shift is only applicable to Swin, as neighborhood attention computes dynamic key-value pairs for every query. Swin results are directly reported from the original paper.

Model	Attention	Tokenizer	Downsampler	Layers	Heads	MLP Ratio	Top-1	# Params	FLOPs
Swin-T	SWSA	Patch	Patch	2,2,6,2	3	4	81.29%	28.3 M	4.5 G
Overlapping Swin-T	SWSA	Conv	Conv	2,2,6,2	3	4	81.78%	30.3 M	4.9 G
NAT-like Overlapping Swin-T	SWSA	Conv	Conv	3,4,18,5	2	3	82.72%	27.9 M	4.3 G
NAT-T	NA	Conv	Conv	3,4,18,5	2	3	83.20%	27.9 M	4.3 G

Table 8: Ablation study on NAT, with Swin-T as the baseline. Through the use of overlapping convolutions, and our NAT design boosts, classification accuracy is significantly while resulting in fewer parameters and FLOPS than Swin-T. Swapping SWSA with NA results in an improvement of almost 0.5% in accuracy.

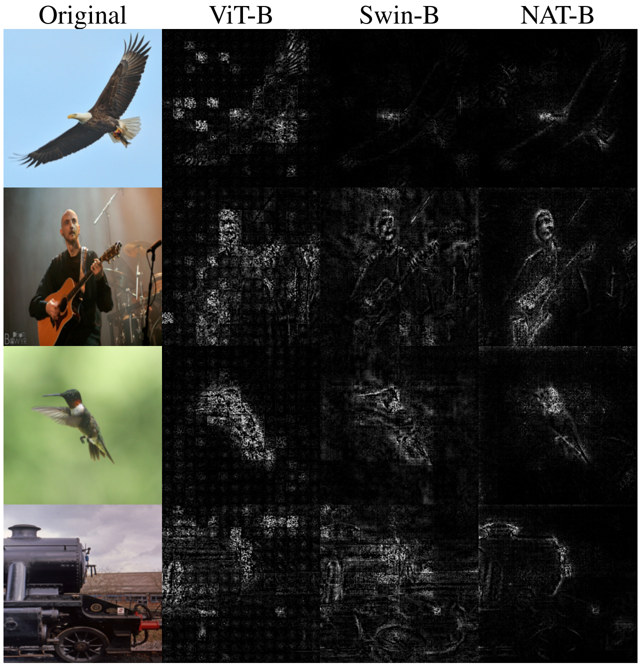


Figure 5: Salient Maps of selected ImageNet validation set images, comparing ViT-Base, Swin-Base, and NAT-Base. The ground truths for these images are: Bald Eagle, Acoustic Guitar, Hummingbird, and Steam Locomotive, respectively.

5. Conclusion

Transformer-based vision models have received significant attention from the research community in computer vision, especially since the introduction of ViT [9]. Some works focused on data efficiency, with minor changes in the architecture [27, 35, 11, 28, 10], while others focused on efficiency and transferrability to downstream tasks [20, 36, 29, 10]. In this paper, we introduced an alternate way of localizing self attention with respect to the structure of data, which computes key-value pairs dynamically for each token, along with a more data efficient configuration of models. This helps create a model that utilizes both the power of attention, as well as the efficiency and inductive biases of convolutions. We've shown the power of such a model in image classification, in which it outperforms both Swin Transformer and ConvNeXt significantly. Additionally, we've shown that it can be seamlessly applied to downstream tasks, where it also outperforms or competes with those existing methods. We will be conducting experiments on even larger NAT variants, as well as ImageNet-21k pretraining in future versions.

Acknowledgement: We thank Picsart AI Research (PAIR), Meta/Facebook AI, and IARPA for their generous support to make this work possible.

References

[1] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakan-

tan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 7

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

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

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 6

[7] 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 conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 6

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3, 9

[10] Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021. 1, 6, 9

[11] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021. 1, 2, 3, 6, 9

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 8

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 4

[15] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *arXiv preprint arXiv:2104.10858*, 2021. 6

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[17] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1

[18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 7

[20] 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. *arXiv preprint arXiv:2103.14030*, 2021. 2, 3, 4, 5, 6, 7, 8, 9

[21] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 4, 7, 8

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 11

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. 1

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[26] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 1

[27] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through at-

tention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 2, 3, 6, 7, 8, 9

[28] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 6, 9

[29] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 2, 3, 4, 9

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017. 1, 2, 3

[31] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 6

[32] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[33] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 7

[34] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 4, 6

[35] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 1, 2, 6, 9

[36] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*, 2021. 9

[37] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 6

[38] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6

[39] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021. 6

[40] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 6

[41] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 7

A. Implementation details

In order to implement Neighborhood Attention, we initially started with a version based on existing operations in PyTorch [22]. Excluding corner pixels (those that cannot have an $L \times L$ window around surrounding them), the rest can be extracted using any raster-scan operator. Typically, any such operator with a stride of 1 will yield the desired neighborhoods for pixels that have $\frac{L-1}{2}$ pixels on each of their 4 sides (as discussed, we constrain L to be an odd number greater than 1). In PyTorch, the `unfold` function is the most straightforward solution. As for the corner pixels, simply repeating some of those extracted windows will yield their expected neighborhoods, as their neighborhood consists of the same pixels. This repetition can be done easily with PyTorch's replicated padding, which pads a tensor by repeating edge cells. The combination of these two operations (`unfold` then `replicated_pad`) on the key-value pair tensors ($H \times W \times C$) will yield two tensors of shape $H \times W \times C \times L \times L$ (one $L \times L$ for each channel and pixel). Neighborhood Attention can be computed by plugging these into (2) in the place of K and V. An upside of this is that the matrix multiplications can be batched and parallelized, just like Self Attention and Window Self Attention.

However, this implementation is highly inefficient, because it needs to temporarily store the extracted windows, and make 2 separate CUDA kernel calls on the very large tensors, just to replicate Neighborhood Attention. Additionally, it is not very flexible and makes the addition of relative positional bias very complicated.

In order to resolve this issue, we wrote custom CUDA kernels for the QK and AV operations separately (leaving operations such as softmax and linear projections to the original kernels for the sake of efficiency). The kernels were written as PyTorch extensions, and therefore can be integrated seamlessly with PyTorch modules. The QK kernel can additionally take in the relative positional biases as a tensor, and apply them *as the attention weights are being computed*, which reduces the number of threads and makes it more efficient. We observed that with these kernels, training a Tiny variant similar to Swin-T takes about 25% the time as the original implementation, and consumes about 15% the amount of memory with 64 samples on a single GPU. We've also verified that its memory usage matches a Swin model with the same configuration, just as they match theoretically. That said, the kernel is still in early stages of development and would require much additional optimiza-

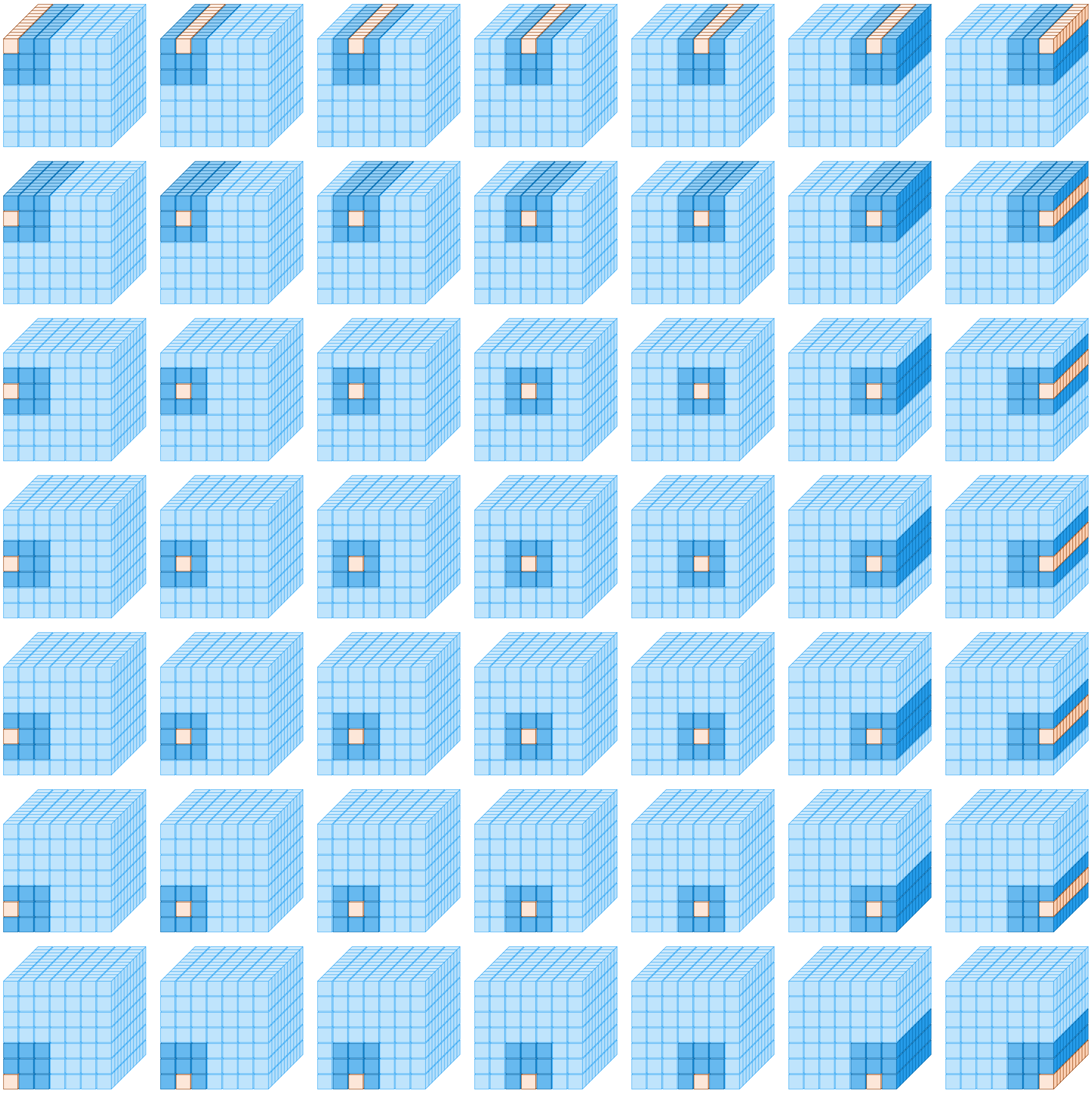


Figure 6: The neighborhood size is expanded at the corners to keep the receptive field size identical to the rest of the feature map. The alternative to this would have been smaller neighborhoods (zero padding at the corners). This choice over zero padding was primarily due to its properties (equivalence to self attention when neighborhood and feature map sizes match), but also due to stronger performance. Additionally, the expanded neighborhood keeps the receptive field the same for all tokens, which is a better design choice from the attention point of view.

tion to reach its optimal performance. Theoretically, the speed of an optimal kernel should be very close to Swin, as the two methods have the same number of floating point operations, and memory usage. We will release an early version of the kernel on our GitHub repository, which not only allows the community to use our method, but also opens up the possibility of contributions from engineers who could optimize our kernel further.