Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

패턴인식

202132032 김형범

2022. 09. 19.

Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

CONTENTS

Transformer 주차별 계획

1 2022.09.19. Transformer 개념 설명

2 2022.09.26. ViT Image classification

3 2022.10.03. BERT 개념 설명

4 2022.10.10. Swin transformer Image classification

5 2022.10.17. DeiT Image classification

6 2022.10.24. MobielViT Image classification

CONTENTS

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

IV

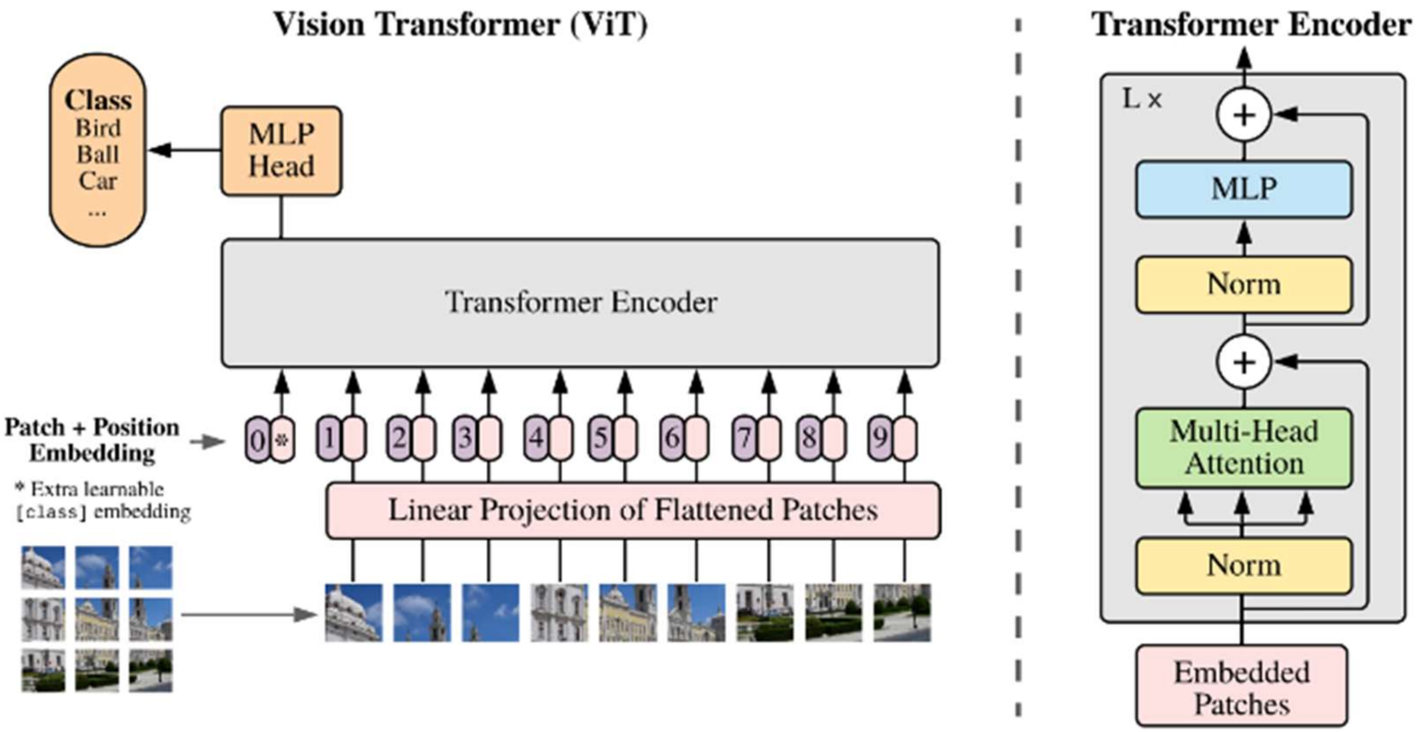
Transformer

1 _ Introduction + contributions

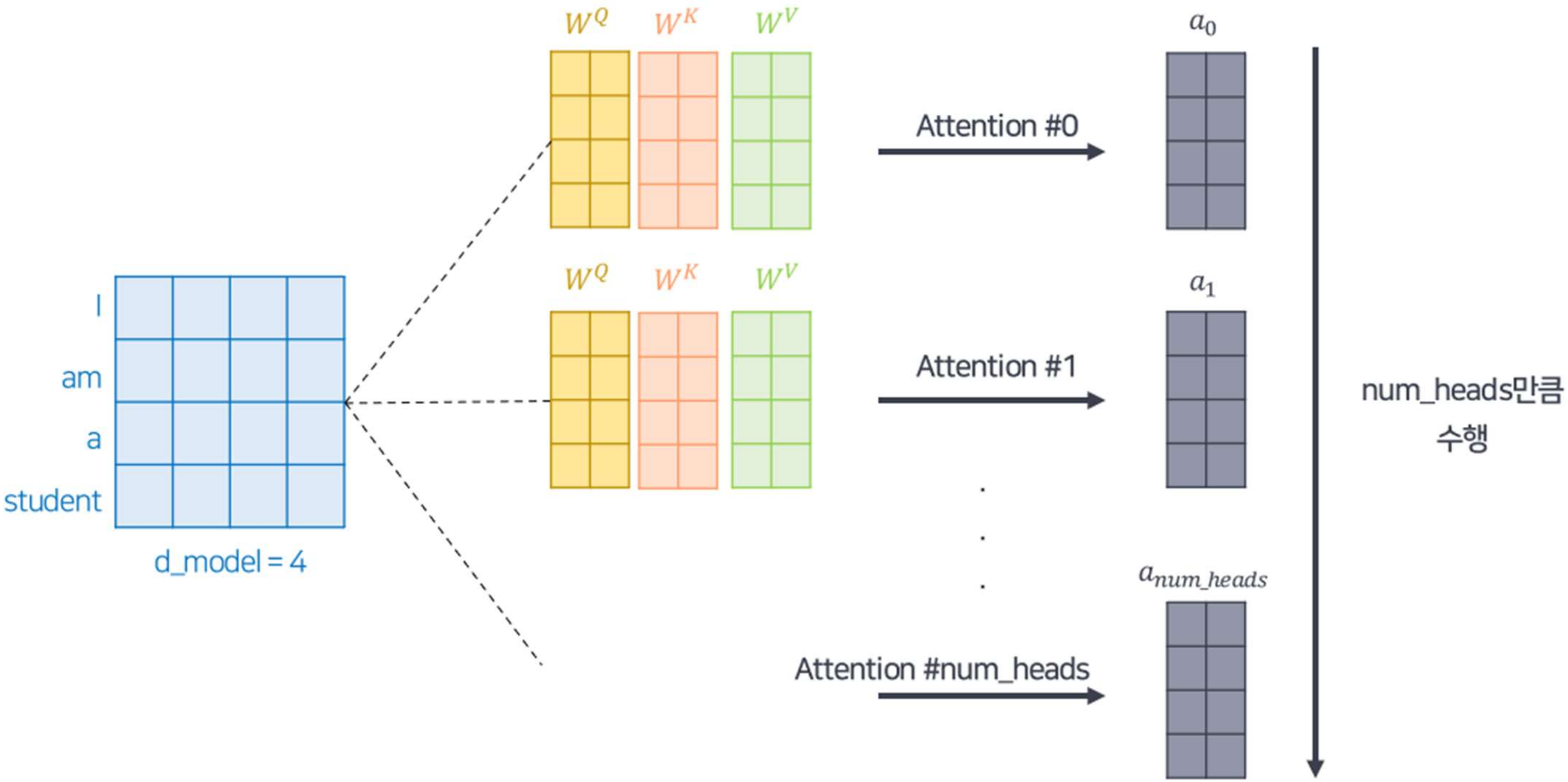
1. ViT to Swin

2. contributions

Introduction + contributions
ViT to Swin



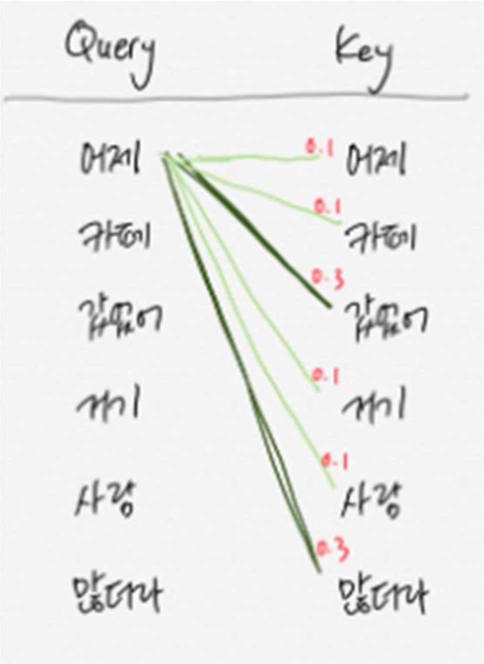
ViT to Swin

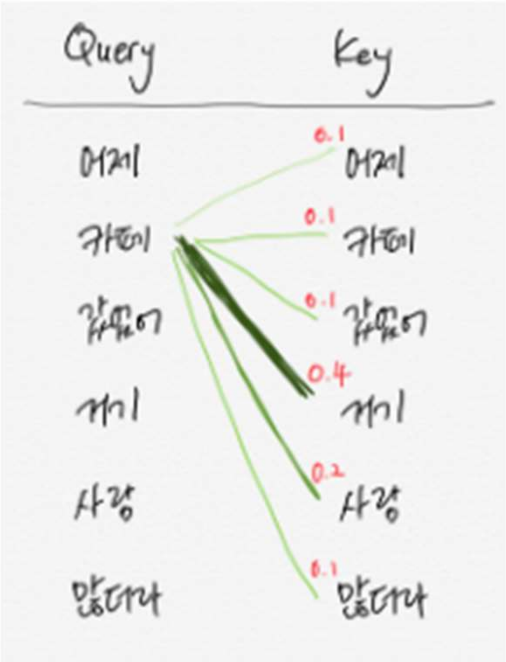


W^Q W^K W^V
W^Q W^K W^V
I
am
a
student
d_model = 4
Attention #0
Attention #1
Attention #num_heads
a_0
a_1
a_{num_heads}
num_heads만큼
수행

Introduction + contributions

ViT to Swin





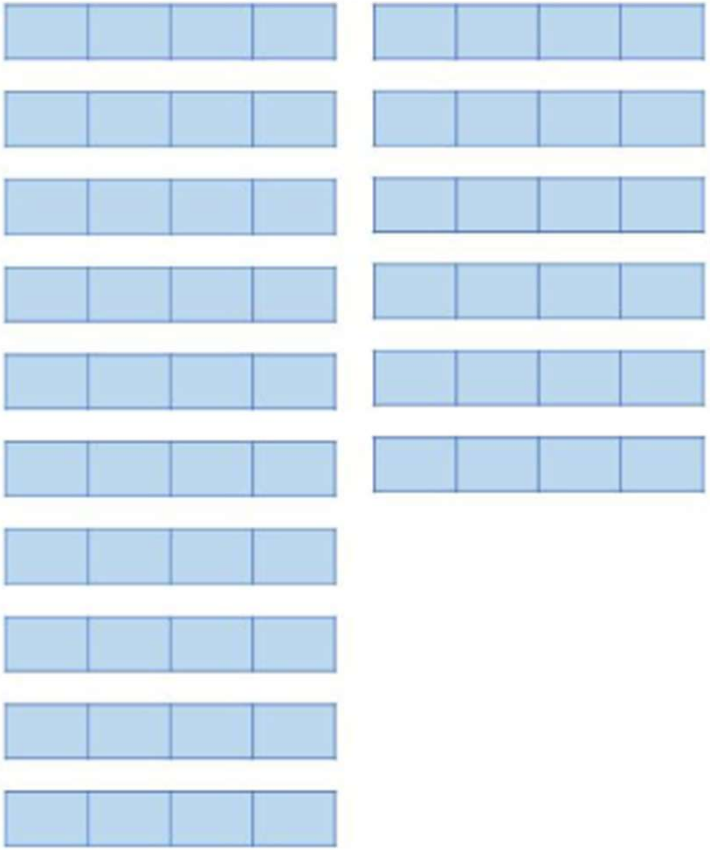
Introduction + contributions
ViT to Swin

8 x 8 px

2 x 2
patch

16 tokens

16 x16
Self-attention



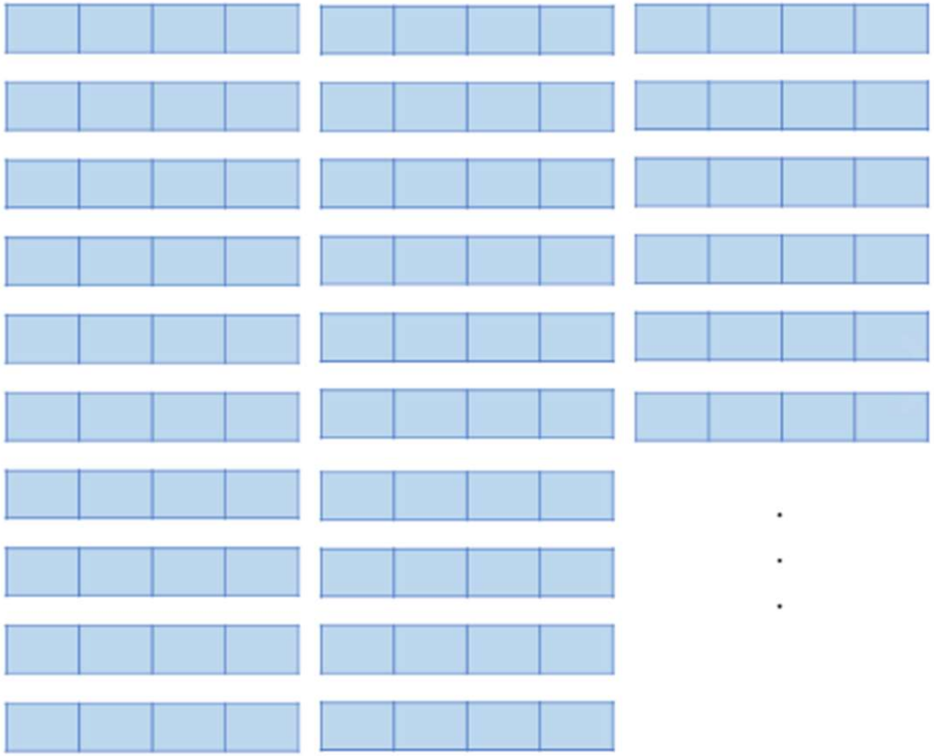
good slide

Introduction + contributions
ViT to Swin

(8 x N) x (8 x N) px

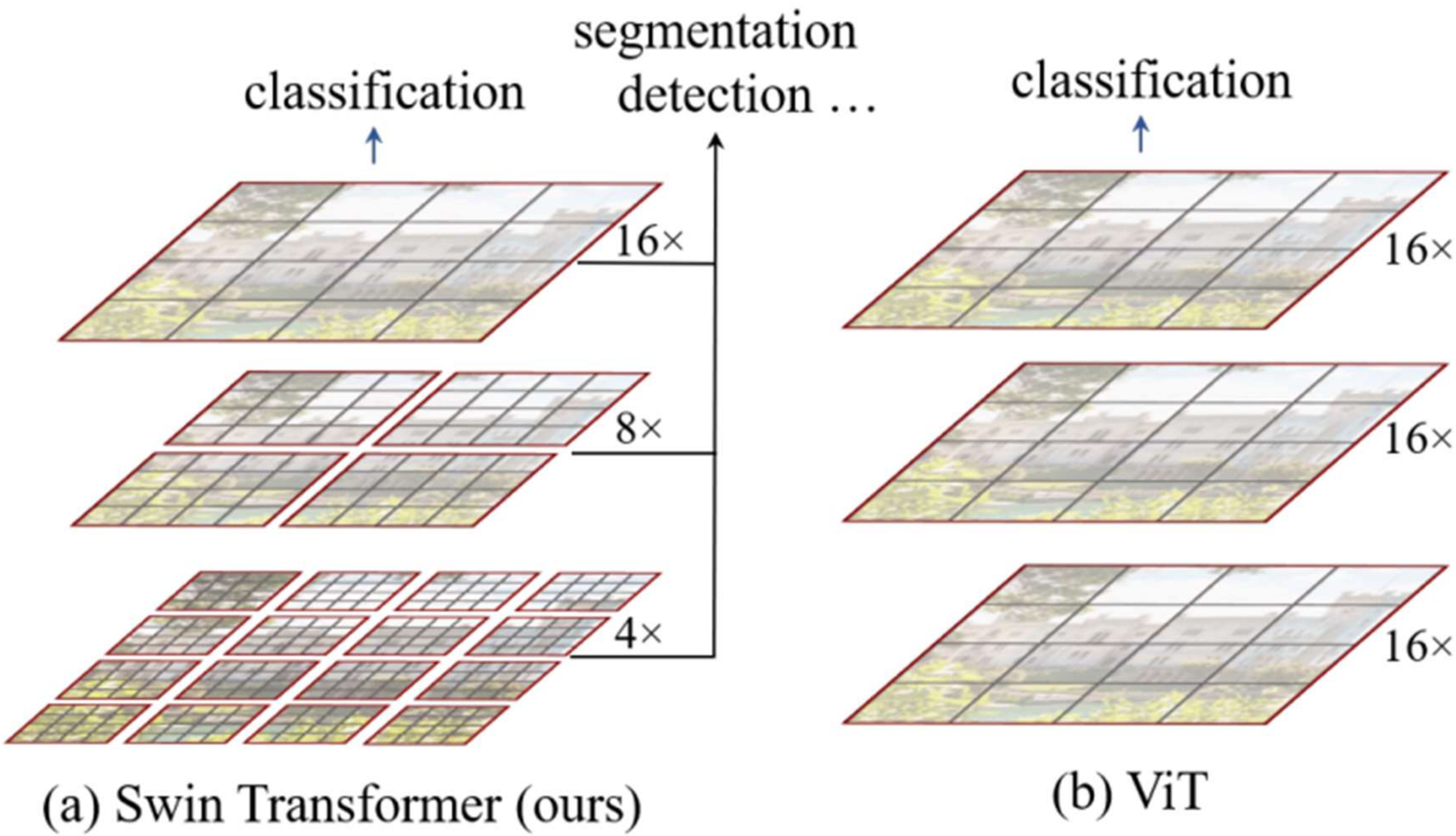
2 x 2
patch

16 x N^2
tokens



($16 \times N^2$) x ($16 \times N^2$)
Self-attention

Introduction + contributions
ViT to Swin



good slide

Contribution

1. NLP 분야에서 사용하는 Transformer는 Vision분야에서 활용하기 어렵다.
→ 일정한 수의 pixel을 patch라고 정의하고 이 patch를 token처럼 최소 처리 단위로 정의하여 해결

2. 영상은 텍스트에 비해 high-resolution이고 Vit의 연산량은 영상의 사이즈에 quadratic하게 증가
→ 계층적인 feature map을 이용하고 feature map의 window 내에서만 self-attention을 적용.

3. Window 내에서만 self-attention을 적용할 경우 윈도우 내에 속한 patch 간의 연관성만 고려할 수 있고 다른 윈도우에 속한 patch간의 연관성을 파악할 수 없다.
→ Shifted window를 도입하여 해결

4. Vision problem에서 Vit보다 뛰어난 성능을 보여준다.

Transformer

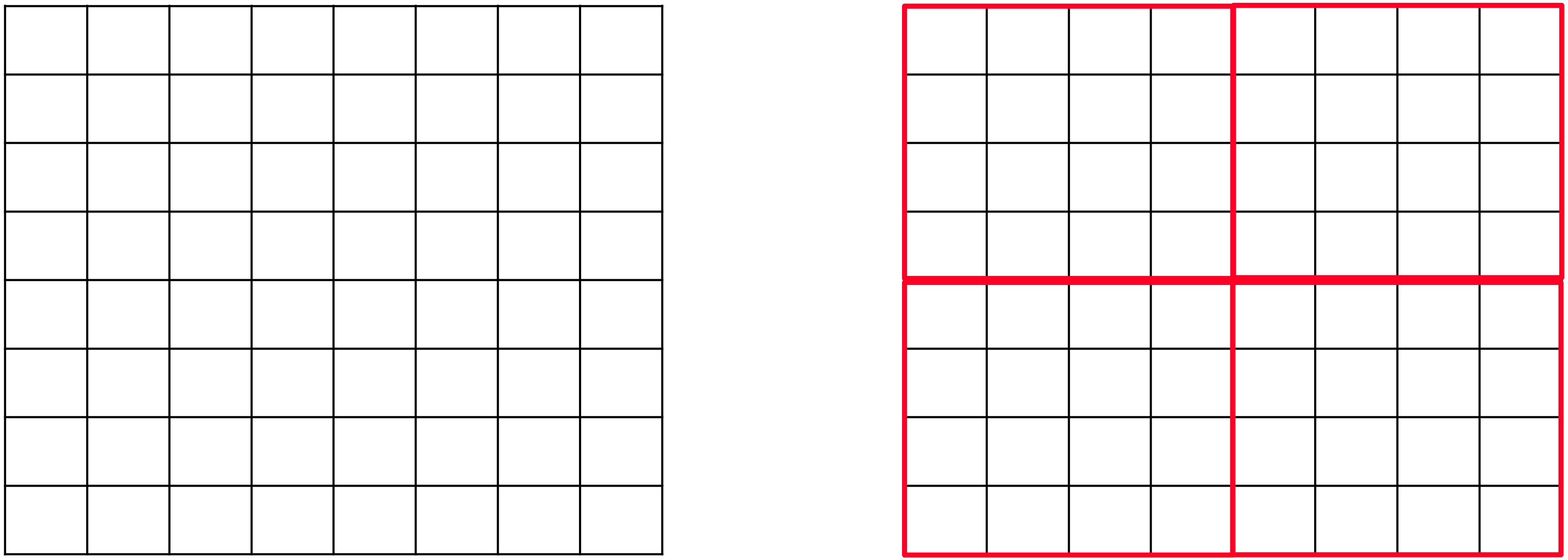
2 _ Method

1. Window

2. Shifted Window

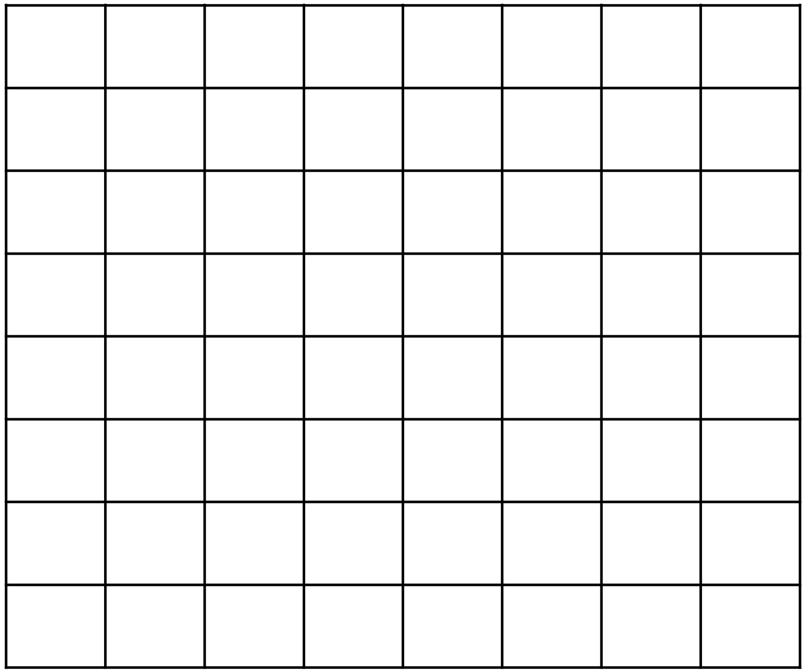
3. Overall Architecture

Method
Window

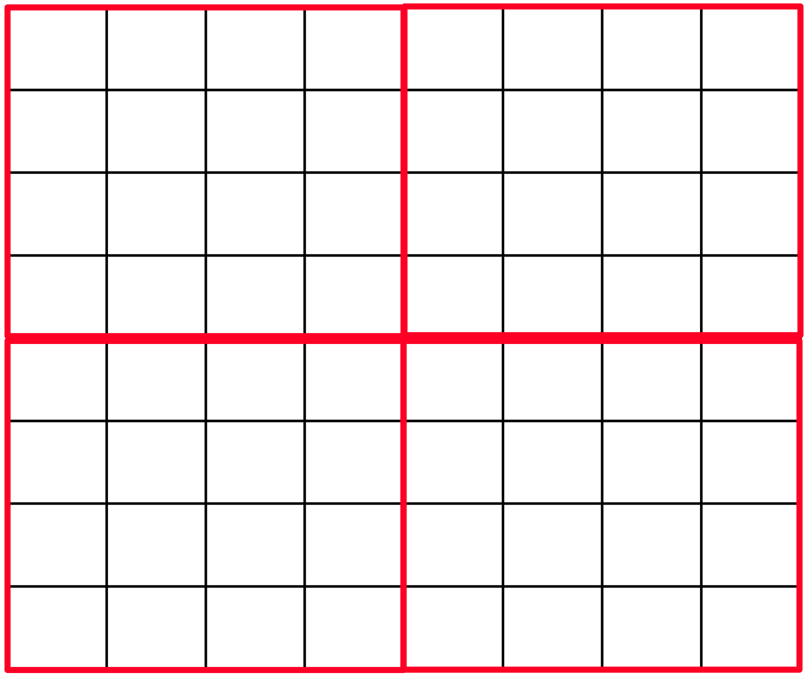


good slide

Method
Window



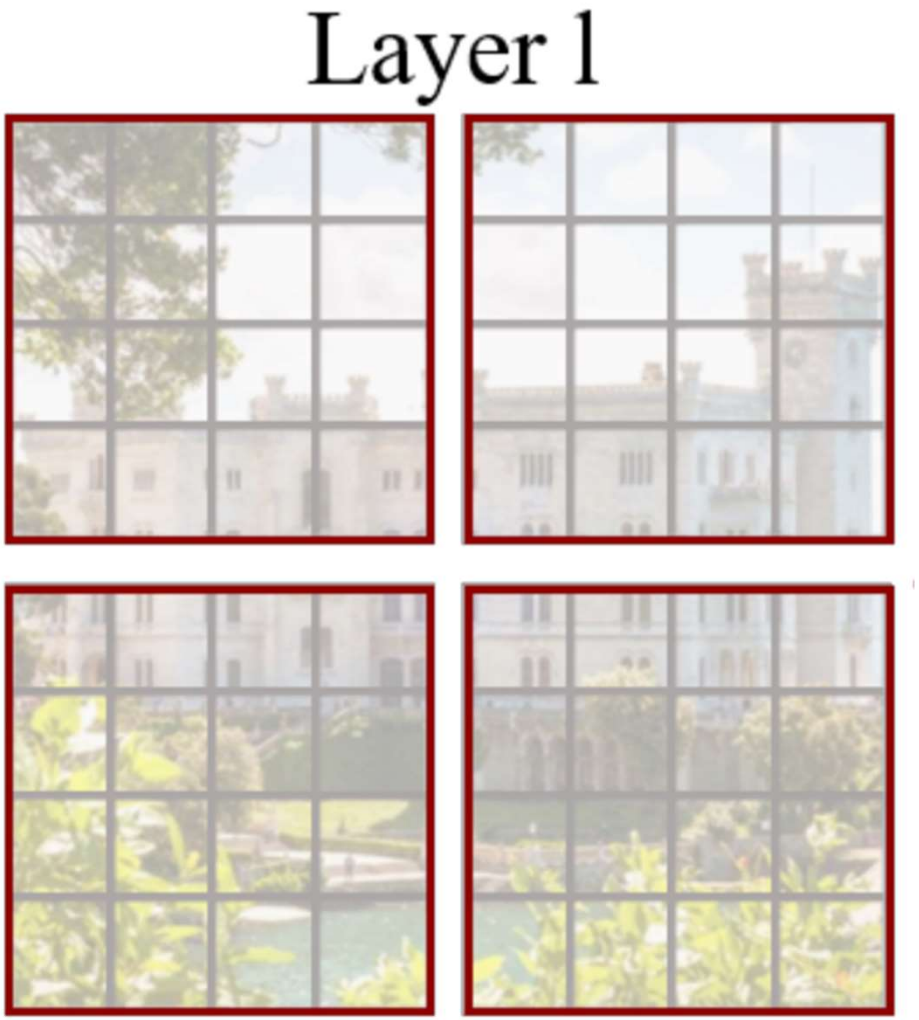
64 x 64 = 4096
Self Attention



16 x 16 x 4 = 1024
Self Attention

CHAPTER I _

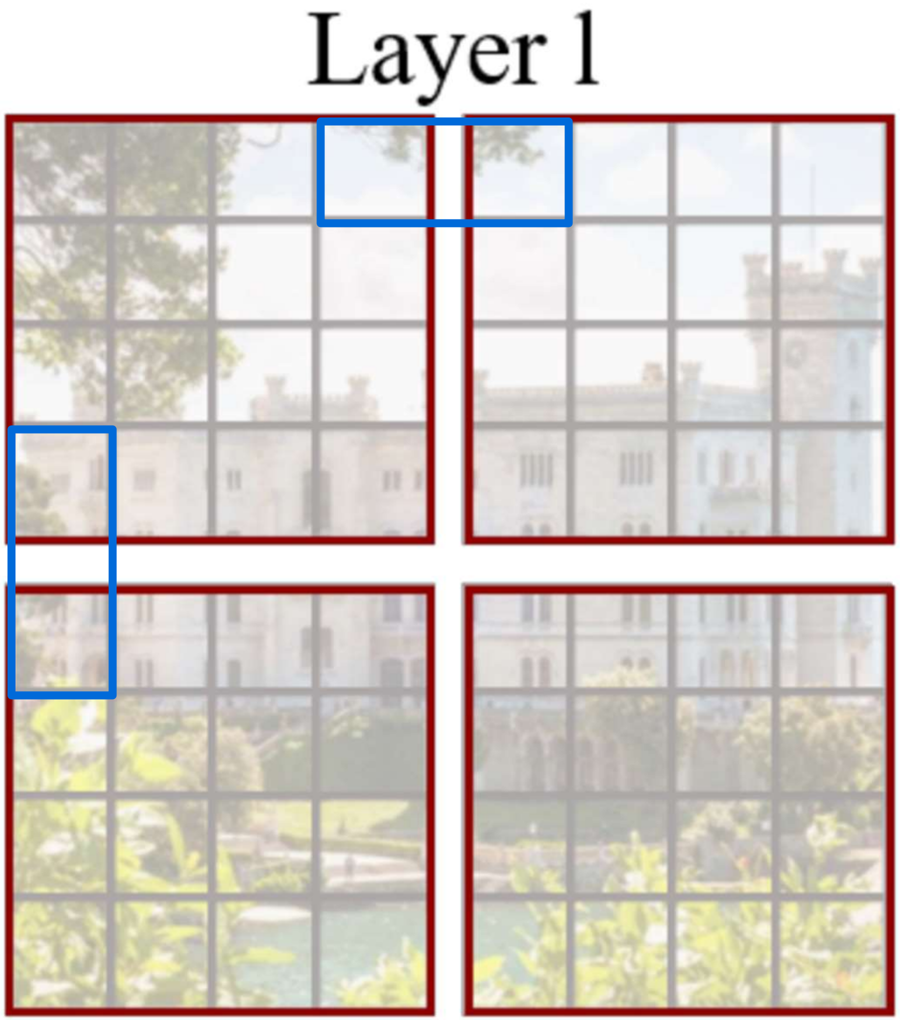
Method
Window



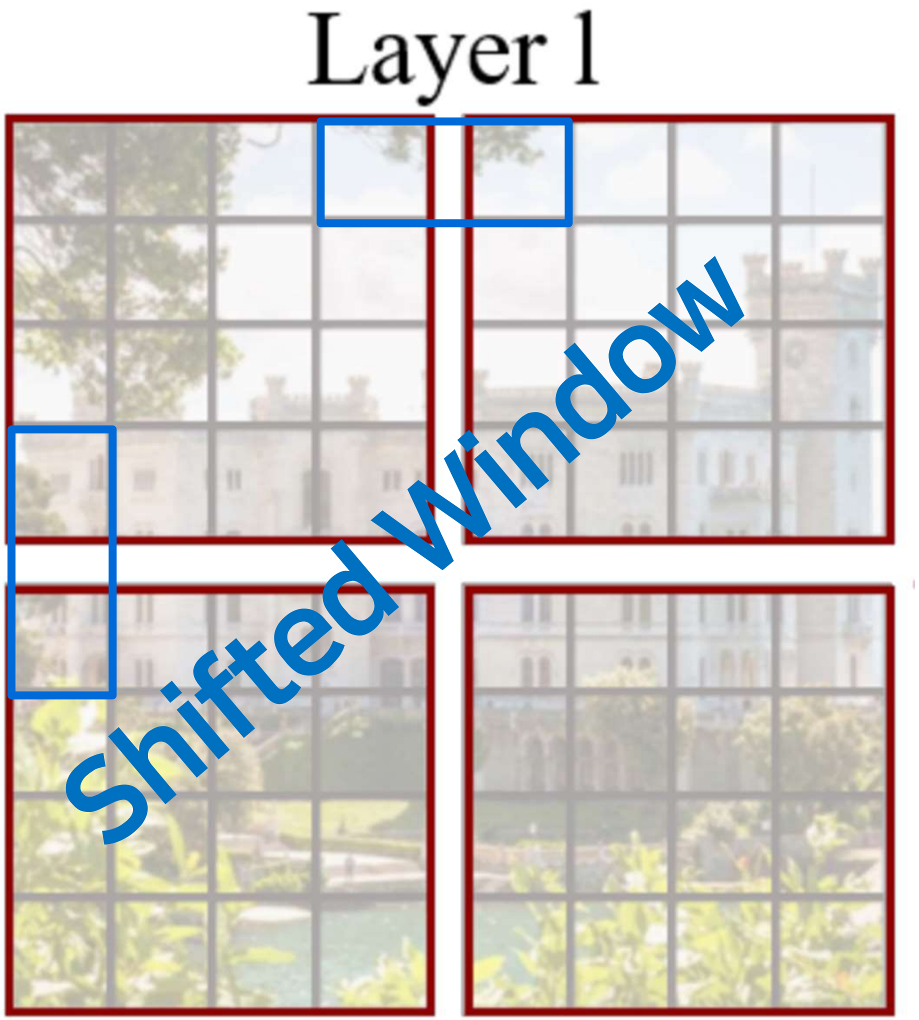
Transformer

good slide

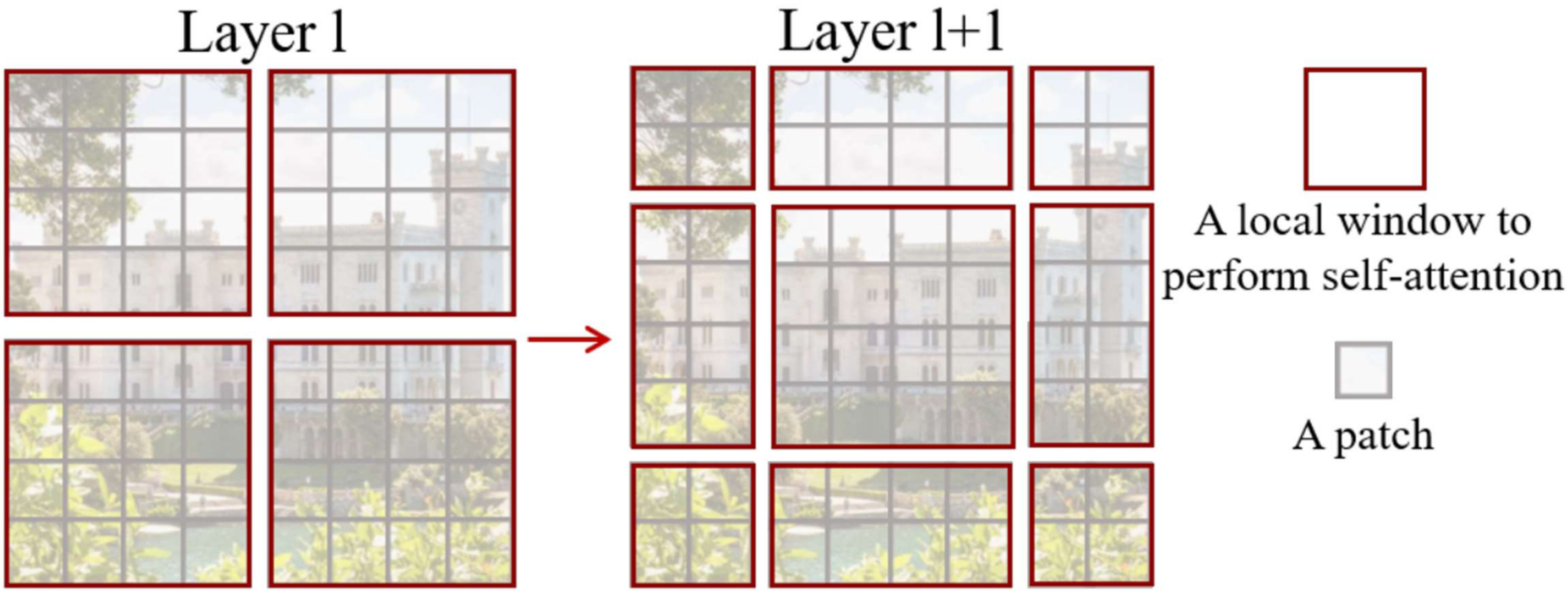
Window



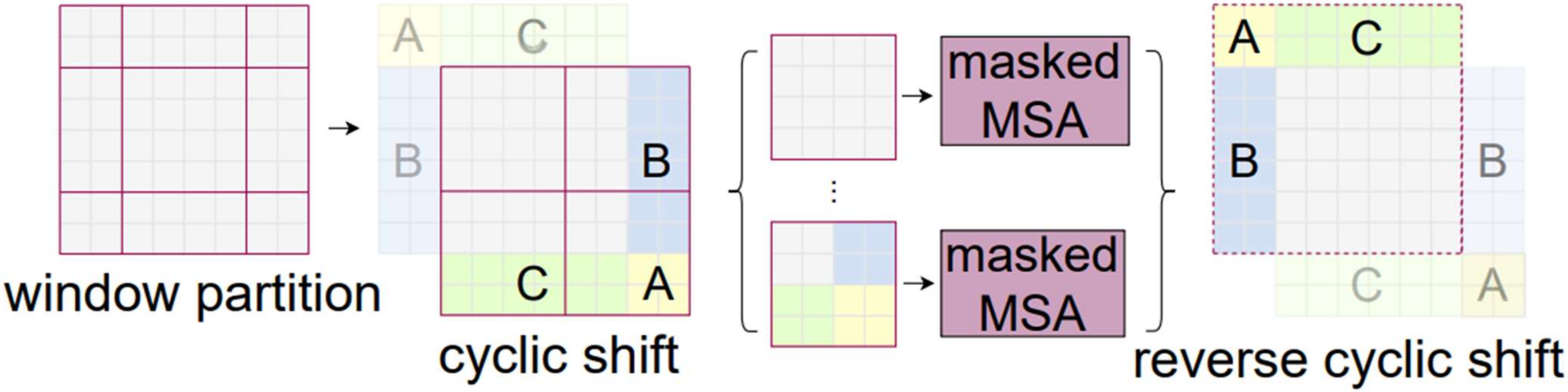
Window



Method
Shifted Window



Method
Shifted Window



Overall Architecture

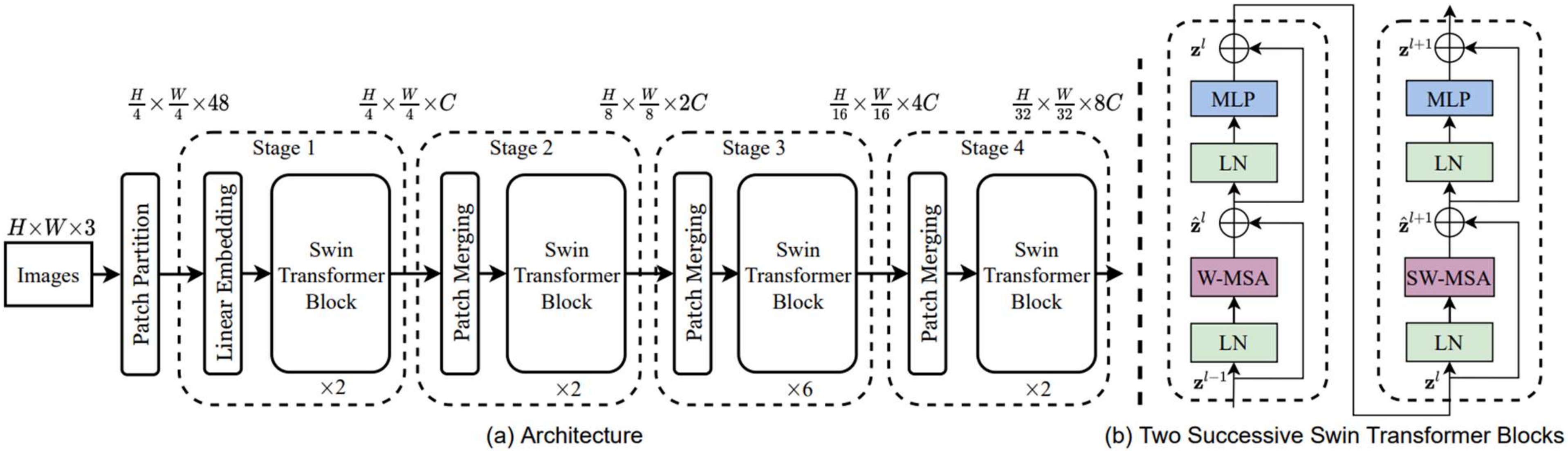
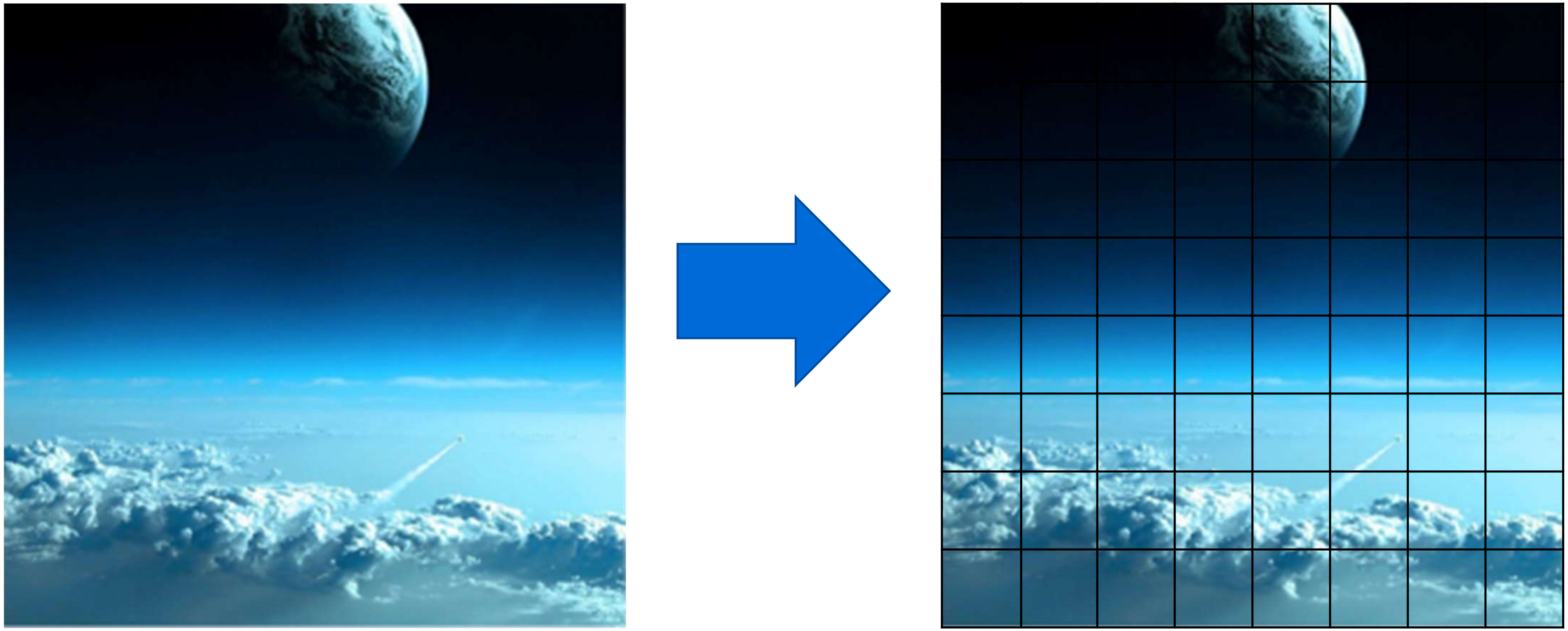


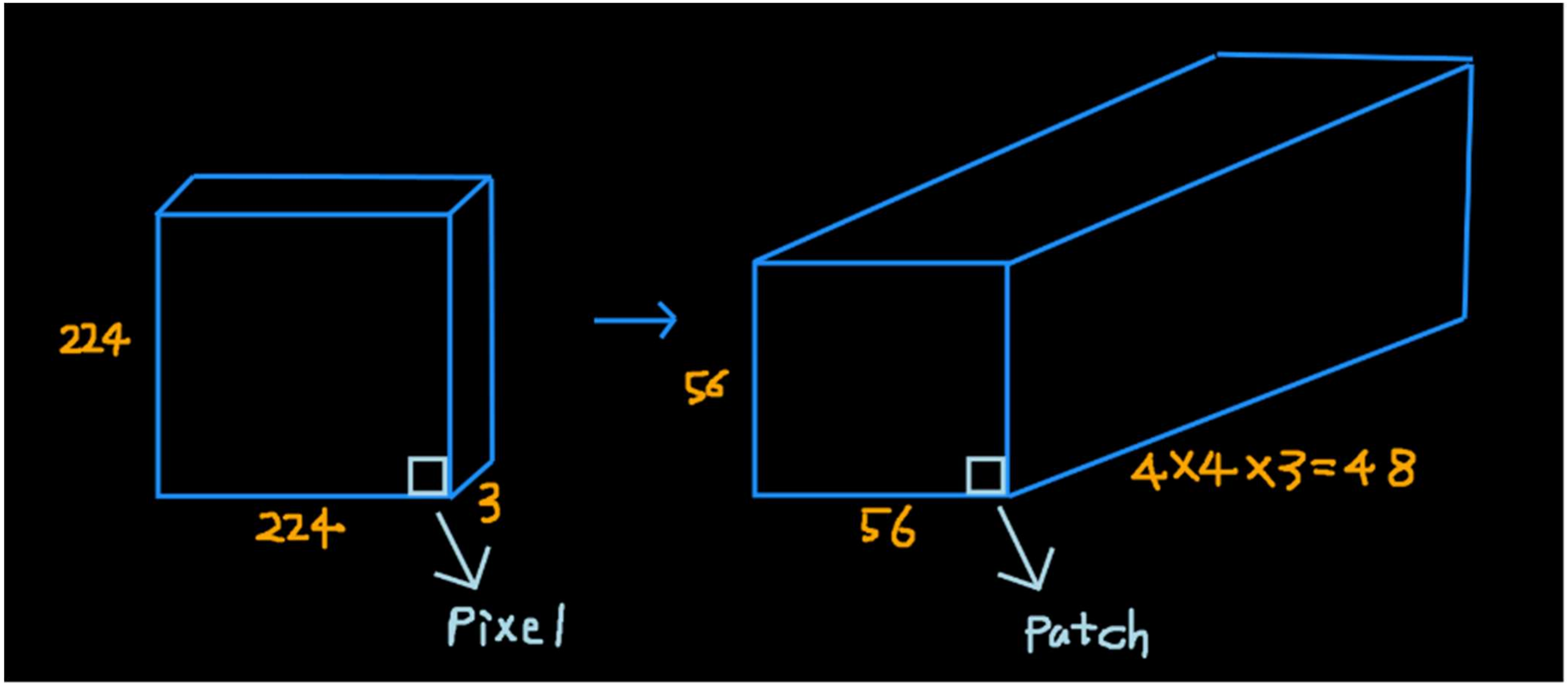
Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Method
Overall Architecture

Patch Partition

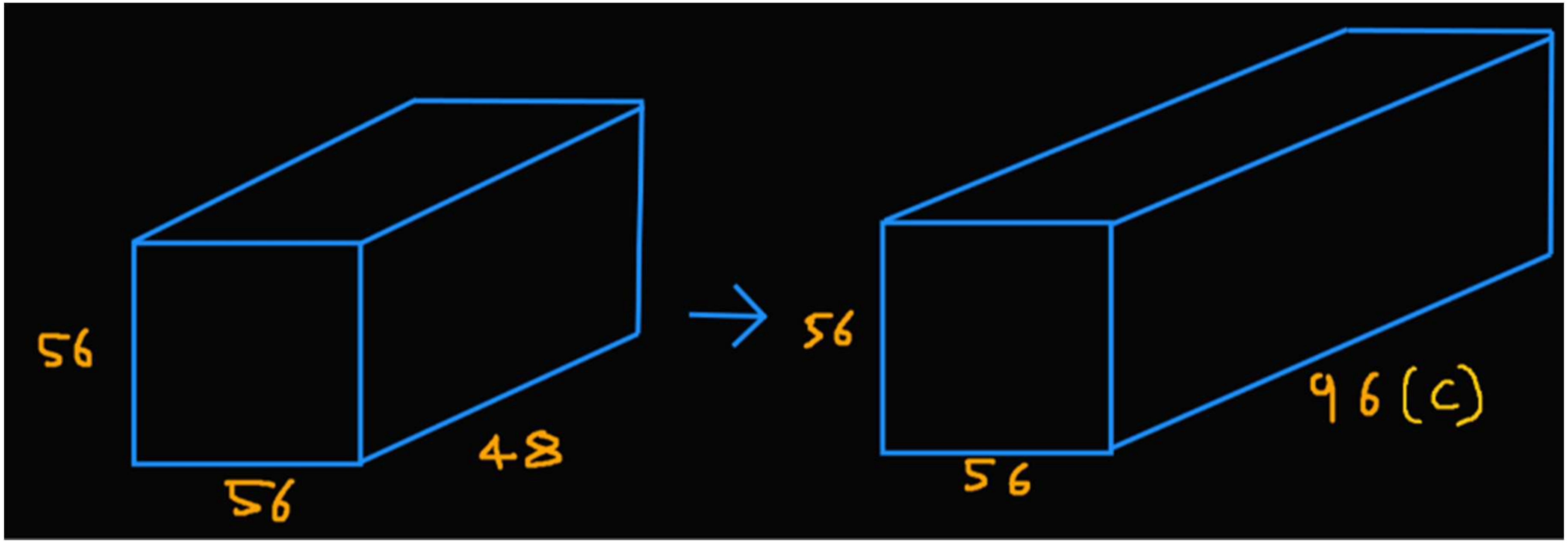


Patch Partition

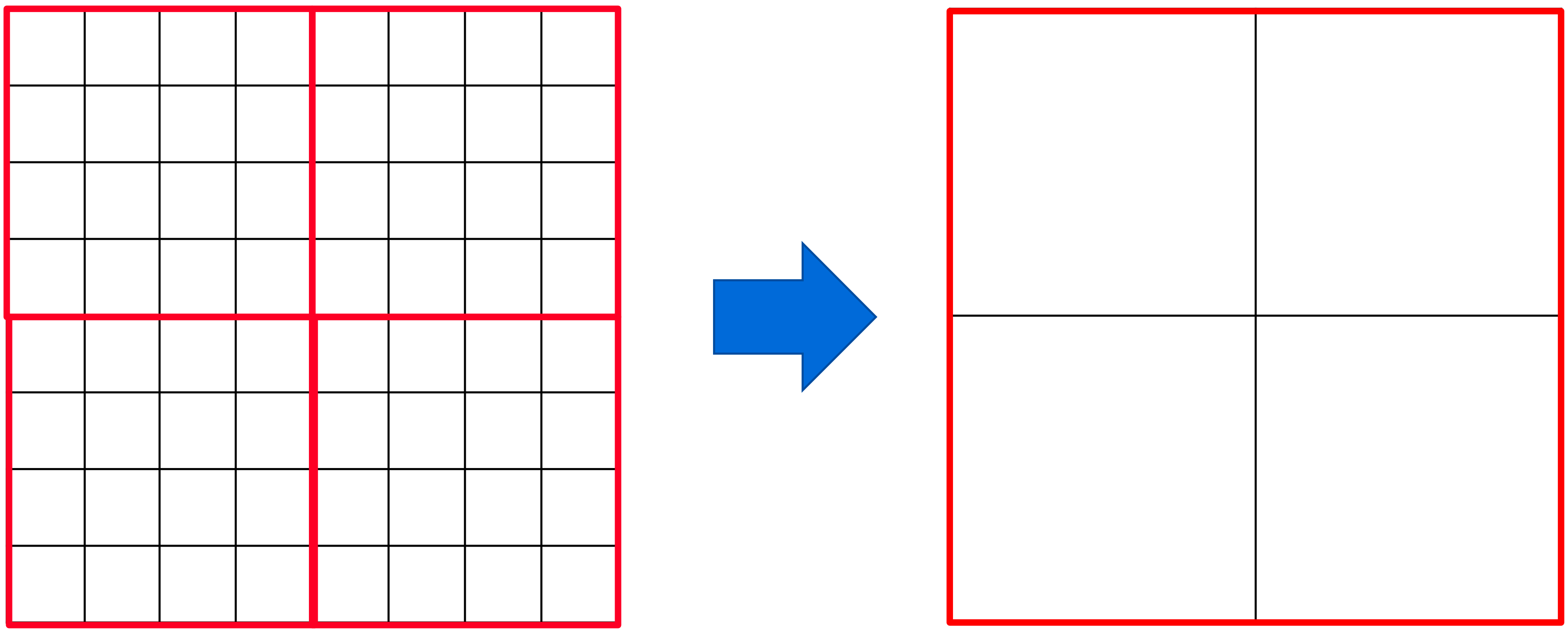


Overall Architecture

Linear embedding

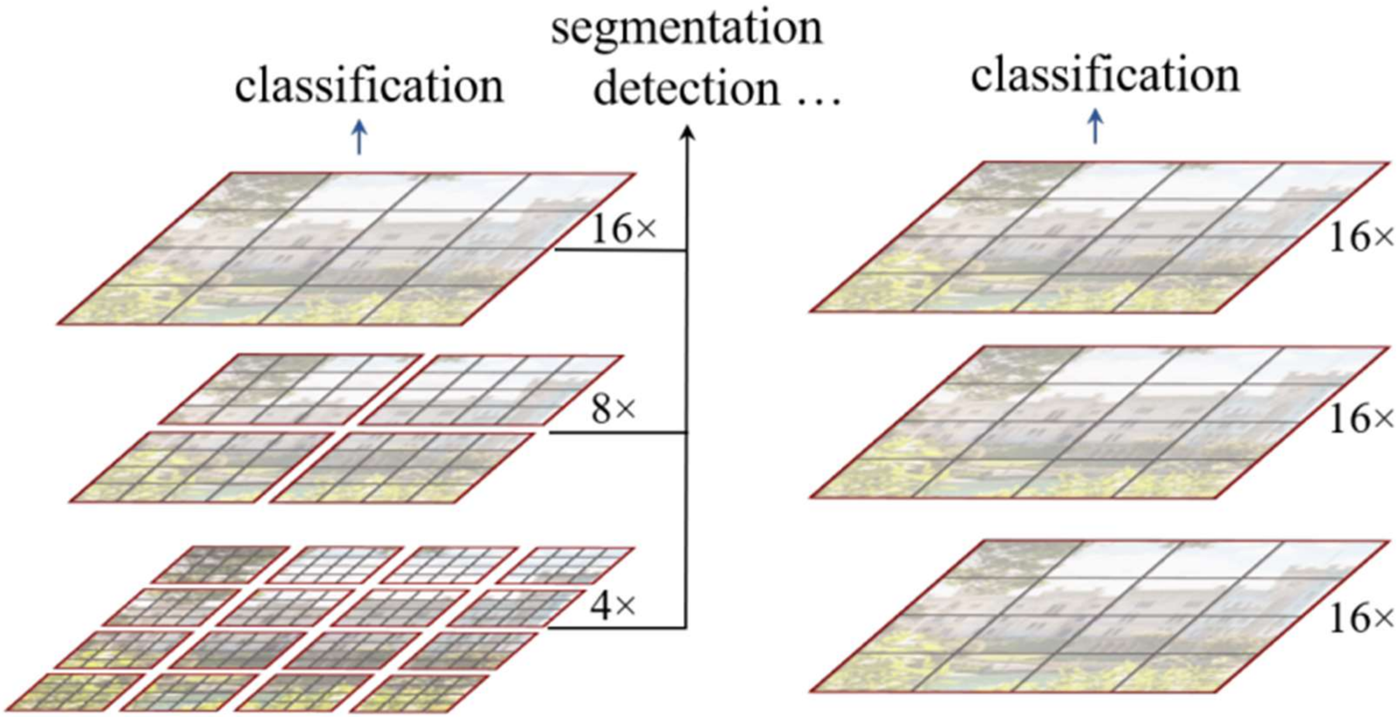


Patch Merging



Overall Architecture

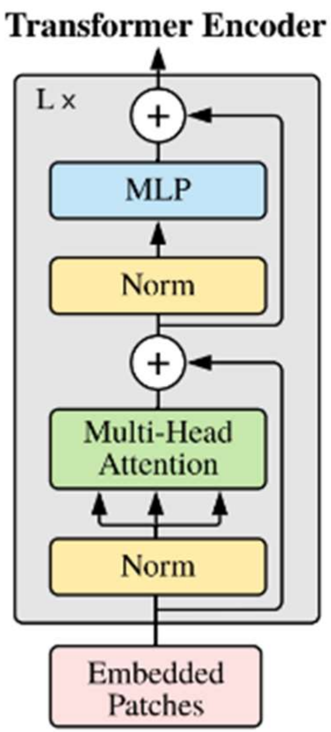
Patch Merging



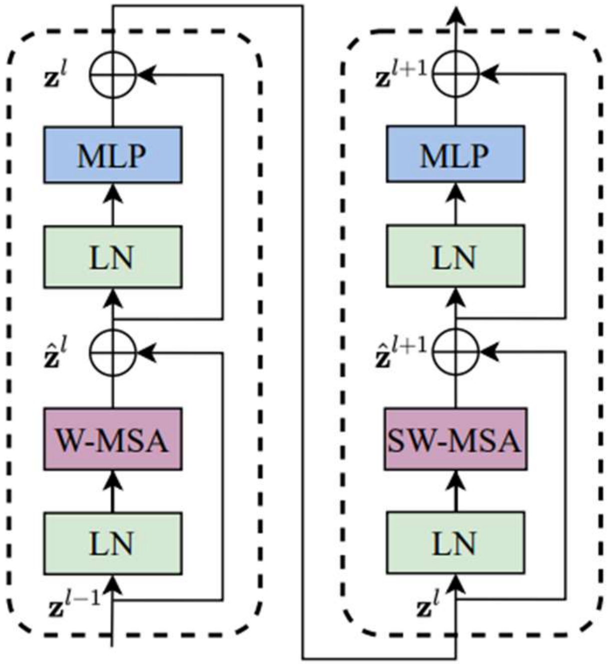
(a) Swin Transformer (ours)

(b) ViT

Swin transformer block



ViT transformer encoder



Swin transformer block

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

3 _ Experiments

1. Model variants

2. Experiments results

Method
Model variants

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

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 96\text{, head } 3\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 96\text{, head } 3\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 128\text{, head } 4\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 192\text{, head } 6\end{bmatrix} \times 2$
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN	concat 2×2, 384-d , LN
		$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 192\text{, head } 6\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 192\text{, head } 6\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 256\text{, head } 8\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 384\text{, head } 12\end{bmatrix} \times 2$
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN	concat 2×2, 768-d , LN
		$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 384\text{, head } 12\end{bmatrix} \times 6$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 384\text{, head } 12\end{bmatrix} \times 18$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 512\text{, head } 16\end{bmatrix} \times 18$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 768\text{, head } 24\end{bmatrix} \times 18$
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN	concat 2×2, 1536-d , LN
		$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 768\text{, head } 24\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 768\text{, head } 24\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 1024\text{, head } 32\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times7, \\ \text{dim } 1536\text{, head } 48\end{bmatrix} \times 2$

Experiments results

(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

Experiments results

(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

Method
Experiments results

Method	mini-val		test-dev		#param.	FLOPs
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}		
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	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.

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.

감사합니다

Thank you

2022. 019 19.

good slide