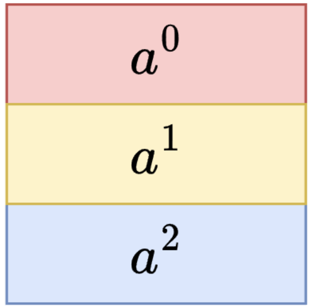
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

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

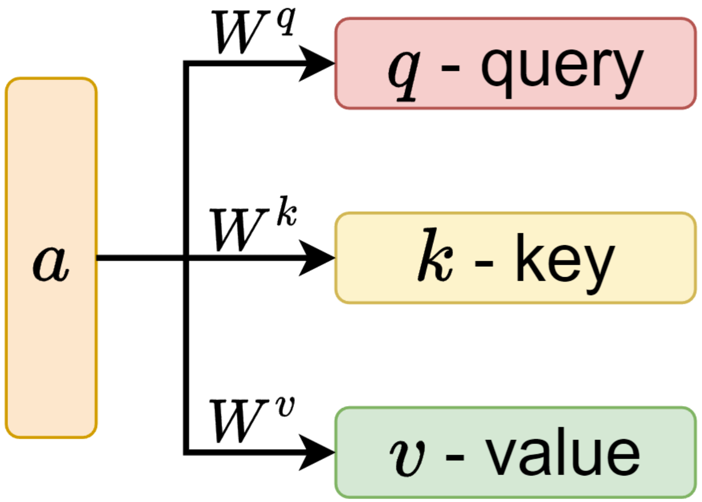
ICCV 2021 Best Paper

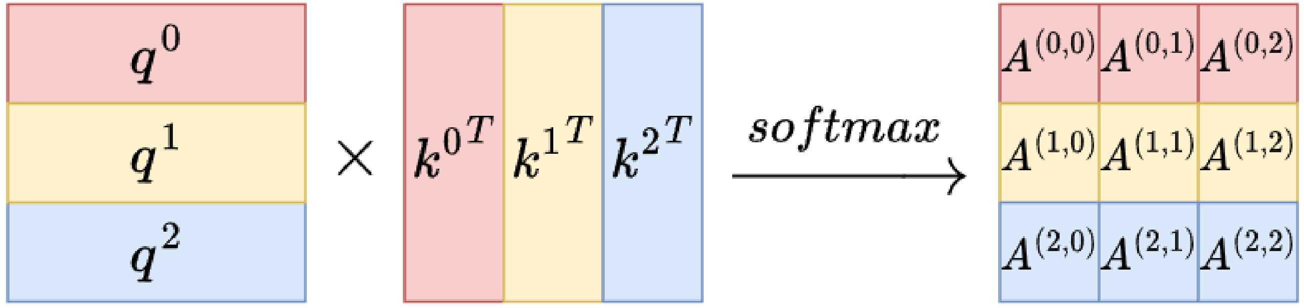
[1] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 10012-10022.

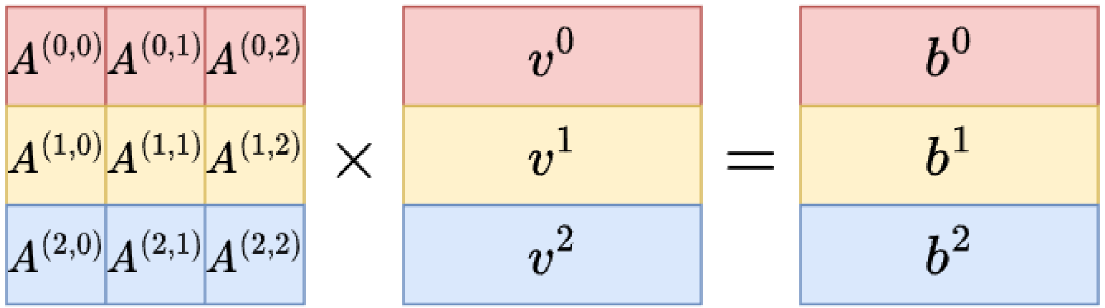
□Transformer – Self Attention



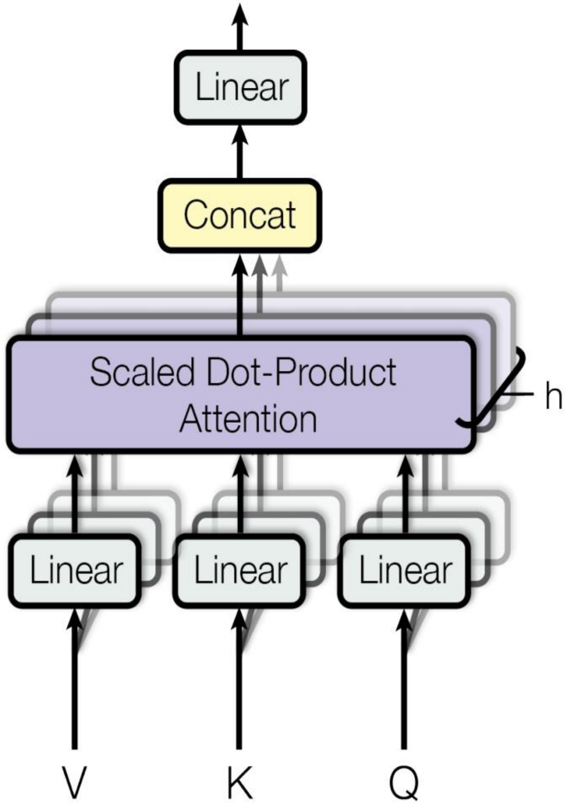
$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

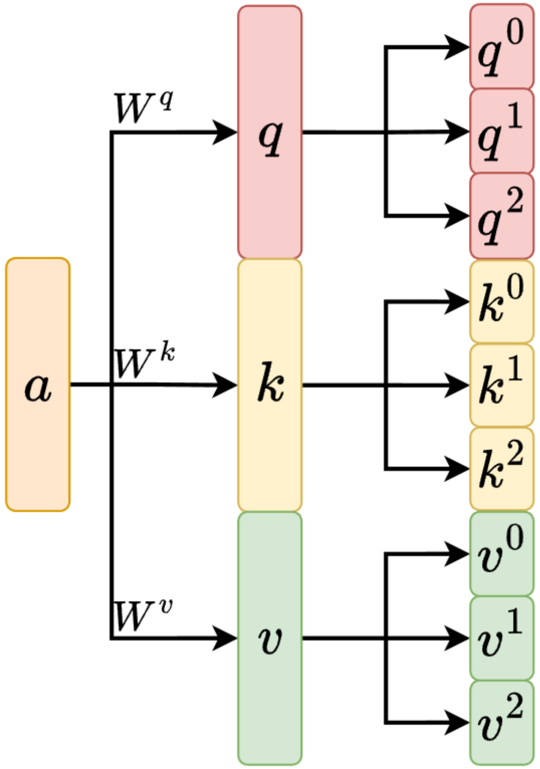






□Transformer – Multi-Head Self Attention

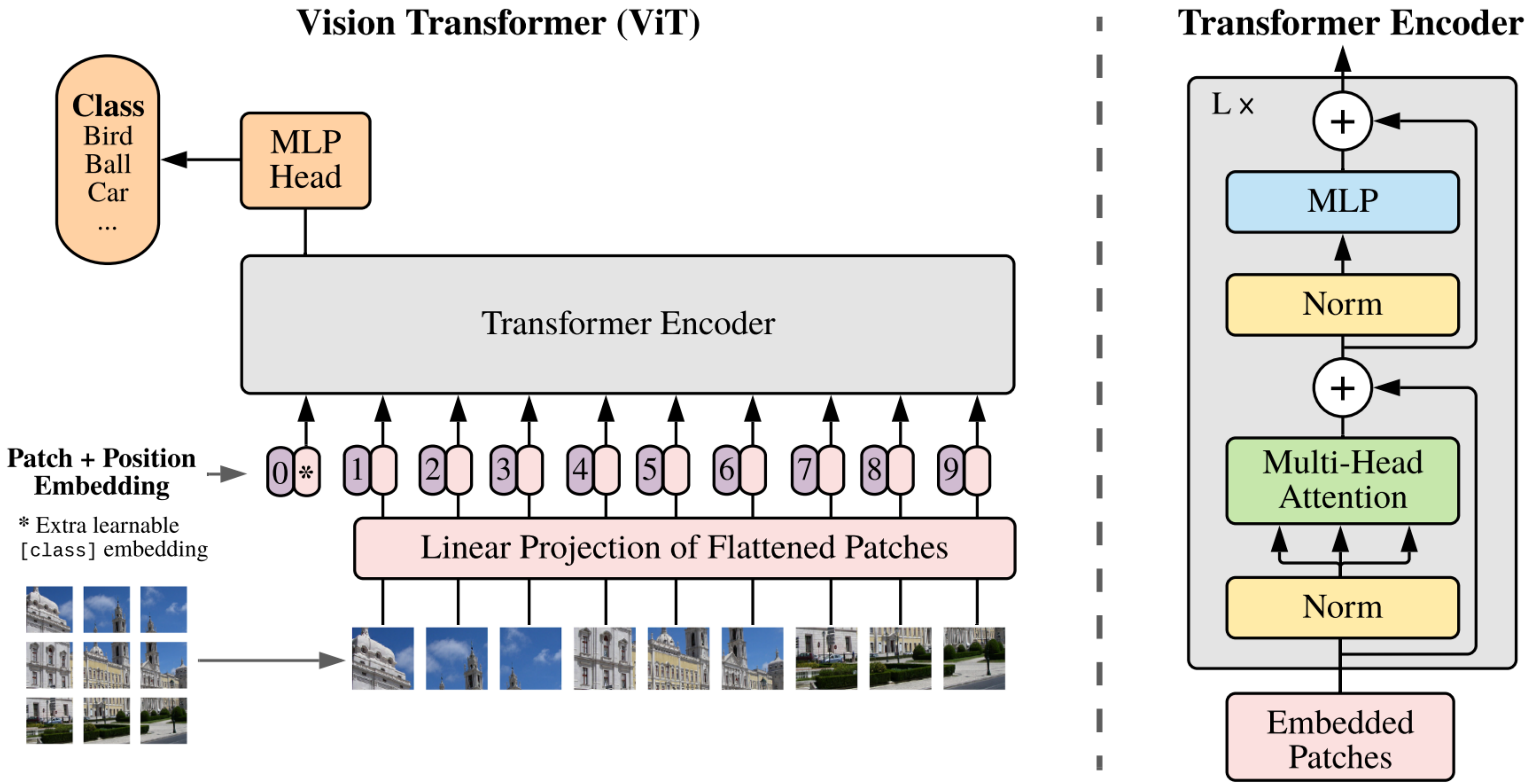




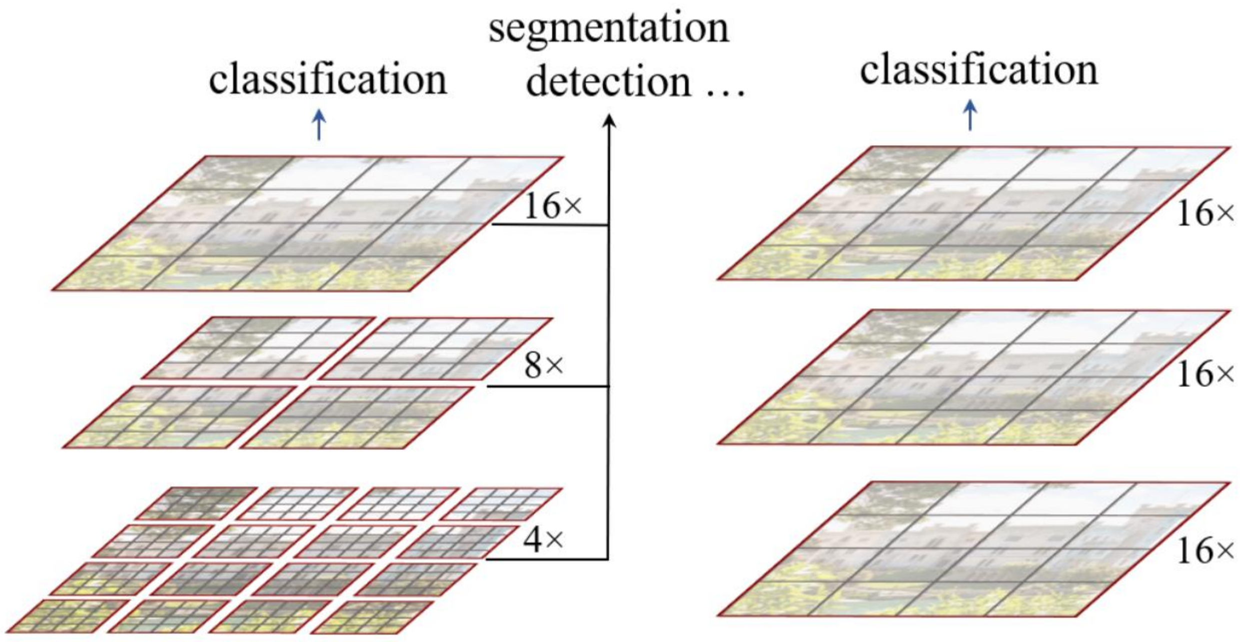
$$MulitHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^O$$

$$\text{where head}_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

◻Vision Transformer



☐Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

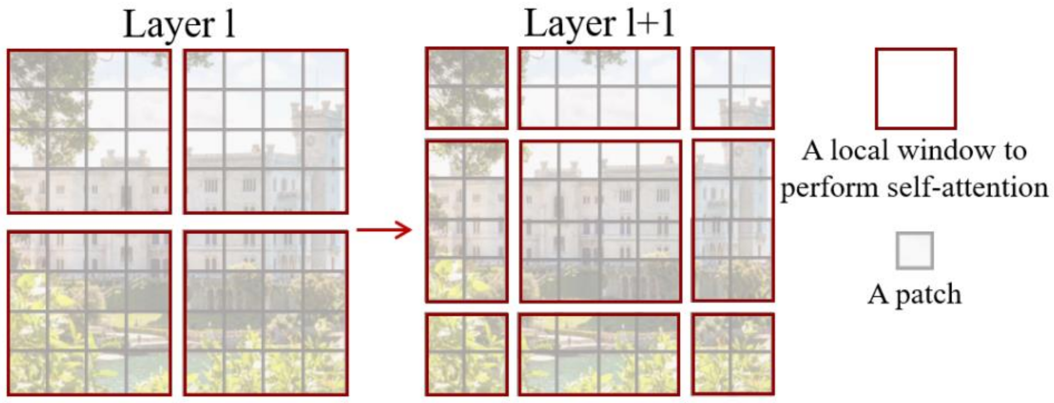


(a) Swin Transformer (ours)

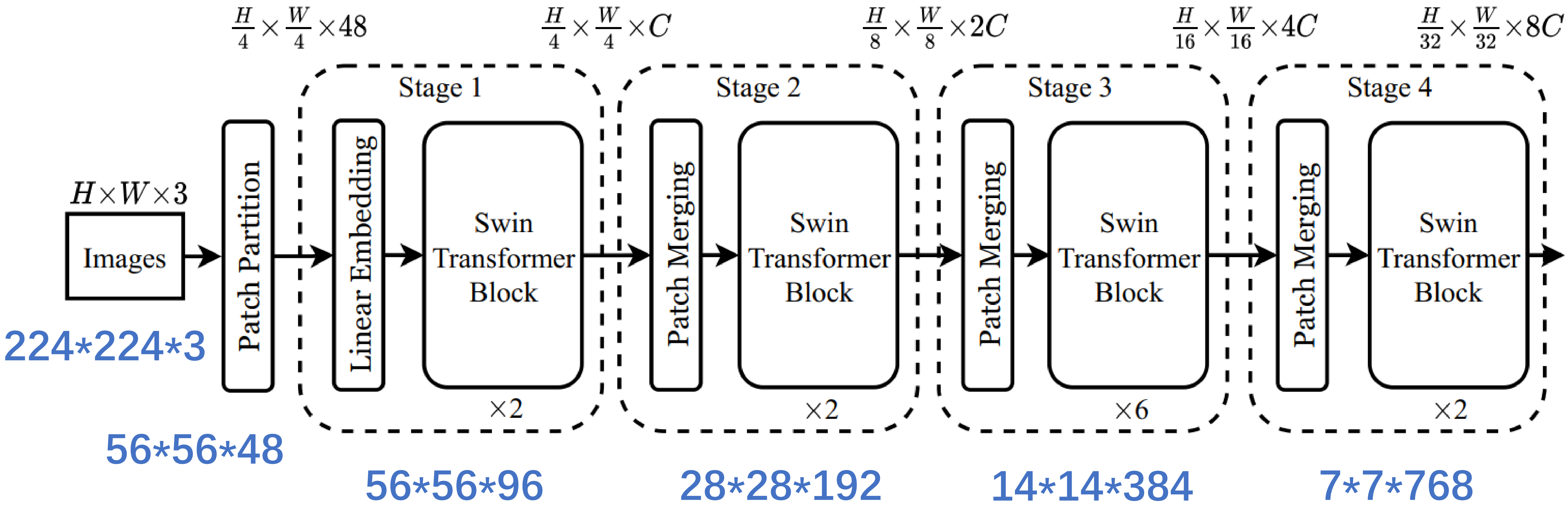
(b) ViT

- ViT
  - 固定下采样率
  - 不具有多尺度特征
- SwinT
  - 多尺度特征输出
  - 窗口降低运算复杂度
  - 移动窗口进行信息互通

In this paper, we seek to expand the applicability of Transformer such that it can serve as a general-purpose backbone for computer vision, as it does for NLP and as CNNs do in vision.

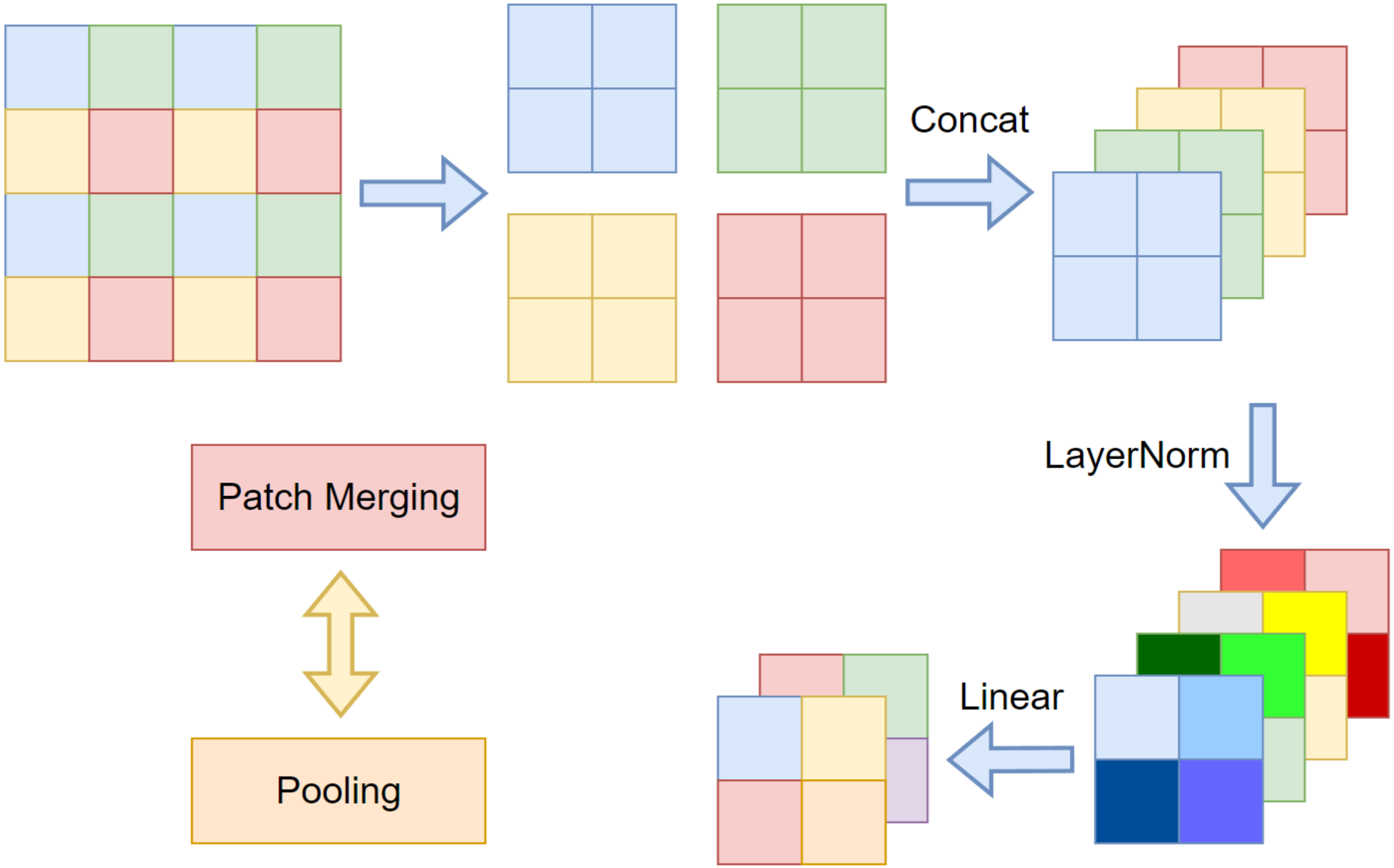


□Swin Transformer - Hierarchical Backbone



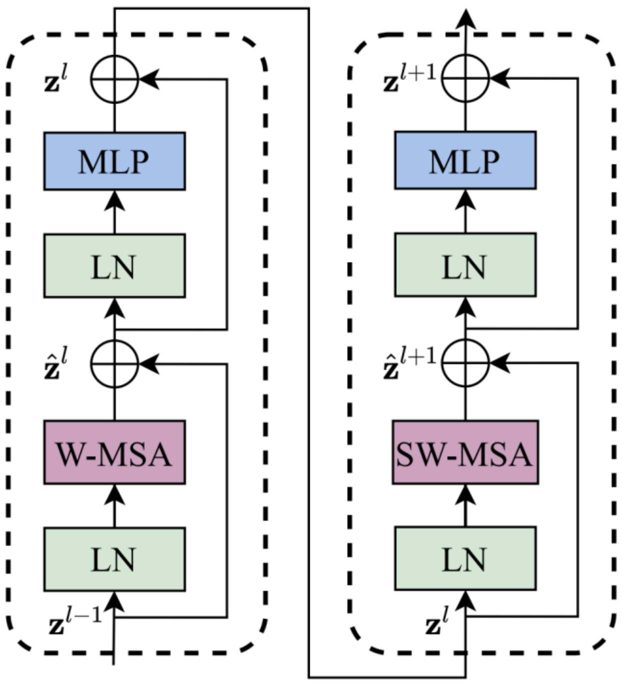
每一阶段的特征图宽高缩小一半，通道数增加一倍。

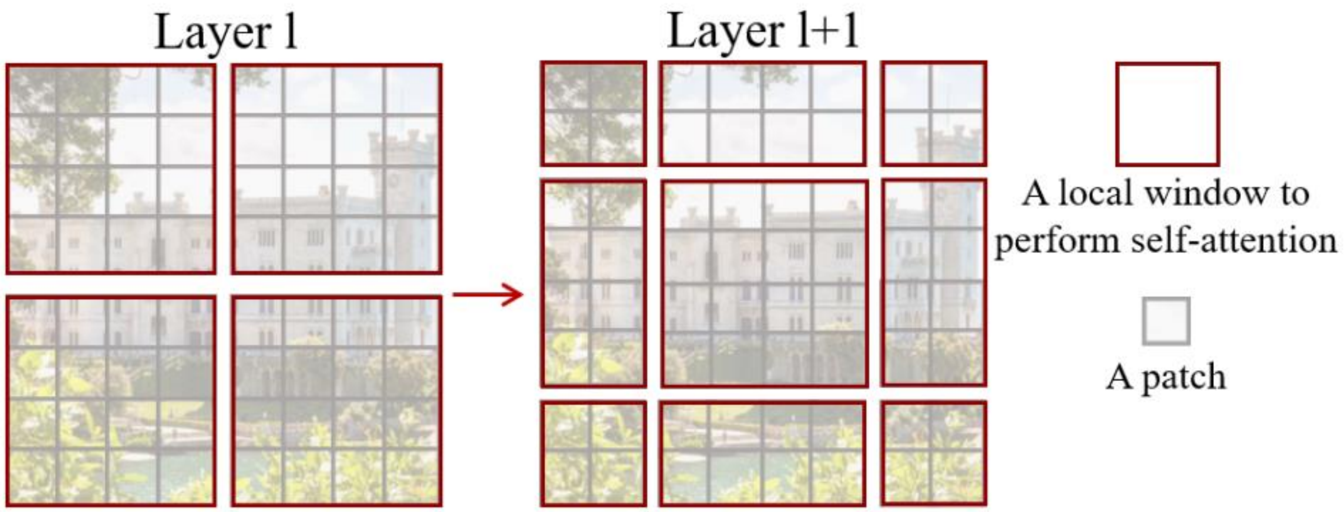
□Patch Merging



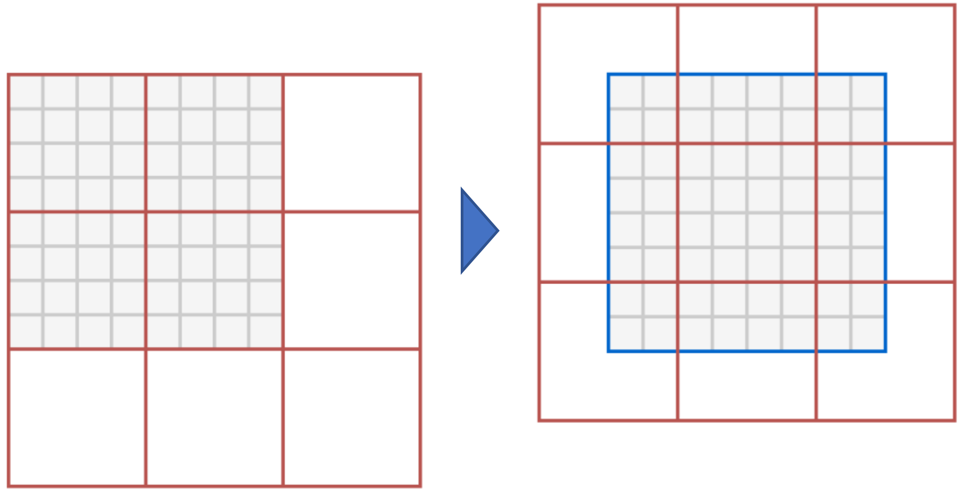
与池化层作用接近，实现多尺度特征

□Two Successive Swin Transformer Blocks



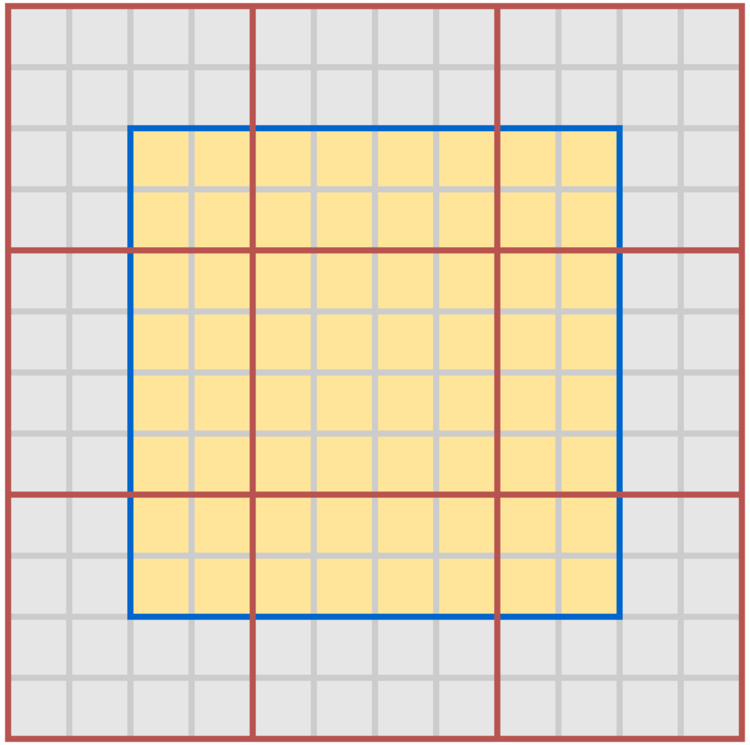


- ◆ 每个窗口堆叠起来当做一个个Batch并行Attention运算
- ◆ 移动窗口后窗口内图像大小不一，怎样继续并行Attention运算？



□Swin Transformer Block

◆Padding

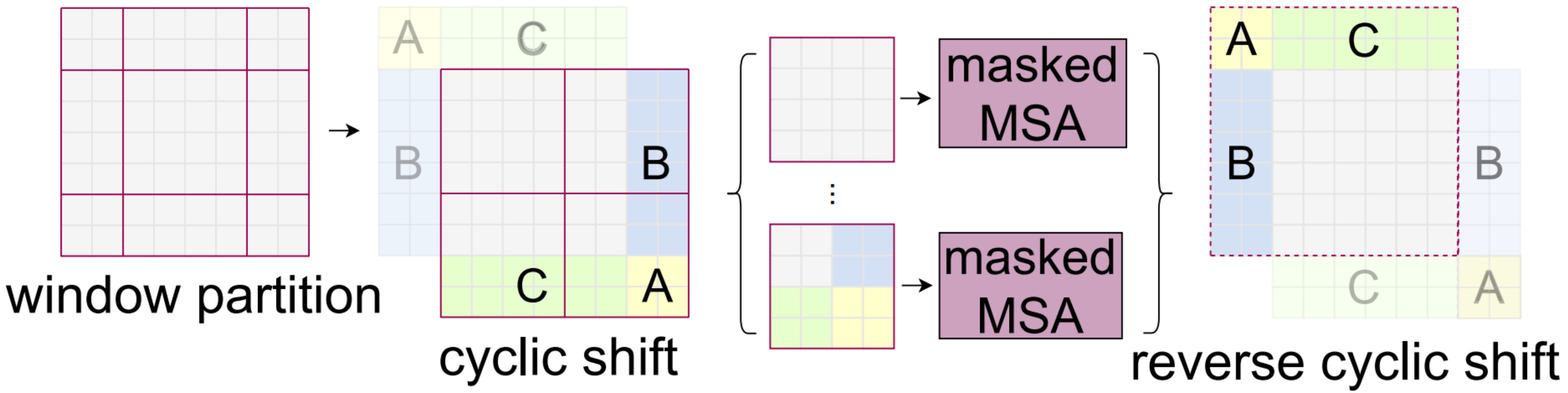


使用Padding存在的问题：

- 4个Batch → 9个Batch
- 边缘处窗口有大量无效运算
- pad位置的注意力需要mask

□Swin Transformer Block – Shifted Window

◆Cyclic Shift

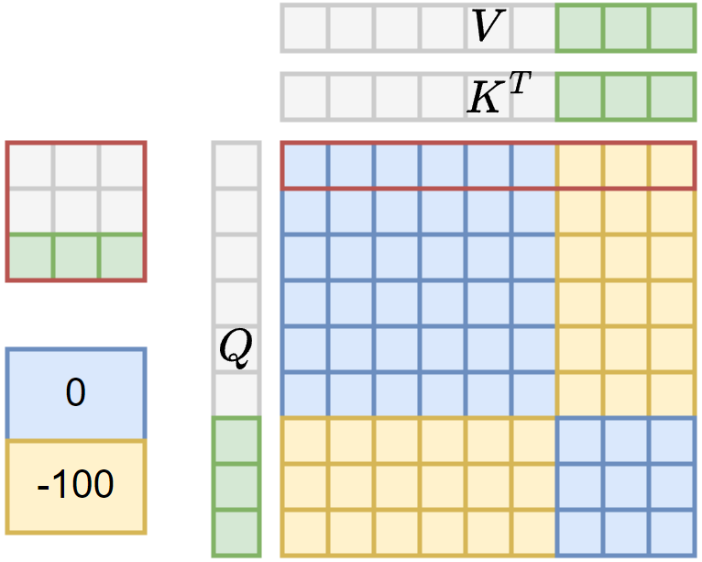


区域原本就连贯的窗口 ⇨ 正常进行Attention运算

因为移位拼接而成的窗口 ⇨ 只在原本连贯的区域内Attention

Swin Transformer Block

◆Cyclic Shift - Mask



Mask

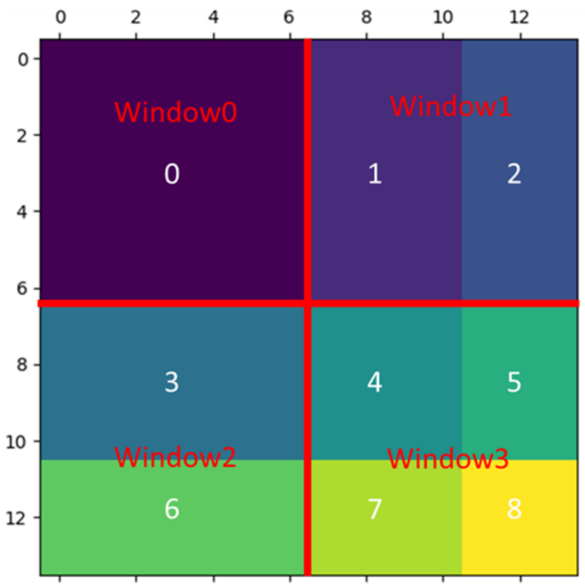
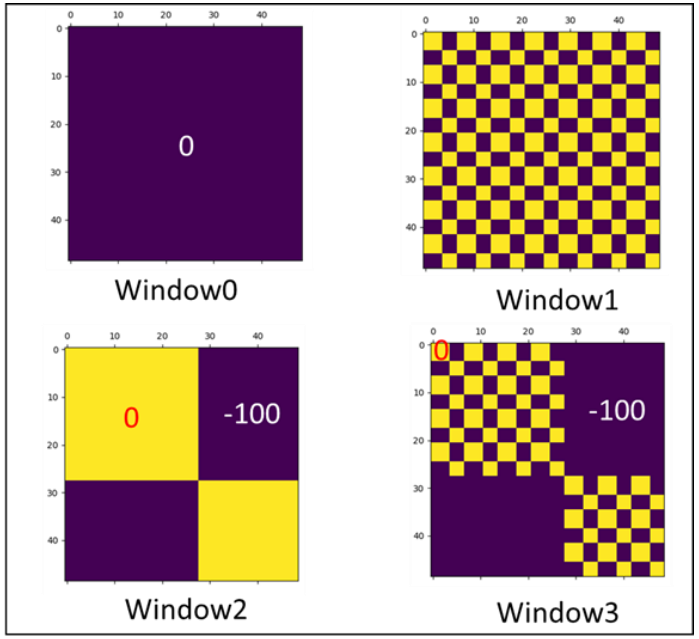


Image Mask
(14x14, window 7x7, shift 3)



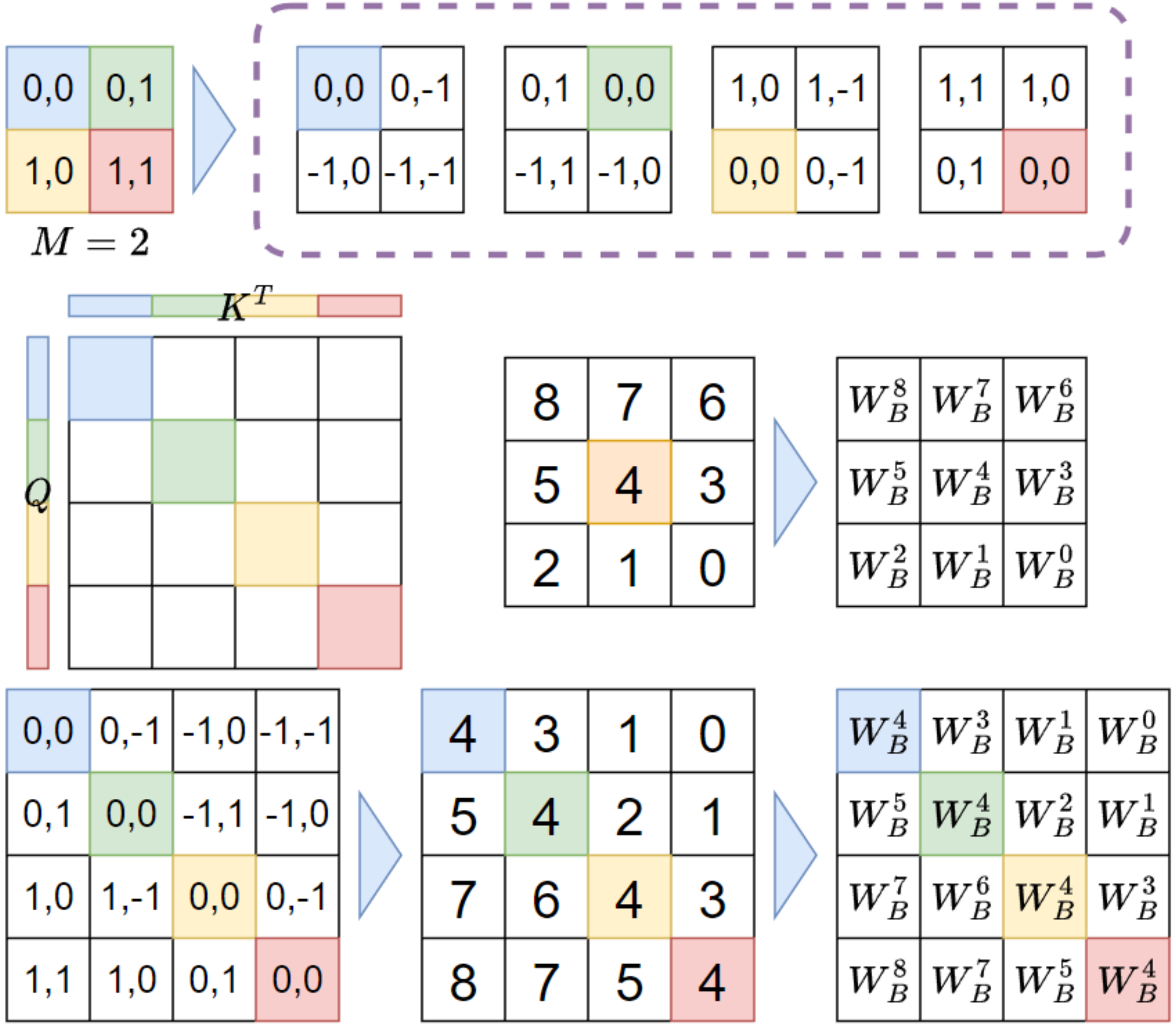
Attn Mask

$$\text{Attention}(Q,K,V)=\text{Softmax}\left(\frac{QK^T}{\sqrt{d}}+M+B\right)V$$

[The Question about the mask of window attention · Issue #38 · microsoft/Swin-Transformer (github.com)]

□Relative Position Embedding

$$\text{Attention}(Q,K,V)=\text{Softmax}\left(\frac{QK^T}{\sqrt{d}}+M+B\right)V$$

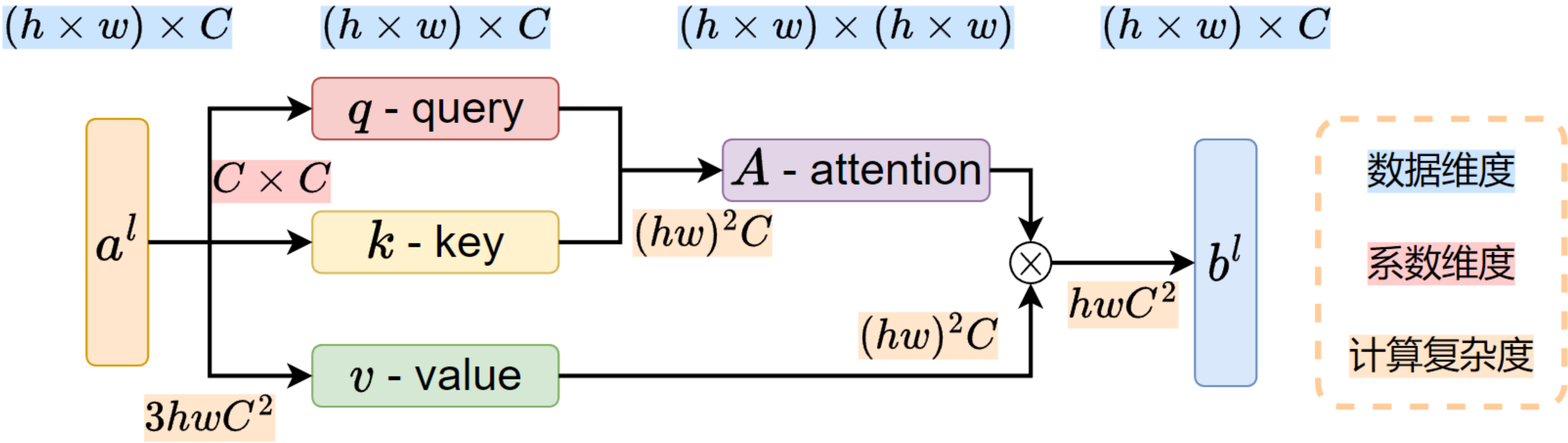


◆ **在$M \times M$的窗口中进行Attention运算**

- QK^T的大小为$M^2 \times M^2$
- 有$(2M-1)^2$种相对位置
- 窗口确定相对位置编码索引
- 依据索引在位置编码Table中查询编码数值构建B
- 具有平移不变性

☐Window Attention的计算复杂度

Multi-Head Self Attention：



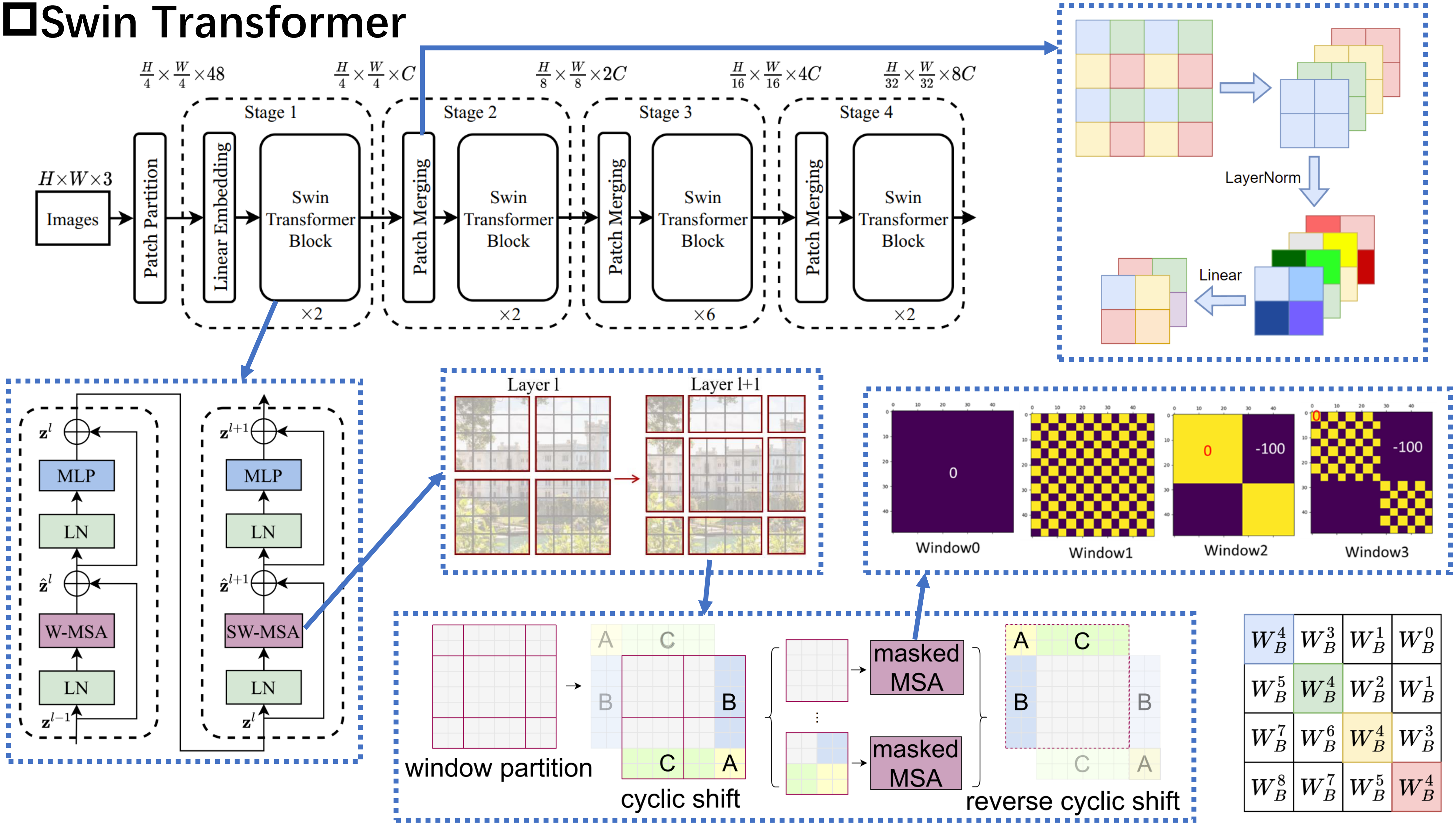
$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2 C$$

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

- h/w – 特征图的高/宽
- C – 通道数，元素的维度
- M – 窗口的边长

$$\Omega(\text{W-MSA}) = (\frac{h}{M} \times \frac{w}{M}) \times [4M^2C^2 + 2(M^2)^2C] = 4hwC^2 + 2M^2hwC$$

Swin Transformer



□Experiments - Image Classification On ImageNet

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

(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

(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~~ 86.4*
Swin-L	384^2	197M	103.9G	42.1	~~86.4~~ 87.3

*： 2021年8月更新

□Experiments – Object Detection On COCO

(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	-

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

☐Experiments – Semantic Segmentation on ADE20K

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. $\ddagger$ indicates that the model is pre-trained on ImageNet-22K.

☐Experiments – Ablation Study

	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).