Swin Transformer

Hierarchical Vision Transformer using Shifted Windows

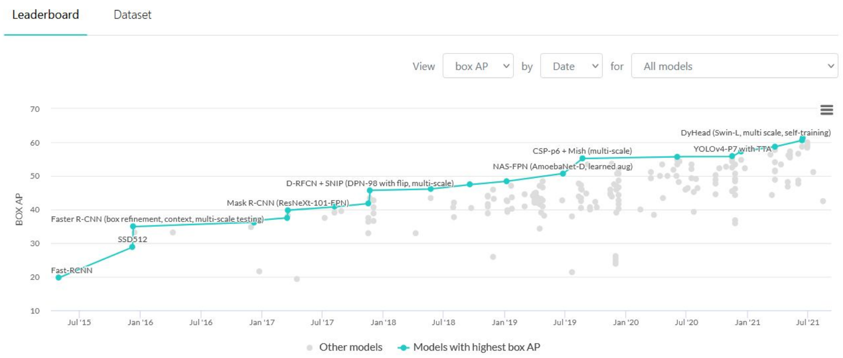
Ze Liu and colleagues in ICCV 2021

Anil Keshwani @ PINLab Reading Group - 3rd November 2021

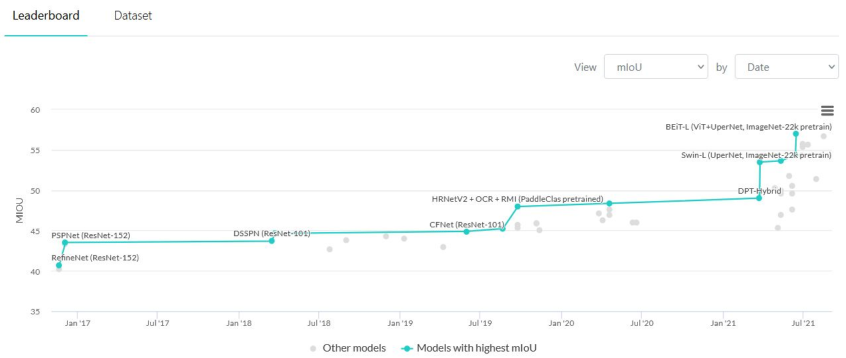
Summary: Swin Transformer

- General purpose computer vision backbone
- Hierarchical transformer with representations computed with shifted windows
- Linear complexity with respect to image size
- Self-attention limited to non-overlapping, local windows
- Tractable for dense prediction tasks (and image classification):
    - Object detection (bbox) - 58.7 box AP (COCO test-dev) - +2.7 box AP over SotA
    - Object detection (mask) - 51.1 mask AP (COCO test-dev) - +2.6 mask AP over SotA
    - Semantic segmentation - 53.5 mIoU (ADE20K val) - +3.2 mIoU over SotA
    - ImageNet Top-1 Accuracy on Classification of 87.8% (SotA 90.9% now)

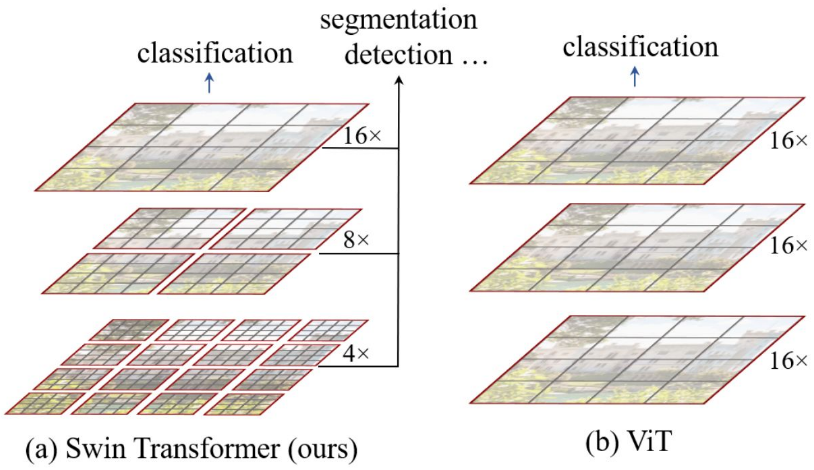
Object Detection on COCO test-dev



Semantic Segmentation on ADE20K val



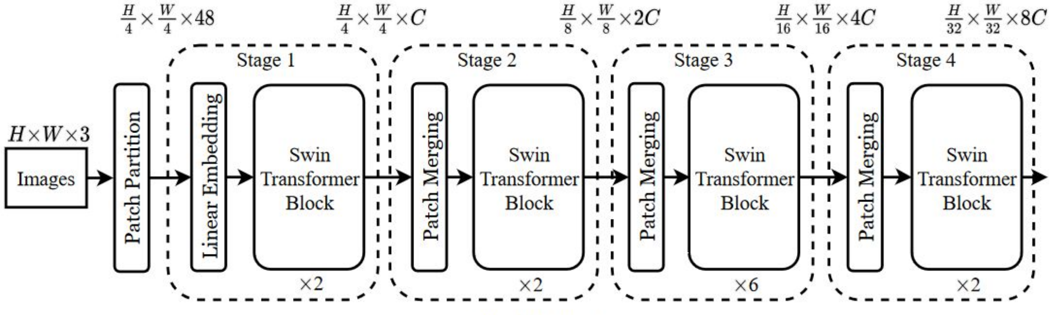
Idea: Central Approach of Swin Transformer



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

- Hierarchical feature maps by merging image patches in deeper (on left: higher) layers
- Compute self-attention only within each local window (on left: red demarcations)
- Approach allows linear complexity, not quadratic complexity with respect to "tokens"
- Dense prediction tasks are tractable
  - i.e. pixel-level predictions for semantic segmentation
- cf: Vision Transformer (ViT) computes
  - Single low-resolution feature map
  - Global self-attention → quadratic complexity
- Complexity equations: Windows contain MxM patches over an h x w x C input image. Fix M=7

Method: Architecture of Swin Transformer “Tiny” (Swin-T)



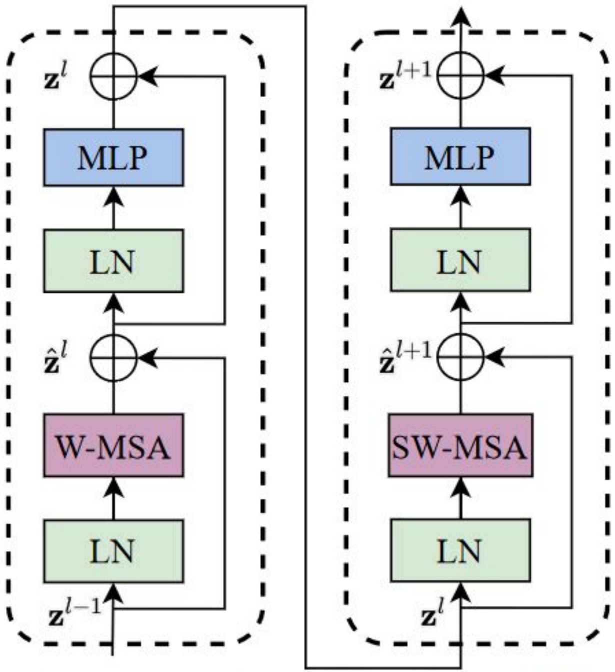
(a) Architecture

- Image split into non-overlapping patches (like ViT) → “tokens”
  - 4 x 4 x 3 = 48 dimensional tokens
- Linear projection → reduce dimensionality to C
- Stage 1 Swin Transformer block: modified self-attention

  → retains token number: (H / 4) x (W / 4)
- Stage 2:
  - Patch Merging of 2 x 2 groups → 2x resolution downsampling
  - Linear Projection of 4C-dimensional features → output dimension set to 2C
  - *Ex. Starting with 4 “patches” we get 1, but with double the Channels, to increase representation capacity*
- Stage 2 is repeated two more times: Stage 3 & Stage 4

  → produces hierarchical feature map resolutions like ResNet/VGG

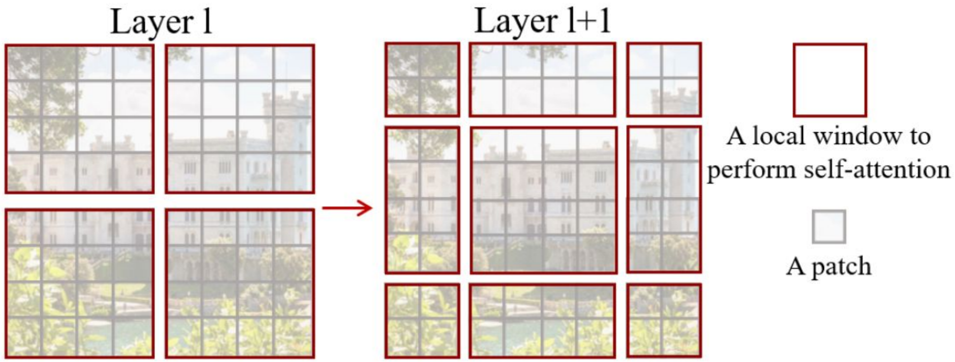
Method: Swin Transformer Block



Two Successive Swin Transformer Blocks

- Like a Transformer Block but…
- Replace Multi-headed Self-Attention with shifted windows version
- Note shift of windows across Blocks
- Swin Transformer Block:
    - Layer Normalisation
    - (Shifted) Window Multi-headed Self-Attention
    - + Residual Connection
    - Layer Normalisation
    - 2-layer MLP with GELU activation
    - + Residual connection

Method: Shifted window partitioning in successive blocks



- Not inter-block connections in window-based self-attention

  → limits modelling power (c.f. global self-attention!)

- "Shifted" window partitioning approach
- Alternate between partitioning configurations in consecutive Swin Transformer Blocks

  → Enables information flow across windows

Example (on left)

- 8 x 8 image partitioned into 2 x 2 windows of size 4 x 4
  - i.e. M = 4
- Next self-attention module has shifted window configuration

  → displace windows by (floor(M / 2), floor(M / 2))

Some Details: Relative Position Bias + Architectural Variants

Relative Position Bias

- Relative positional embeddings used
- Relative position performs better than absolute empirically

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d} + B)V,$$

- Actually B not M^2 x M^2 ; smaller (2M - 1) x (2M - 1)

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture.

Architecture Variants

- Base model: Swin-B with model size and complexity on par with ViT-B
- Swin-T, Swin-S and Swin-L are 0.25x, 0.5x and 2x size and complexity
  - Swin-T comparable to ResNet-50; Swin-S to ResNet-101
- Window size, M = 7 (default; all experiments)
- C: Channels from hidden layers in Stage 1

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 =$\{2, 2, 18, 2\}$

Experiments: Results for Image Classification

Regular ImageNet-1K training: 1.28M training images and 50K validation images from 1K classes

- No repeated augmentation or exponential moving averaging, as is used for ViT
- 1.5% for Swin-T (81.3%) over DeiT-S (79.8%) using 224 x 224 input
- +1.5%/1.4% for Swin-B (83.3%/84.5%) over DeiT-B (81.8%/83.1%) using 224 x 224 or 384 x 384 input
- Note: No architecture search, like e.g. EfficientNet

Pre-training on ImageNet-22K (14.2 million images; 22K classes) + fine-tuning on ImageNet-1K

- Swin-B obtains 86.4% top-1 accuracy, which is 2.4% higher than ViT with similar inference latency

(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].

Experiments: Results for Object Detection

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	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-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	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

COCO 2017: 118K training, 5K validation and 20K test-dev images

- Swin-T architecture brings consistent +3.4~4.2 box AP gains over ResNet-50; with slightly larger model size, FLOPs and latency

Comparison with DeiT-S (with Cascade Mask R-CNN)

- Swin-T has +2.5 box AP and +2.3 mask AP higher than DeiT-S with similar model size (86M vs. 80M)
- significantly higher inference speed (15.3 FPS vs. 10.4 FPS)

Lower inference speed of DeiT mainly due to its quadratic complexity to input image size

Experiments: Results for Semantic Segmentation

ADE20K: 150 semantic categories; 25K images in total, with 20K for training, 2K for validation, and another 3K for testing

- Swin-S +5.3 mIoU over DeiT-S with similar computation cost (49.3 vs. 44.0)
- Swin-S also +4.4 mIoU higher than ResNet-101, and +2.4 mIoU higher than ResNeSt-101
- Swin-L with ImageNet-22K pretraining surpasses SETR (previous SotA) by +3.2 mIoU

Much smaller parameter size of DeiT-S maybe seems cheeky; authors equated FLOPs not model size.

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.

Experiments: Ablations

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture. w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*; abs. pos.: absolute position embedding term of ViT; rel. pos.: the default settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

Shifted Windows

- +1.1% top-1 accuracy on ImageNet-1K, +2.8 box AP/+2.2 mask AP on COCO, and +2.8 mIoU on ADE20K
- …even without shifted windows, performance is good, just not as good!
- Not much latency overhead paid for using shifted windows with cyclic-shifted for efficient batching

Positional Embeddings

- Positional embeddings outperform absolute
- Adding absolute with relative reduces performance marginally for object detection and segmentation (but increase classification slightly; 0.4%)
    - e.g. 46.1 versus 44.0 in ADE20K mIoU

Efficient batch computation

Additional and undersized windows created by shifting of windows → problems with batching and latency. Solved with cyclic-shifting (see paper)

Conclusions

- The Swin Transformer introduces some of the “inductive biases” inherent to CNNS in the ViT approach and architecture via the Patch Merging module

  → Model becomes hierarchical and resembles common backbones with increasing channel width with higher layers

- This model effectively implements Local Attention (not novel), but very effectively
- Shifting windows further circumvents utility of global attention and improves performance (evidenced via ablation)
- Local attention (M = 7) linearises complexity

  → opens up dense prediction tasks, where ViT falls down e.g. segmentation

- Relative positional embeddings may enable translation invariance
  - Why does this improve dense task performance but hurt classification?

Resources

Swin Transformer

- Paper: https://arxiv.org/pdf/2103.14030.pdf
- Code: https://github.com/microsoft/Swin-Transformer/

Datasets

- COCO Detection Evaluation (including Metrics): https://cocodataset.org/#detection-eval
- ADE20K Dataset: https://groups.csail.mit.edu/vision/datasets/ADE20K/

Baselines

- Data-efficient Image Transformer: https://paperswithcode.com/method/deit
- Vision Transformer: https://paperswithcode.com/method/vision-transformer
    - See also https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html

State of the Art

- ImageNet Classification SotA on Papers with Code: https://paperswithcode.com/sota/image-classification-on-imagenet
- ADE20K val Semantic Segmentation SotA on Papers with Code: https://paperswithcode.com/sota/semantic-segmentation-on-ade20k-val
- Object Detection on COCO test-dev SotA on Papers with Code: https://paperswithcode.com/sota/object-detection-on-coco

Thanks, That's a wrap!