Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Under Review, *ICCV'21*

Presenter: Sungwon Hwang

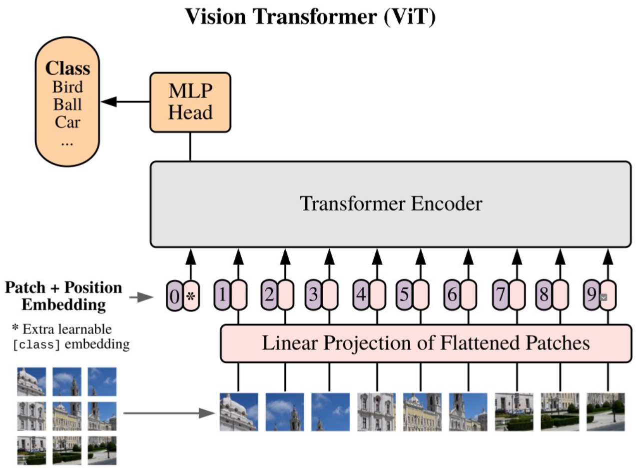
June 21, 2021

Ze Liu[†*] Yutong Lin[†*] Yue Cao[*] Han Hu[*‡] Yixuan Wei[†]
Zheng Zhang Stephen Lin Baining Guo
Microsoft Research Asia
{v-zeliu1,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com

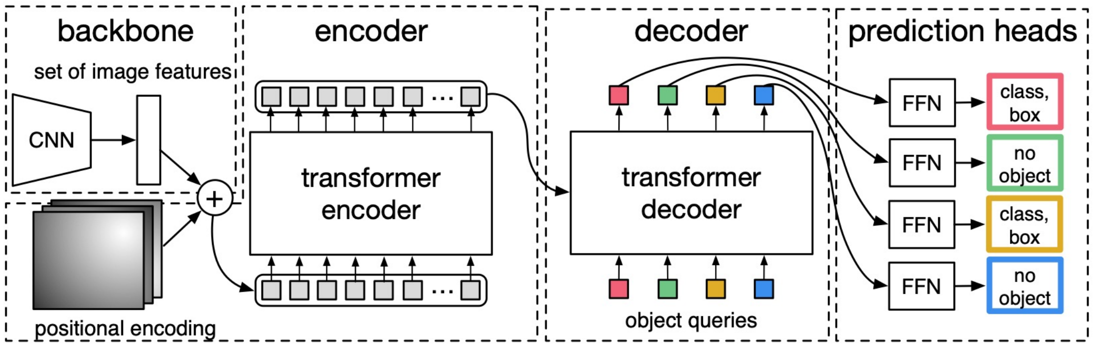
Introduction

Advents of different non-global self-attention for Vision

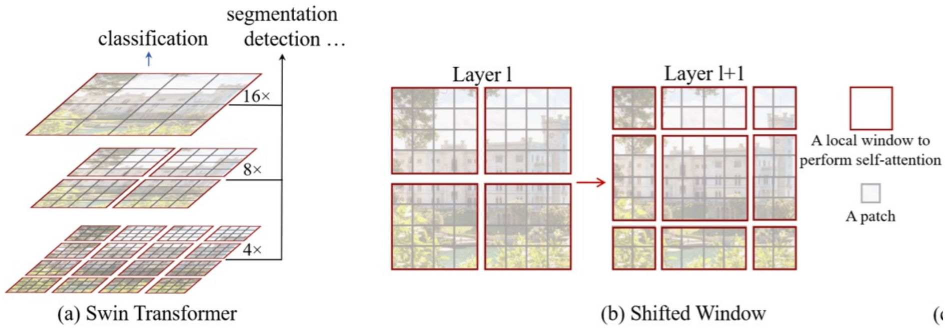
- **Global Attention**
  - ViT (Vision Transformer)



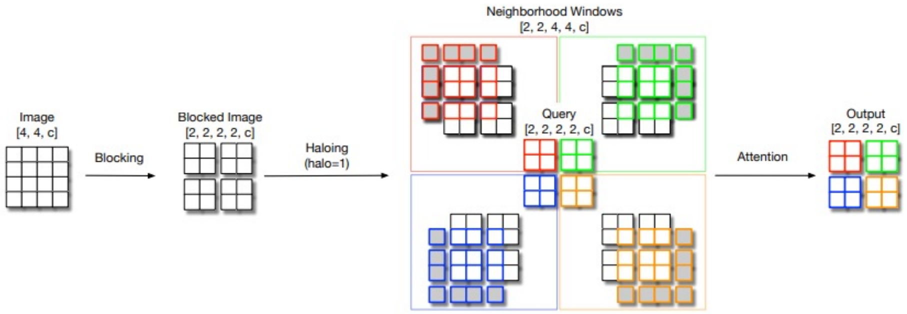
  - DETR (Detection Transformer)



- **(Spatially) Local Attention**
  - Swin (Sliding window) Transformer



  - (HaloNet) Scaling Local Self-Attention



Introduction

Two main differences in two modalities: Vision vs. NLP

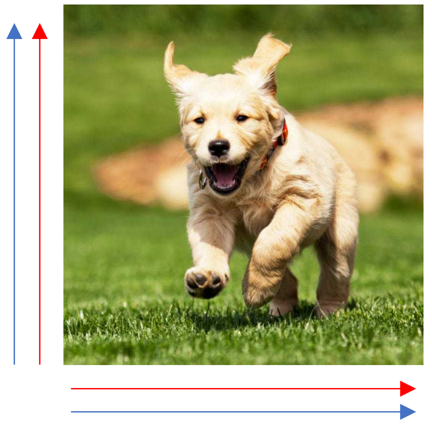
- Scale of unit information of interest

NLP	Vision
Tokens are all of a fixed scale	Locality are of variable scale

- Quadratic Size increment of image vs. Linear size increment of passages

Key contribution of this paper

Shifting (Not Sliding) windows



Passage 1

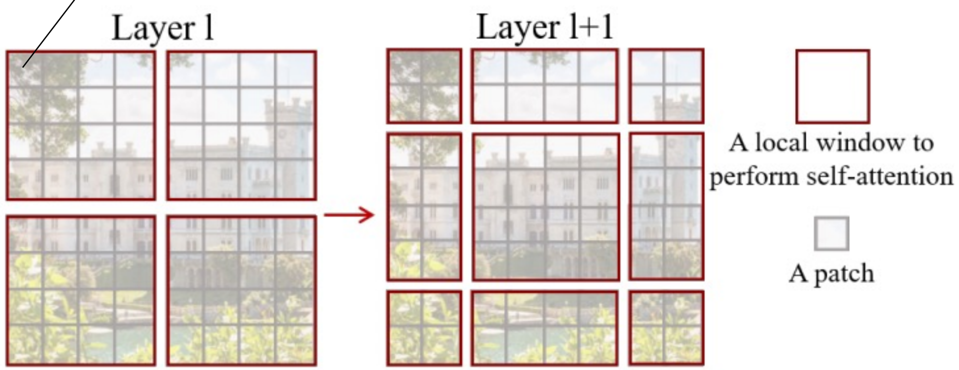
Food has always been considered one of the most salient markers of cultural traditions. When I was a small child, food was the only thing that helped identify my family as Filipino American. We ate *pansit lug-lug* (a noodle dish) and my father put *patis* (salty fish sauce) on everything. However, even this connection lessened as I grew older. As my parents became more acculturated, we ate less typically Filipino food. When I was twelve, my mother took cooking classes and learned to make French and Italian dishes. When I was in high school, we ate chicken marsala and shrimp fra diablo more often than Filipino dishes like *pansit lug-lug*.

Methodology

Local Self-Attention in Shifting Windows

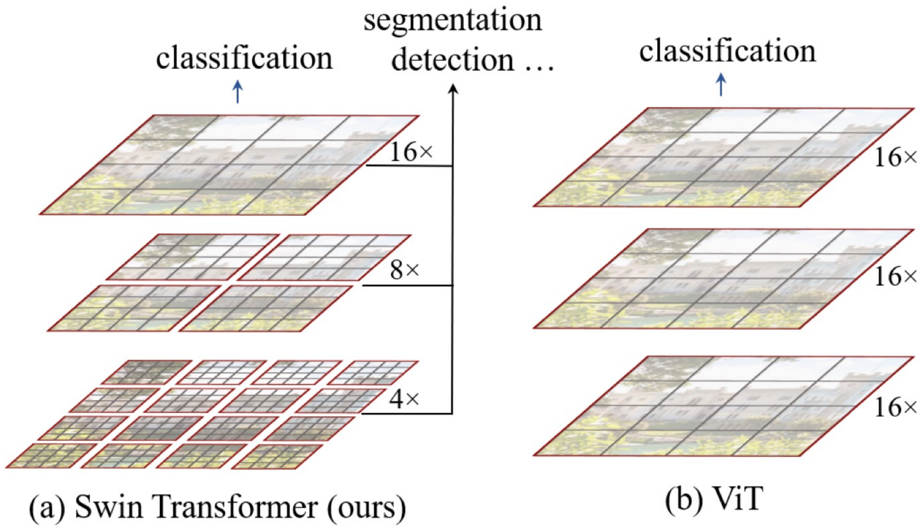
- Windowed Multi-head Self Attention (W-MSA) & Shifted Windowed Multi-head Self Attention (SW-MSA)
  - Local expansion of receptive field
  - Computational Complexity: $O\left((M^2)^2 C \cdot \frac{w}{M} \cdot \frac{h}{M}\right) = O(M^2 whC)$
  - Relative Position Bias added to every self-attention instead of positional embeddings
  - Connection across windows (Increment of receptive field)

Patch: (2, 2, c) -> (1, 1, 4·C)



W-MSA

SW-MSA



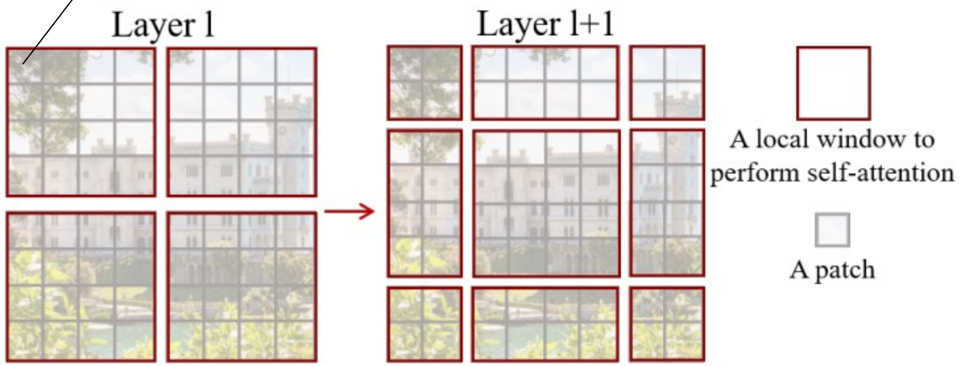
$$\mathrm{Attention}(Q, K, V) = \mathrm{SoftMax}(QK^T/\sqrt{d} + B)V,$$

Methodology

Local Self-Attention in Shifting Windows

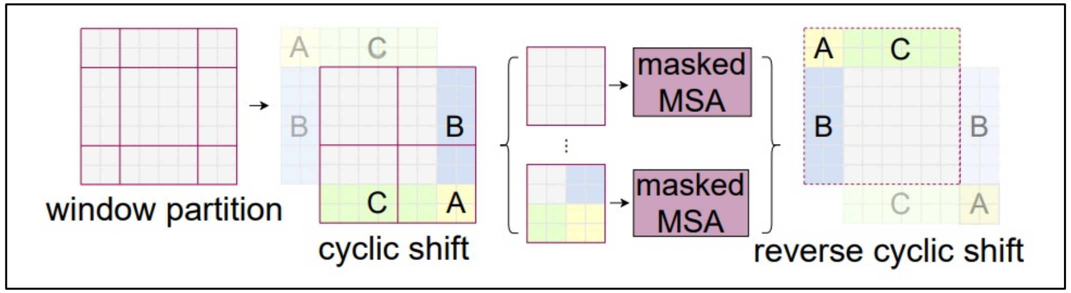
- Windowed Multi-head Self Attention (W-MSA) & Shifted Windowed Multi-head Self Attention (SW-MSA)
  - Local expansion of receptive field
  - Computational Complexity: $O\left((M^2)^2 C \cdot \frac{w}{M} \cdot \frac{h}{M}\right) = O(M^2 whC)$
  - For sub-windows, Padding noticeably increases computational complexity

Patch: (4, 4, c) -> (1, 1, 16·C)



W-MSA

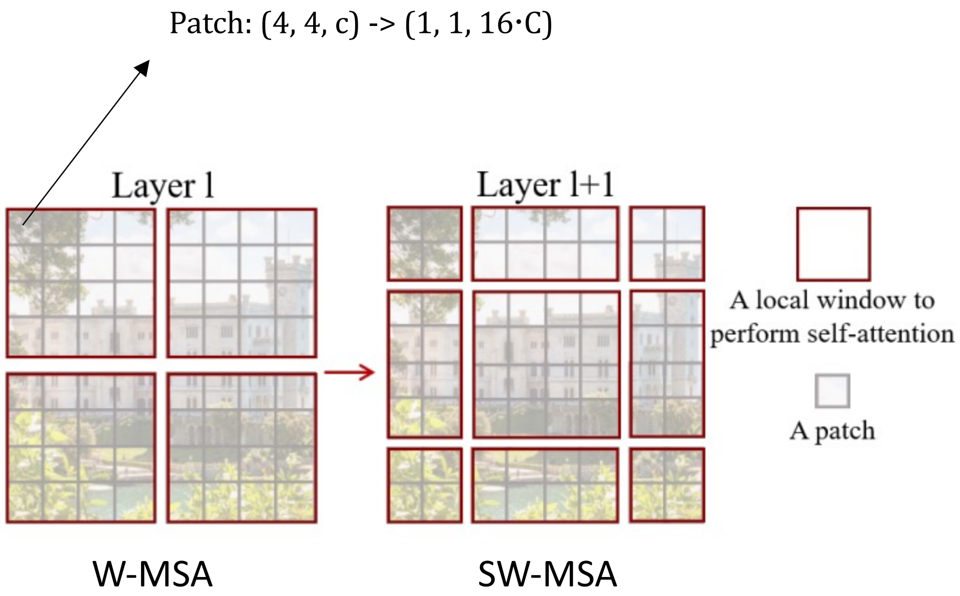
SW-MSA

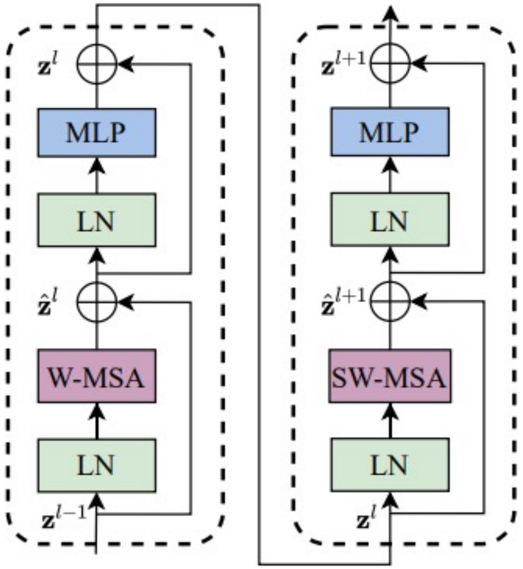


Methodology

Local Self-Attention in Shifting Windows

- Windowed Multi-head Self Attention (W-MSA) & Shifted Windowed Multi-head Self Attention (SW-MSA)
  - Local expansion of receptive field
  - Computational Complexity: $O\left((M^2)^2 C \cdot \frac{w}{M} \cdot \frac{h}{M}\right) = O(M^2 whC)$
  - For sub-windows, Padding noticeably increases computational complexity

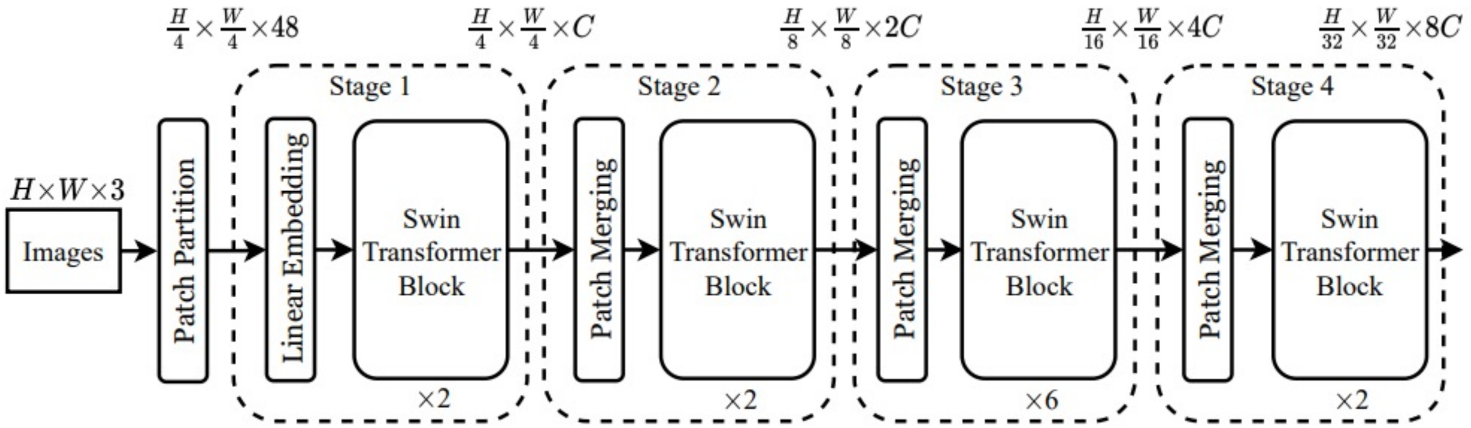




(b) Two Successive Swin Transformer Blocks

Methodology

- Overall Architecture
  - Patch Merging (down-sampling): 2x2 => 1d-concat => 4C to 2C mlp
  - Number of patches per window (M) fixed to 7 for all stage



(a) Architecture

Ablation Study

Ablation study

- Different types of injection of positional information
  - w/o app: no application of embedding in the first scaled-dot product term

	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**

Result

Performing well on LOCALIZATION TASKS

- Classification
  - ViT is claimed to work well when PRETRAINED WITH LARGER Dataset…?

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [47]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [47]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [47]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [57]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [57]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [57]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [57]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [57]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [60]	224^2	22M	4.6G	940.4	79.8
DeiT-B [60]	224^2	86M	17.5G	292.3	81.8
DeiT-B [60]	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	83.3
Swin-B	384^2	88M	47.0G	84.7	84.2

1	ViT-G/14	90.45%	1843M	˅
2	ViT-MoE-15B (Every-2)	90.35%	14700M	˅

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [37]	384^2	388M	204.6G	-	84.4
R-152x4 [37]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	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	86.0
Swin-L	384^2	197M	103.9G	42.1	86.4

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [65] and a V100 GPU, following [60].

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Result

Performing well on LOCALIZATION TASKS

- 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

(c) System-level Comparison

Method	mini-val		test-dev		#param.	FLOPs
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}		
RepPointsV2* [11]	-	-	52.1	-	-	-
GCNet* [6]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [12]	-	-	52.7	-	-	-
SpineNet-190 [20]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [75]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [58]	54.4	-	55.1	-	77M	410G
DetectoRS* [45]	-	-	55.7	48.5	-	-
YOLOv4 P7* [3]	-	-	55.8	-	-	-
Copy-paste [25]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Result

Performing well on LOCALIZATION TASKS

- Semantic Segmentation

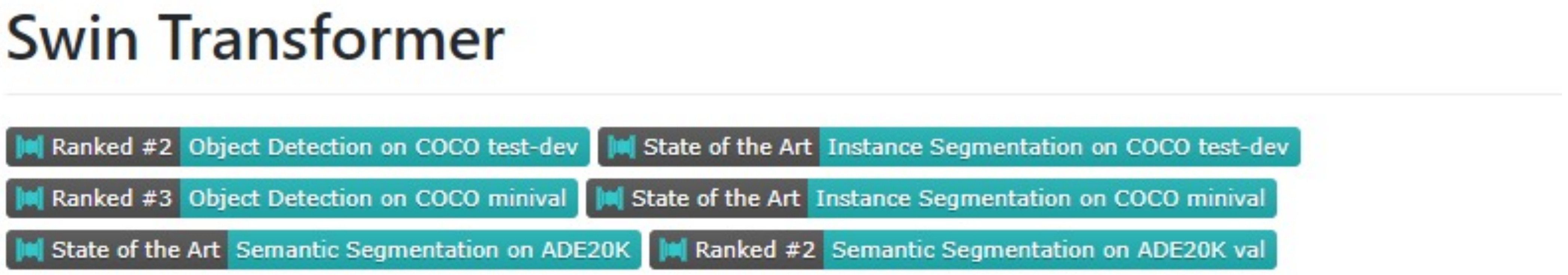
ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [22]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [10]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [23]	ResNet-101	45.9	38.5	-		
DNL [68]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [70]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [66]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [70]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [10]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [10]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [78]	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.

Result

Overall Architecture

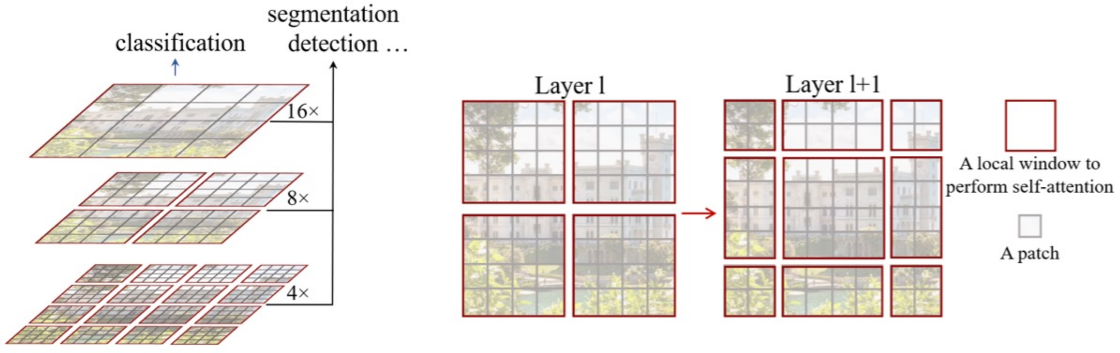
- SWIN: Strong in Localization specifically
  - ViT-based models outperform Swin in image classification
  - Swin outperforms in localization tasks such as detection & segmentation.



Thoughts: Why it works well?

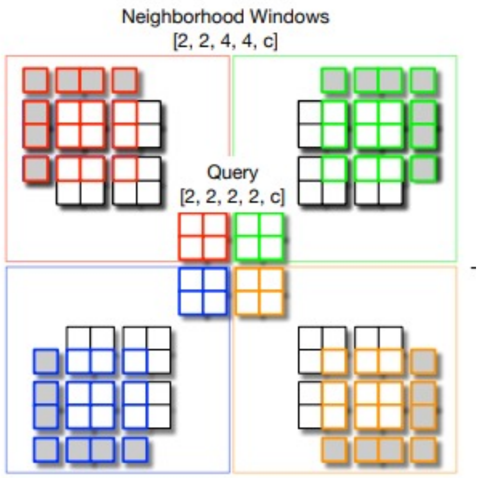
Efficient Diffusion of Receptive Field when using Local Self-attention

- Comparison with HaloNet #1: Receptive Field
    - Halonet: Gate to communication to other blocks (windows for Swin) is determined by h, but it’s too small
    - Swin: Each quarter of patches in the previous window attend to new neigbors whose size is equivalent to 3 times the quarter.
- Comparison with HaloNet #2: Pixel-wise vs. patch-wise
    - HaloNet: Pixel-wise local self-attention
    - Swin: Patch-wise local self-attention



(a) Swin Transformer

(b) Shifted Window



HaloNet Model	b	h	r_v	r_b	Total Layers	l_3	s	d_f	Params (M)	EfficientNet Params (M)	EfficientNet Image Size (M)
H0	8	3	1.0	0.5	50	7	256	–	5.5	B0: 5.3	224
H1	8	3	1.0	1.0	59	10	256	–	8.1	B1: 7.8	240
H2	8	3	1.0	1.25	62	11	256	–	9.4	B2: 9.2	260
H3	10	3	1.0	1.5	65	12	320	1024	12.3	B3: 12	300
H4	12	2	1.0	3	65	12	384	1280	19.1	B4: 19	380
H5	14	2	2.5	2	98	23	448	1536	30.7	B5: 30	456
H6	8	4	3	2.75	101	24	512	1536	43.4	B6: 43	528
H7	10	3	4	3.5	107	26	600	2048	67	B7: 66	600