Swin on Axes: Extending Swin Transformers to Quadtree Image Representations

Marc Oliu[1,2] mosi@create.aau.dk
Kamal Nasrollahi[1,2] kn@create.aau.dk
Sergio Escalera[1,3] sescalera@ub.edu
Thomas B. Moeslund[1] tbm@create.aau.dk

Aalborg Universitet[1]
Fredrik Bajers Vej 7K, Aalborg Øst

Universitat de Barcelona and Computer Vision Center[3]
Gran Via de les Corts Catalanes 585, Barcelona
Campus UAB, Edifici O, Cerdanyola del Vallès

Milestone Systems[2]
Banemarksvej 50, Brøndby

Abstract

In recent years, Transformer models have revolutionized machine learning. While this has resulted in impressive results in the field of Natural Language Processing, Computer Vision quickly stumbled upon computation and memory problems due to the high resolution and dimensionality of the input data. This is particularly true for video, where the number of tokens increases cubically relative to the frame and temporal resolutions. A first approach to solve this was Vision Transformers, which introduce a partitioning of the input into embedded grid cells, lowering the effective resolution. More recently, Swin Transformers introduced a hierarchical scheme that brought the concepts of pooling and locality to transformers in exchange for much lower computational and memory costs. This work proposes a reformulation of the latter that views Swin Transformers as regular Transformers applied over a quadtree representation of the input, intrinsically providing a wider range of design choices for the attentional mechanism. Compared to similar approaches such as Swin and MaxViT, our method works on the full range of scales while using a single attentional mechanism, allowing us to simultaneously take into account both dense short range and sparse long range dependencies with low computational overhead and without introducing additional sequential operations, thus making full use of GPU parallelism.

1. Introduction

Swin Transformers [9] are a type of neural topology based on transformer networks [17]. Contrary to conventional transformers and their Computer Vision counterpart, Vision Transformers (ViT) [4], Swin employs a hierarchical partitioning of the image in order to define local regions upon which the attention mechanism is applied. The image resolution is successively halved after a specific number of transformer blocks, which increases reach of the attention mechanism without actually increasing the size of the individual attention matrices. The main problem with such an approach is that long range dependencies are taken into consideration in an indirect fashion. It is only thanks to its window shifting mechanism that, through the use of multiple Swin Transformer blocks, information can slowly diffuse to distant regions of the image.

In this work we propose a reformulation of the method, exposing it as a sequential computation over an quadtree representation of the input images. We then propose a new flavor of Swin transformers that naturally emerges from said reformulation, which we refer to as Swin on Axes (Swinax). Compared to the first, Swinax can jointly consider token relationships at multiple scales, the dilation factor of the attentional mechanism increasing along with said scale. It can thus achieve the same degree of sparsity and computation of long range dependencies Swin achieves over multiple layers, but it does so in a single transformer block, with the computations being performed in parallel.

2. Previous work

Transformers [17] were first introduced in 2017 as an attentional mechanism that allows for the passing of information between a collection of tokens $X = \{x^{(1)}; ..; x^{(p)}\}$ in language models. The attentional mechanism in question extracts a series of keys $K_X = \left[f_K(x^{(1)}); ..; f_K(x^{(p)}))\right]$, queries $Q_X = \left[f_Q(x^{(1)}); ..; f_Q(x^{(p)}))\right]$ and values $V_X = \left[f_V(x^{(1)}); ..; f_V(x^{(p)}))\right]$, then proceeding to apply the attentional mechanism shown below for a given attention head j:

$$Att^{(j)}(X) = Smx\left(\frac{1}{\sqrt{d_K}} K_X^{(j)} {Q_X^{(j)}}^T + B^{(j)}\right) V_X^{(j)} \quad (1)$$

Here, d_K is the dimensionality of keys $K_X^{(j)}$, $B^{(j)}$ corresponds to the relative position biases and $Smx(\cdot)$ is the

Softmax activation function. A given attentional layer has h attention heads, with the final output of the layer being a linear combination of the latter:

$$Y = \left[Att^{(1)}\left(\cdot\right),\ ..,\ Att^{(h)}\left(\cdot\right)\right] W_O \tag{2}$$

Where W_O is the output weights matrix. Apart from the attentional mechanism, a transformer block is typically followed by a two-layer fully connected network updating the token representations, with layer normalization after both the multi-head attention and fully connected sub-network. The original work proposed an encoder-decoder architecture, where a series of transformer blocks are stacked to produce an encoder processing an input token sequence, and similar set of such layers produces a decoder. Contrary to the encoder, the decoder consists of two attention mechanisms: the first one is the regular self-attention block, where the attentional mechanism looks at other tokens within the same sequence. The second one updates the token representations of the decoder based on those of the encoder in what is commonly known as cross-attention.

Latter variations introduced encoder-only [3] and decoder-only [11] models. The former model either the relationship between elements in a sequence by reconstructing masked input tokens, or directly predict a target label based on the input sequence. The latter, on the other hand, are trained in an auto-regressive fashion by masking the self-attention mechanism, ensuring that each token will have access only to itself and the previous tokens. tt The first significant success when applying transformer models to computer vision were Vision Transformers (ViT) [4]. Here, the authors proposed an encoder-only topology where an input image is broken down into non-overlapping cells, typically using a 16×16 grid. Each of the cells is linearly projected into a feature vector representing a token, with an additional classification token added to the sequence. This token sequence is then fed to the encoder-only transformer, with the final output of the classification token being used as overall model output.

While this type of model proposes a robust tokenization approach, it still runs into the problem of a high memory usage in the order of $O\left(p^2\right)$, where $p = 16^2$ is the number of tokens, and the corresponding high computational cost required to generate the attention matrix. While this would still correspond to a relatively small attentional matrix when compared to some language models, a large batch size is usually required in order to stabilize the training of computer vision models. A common way to solve this problem is to apply restrictions on the considered tokens when computing the attentional mechanism.

Local attention enforces locality during the computation of the attentional mechanism, with Swin transformers [9] being the most well known example. In Swin, the grid of tokens is partitioned into non-overlapping $W \times W$ windows, where typically $W = 7$. The attentional mechanism is applied to each window, with the window being displaced by 50% of its width at alternating layers. This allows the model to limit the size of the attentional matrices while still allowing the tokens falling near the edges to diffuse throughout the image. This reduces the memory usage from $O\left(p^2\right)$ to $O\left(pW^2\right)$. Swin also introduces a linear down-sampling function where, after a set of d transformer blocks, neighboring tokens in a non-overlapping grid of 2×2 cells are linearly combined together. This further reduces the memory and computational costs by lowering the number of tokens fed to subsequent transformer blocks, while also indirectly increasing the receptive field of the attention mechanism.

Neighborhood Attention (NA) [7] considers a local token neighborhood for each of the target tokens during attention computation, resulting in a model equivalent to the marking of all elements outside the first k diagonals in the attention matrix. In order to efficiently implement it, the authors provide low level kernels to perform the attention computation, bypassing the otherwise prohibitive memory and computational cost restrictions that would result from implementing is as a series of linear algebra operators. The advantage of this approach is its intrinsic ability to diffuse information across the whole image, due to its lack of hard region boundaries between image segments. Similarly to Swin, a linear down-sampling step is introduced after every few attentional blocks.

A more recent work proposes Dilated Neighborhood Attention (DiNA) [6], introducing dilation to NA. Similar to the approach it is based on, this is achieved through a low level CUDA kernel implementation. The dilation factor then becomes a per layer model hyper-parameter.

Sparse attention aims to combine sparse global attention patterns with denser local ones by introducing strategies that reduce the memory and computational costs. Sparse Transformers [2] was the first work to introduce such an approach. The authors noted that ViT first learned locality patterns, factorizing across the vertical and horizontal axes on latter layers. Based on that, they implemented hand crafted factorizations of the attentional mechanism, weaving these insights into the architecture.

Longformers [1] use a combination of sliding window attention, optionally with dilation, and global attention. They do so by considering a select subset of tokens as globally connected, serving as shortcuts for the diffusion of information across the token sequence. Routing Transformers [12] use a learned attention connectivity instead. To do so, a series of k centroids are learned and used during k-means clustering to group the queries and keys of the attention mechanism. This restricts the attention matrix to mappings between queries and keys belonging to the same cluster. Note that nothing prevents a given token from having its query and key belong to different clusters, potentially giving the attention matrix a single component graph connectivity.

MaxViT [16], an approach based on Swin transformers, proposes decomposing a token grid $X \in \mathbb{R}^{<H \times W \times C>}$ into a batched form $X \in \mathbb{R}^{<\frac{H}{N} \times \frac{W}{N} \times N \times N \times C>}$. In this form, applying an attention mechanism over axes $N \times N$ corresponds to the local windowed attention commonly seen in Swin. Applying it over axes $\frac{H}{N} \times \frac{W}{N}$ instead corresponds to a dilated attention over the whole picture, where the dilation factor is N. By applying two successive attention blocks, one over the local windows and the other dilated over the whole input, the approach considers both dense local and sparse global attention.

QuadTree [15] uses a quad-tree based attentional mechanism, where keys, queries and values are iteratively downsampled to create a pyramid of features at different resolutions. The attentional matrix is first computed on the coarsest level and used to decide the top-K tokens to consider when computing the attentional matrix for the next, finer level of the pyramid. Messages are passed from all levels of the pyramid to eack key token. While this approach offers great flexibility both in terms of granularity and distribution of the selected tokens, this is achieved through a large increase in the number of sequential operations, reducing model parallelism.

Our work falls on the family of sparse attention methods, proposing an attentional mechanism where the attention pattern gradually sparsifies as the distance between token pairs increases. It shares some similarities with MaxViT [16], but using a more general, principled approach that lends it a higher degree of flexibility.

3. Method

Our aim is to provide a computationally efficient attentional mechanism capable of adapting to the image resolution by automatically adjusting the sparsity based on the distance between tokens. This is particularly important for tasks requiring both a detailed local understanding of image elements (eg. object recognition), while also requiring a more general understanding (eg. image composition, instance interaction). As seen in Sec. 2, previous approaches rely either on a manual selection of shortcut tokens [1, 12], a two-stage local/global attentional mechanism [16] or the iterative, sequential refinement of the attention matrix [15]. These approaches either do not scale well with the image resolution, introduce additional sequential operations that hamper parallelism, or both.

To tackle these issues, we propose a reformulation of Swin transformers, where the segmentation of the input into local attentional regions and linear pooling of tokens are seen as regular attention and dimensionality reduction operations applied to a quadtree representation of the input image. As we show in this section, this allows for an equivalent, more straightforward application of the approach, as well as allowing for other attentional operations such as attention dilation (Sec. 3.3) and multi-scale attention (Sec. 3.4). An implementation of the approach is made publicly available. [1]

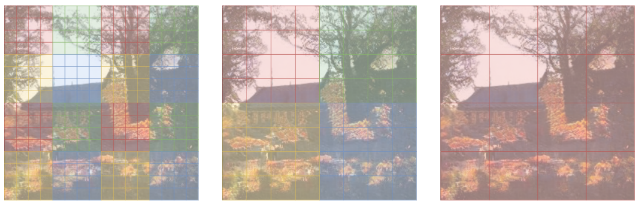
Figure 1. Attention matrices of a Swin transformer at different resolutions. Each grid cell represents a token, while colored regions correspond to the attention regions. The number of attention regions is reduced as cell grids are linearly combined during pooling. The size of attention regions remains constant.

3.1. Hierarchical structure of Swin

Swin defines a window of constant size $W \times W$ which is applied over the input token grid in a tiled, non-overlapping fashion, as shown in Fig. 1. This window is used to decide which cells of the grid will interact with each other during the computation of the attention mechanism, creating a local connectivity pattern within a given transformer block.

After successively applying a series of such layers, the resolution of the grid is reduced to half its original size by linearly projecting the features in each 2×2 non-overlapping neighborhood into a single output feature vector. This reduces the overall number of cells by 75%. Further transformer blocks are applied after that. While these maintain the same window size and thus the number of cells considered and size of the attention matrix, the receptive field effectively doubles in size due to the down-scaling of the grid.

3.2. Quadtree representation

An intuitive way of representing the data input to a Swin transformer block is to define $X \in \mathbb{R}^{<B \times N_W \times W \times W \times C>}$, where B is the batch size, N_W is the number of non-overlapping attention neighborhoods, $W \times W$ represents the spatial resolution of said neighborhoods and C corresponds to the feature length of a token. During pooling, we would need to transpose and reshape the matrix to $\mathbb{R}^{<B \times N_W/4 \times W \times W \times 4C>}$ before applying the linear transform to each token.

Here we propose a different approach where the input data is structured as a quadtree [5], a hierarchical structure that is perfectly suited to our purposes. Similarly to its higher dimensional variant, the octree [10], it splits a two-dimensional space into four quadrants, all elements being

[1]Code: Will be made publicly available on publication.

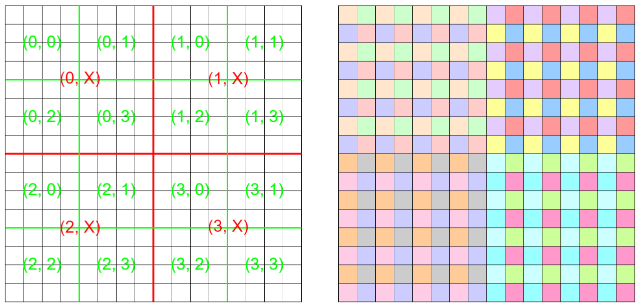

Figure 2. **Left:** Coordinates of cells and subcells for the quadtree representation of the input cell grid. **Right:** Checkerboard pattern of the dilated attention mechanism, obtained by shifting the selected axes for the attention window once to the left ($x2$ dilation). Each color denotes the group of tokens included in a single attention window, for a total of 16 windows.

distributed among them based on their spatial coordinates. Each of the quadrants is further subdivided, the process iteratively repeating itself until we either have a sufficiently small number of elements on the leaf partitions, or a single element remains. For highly regular lattice structures such as images, the distribution of elements among quadrants is homogeneous. This allows us to encode the structure as a high dimensional tensor whose dimensions consist of four elements, each corresponding to a quadrant. To do so, we first expand the height and width dimensions of the input images into a series of size 2 axes, then reshuffle and reshape:

$$\begin{aligned} \mathbb{R}^{<B\times H\times W\times C>} &\xrightarrow{reshap.} \mathbb{R}^{<B\times h_1\times..\times h_n\times w_1\times..\times w_n\times C>} \\ &\xrightarrow{transp.} \mathbb{R}^{<B\times h_1\times w_1\times..\times h_n\times w_n\times C>} \\ &\xrightarrow{reshap.} \mathbb{R}^{<B\times a_1\times..\times a_n\times C>} \end{aligned} \tag{3}$$

For $2d$ images, this results in a series of axes $\{a_1, .., a_n\}$ of size 4, where a_i results from vectorizing dimensions $h_i \times w_i$. Note that the above transform can just as easily be applied to three-dimensional data, such as video or volumes. There, each dimension a_i would be of size 8, the resulting tensor encoding an octree. Under this representation, axis a_i partitions an input $2d$ image into four equally sized chunks corresponding to the four quadrants of a quadtree, while the successive dimensions each partition the previous quadrants into four more segments. This is illustrated by the leftmost image in Fig. 2.

Under this representation, the attention computation and linear down-sampling performed by Swin become straightforward. Below, an example is shown for a cell grid of size 64 ($d = 6$ total axes), where the attention window is of size $W = 8$ ($w = 3$ axes).

$$X = \mathbb{R}^{<B\times a_1\times a_2\times a_3\times {\color{blue}a_4\times a_5\times a_6}\times {\color{red}C}>} \tag{4}$$

Here, the dimensions colored in blue correspond to the attention window, while those colored in red are the features. The other dimensions correspond to the batch size. After applying k transformer blocks on the data, Swin performs a linear transform on non-overlapping neighborhoods of size 2x2, resulting in a reduction of the image size. In our representation, this corresponds to shifting the selected axes for the attention window to the left:

$$X = \mathbb{R}^{<B\times a_1\times a_2\times {\color{blue}a_3\times a_4\times a_5}\times {\color{red}a_6\times C}>} \tag{5}$$

The feature dimensions, marked in red, can then be linearly transformed and replaced by a single output feature dimension.

3.3. Dilated attention

A direct extension to Swin comes from considering what would happen if we shifted the selection of attention window axes without subsuming the rightmost axis into the feature representation. That is, if we consider the following assignment of axes:

$$X = \mathbb{R}^{<B\times a_1\times a_2\times {\color{blue}a_3\times a_4\times a_5}\times a_6\times {\color{red}C}>} \tag{6}$$

Here, B and axes a_1, a_2 and a_6 jointly define the batch size, while axes a_3, a_4, and a_5 define the attention window. Applying a transformer block over the above representation is equivalent to doubling the size of the attention window while maintaining the number of tokens, which is achieved through an implicit $2x$ dilation of the attention regions. This is illustrated by the rightmost image in Fig. 2. Displacing the selection of window axes further to the left would equate to a $4x$ dilation of the attention mechanism.

With this approach, we can compute larger attention regions without having to sub-sample the image nor increase the computational and memory costs. This allows us to consider both local and global attention before sub-sampling the grid cells, increasing the flexibility of Swin. The shifting of the attention windows, as performed on Swin at alternate attention blocks, also becomes unnecessary. This is due to subsequent dilated layers already allowing communication across attentional window boundaries by increasing the receptive field.

3.4. Multi-scale attention

We can further exploit the quadtree representation by applying the attention mechanism to multiple axes simultaneously. The simplest approach is to consider each axis independently, resulting in d attention matrices of size 4×4. The resulting attention mechanism shares information more densely among tokens that are closed in the quadtree representation, with the dilation factor increasing for tokens that are further away. This is shown in the middle diagram of Fig. 3. A more general option is to apply the same approach

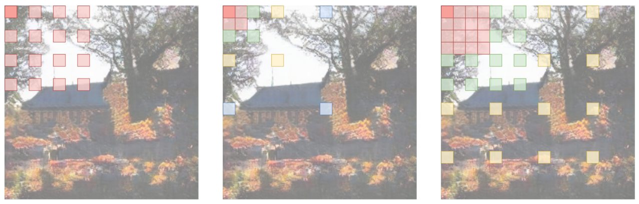

Figure 3. Attentional neighborhood of a token. Each color represents a different attention scale. **Left:** Simple attention with an x2 dilation factor. **Middle:** Multi-scale attention applied over each axis of the quadtree representation. **Right:** Multi-scale attention applied over the quadtree axes using a sliding window of size 2.

over a sliding window moving across the axes, resulting in larger individual attention matrices. The former approach would then correspond to using a sliding window of size 1. For a sliding window of size 2, we would use each element of the set $A = \{(a_1, a_2),\ (a_2, a_3),\ ..,\ (a_{n-1}, a_n)\}$ as the attention window. This is illustrated on the rightmost diagram of Fig. 3.

The keys $K_X^{(j)}$, queries $Q_X^{(j)}$ and values $V_X^{(j)}$ are computed once for all tokens and shared among the different attention windows. A single set of relative biases is also learned using the same approach as in [9]. The final value for any given attention head is obtained by multiplying the attention tensors $\widetilde{A}^{(j,k)}$ with the values tensor $V_X^{(j)}$, then adding the resulting matrices together. This corresponds to independently applying the attentional mechanism over multiple dilation factors, instead of applying it to multiple scales separately. We jointly compute the Softmax activation over the different attention tensors:

$$\begin{aligned}\overline{A}^{(j,k)} &= \frac{e^{\hat{A}^{(j,k)}}}{Z^{(j)}} \\ Z^{(j)}_{abcd} &= \mathbb{1}_{abcd} \otimes \left(\sum_{i=1}^{d-w+1} e^{\hat{A}^{(j,i)}} \right)_{abcduv}\end{aligned} \tag{7}$$

Where d is the dimensionality of the quadtree representation, $\hat{A}^{(j,k)}$ are the the attention matrices before applying Softmax and $\otimes$ denotes the Einstein summation operator, with the subscripts for both its terms corresponding to the pairings between axes. The resulting algorithm for the computation of a multi-scale attention head j is shown in Alg. 1.

4. Complexity

Lets define p as the number of tokens, typically $p = 2^{2d}$ for an image size restricted to being a power of 2 in both height and width, and W as the attentional window size. The total number of scales N_S over which the attentional mechanism is applied is given by:

$$N_S = 1 + log_2 \left(\frac{\sqrt{p}}{W} \right) \tag{8}$$

The number of tokens each one is paired with is then given by $N_P = W^2 N_S$. Note that each scale past the first has a 25% overlap in terms of tokens relative to the previous (smaller) one. When efficiently implementing the attentional mechanism through a custom kernel, we can discard these redundant tokens from the computation. This results in the following number of pairings per token:

$$N_T = W^2 \left(1 + \frac{3}{4} log_2 \left(\frac{\sqrt{p}}{W} \right) \right) \tag{9}$$

The proposed attentional mechanism is slightly more costly when compared to Swin. Whereas MSA incurs a quadratic cost relative to the number of tokens p, and the Windowed MSA introduced by Swin incurs a linear cost, our proposed Multi-Scale MSA has a slightly above linear cost:

$$\begin{aligned}\Omega(\text{MSA}) &= 4pC^2 + 2p^2C \\ \Omega(\text{W-MSA}) &= 4pC^2 + 2W^2pC \\ \Omega(\text{MS-MSA}) &= 4pC^2 + 2W^2 \left(1 + \frac{3}{4} \log_2 \frac{\sqrt{p}}{W} \right) pC\end{aligned} \tag{10}$$

Here C is the feature length of a token. Of note is that while the cost is higher than Swin, the window size W can be smaller for our model and still obtain comparable performance, the parameter acting as a trade-off between local dense and global sparse attention. As discussed in the experimental section, our approach would in fact be slightly less computationally expensive than Swin when using the same architecture and input sizes.

5. Experiments

In order to evaluate our approach, we have chosen Swin as a baseline, replacing the attentional mechanism by our sparse multi-scale formulation. Two datasets are chosen, evaluating our performance on both a localized task like object recognition, as well as another where our approach is expected to show significant improvements due to the long range relationships displayed on the dataset. We have chosen Scene Recognition for such a purpose.

5.1. Architecture

We have based our model on the Swin Tiny (Swin-T) [9] architecture, using the same layers, attentional heads, and feature size. This corresponds to 12 attentional layers, each consisting of an attentional mechanism followed by a fully connected network with a single hidden layer. These are grouped in a sequence of $2-2-6-2$, with a 2×2 linear pooling doubling the number of features per token in-between each attentional block. Each block also doubles the number of attentional heads, with the first one consisting of 3 heads per layer. The input image is divided into

Algorithm 1 Multi-scale attention mechanism

```
**Require:** $K_X^{(j)},\ Q_X^{(j)},\ V_X^{(j)}$
**Require:** $A = \{(a_1, a_2),\ ..,\ (a_{n-1}, a_n)\}$
  **for** $k,\ (u,\ v) \in enum(A)$ **do**
      $\hat{A}^{(j,k)}_{..uv..\hat{u}\hat{v}} \leftarrow (K_X^{(j)})_{..uv..z} \otimes (Q_X^{(j)})_{..\hat{u}\hat{v}..z}$
      $\hat{A}^{(j,k)} \leftarrow \frac{1}{\sqrt{d_K}} \hat{A}^{(j,k)} + B^{(j,k)}$
  **end for**
  $\left[Z^{(j,1)},\ ..,\ Z^{(j,k)}\right] \leftarrow Smx\left(\left[\hat{A}^{(j,1)},\ ..,\ \hat{A}^{(j,k)}\right]\right)$
  $Att^{(j)} \leftarrow \mathbf{0}$
  **for** $(u,\ v) \in A$ **do**
      $T_{..uv..z} \leftarrow Att^{(j)} + A^{(j,k)}_{..uv..\hat{u}\hat{v}} \otimes (V_X^{(j)})_{..\hat{u}\hat{v}..z}$
      $Att^{(j)} \leftarrow Att^{(j)} + T$
  **end for**
```

tokens of size 4×4, which are then linearly projected into feature vectors of size 96. GELU activations are used for the fully connected layers, as well as layer normalization both after the attentional mechanism itself and after the fully connected layers, with residual connections in both cases. An average pooling is performed on the output of the last attentional block, followed by a Softmax classification layer.

The modifications to said architecture are two-fold. Firstly, a new layer is added at the very beginning, before tokenization, which brings the input image into a quad-tree representation as per Eq. (3). Secondly, the windowed attention is removed, along with the window shifting, and replaced by our multi-scale attention mechanism, as explained in Sec. 3.4. We use a sliding window of size 2, resulting in 4×4 attentional windows on each scale.

5.2. Datasets

We have chosen two datasets for our experiments, one for validating the accuracy of our approach on tasks that do not depend on sparse, long range relationships, and another one that does. This allows us to evaluate the main contribution of our approach, with the first dataset serving as a baseline to determine whether said contribution negatively impacts its applicability to other domains. For both datasets, we follow the same data augmentation approach as [9] uses for Imagenet-1K.

Imagenet-1K is an object recognition dataset consisting of 1000 classes. It is split into 1.28M train, 50K validation and 100K test images, with the labels for the latter not being publicly available. Similarly to [9], we train our model on training partition and report our results on the validation set. This task requires the recognition of a single object located somewhere within the image, a task that can be solved through the extraction of successively more complex features in a localized fashion. As such, it serves as a baseline to determine whether our multi-scale approach negatively impacts the performance of the model on datasets where long range sparse attention is not required.

Places365-Standard is a scene recognition dataset consisting of 365 classes, displaying indoors as well as both urban and nature outdoors scenes. It is split into 1.8M train, 36.5K validation and 328.5K test images, with the labels for the latter not being publicly available. We follow the same approach as with Imagenet-1K, training on the training partition and reporting our results on the validation set. For many scene categories, the category a given image belongs to depends on both the presence of a variety of elements spread across the scene as well as their overall composition, making it a good evaluation benchmark for our approach. This is the case, instance, when distinguishing between an art gallery and museum, a bookstore and library, or the different types of gardens.

5.3. Results

From the results in Tab. 1, we first note the number of parameters and computational cost of our model. Using the same architecture as Swin-T, our attentional mechanism does not add to the number of parameters, keeping our approach one of the smallest in the State-of-the-Art. While there is a 22% increase in the computational cost, this still leaves the approach as one of the fastest, surpassed only by the baseline. This is despite the input image being 30% larger in terms of area, our method thus being more efficient than the baseline when taking into account the input size.

This improvement in computational efficiency is explained by the size of the attention window. While Swin applies a 7×7 window at a single scale, our approach uses 4×4 windows applied to all scales of the quad-tree representation. For Swin, this means each token pairs with 49 others. In the case of Swinax, the number of pairs varies depending on the token grid resolution: For the first two layers, with a grid size of 64×64, there are 5 possible scales, resulting in each token being paired with 64 others. This number quickly drops due to the pooling mechanism: The next two pair each token with 52 others, the following six only with 40, and 28 on the last two. Since the number of attentional heads and features per token are doubled after each pooling, the computational cost is substantially reduced as we progress further along the architecture.

As shown in Tab. 1, the accuracy on Imagenet-1K experiences a slight drop of 0.1% when compared to Swin-T. We attribute this to two main reasons. Firstly, the dataset does not depend much on long range relationships: Being an object recognition dataset, a classical approach based on iterative local feature extraction and pooling, as is the case with Swin, suffices in order to correctly categorize an image. Secondly, the same contributions allowing for long range dependencies also imply a lower local token density: Only a 4×4 dense token region is considered, the sparsity exponentially increasing as the attention window moves through

Model	Pre-training	Resolution		Accuracy		FLOPs	Params
		Pre.	Fine.	IN-1k	Places	(G)	(M)
Direct Training							
iSQRT-Conv [18]	-	-	224	77.9%	56.32%	6.3	56.9
WaveMix [8]	-	-	256	–	56.45%	–	45.0
WaveMix [8]	-	-	224	74.9%	-	–	28.9
Swin-T [9]	-	-	224	81.3%	57.20%	4.5	29.0
Swinax (ours)	-	-	256	81.2%	**58.41**%	5.5	29.0
Supervised pre-training							
Swinax (ours)	IN-1k	256	256	81.2%	58.7%	5.5	29.0
ViT-B [4]	IN-1k	224	224	82.3%	57.9%	18.0	86.9
ViT-L [4]	IN-1k	224	224	82.6%	59.4%	62.0	307.0
Hiera-B [13]	IN-1k	224	224	84.5%	58.9%	9.0	52.0
Hiera-L [13]	IN-1k	224	224	86.1%	**59.6**%	40.0	214.0
EffNet-B6 [19]	IN-1k	528	528	84.8%	58.5%	19.5	43.0
EffNet-B7 [19]	IN-1k	600	600	85.2%	58.7%	38.4	66.3
EffNet-B8 [19]	IN-1k	672	672	85.5%	58.6%	63.7	87.4
ViT-B [4]	IN-21k	224	384	84.0%	58.2%	55.6	86.9
ViT-L [4]	IN-21k	224	384	85.2%	59.0%	191.5	307.0
EffNet-B6 [20]	JFT 300M	528	528	86.4%	58.8%	19.5	43.0
EffNet-B7 [20]	JFT 300M	600	600	86.9%	59.2%	38.4	66.3
EffNet-L2 [20]	JFT 300M	475	475	**88.2**%	59.4%	172.6	480.3

Table 1. Comparison of different State-of-the-Art approaches on Places365-Standard (Places), both for directly trained models as well as models pre-trained on other datasets. Results on Imagenet-1K (IN-1k) are provided for comparison against the Swin-T baseline. **Pre:** Image resolution during pre-training. **Fine:** Image resolution during fine-tuning.

the various scales. This can be counter-productive when the sparser long range information obtained in exchange does not contribute much to solving the task, especially when the input image resolution is slightly larger (256×256 for Swinax, as opposed to 224×224 for Swin), making the effective dense window even smaller.

One could ask instead why the drop in accuracy is so small despite that. This is because it is still possible for the model to consider a larger window of dense tokens, albeit in a roundabout way. Information can be passed sparsely on an 8×8 window, each token receiving information on only 25% of the tokens, but with the set of tokens in any of the quadrants jointly encoding information on all of them. The next layer can then propagate said information among all tokens within the quadrant at a denser scale. The reverse is also true: Information can be propagated densely first, and at a sparser scale afterwards. So long as there is more than one attentional layer between poolings, larger dense regions can be considered through compositing of said layers. This is not unlike how the receptive field of a convolutional network increases with the number of layers, but with the increase being exponential instead of linear. We consider the results obtained on Imagenet-1K to show that, despite the method being designed with sparse long range dependencies in mind, it still performs competitively otherwise.

For Places365-Standard, our approach obtains outstanding results, as shown in Tab. 1. We surpass the State-of-the-Art for direct supervised training by almost 2%, going from the 56.45% accuracy of the previous best reported approach to 58.41% for our method, and doing so with both a smaller model and lower computational cost. When compared to the Swin-T baseline, we obtain a 1.21% increase in accuracy. Furthermore, our accuracy is comparable to that of other models trained using supervised pre-training on other datasets, even surpassing ViT-B, despite not performing any pre-training ourselves. If we consider that said pre-trained methods all consider higher input resolutions, bigger models and higher computational costs, it becomes clear that our method is much better at scene recognition: We require both less data and computational resources to perform comparably. A third family of methods utilizing weakly supervised pre-training exists [14], achieving accuracies in the range of 59-61% on Places365. That said, those are trained on massive datasets with billions of images.

When pre-training on Imagenet1K, the accuracy further increases to 58.7%, surpassing or matching a majority of models despite the much smaller number of trainable parameters and computational cost. From among the models pre-trained on Imagenet1K, only Hiera-B/L and ViT-L obtain better accuracies, the smallest of which having 80% more parameters.

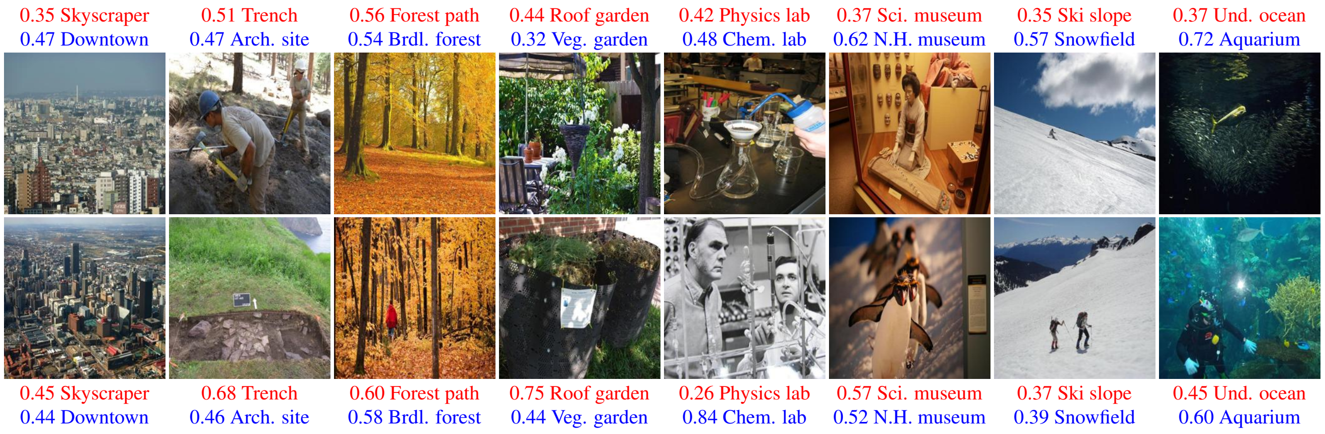

Figure 4. Sample predictions on the Places365 validation set for both Swin-T and Swinax illustrating common classification errors successfully addressed by our approach. Samples are randomly drawn from among the off-diagonal entries of the confusion matrix where Swinax reports the steepest decline in misclassification.

5.4. Qualitative analysis

In 4 we show common mistakes made by Swin-T on the places365 dataset that are successfully addressed by our approach. The downtown, archaeological site and snowfield categories, in particular, clearly display the main advantage of our approach: the improved ability to consider a sparse, global context. For the Downtown category, Swin-T tends to misclassify images as belonging to the skyscraper category due to the presence of multiple such buildings. Swinax, on the other hand, considers the general context, taking into account the large variety of buildings present. For archaeological sites, Swin seems to be have trouble picking up sparse, individually irrelevant clues necessary to distinguish a dig site from a regular trench, such as marker flags, delimiting strings and spot markers. This is also the case when distinguishing between a tilted snowfield and an actual ski slope, the difference being on the boundary and circuit markers.

When distinguishing between the underwater ocean and an aquarium, Swin seems to focus on the presence of a wide array of elements, such as schools of fish, rocks, underwater flora and divers, but fails on the more subtle clues generally spread across the full image, such as glass reflections and a view of the water surface. Similarly, distinguishing between roof gardens and vegetable gardens can be a difficult task, many of the elements being shared between both classes. The clues to disambiguate between both are often large and spread across the image, such as the presence of background walls and floor types indicating the image does not belong to a rooftop.

6. Conclusions and future work

We have introduced Swinax, a new multi-scale attentional mechanism based on a quad-tree representation of the input images, obtaining State-of-the-Art results for scene recognition while being computationally efficient. Our approach has shown that, despite being designed for datasets displaying sparse long range dependencies, it remains competitive for other tasks not requiring it, thus serving as a multi-purpose machine learning algorithm. Compared to similar approaches such as Swin, we have shown our method to be computationally more efficient given an equivalent input size, and to allow for a scale-agnostic pipeline, breaking the relationship between the sizes of the input image and attentional window.

It would be interesting to study the applicability of Swinax to tasks such as semantic segmentation, instance segmentation and depth estimation. Said tasks all involve long range relationships between elements, while also depending on dense local features in order to generate high quality output maps. Swinax is ideally suited for such tasks, its multi-scale attentional mechanism allowing for both dense local and sparse long range attention. It also allows for compact, computationally efficient attention while still maintaining a high resolution token grid. Another potential improvement to the approach is factorizing the input images into factors other than 2, allowing for more flexibility regarding the input image sizes. While there is nothing in principle preventing said extension, this would result in uneven dilation factors at different scales, the impact of which requires further experiments to determine.

7. Acknowledgements

This work is supported by Milestone Systems' industrial post grant 1045-00036B, co-financed by The Innovation Fund of Denmark, by the Spanish project PID2022-136436NB-I00 and by ICREA under the ICREA Academia programme.

References

[1] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 2, 3

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 2

[3] 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. 2

[4] 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. In *Int. Conf. Learn. Represent.*, 2021. 1, 2, 7

[5] Raphael A Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4:1–9, 1974. 3

[6] Ali Hassani and Humphrey Shi. Dilated neighborhood attention transformer. *arXiv preprint arXiv:2209.15001*, 2022. 2

[7] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6185–6194, 2023. 2

[8] Pranav Jeevan, Kavitha Viswanathan, and Amit Sethi. Wavemix-lite: A resource-efficient neural network for image analysis. *arXiv preprint arXiv:2205.14375*, 2022. 7

[9] 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 *Int. Conf. Comput. Vis.*, pages 10012–10022, 2021. 1, 2, 5, 6, 7

[10] Donald JR Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Rensselaer Polytechnic Institute Image Processing Laboratory, 1980. 3

[11] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018. 2

[12] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *ACL*, 9:53–68, 2021. 2, 3

[13] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. *arXiv preprint arXiv:2306.00989*, 2023. 7

[14] Mannat Singh, Laura Gustafson, Aaron Adcock, Vinicius de Freitas Reis, Bugra Gedik, Raj Prateek Kosaraju, Dhruv Mahajan, Ross Girshick, Piotr Dollár, and Laurens Van Der Maaten. Revisiting weakly supervised pre-training of visual perception models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 804–814, 2022. 7

[15] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. *arXiv preprint arXiv:2201.02767*, 2022. 3

[16] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *Eur. Conf. Comput. Vis.*, pages 459–479. Springer, 2022. 3

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Adv. Neural Inform. Process. Syst.*, 30, 2017. 1

[18] Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li. Deep cnns meet global covariance pooling: Better representation and generalization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(8):2582–2597, 2020. 7

[19] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 819–828, 2020. 7

[20] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10687–10698, 2020. 7