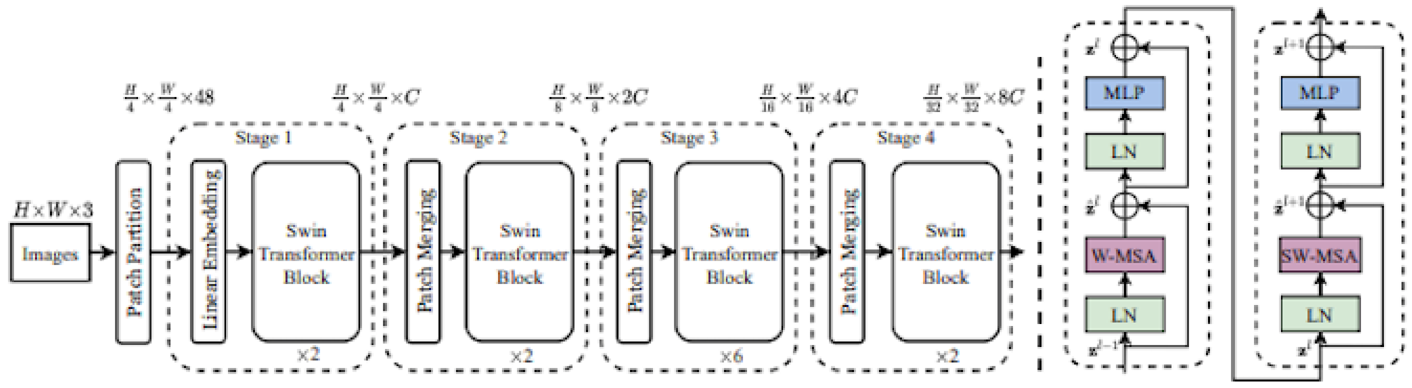
BOLSTER

Blog / Research Labs
Research

SWIN Transformers: The Best of Two Worlds

January 23, 2024 | 6 min read

Adithya Singh



The search for a transformer based backbone model suitable to handle vision tasks has been going on for a while now. There have been multiple transformers proposed to solve different challenges in adopting SWIN transformers for computer vision.

A critical challenge that impedes the performance of transformers for vision tasks is the difference in variation of scale in images and text data. **Images as of now are acquired in high resolution**, which makes variation of scale in different visual entities even more challenging.

Convolutional Neural Networks (CNNs) have been the go-to neural networks to solve problems in computer vision. CNNs offer the learning through visual kernels which learn the underlying feature maps to find the feature patterns using the convolutional operations.

CNNs solve the problem of scale variation using different techniques like downsampling through max-pool layers and feature pyramids. The feature pyramids allow the network to process input images at different scales. This allows for detection of smaller level features like small scale entities in object detection or pixel level features for image segmentation tasks.

When it comes to NLP tasks, the variation in different words of a input sequence is far less than the vision tasks. For example in english language, the largest word is composed of 45 letters and smallest word is 1 letter.

In computer vision, the variation between objects can be up to 1000 pixels. Clearly for transformers to be effective in solving the computer vision tasks efficiently, it is important for them to be able to handle scales. The transformers however, have not been invented on similar theoretical base. Hence, it is where the model for SWIN transformer comes in to form a bridge between the two theoretical bases.

Motivation Behind Shifted Window-Based Transformer

Transformers have been able to change the landscape of machine learning in past years. For NLP tasks, Transformers are now the default family of models to accomplish tasks from text classification, sentiment analysis to even generating answers when prompted with questions.

The advantage of transformers have been their ability to contextualize very long sequences of texts, with the help of a mechanism called self-attention. The idea is to process the text as an input sequence and divide it into smaller tokens of texts.

Every single token in this sequence is then compared to every other token to find the relationships among them. The model learns to associate appropriate weights to all the tokens in the sequence that contextually correspond to current token the most. This is where transformers are really good at capturing global contexts.

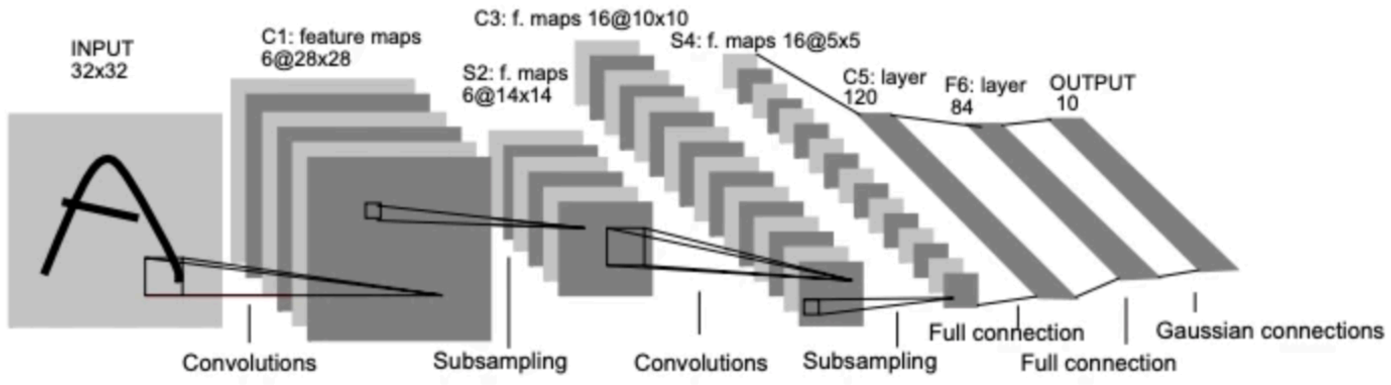


Fig 1. Convolutional Neural network – LeNet

For computer vision tasks, CNNs have been the state of the art model. They have the capability to capture deep visual patterns imperceptible even to the human eye; these patterns known as features are aggregated by CNNs to solve vision related tasks.

CNNs rely on convolution kernels as the feature extractor. These kernels process a local patch of an image at a time as can be seen in figure 1. This tendency of the patches to focus only on a small receptive field out of the entire image at a time, limits it’s capability to gather the understanding of the global context in images.

Transformers on the other hand are great at gathering global contexts. Transformers also have a much larger learning capacity than traditional CNNs. Due to these capabilities, researchers have been trying to explore methods to bridge the gap between transformers and vision tasks. The Vision Transformer is the pioneer work towards achieving this goal.

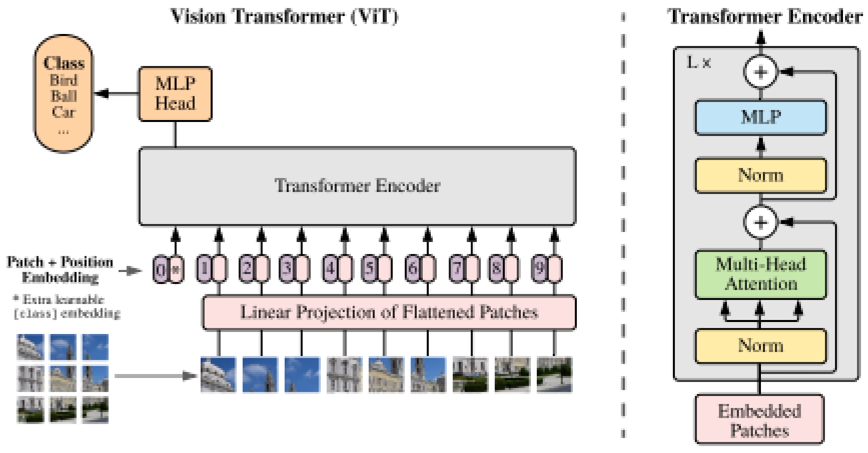


Fig 2. Vision Transformer

The vision transformer processes an image by dividing input image into smaller patches as seen in Figure 2. These patches are treated the same way as tokens in a text sequence and are called visual tokens.

The patches are fed sequentially to the vision transformer model and the weights are adjusted by finding the relationship between the patches. Since each patch is compared to every other patch to find the relevant contexts, the processing time complexity of vision transformer increases quadratically with image dimensions. This makes it unscalable to process images of high resolution. Furthermore, image patches are only computed at fixed scale whereas visual entities in an image are highly variable in scale. This makes it difficult to train vision transformers on tasks dealing with small visual entities like object detection and pixel level patterns like image segmentation.

SWIN Transformers addresses both these challenges by proposing hierarchical processing of image tokens using shifted window patches with linear complexity to image size. The SWIN transformer processes input image tokens at multiple levels while keeping the number of patches generated same.

Layer Architecture

The key contribution of SWIN transformer is the shifted window mechanism. The solution is two pronged as it solves both the challenges, it helps the model to extract features at variable scales and also restricts the computational complexity with respect to image size to linear. The shifted windows mechanism divides the image into windows, each window is further divided into patches.

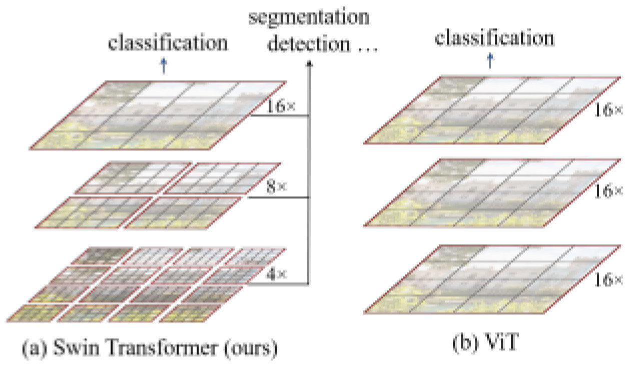


Fig 3. Hierarchical feature representation by SWIN vs fixed scale feature representation by ViT model

Each window block is divided into these patches and fed to model in same way the vision transformer processes the entire input image. The self attention block of the transformer computes the key-query weight for these patches within these windows.

This helps the model emphasize on smaller scale features, however this raises another challenge, since the relationship between the patches are computed within the windows self attention mechanism is unable to capture the global context which is a key feature in transformers. The problem of linear computational complexity also remains unaddressed.

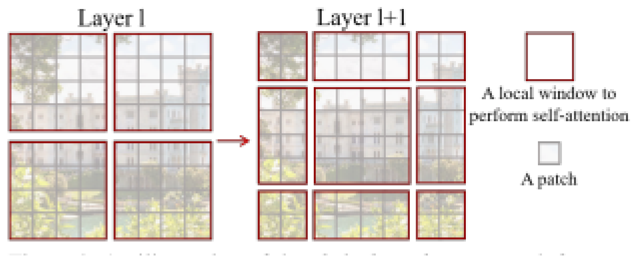


Fig 4. Window partitioning in successive layers of SWIN

In **first layer**, the image is divided into windows denoted in the figure by red boxes. Each window is further divided into patches denoted by gray boxes.

In **second layer**, this window is shifted as depicted and these windows are overlapping with the windows divided in previous layer. As it can be noted, the number of patches remain fixed, which ensures that the computational complexity remains linear with image size.

Moreover the overlapping nature of the windows ensures that the patches which were not paid attention to in previous layer due to being in different window, can be emphasized on in current layer. This ensures that global context is captured through these connections bridged from the previous layer.

First the input RGB image is split into non-overlapping windows, each patch is treated as a token and converted to raw pixel values as its feature set. The patch size is 4×4, hence the feature set dimensions become 4x4x3=48. A linear embedding layer converts the patch into embedding token of dimension C. These tokens are fed through several transformer blocks, these blocks maintain the number of patches to be H/4 x W/4. This set of transformer blocks is called stage 1.

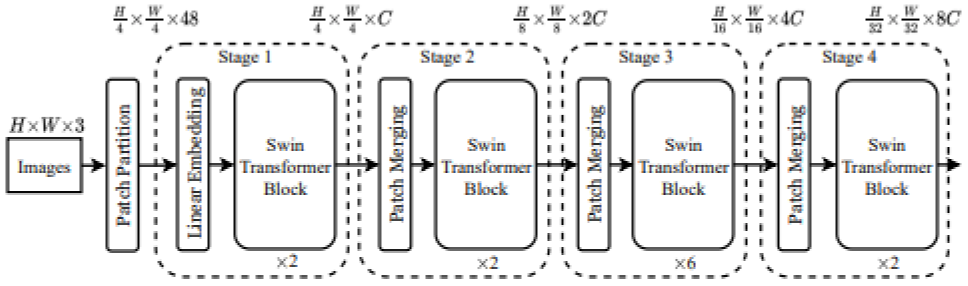

$\frac{H}{4}\times\frac{W}{4}\times 48$
$\frac{H}{4}\times\frac{W}{4}\times C$
$\frac{H}{8}\times\frac{W}{8}\times 2C$
$\frac{H}{16}\times\frac{W}{16}\times 4C$
$\frac{H}{32}\times\frac{W}{32}\times 8C$
Stage 1
Stage 2
Stage 3
Stage 4
$H\times W\times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
Patch Merging
×2
×2
×6
×2

Fig 5. SWIN Architecture

Next up is the **patch merging layer**; the first patch merging layer concatenates the features of each group of 2×2 neighboring non-overlapping windows and reduces the number of tokens with new output dimension as 2C. SWIN transformer blocks processes these tokens to extract the features, however the resolution of patches this time is H/8 x W/8.

This entire block of patch merging and feature extraction is referred to as Stage 2. There are two more such blocks which are referred to as Stage 3 and Stage 4, and each of

them reduces the token numbers to half of input resolution which is H/16 x W/16 and H/32 x W/32 respectively.
These stages allow the tokens size to be variable across the layers, which helps with reliable feature extraction of visual entities on different scales and as explained before, the shifting nature of these windows helps with establishing self-attention connectors with all the patches across different windows.

Shifted Window Multi-Head Self Attention

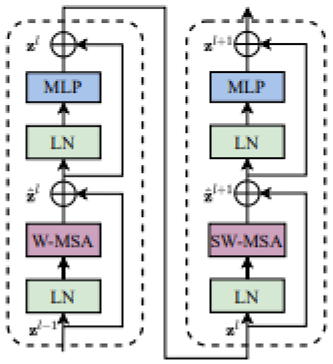


Fig 6. SWIN Transformer block

SWIN transformers replaces the multi-head self attention usually used in the transformers by the shifted window multi-head self attention as seen above. The rest of the layers like MLP and Layer Normalization are kept the same as in traditional transformers.
This implies the self attention is only performed in the local windows which are further divided into patches of size MxM. It ensures that the computational complexity of the model remains linear, as long as M is a constant. In SWIN transformer's case the M is kept to be constant at 7. Following equation can depict the difference between computational complexity of the traditional multi-head self attention and the shifted window self attention against the image size hxw

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \qquad (1)$$
$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \qquad (2)$$

It can be clearly seen that the multi-head self attention MSA varies quadratically with the image dimension hxw. Whereas, shifted window multi-head self attention W-MSA varies linearly with the image dimensions hxw if the variable M is constant. This makes SWIN transformer more suitable for training on images with higher resolutions.

The downside to window-based self attention is the lack of self-attention connections outside the local window, which limits the model's capability to capture the global contexts. The SWIN tackles this problem by shifting the windows with respect to each other in successive SWIN blocks. There are two window partitioning configuration which are alternated between two blocks of a particular stage.

The first block partitions the image regularly starting from the leftmost pixel, and dividing the 8×8 feature map into a 2×2 window grid where each window is 4×4 dimension (M=4). For the next block this window is shifted by M/2, M/2 pixels with respect to windows in previous block. This shifting now allows cross window connections which helps in gaining a global context for the relationships between the patches across the windows. To illustrate the processing of the image input sequence by the two blocks following equations come in handy.

$$
\begin{aligned}
&\hat{\mathbf{z}}^{l} = \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1}, \\
&\mathbf{z}^{l} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l}\right)\right) + \hat{\mathbf{z}}^{l}, \\
&\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^{l}\right)\right) + \mathbf{z}^{l}, \\
&\mathbf{z}^{l+1} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1}, \qquad (3)
\end{aligned}
$$

The zl' and zl are feature map outputs by the MSA block and the MLP block respectively, zl and zl+1 are the outputs from two consecutive layers of SWIN transformer blocks. One issue with shifted window partitioning is visible in the figure 4, where the second configuration has more windows as compared to the first configuration, it can also be seen that the some windows are smaller in second configuration.

SWIN transformer architecture solves this problem using cyclic shifting windows, where the windows on the fringes are padded with each other. This is called the cyclic shift padding depicted in figure ___

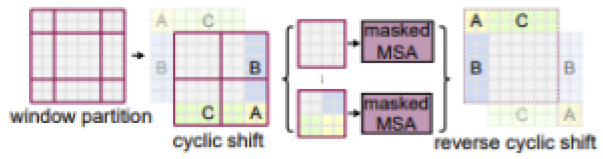


Fig 7. Cyclic shift padding

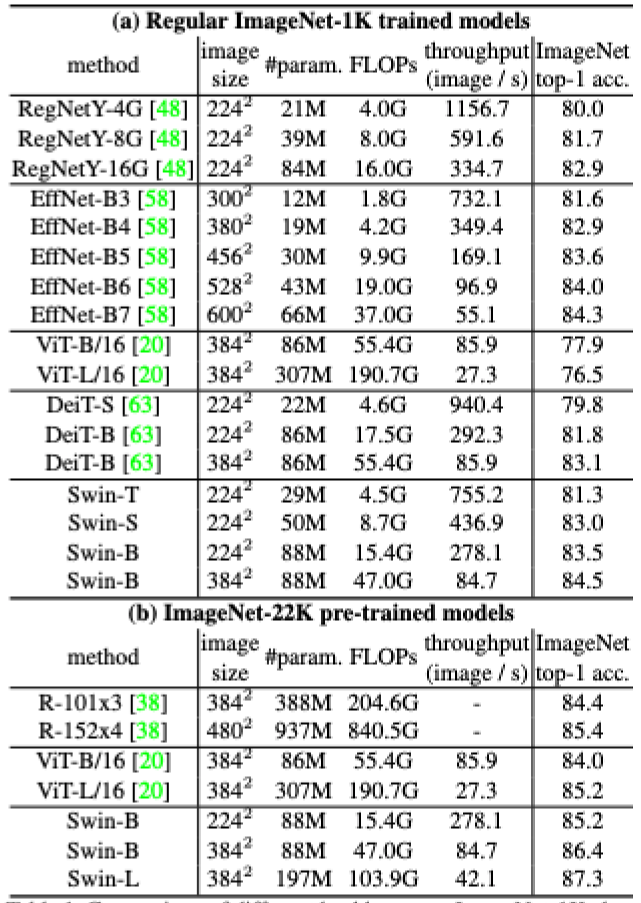
Results and Experiments

The performance of the SWIN transformer model has been analyzed and compared against state of the art CNNs and Vision Transformers for all mainstream computer vision tasks like image classification, object detection and image segmentation.

There are 4 variants of the SWIN transformer architecture, which vary in number of layers and the input dimension C of the input token sequence after linear projection. The different variants can be categorized as below

- Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers =$\{2, 2, 18, 2\}$

For image classification, the SWIN transformer has been evaluated against the state of the art CNNs like RegNet and EfficientNet and also against transformers like ViT and DeIT. The benchmark dataset utilized for this comparison is the widely accepted ImageNet 1k, which has 1.28million training images along with 50k evaluation images for about 1000 classes. SWIN-T outperforms DeIT-S architecture by 1.53% for 224×224 input for the top-1 accuracy metric



(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

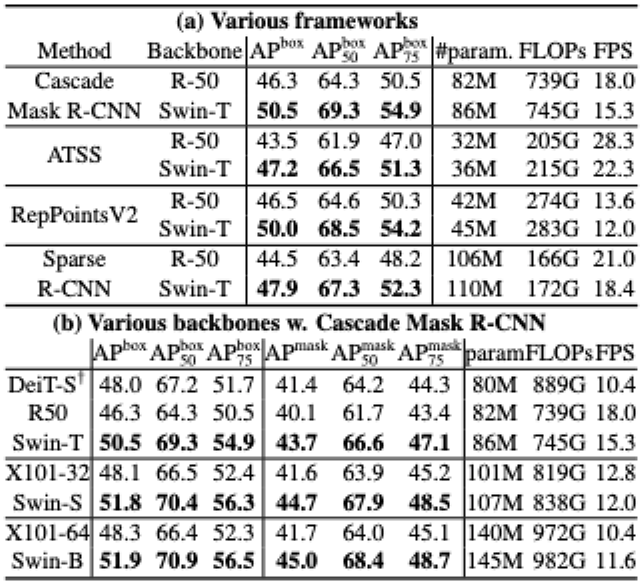
(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

For object detection, COCO 2017 dataset benchmark is used to judge the performance of SWIN and compared against other backbones. For object detection, only the state of the art model backbones like ConvNeXt and DeIT are compared with SWIN.

The framework used to predict bounding boxes with these backbones is the cascade mask-RCNN. SWIN-B achieves a high box detection average precision of 51.9AP, which is a 3.6 point gain over what ResNeXt101-64x4d exhibits. On the other hand SWIN-T outperforms the DeIT-S model of similar size by 2.5 box AP points.



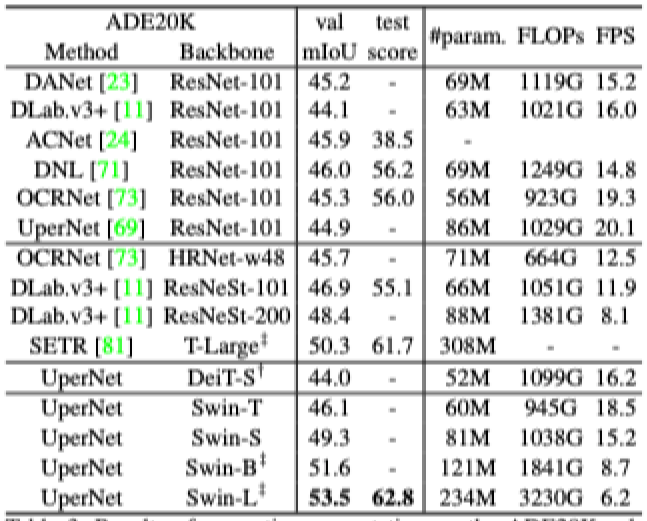
(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse	R-50	44.5	63.4	48.2	106M	166G	21.0
R-CNN	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

For Image segmentation, the benchmark dataset ADE 20k is used which consists of about 150 classes distributed across 20k training images, 2k validation images and 3k testing images. SWIN is compared to DeIT and ResNet 101 models. SWIN-S exhibits 5.3mIoU points higher the DeIT-S model of same computation cost. It also outperforms the ResNet-101 by 4.4 mIoU points.



ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

Conclusion: Vision Transformers for Cyber Defense

Vision Transformers are an emerging method in solving computer vision tasks due to the immense success of transformers in the NLP domain. However, vision transformers are impeded by some fundamental issues arising out of differences between text data and image data.

The variation of scale between various visual entities is higher in the image data. Vision Transformers also use the same fixed scale token strategy used by transformers in NLP domain, unsuitable for extracting feature patterns for the small scale visual entities. Vision transformer have quadratic computational complexity against image size, making it unsuitable for processing high resolution image data.

SWIN transformers solve this problem by proposing images to be broken in windows, these windows are further broken down into smaller patches, these patches are fed as an input sequence and attention is calculated only by creating self attention connection locally. The self-attention connection with patches in other windows is achieved by shifting the windows in next block and allow the flow of connection from previous block.

Between the stages, the window size is decreased to allow creation of hierarchical maps. Since the patches are fixed in number, the computational complexity against the image dimensions is restricted to linear, making SWIN more suitable than traditional transformer.
To learn more about Bolster's Research Team, and to see how generative AI can help protect your business from cyber attacks, ***speak with the Bolster team today****!*

References

1. **Vision Transformers**

2. **SWIN transformer**

3. **DeIT**

4. **ConvNeXt**

5. **ResNet**

6. **LeNet**