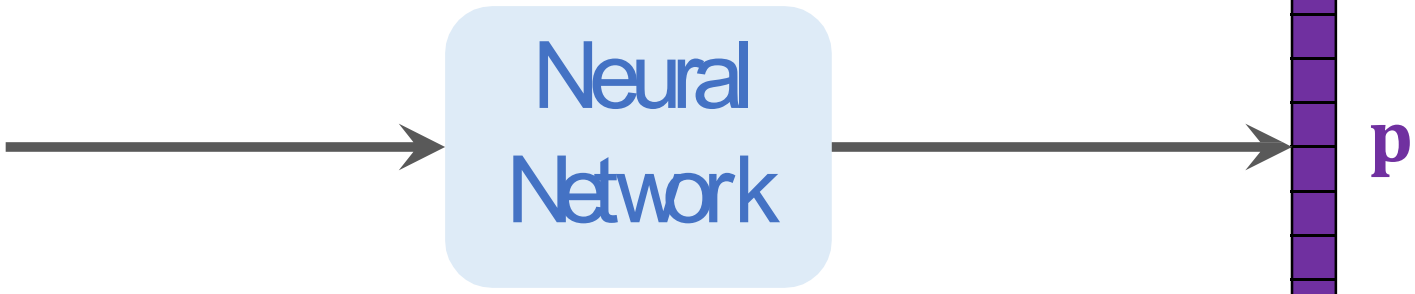
Vision Transformer (ViT)

Shusen Wang

Stevens Institute of Technology

http://wangshusen.github.io/



p

Neural Network

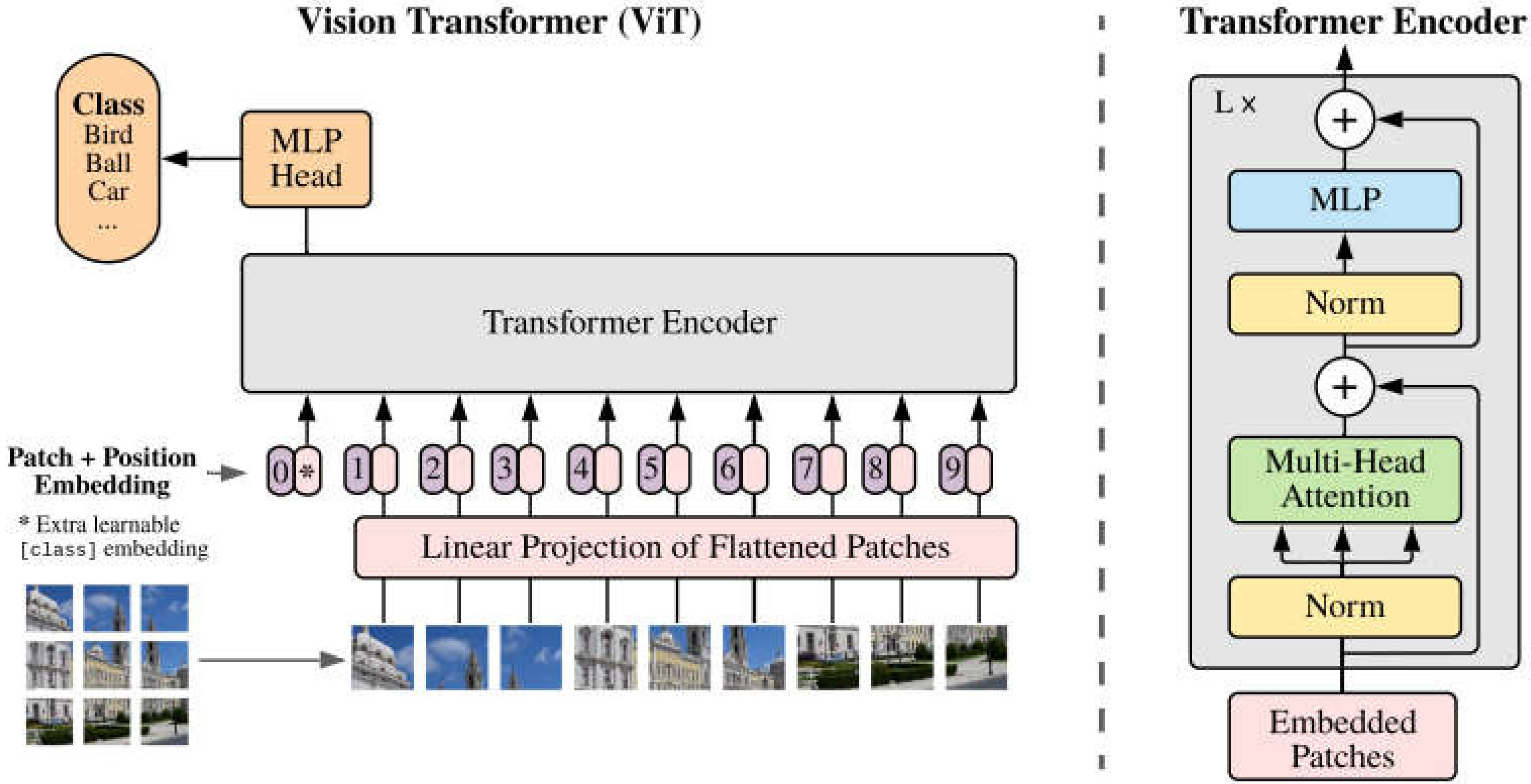
p

Image Classification

- CNNs, e.g., ResNet, were the best solutions to image classification.
- Vision Transformer (ViT) [1] beats CNNs (by a small margin), if the dataset for pretraining is sufficiently large (at least 100 million images).
- ViT is based on Transformer (for NLP) [2].

Reference

1. Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In *ICLR*, 2021.
2. Vaswani et al. Attention Is All You Need. In *NIPS*, 2017.

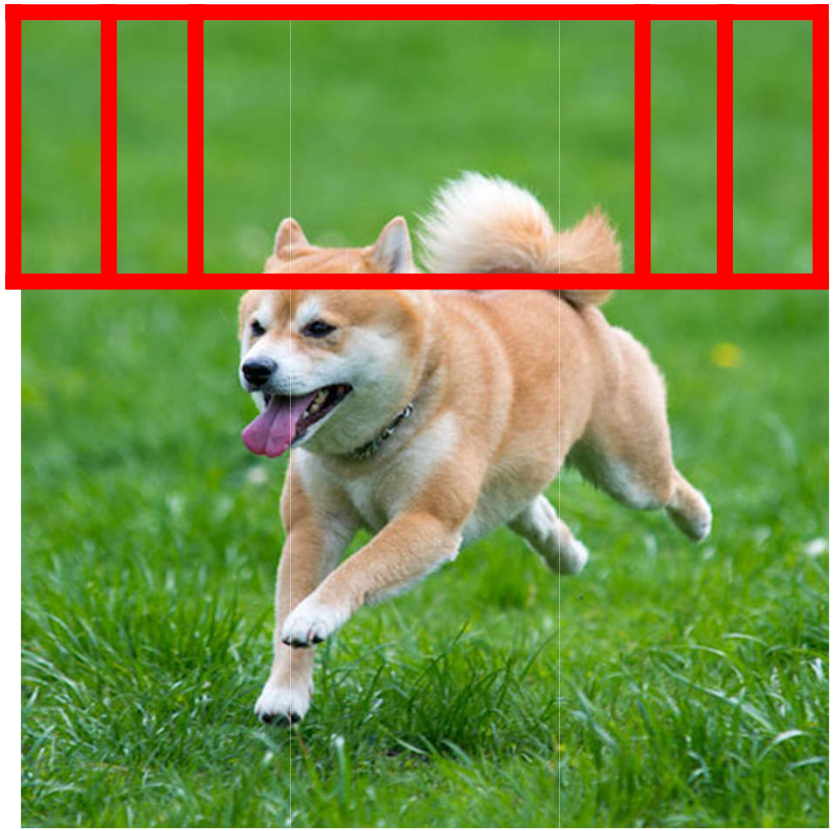

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP
Head
Transformer Encoder
Patch + Position
Embedding
* Extra learnable
[class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head
Attention
Norm
Embedded
Patches

Split Image into Patches

Split Image into Patches

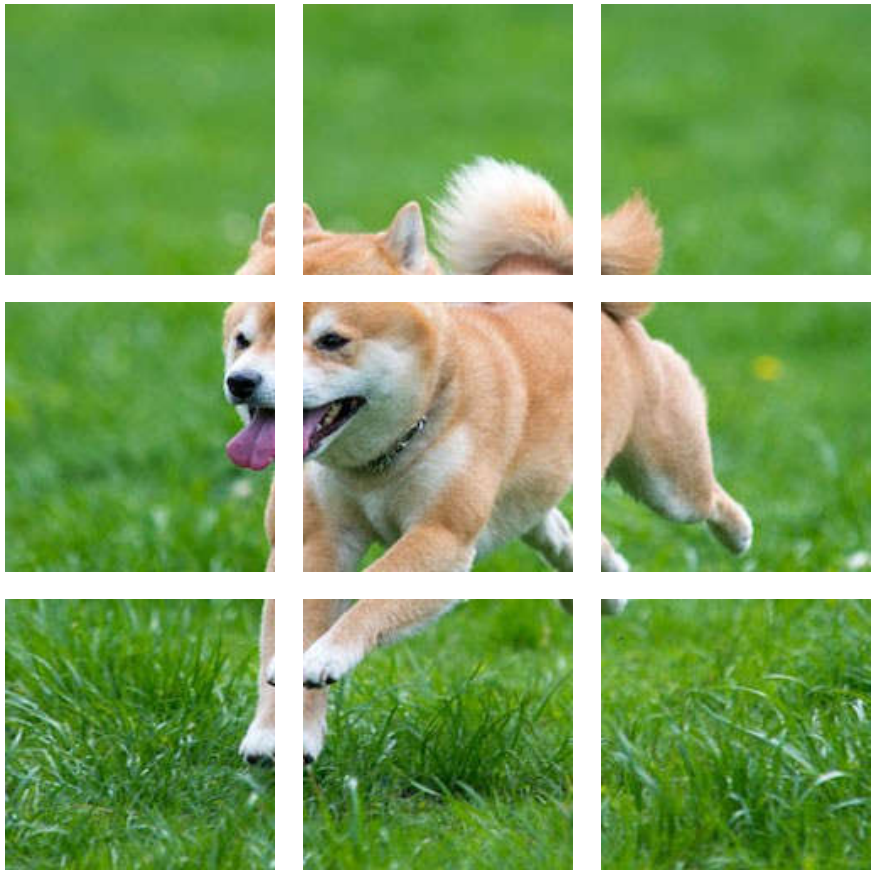
- Here, the patches do not overlap.

Split Image into Patches



- Here, the patches do not overlap.
- The patches can overlap.
- User specifies:
  - patch size, e.g., 16×16;
  - stride, e.g., 16×16.

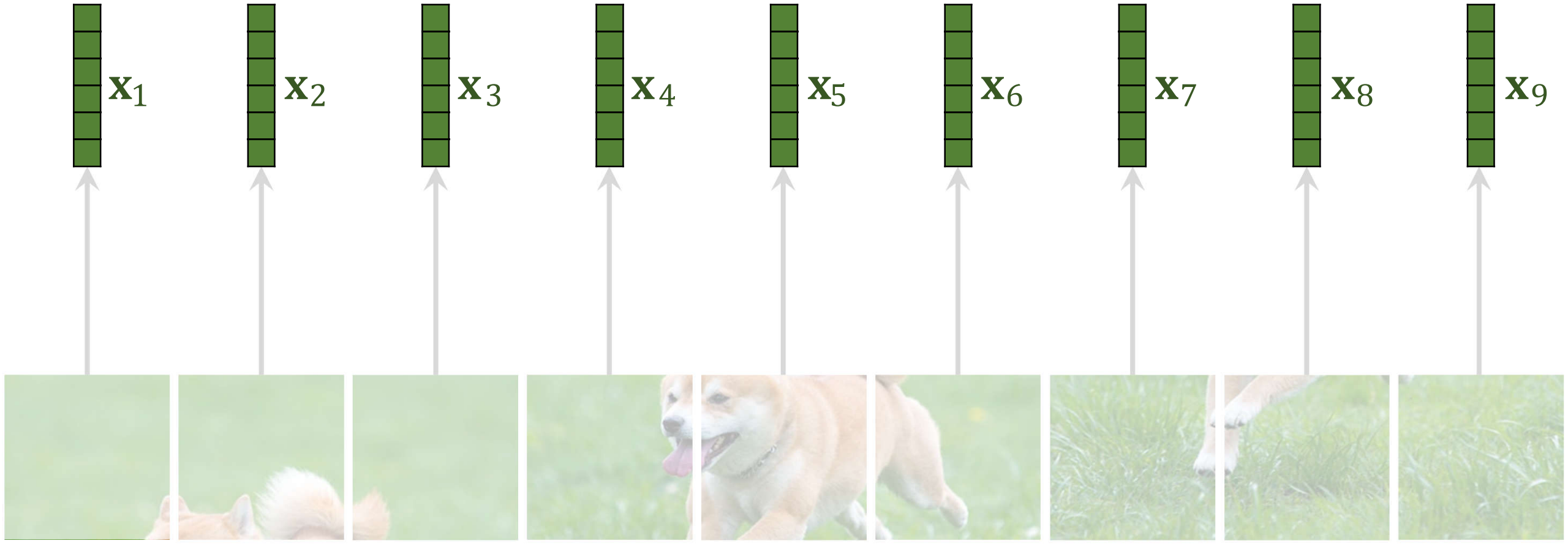
Vectorization

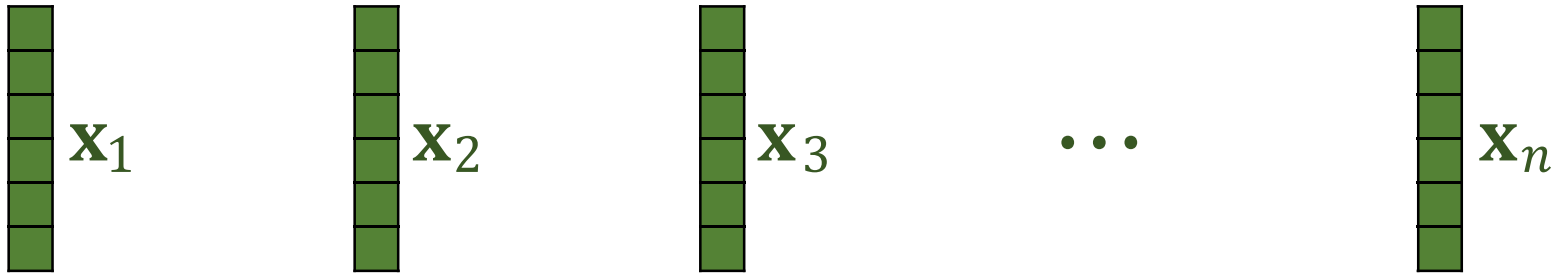


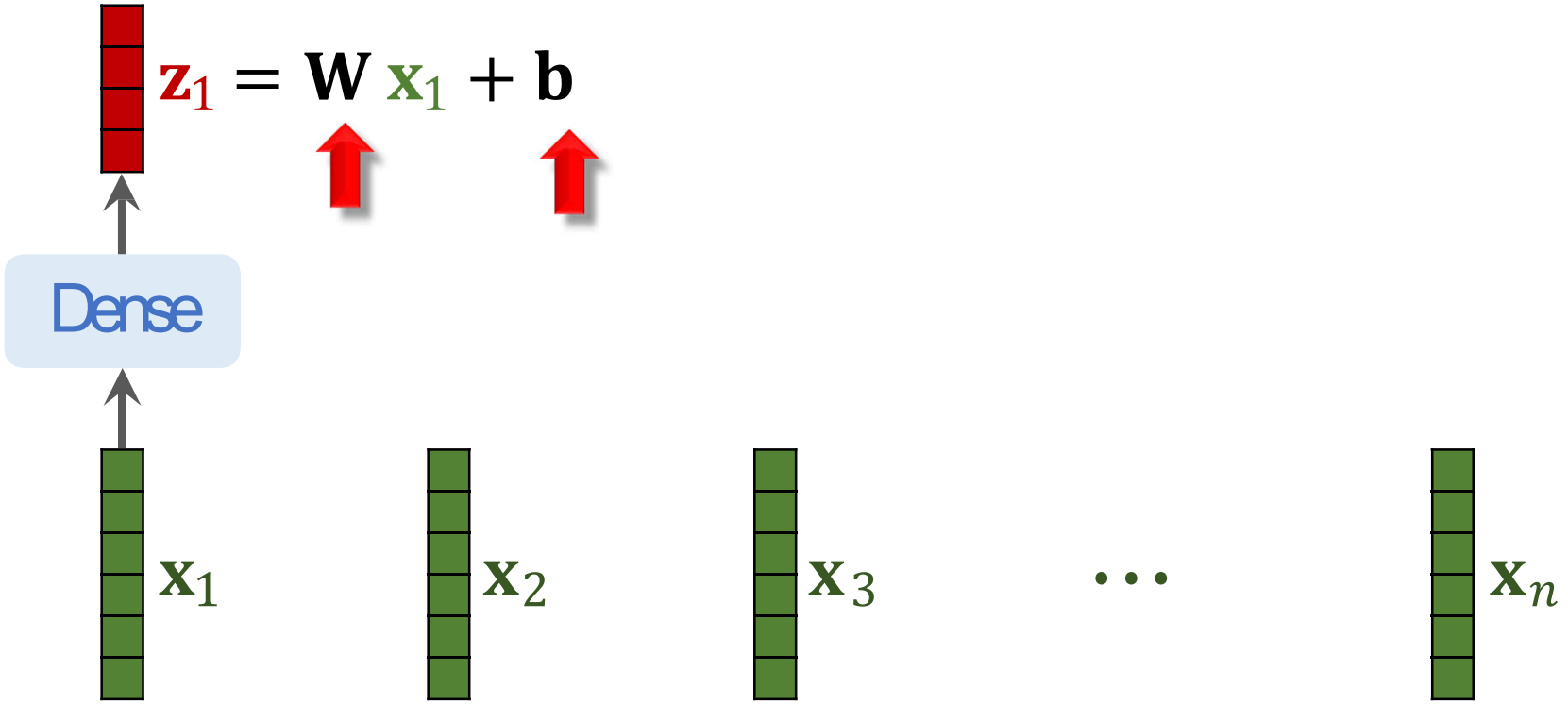
Vectorization

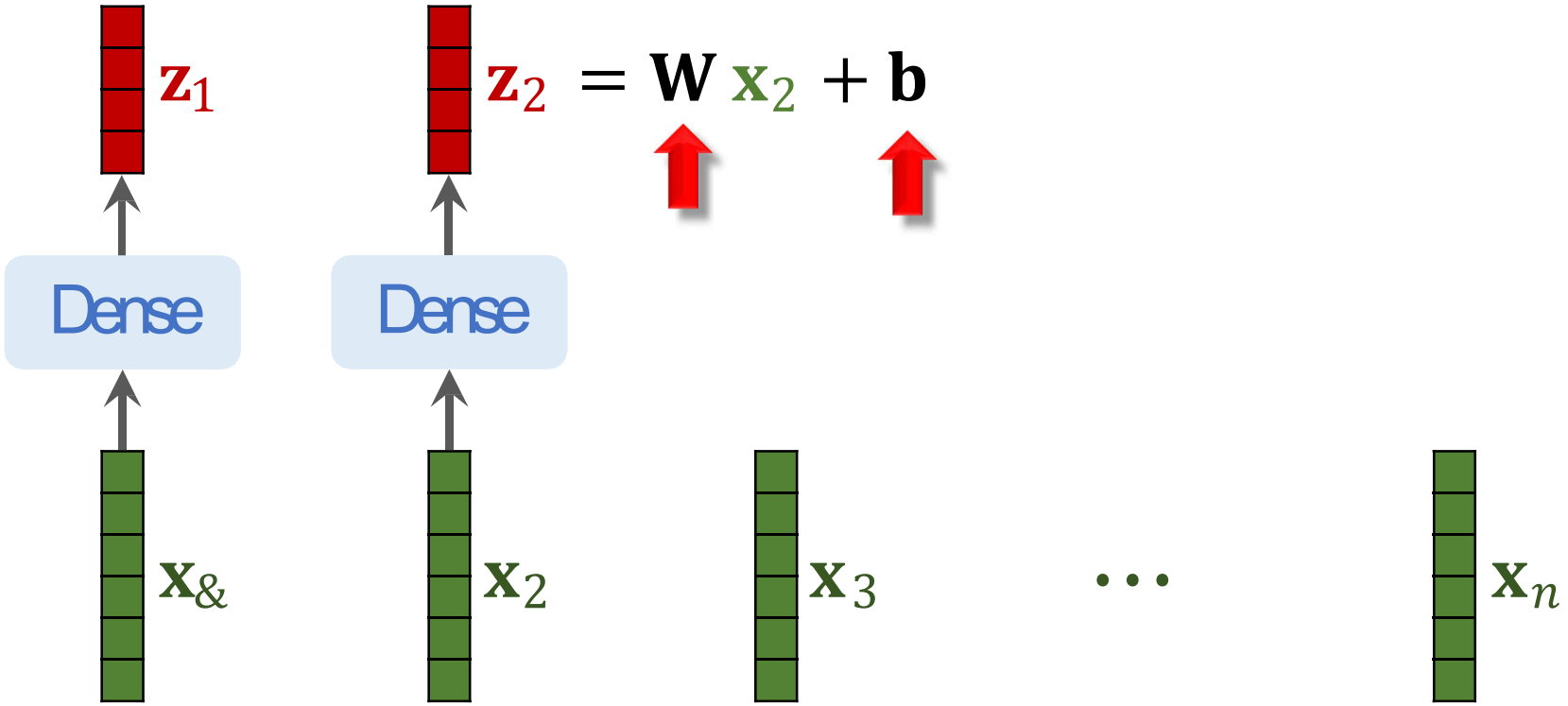
If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2\ d_3 \times 1$.

16x16x3 = 786x1



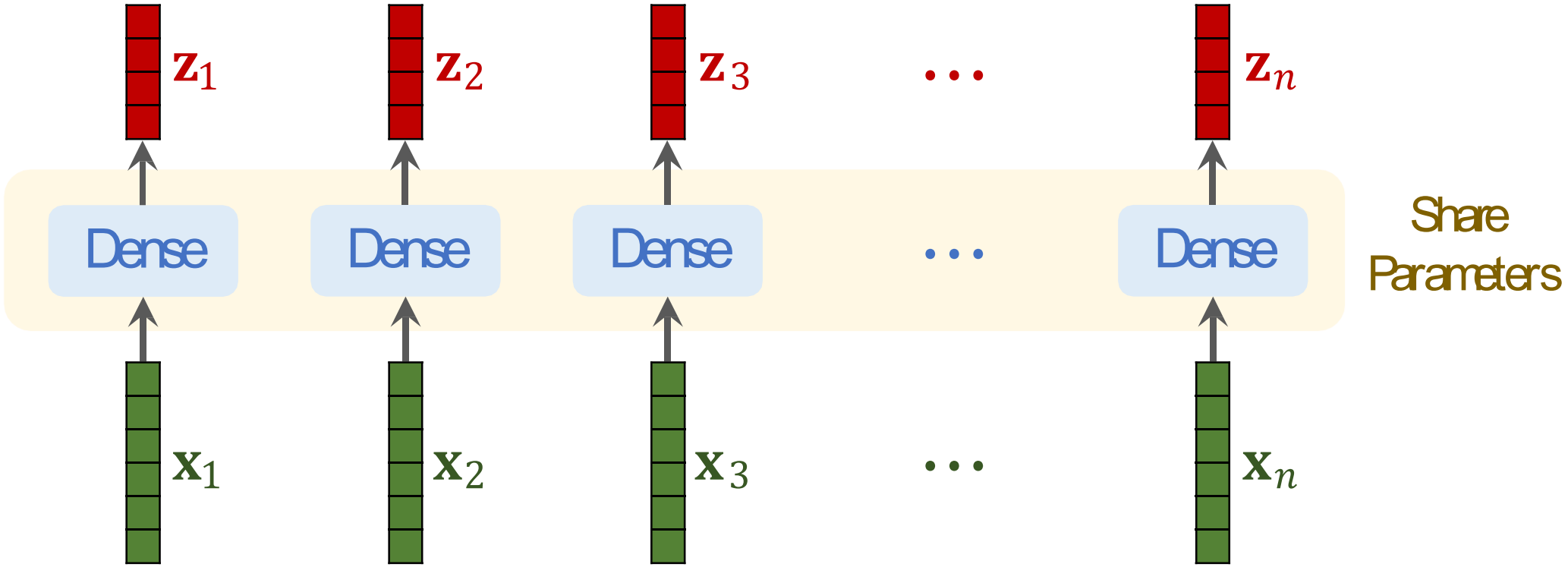

$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$


$\mathbf{z}_1 = \mathbf{W}\,\mathbf{x}_1 + \mathbf{b}$
Dense
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$


$\mathbf{z}_1$
$\mathbf{z}_2 = \mathbf{W}\,\mathbf{x}_2 + \mathbf{b}$
Dense
Dense
$\mathbf{x}_\&$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$

196 x786 -> 196 x 512

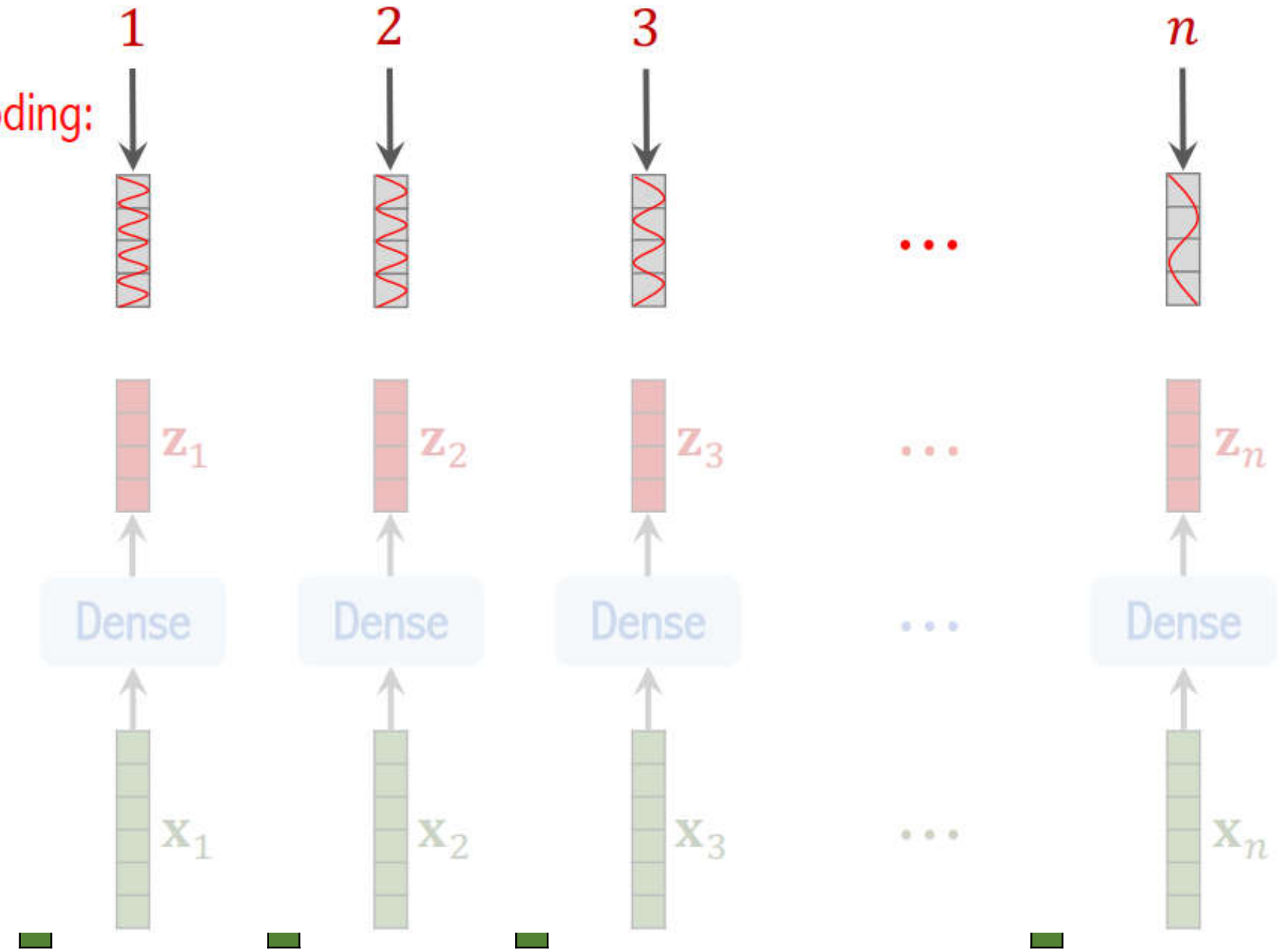
Retain important information



Positional Encoding

- Encoder take all 196 vectors of size 512x1 as input at once.
- Encoder does not able to make which vector come first and which come later, so need a position encoding
- Understand the arrangement of patches, which is essential for recognizing objects and understanding the visual context of the entire image

Positional Encoding:

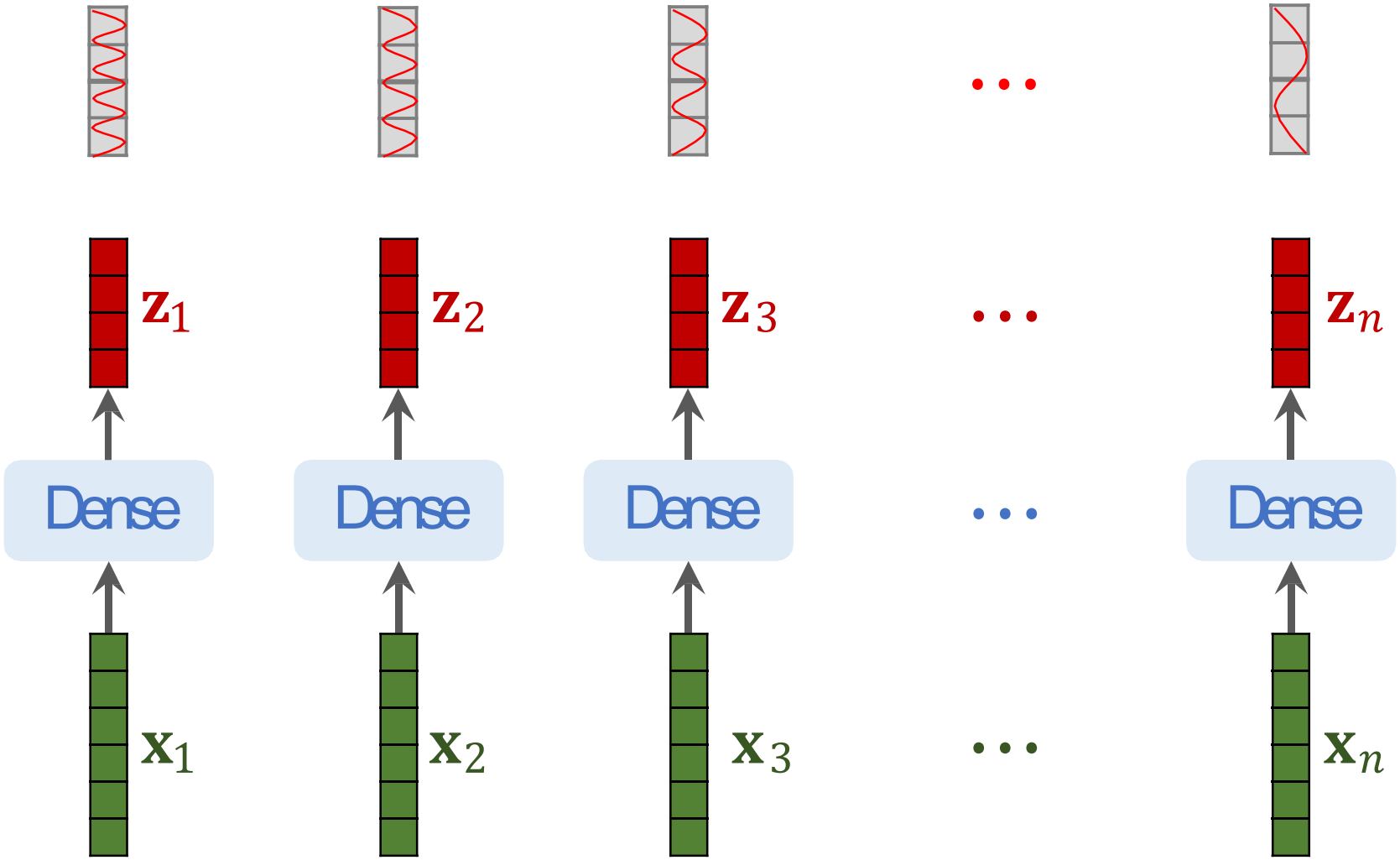


Positional Encoding

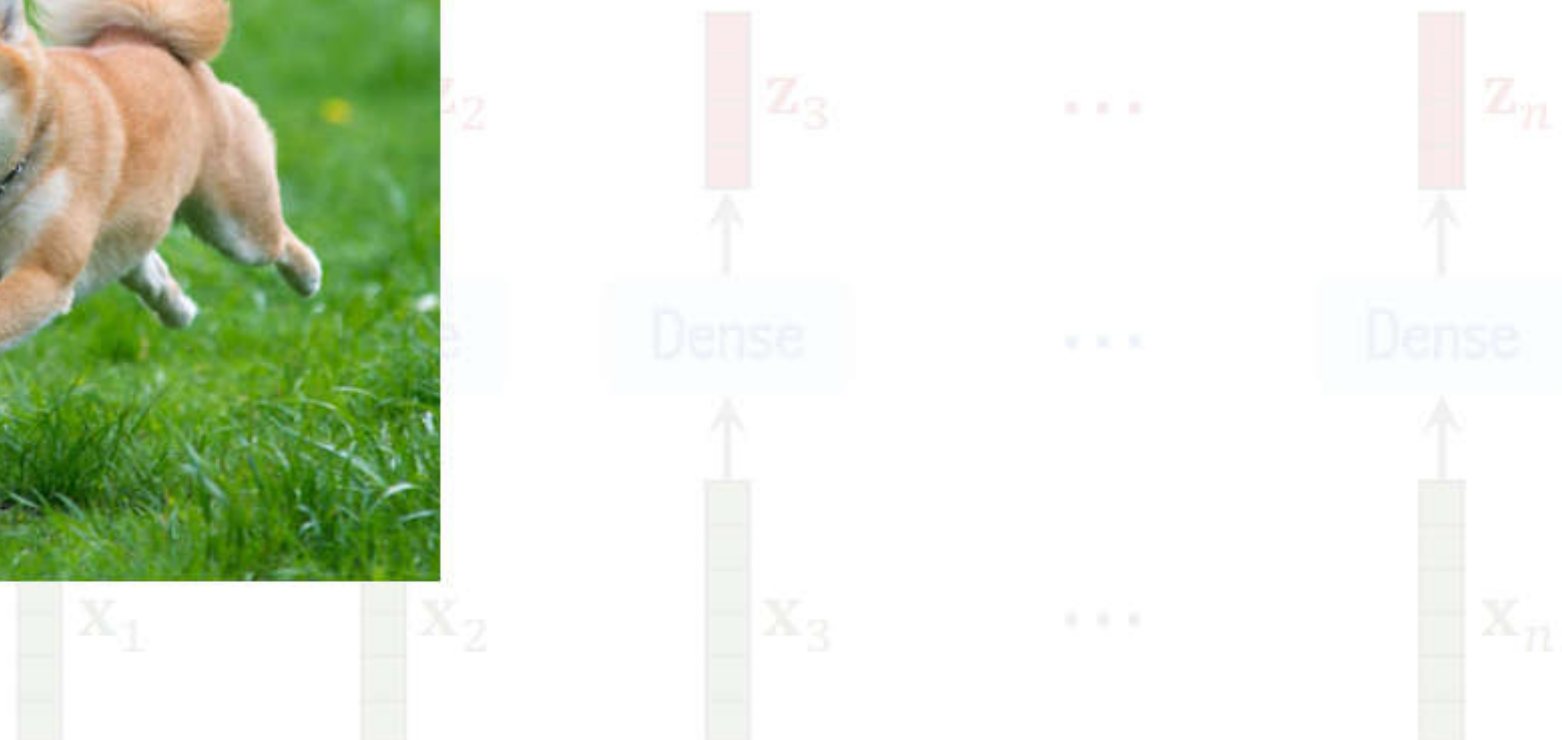
- **- Element wise addition of position encoding vector with the embedding vector.**

- Patch representation: [0.1 0.2 0.3 0.4]
- Positional encoding: [1.0 2.0 3.0 4.0]
- Elementwise addition : [1.1 2.2 3.3 4.4] = [feature + postion]

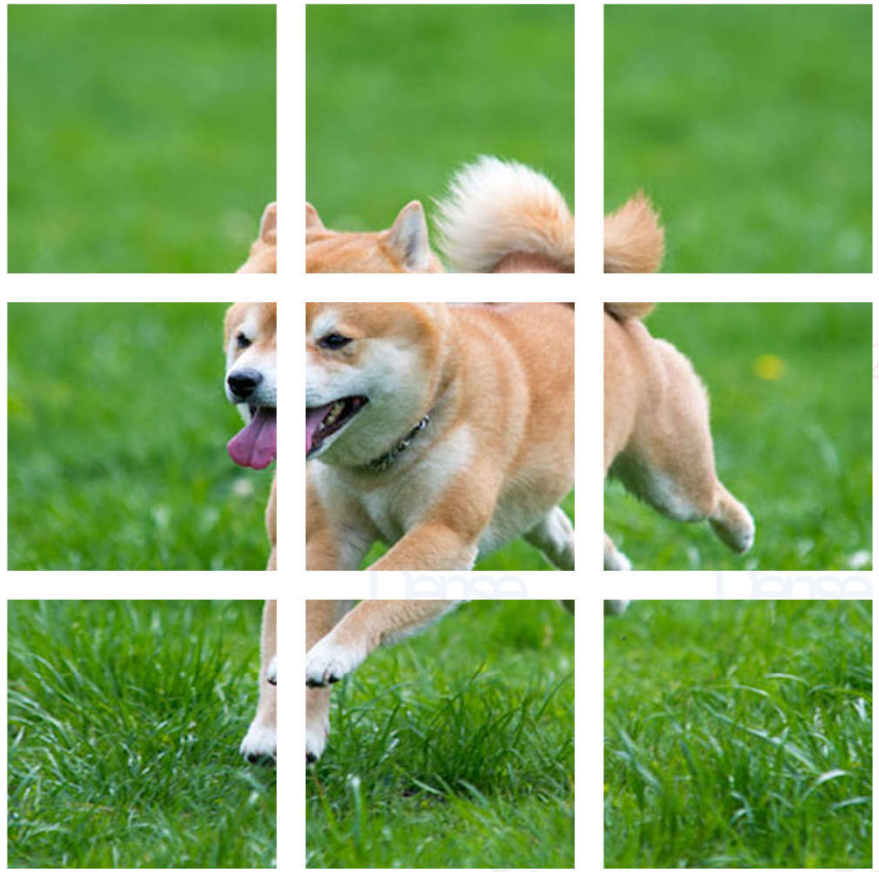
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.



Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

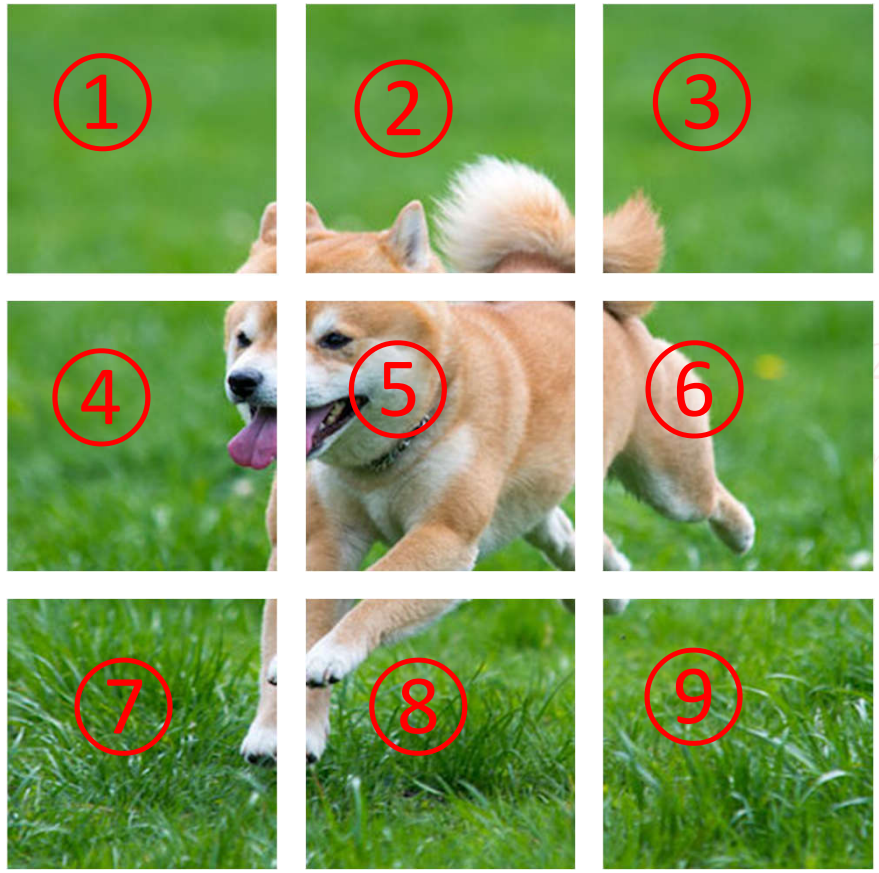


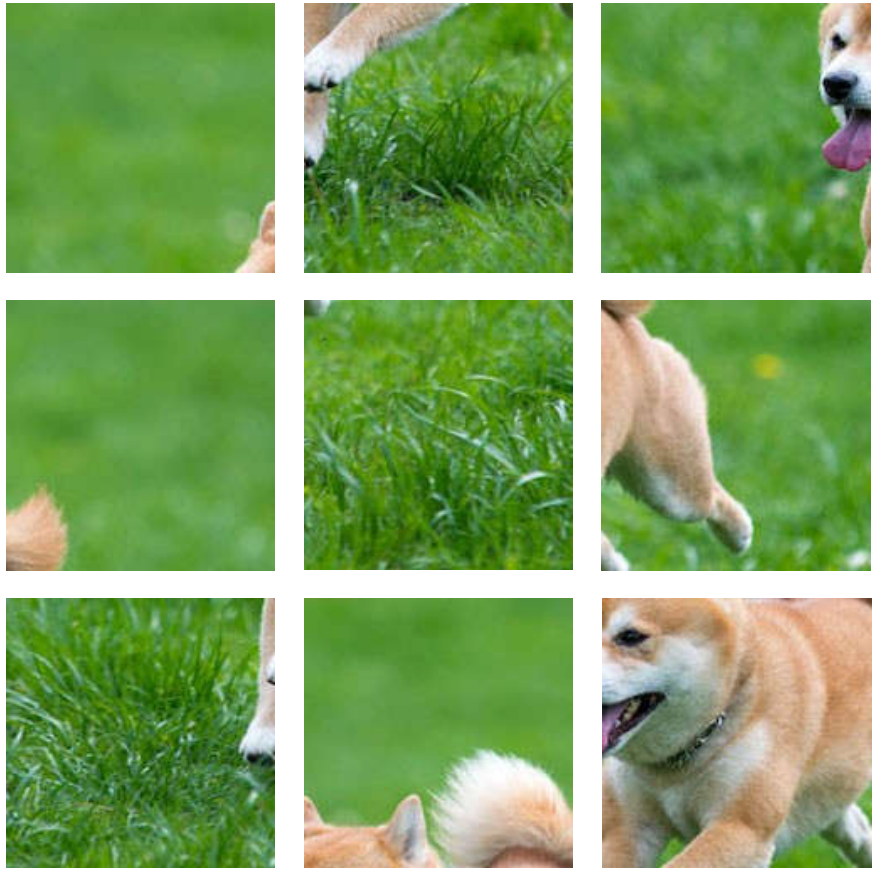
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

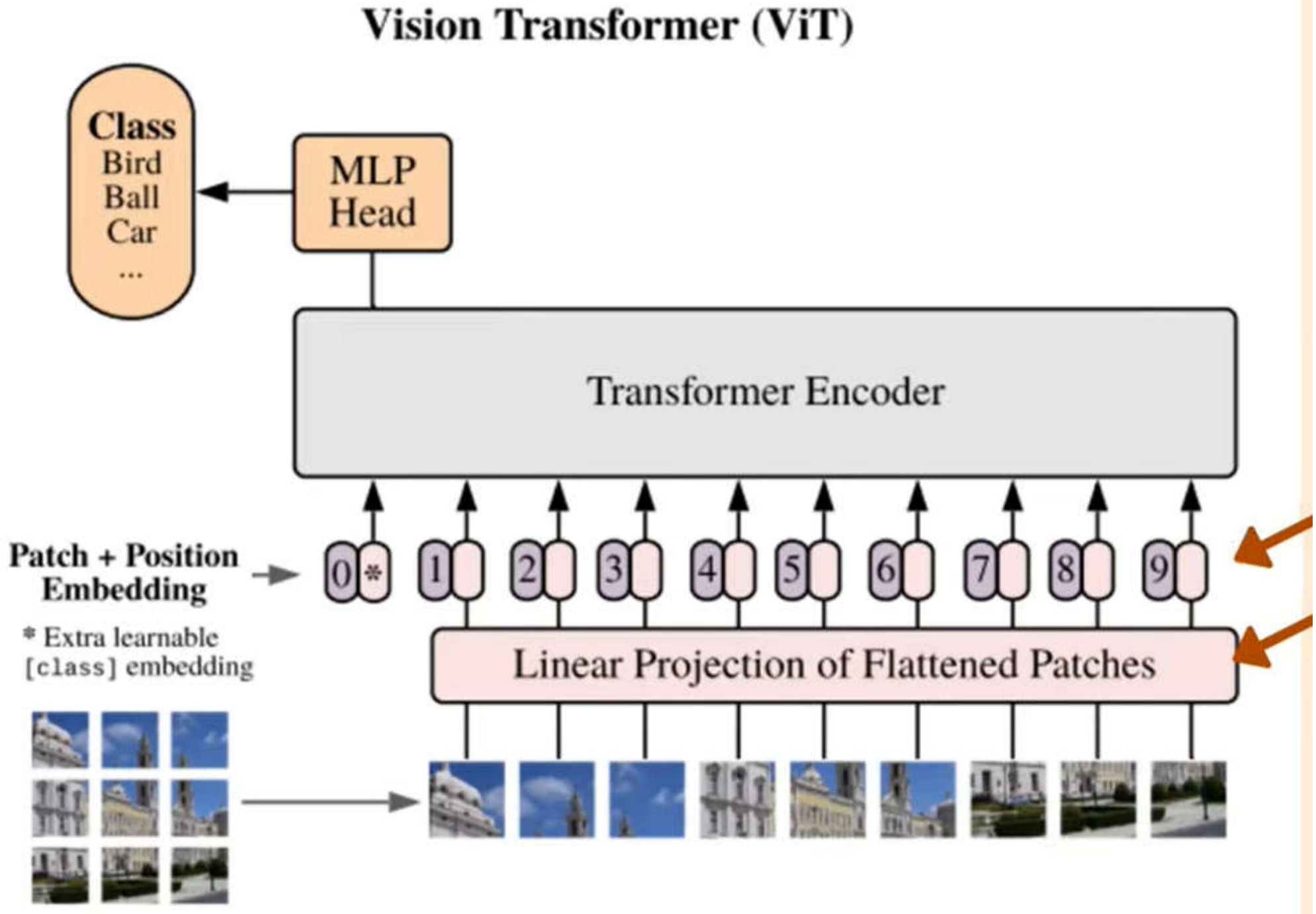


Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

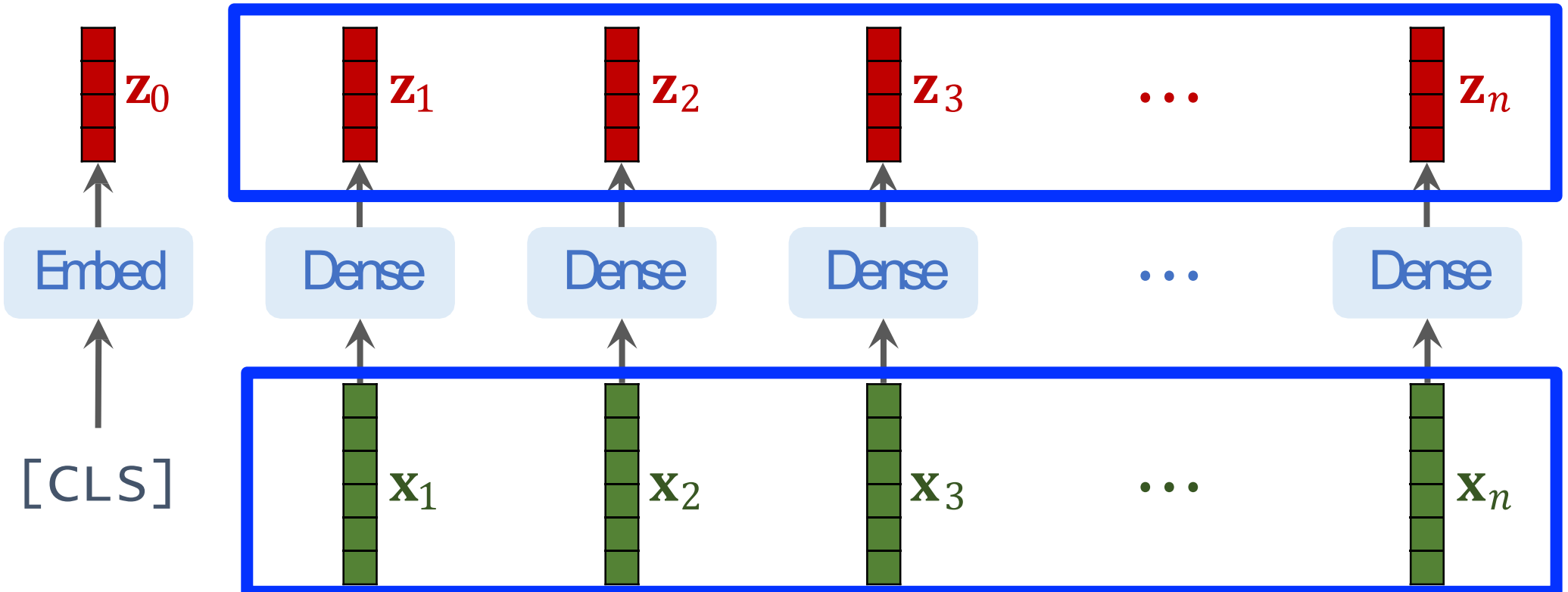
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

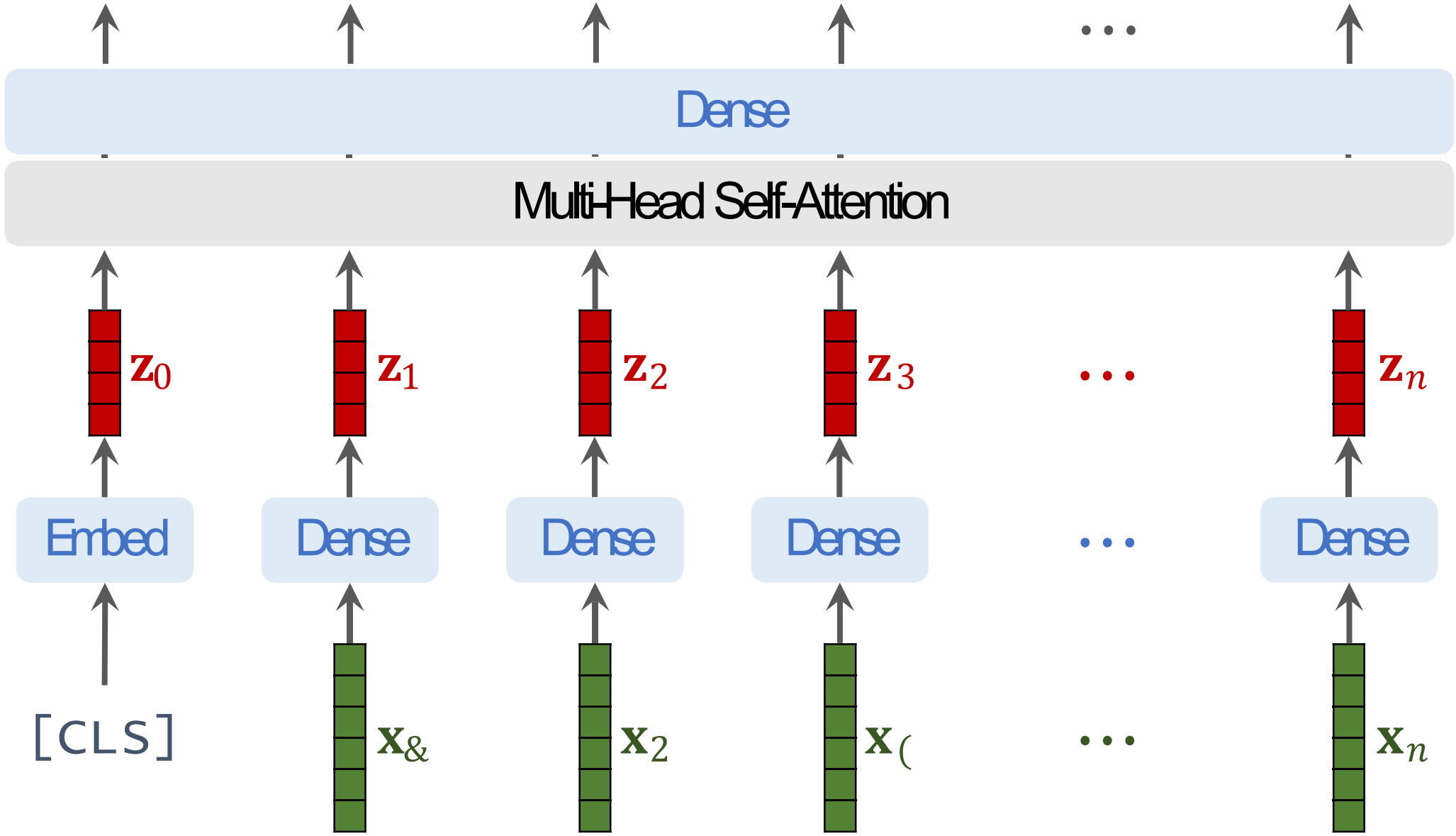


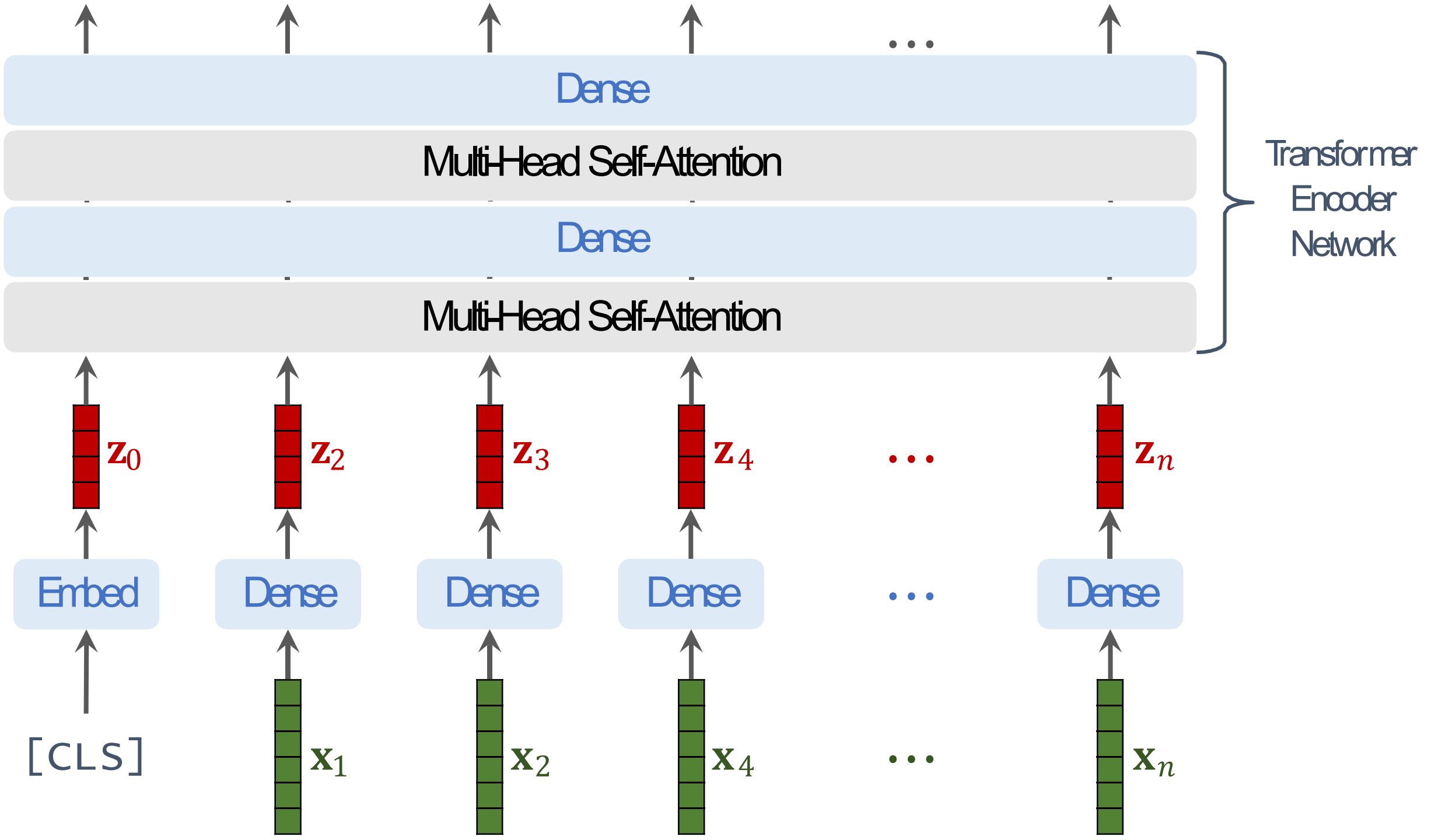


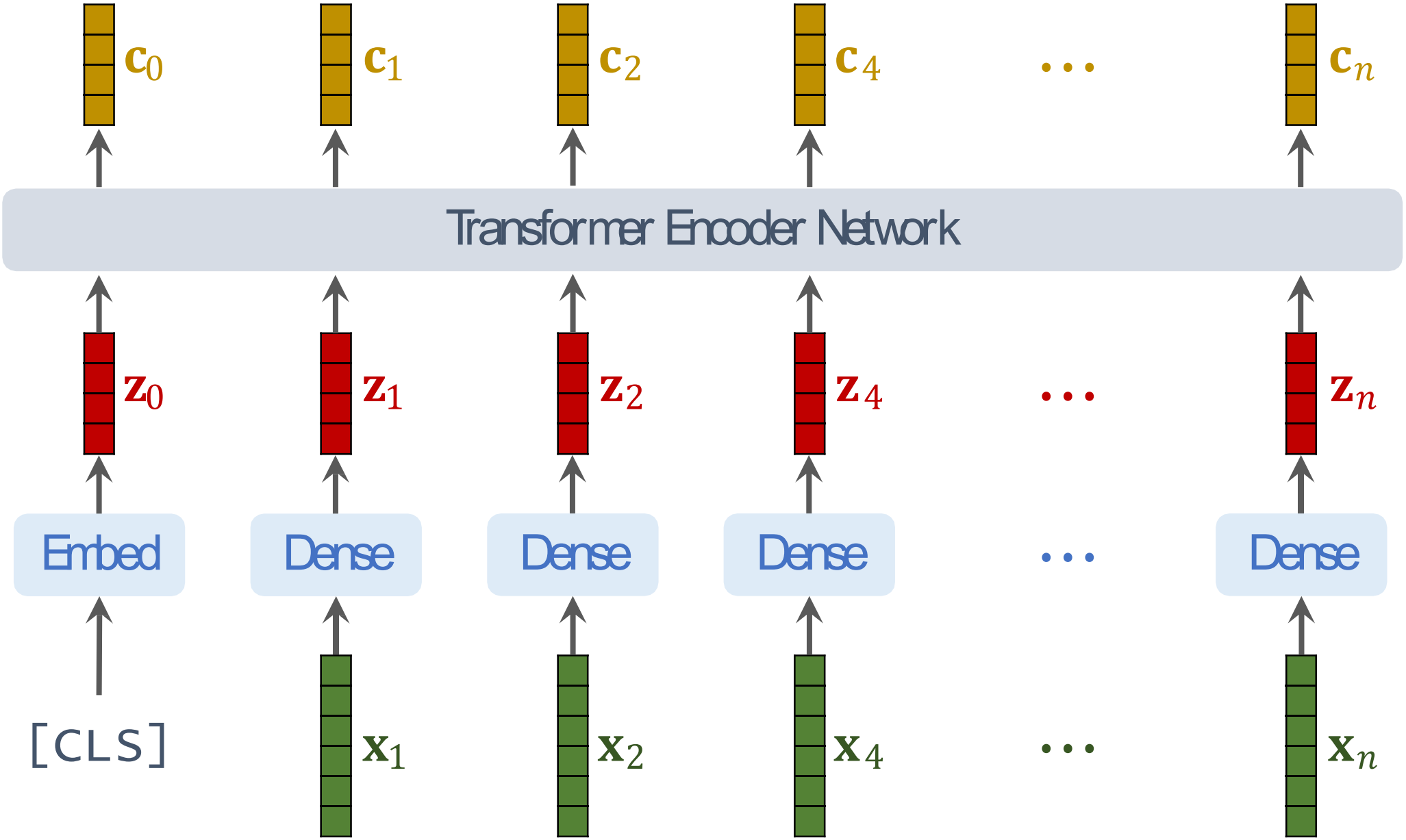

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP
Head
Transformer Encoder
Patch + Position
Embedding
* Extra learnable
[class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

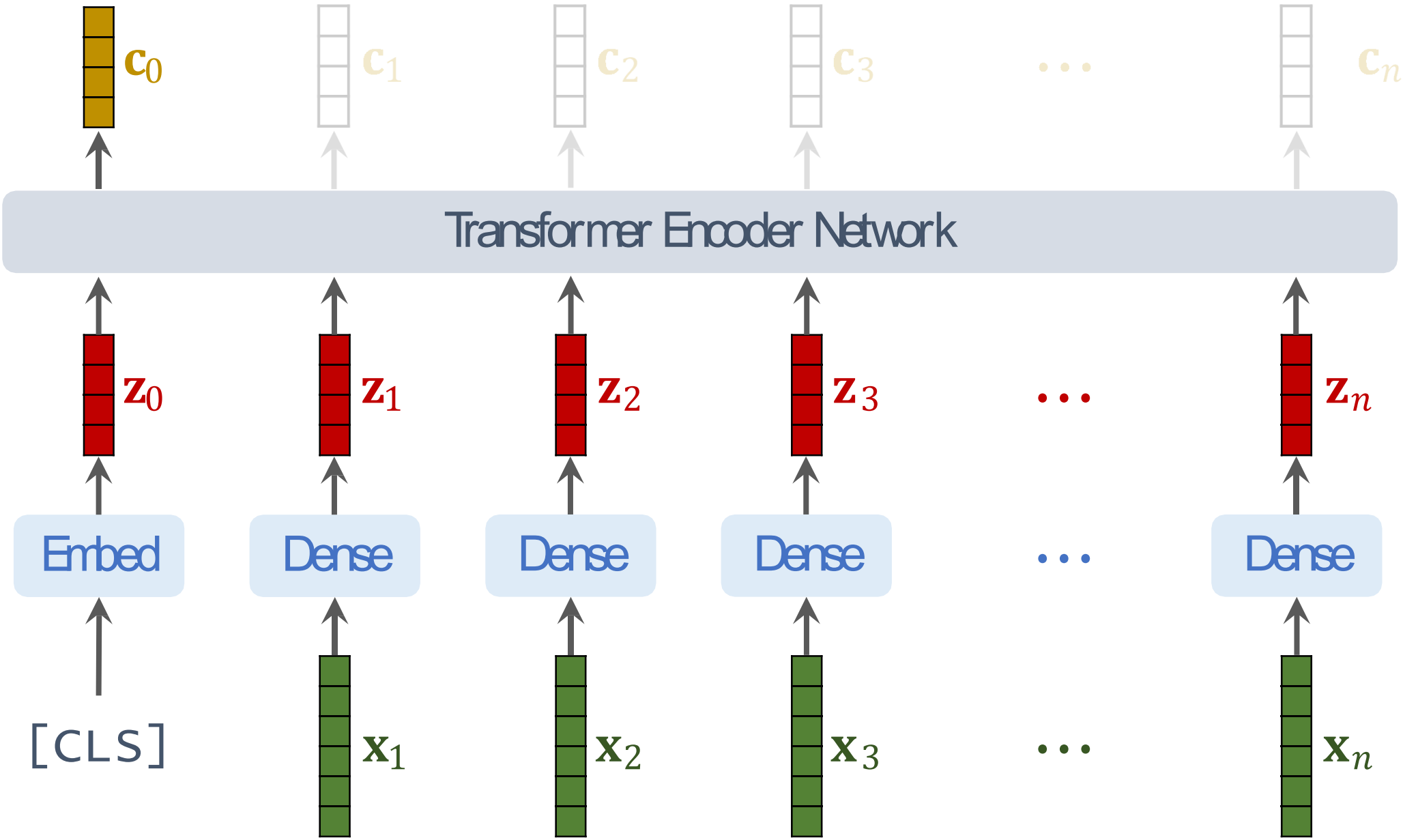
- **The [CLS] token** serves as a **placeholder for the overall image representation**.
- **Benefits of Using the [CLS] Token**☹ Unified Representation) The [CLS] token serves as a single vector representation of the entire image, capturing global context and high-level features.
- **Simplifies Classification**: Instead of combining information from all patch tokens for classification, ViT only needs to focus on the [CLS] token, making it more efficient.

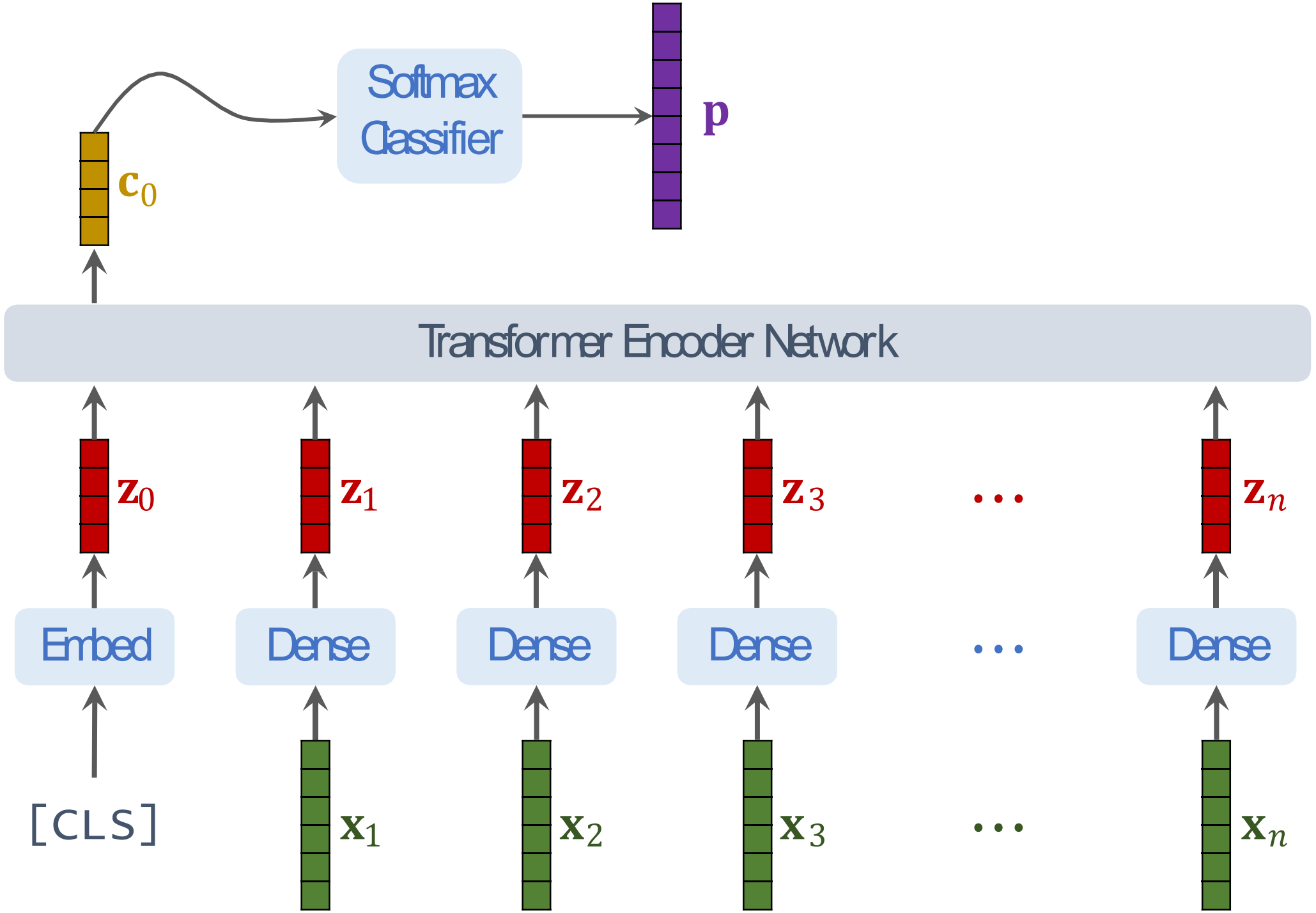


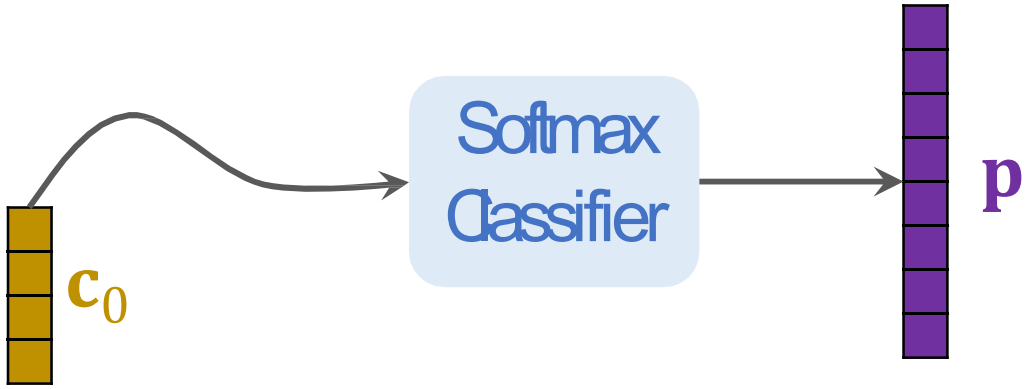

Dense
Multi-Head Self-Attention
z_0
z_1
z_2
z_3
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
$x_{\&}$
x_2
$x_($
x_n

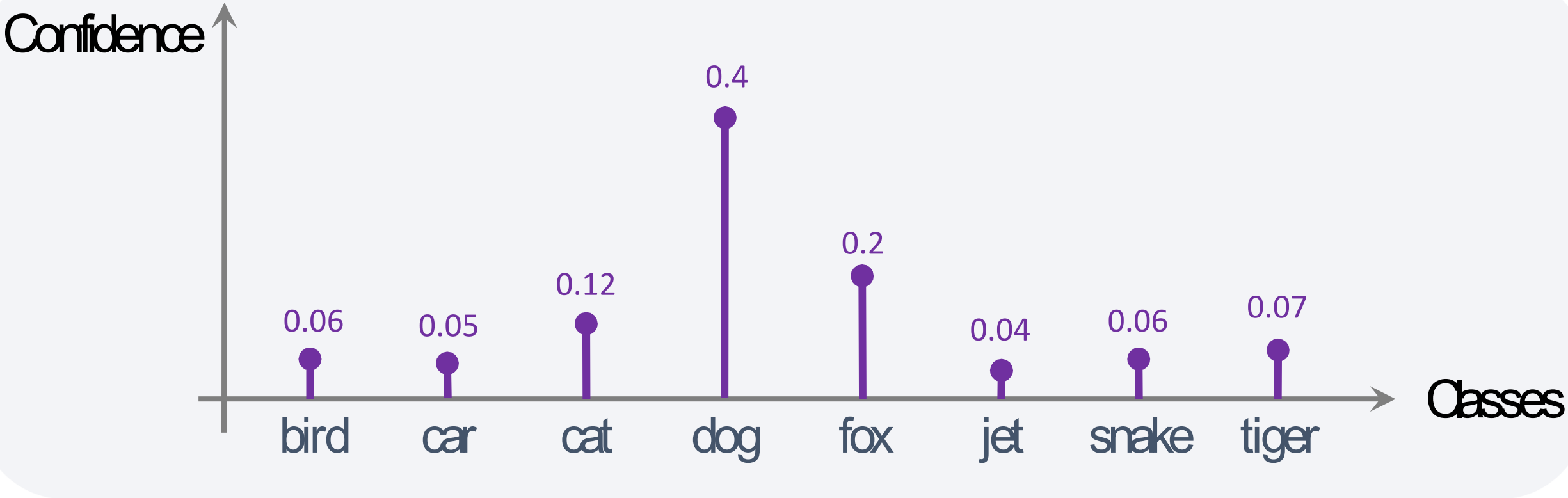

Dense
Multi-Head Self-Attention
Dense
Multi-Head Self-Attention
Transformer Encoder Network
z_0
z_2
z_3
z_4
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
x_1
x_2
x_4
x_n

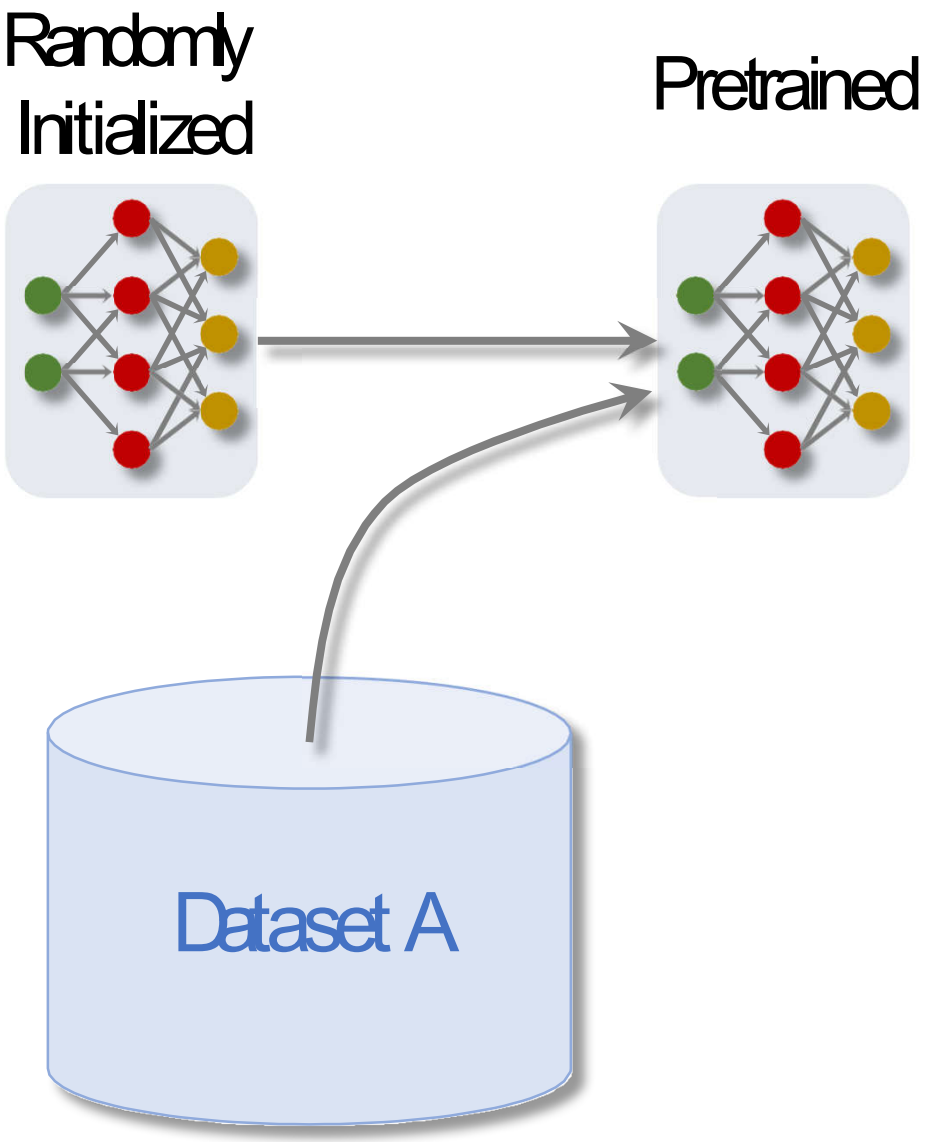

$\mathbf{c}_0$
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathbf{c}_4$
...
$\mathbf{c}_n$
Transformer Encoder Network
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_4$
...
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
...
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_4$
...
$\mathbf{x}_n$

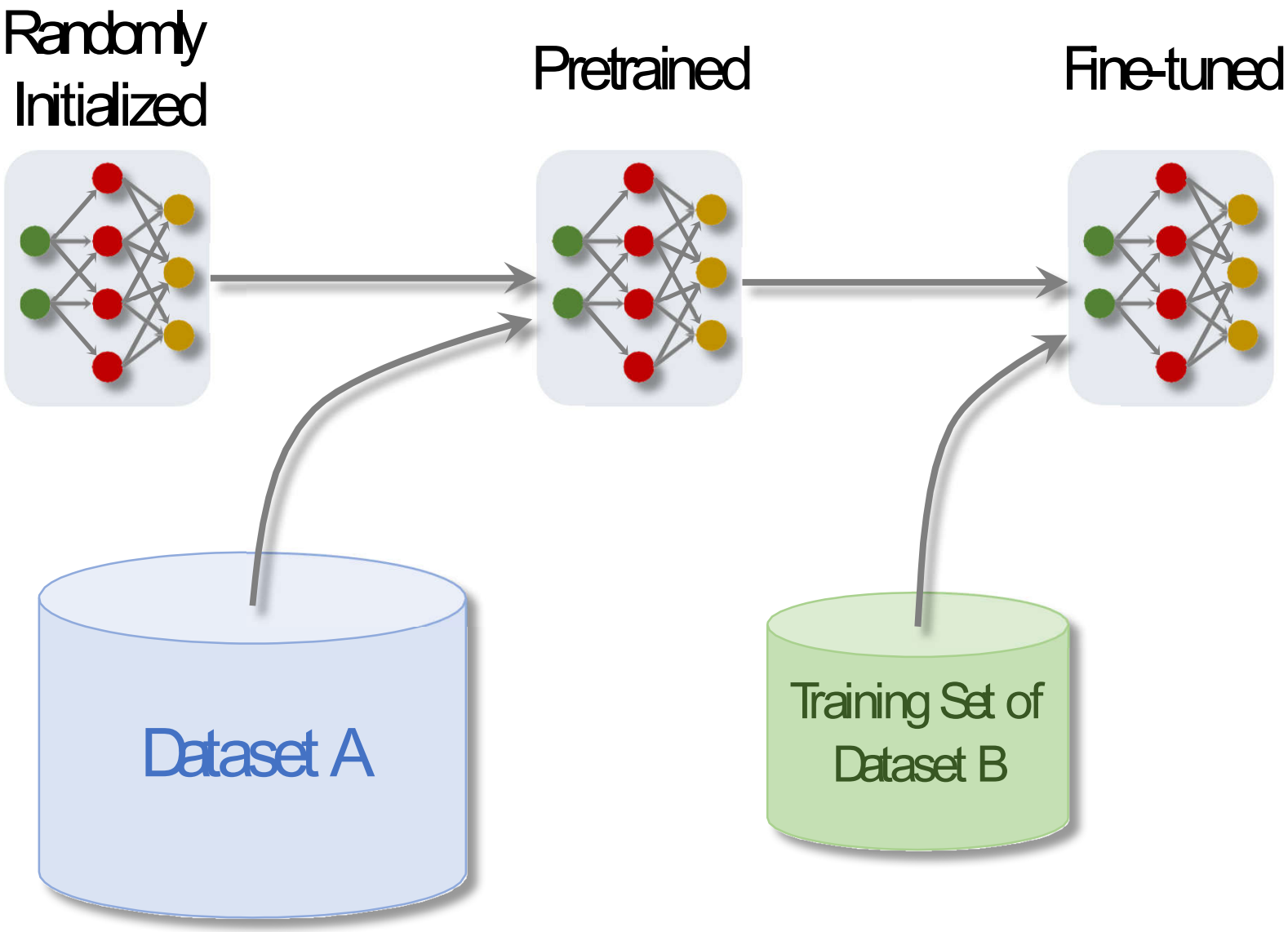

c_0
c_1
c_2
c_3
c_n
Transformer Encoder Network
z_0
z_1
z_2
z_3
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
x_1
x_2
x_3
x_n

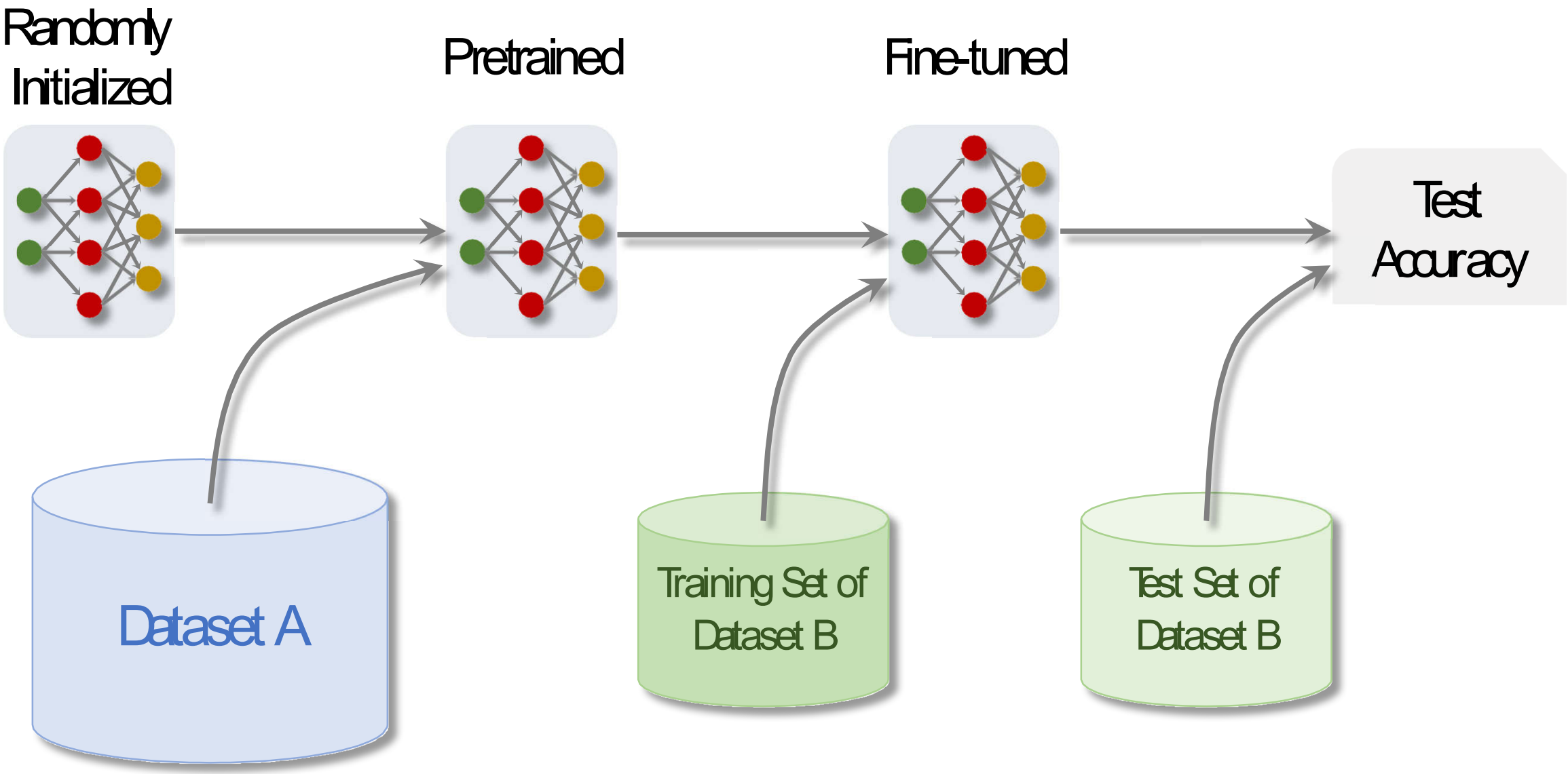

Softmax Classifier
p
$\mathbf{c}_0$
Transformer Encoder Network
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$


Softmax
Classifier
$\mathbf{c}_0$
$\mathbf{p}$


Confidence
0.06
0.05
0.12
0.4
0.2
0.04
0.06
0.07
bird
car
cat
dog
fox
jet
snake
tiger
Classes


Randomly Initialized
Pretrained
Dataset A


Randomly Initialized
Pretrained
Fine-tuned
Dataset A
Training Set of Dataset B


Randomly Initialized
Pretrained
Fine-tuned
Test Accuracy
Dataset A
Training Set of Dataset B
Test Set of Dataset B

Datasets

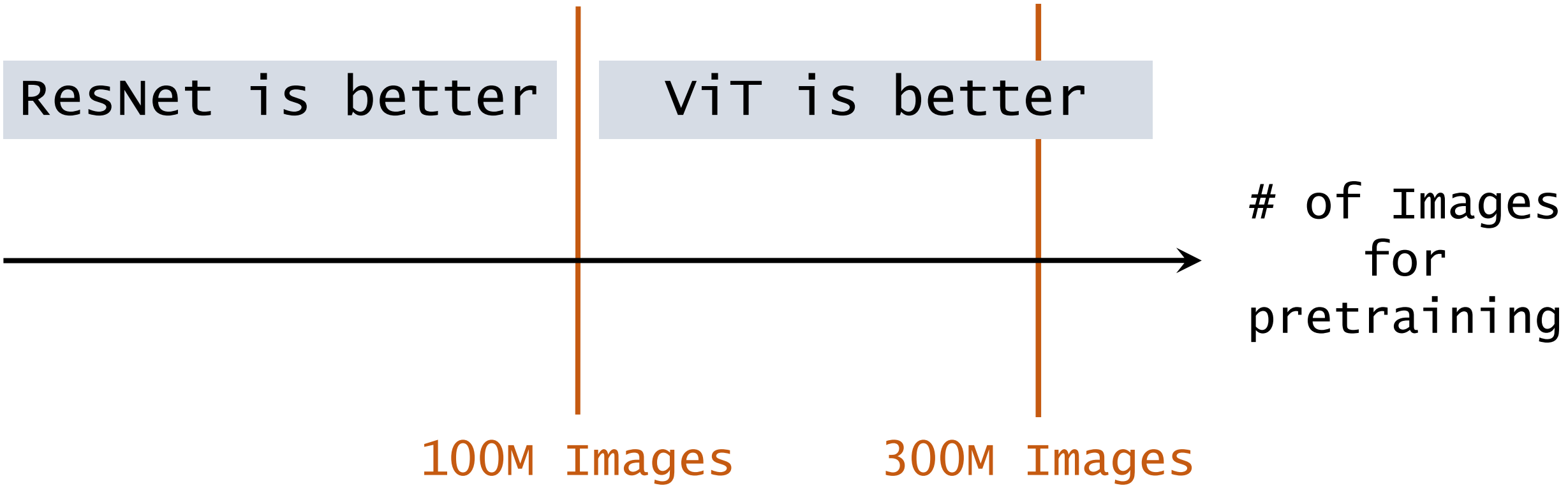
	# of Images	# of Classes
ImageNet (Small)	1.3 Million	1 Thousand
ImageNet-21K (Medium)	14 Million	21 Thousand
JFT (Big)	300 Million	18 Thousand

Image Classification Accuracies

- Pretrain the model on Dataset A, fine-tune the model on Dataset B, and evaluate the model on Dataset B.

- Pretrained on ImageNet (small), ViT is slightly worse than ResNet.
- Pretrained on ImageNet-21K (medium), ViT is comparable to ResNet.
- Pretrained on JFT (large), ViT is slightly better than ResNet.

Image Classification Accuracies



Swin Transformer

Resource:
https://medium.com/thedeephub/building-swin-transformer-from-scratch-using-pytorch-hierarchical-vision-transformer-using-shifted-91cbf6abc678

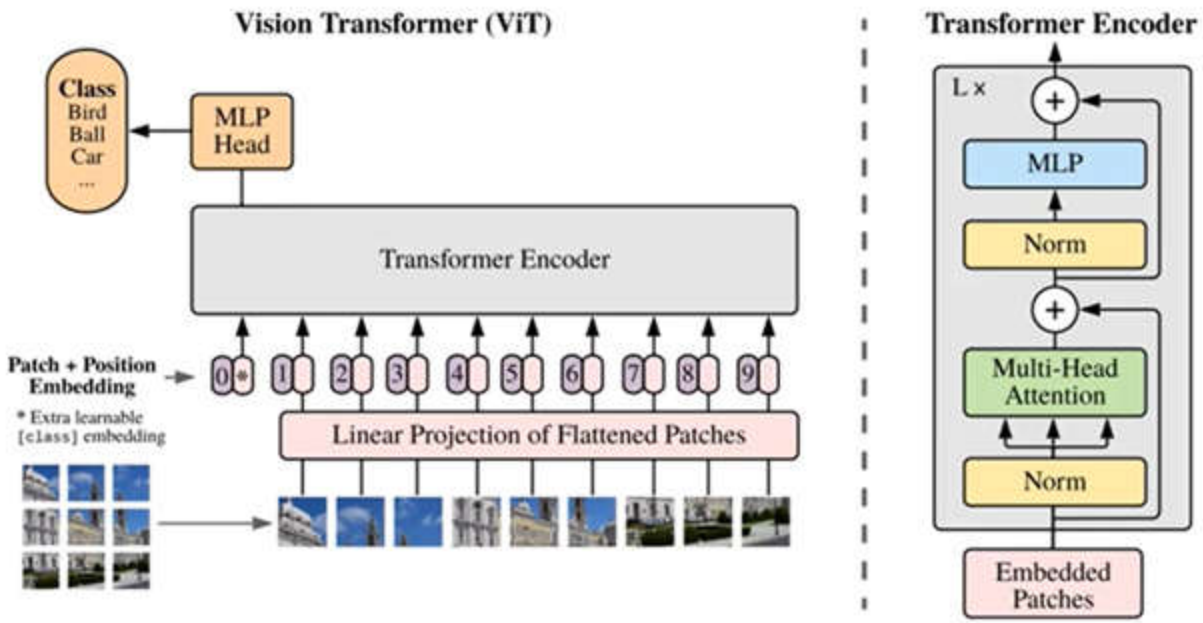
https://www.youtube.com/watch?v=ORWdELQ1h9M&t=308s&ab_channel=CodeWithAarohi

https://www.youtube.com/watch?v=iTHK0FDWJys&list=PL9iXGo3xD8jokWaLB8ZHUkjjv5Y_vPQnZ&index=9&ab_channel=AIHMP

https://www.youtube.com/watch?v=qUSPbHE3OeU&list=PLHYTrZz5TkDn-_g6A4f-YyQXGrGBslwtF&ab_channel=SoroushMehraban

Vision Transformer (ViT)

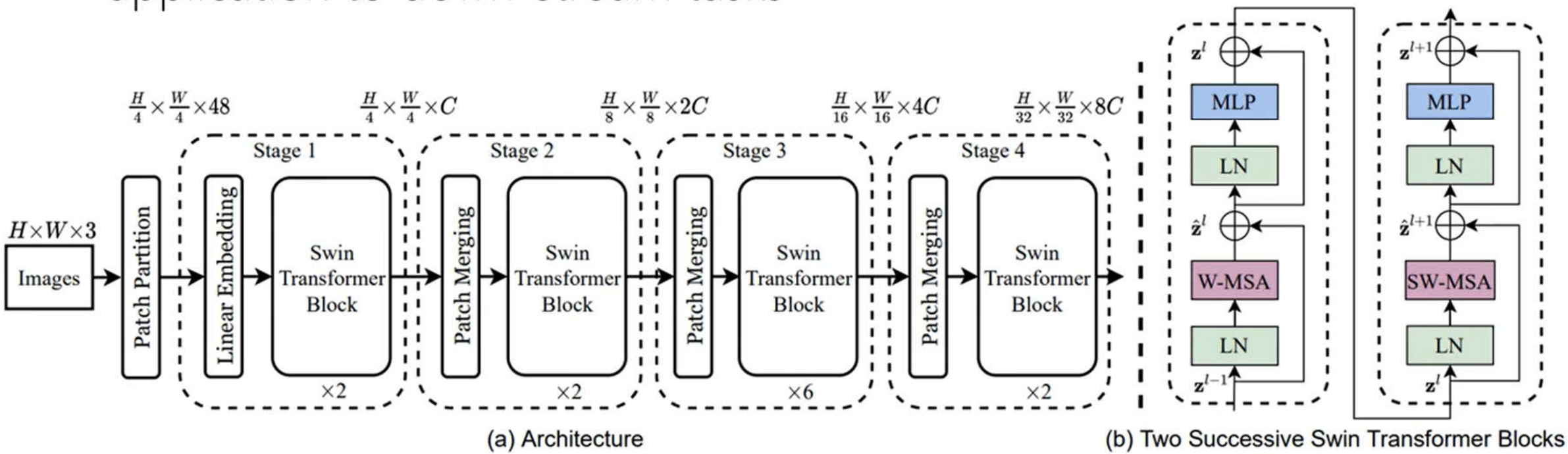
- by Google Brain (2020.10)



Alexey Dosovitskiy et al. an Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR' 2021

Architecture instantiations

- Resolution of each stage is set similar as ResNet, to facilitate application to down-stream tasks



Swin Transformer

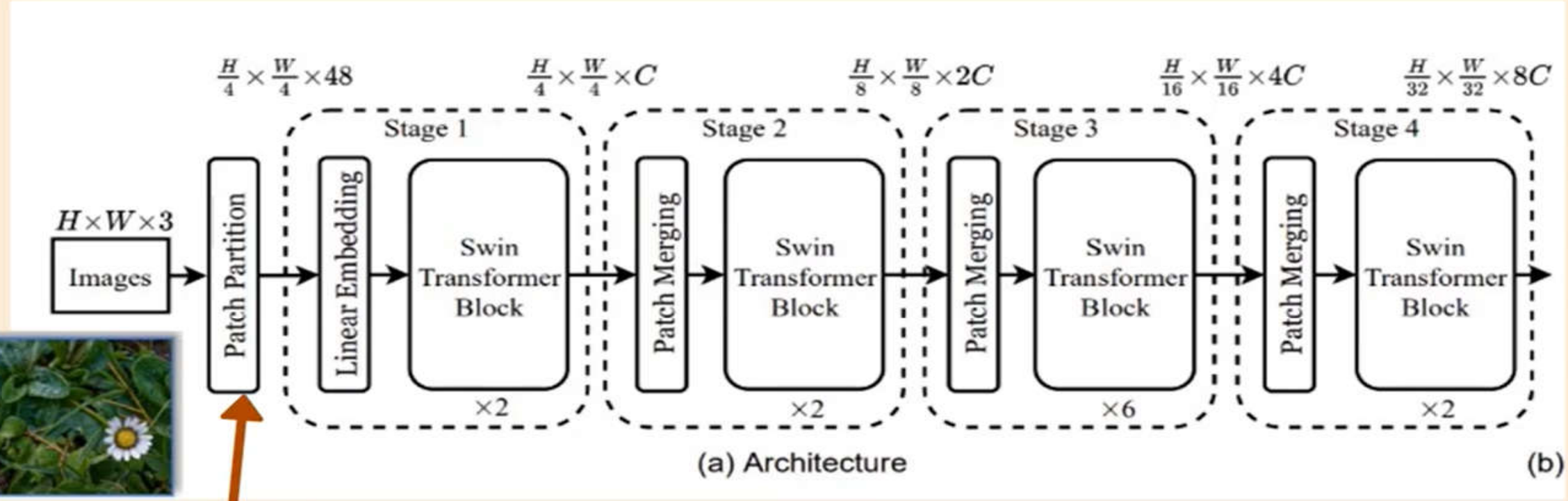
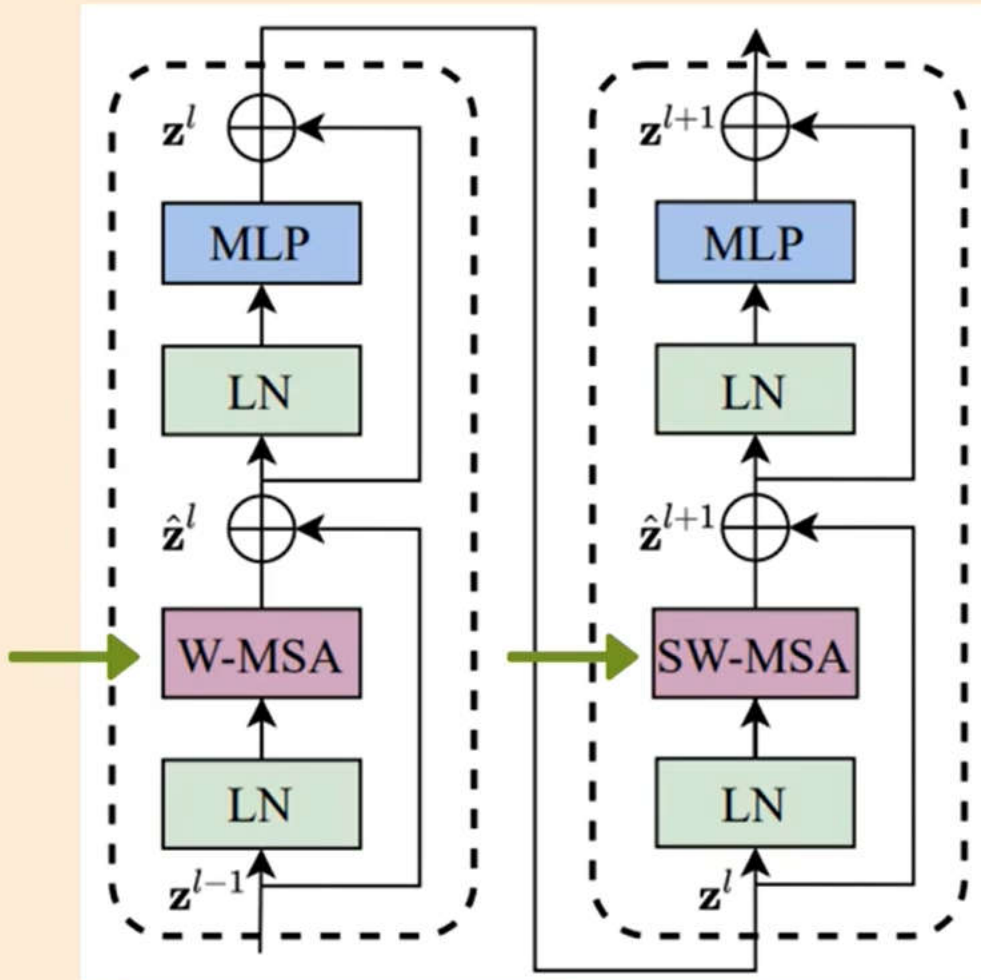
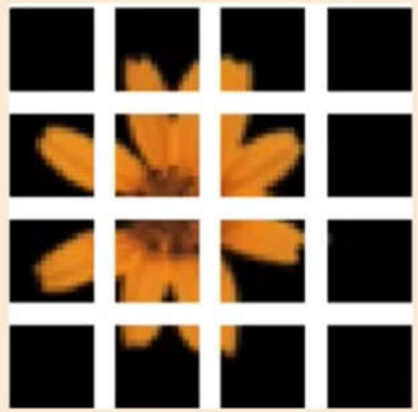


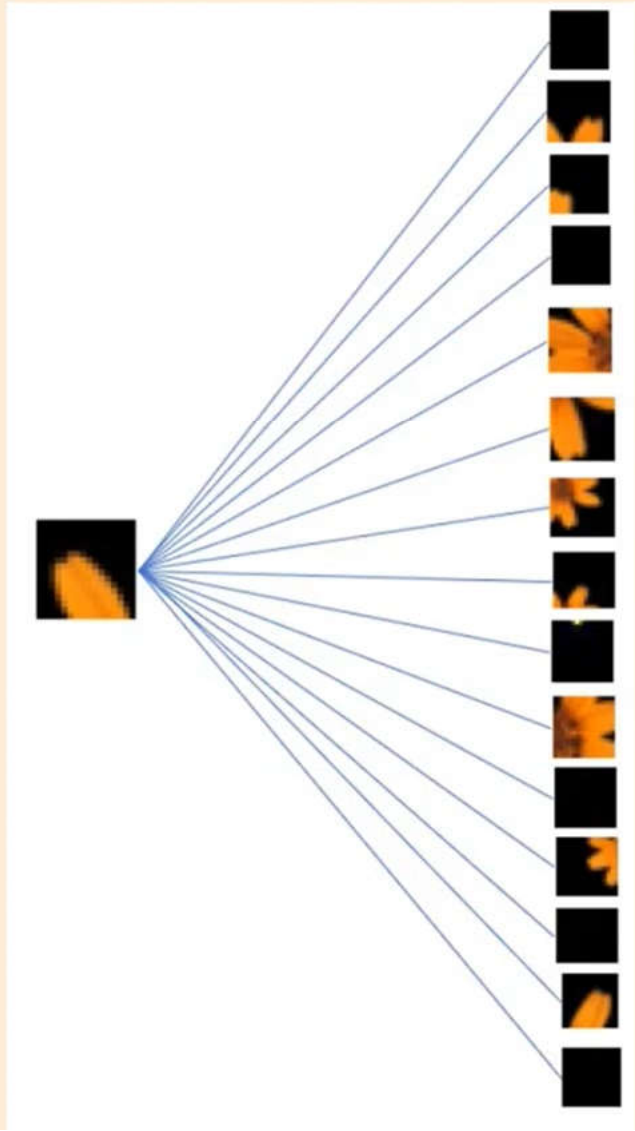
Image size = 224*224
Patch size: 4 pixels x 4 pixels
Number of patches per row: 56.0
Number of patches per column: 56.0
Total patches: 3136.0
Every patch have 48 entries or values(4*4*3).

Swin Transformer



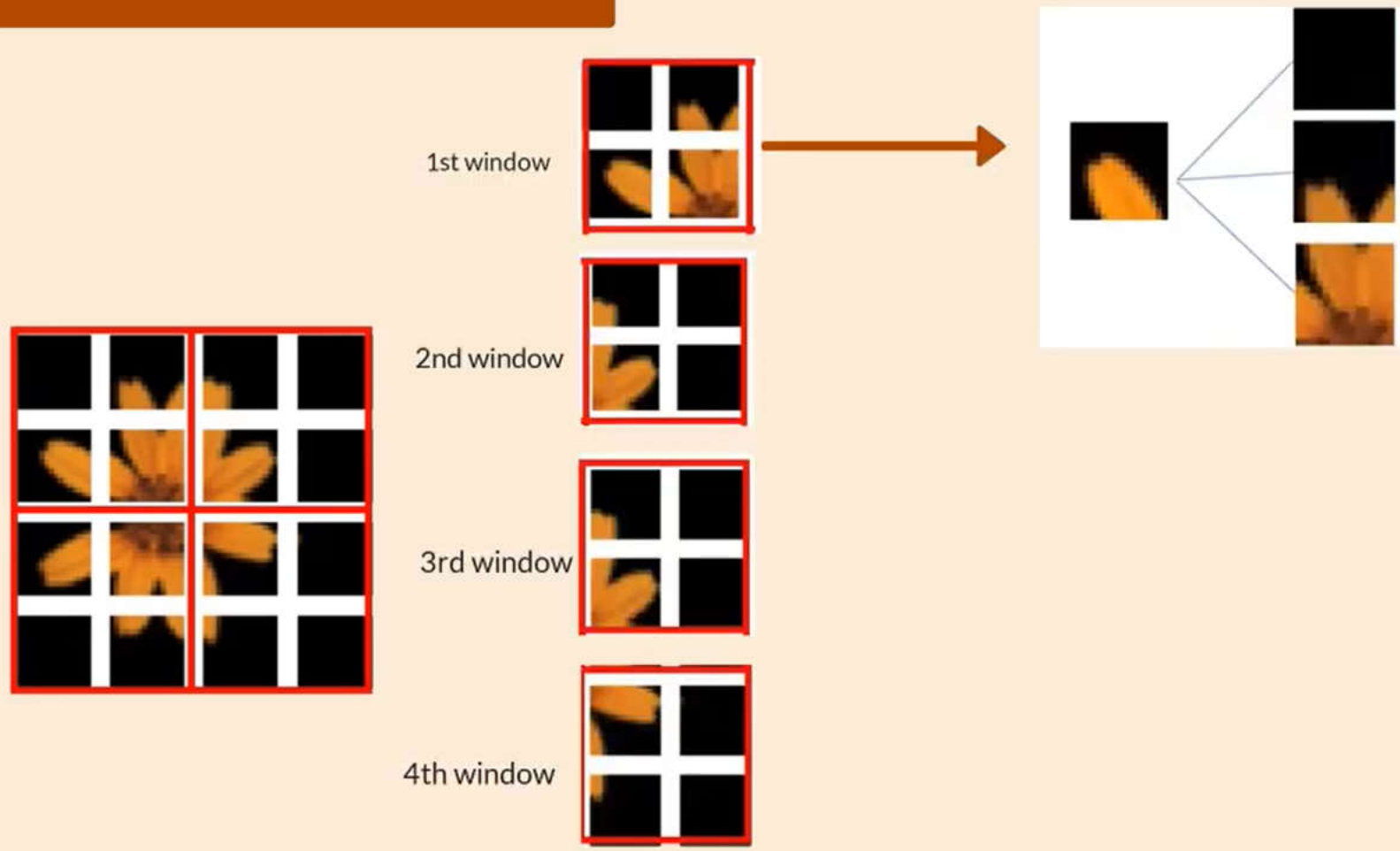
How attention works in ViT?

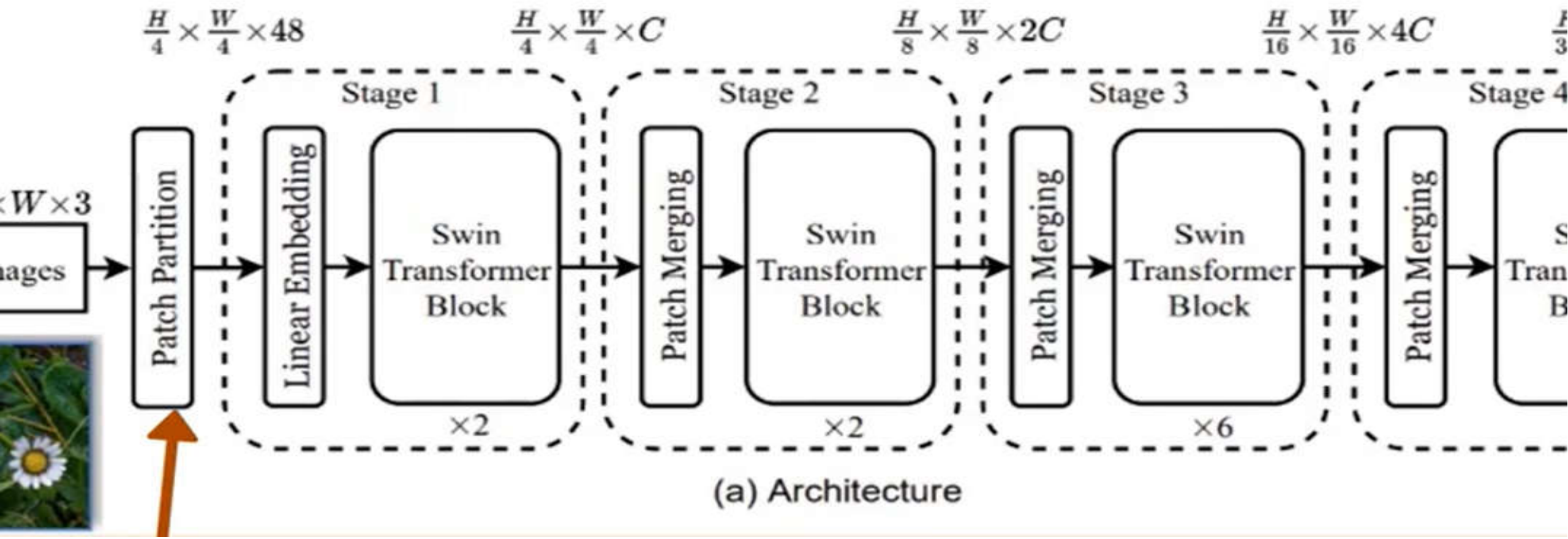


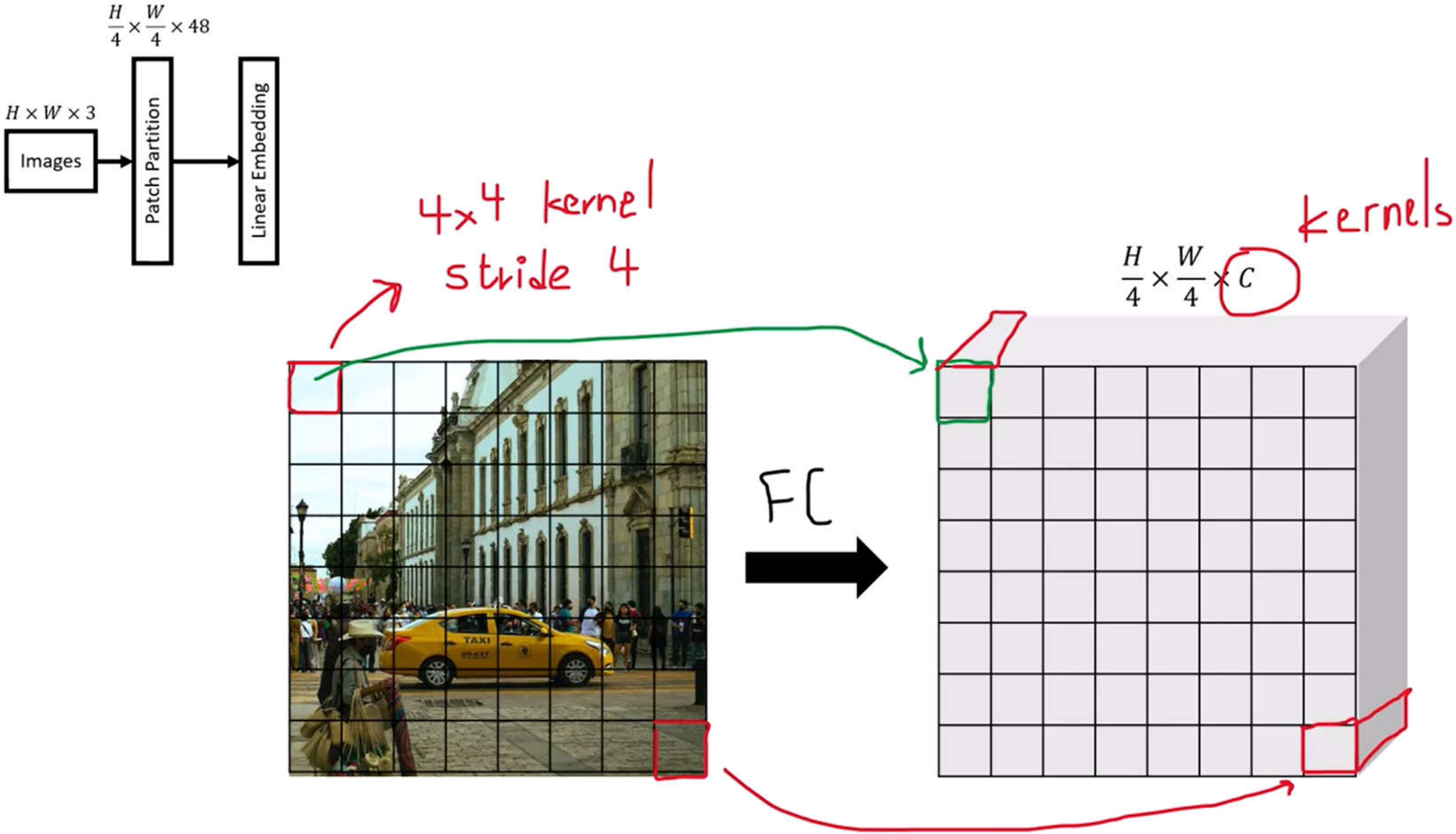


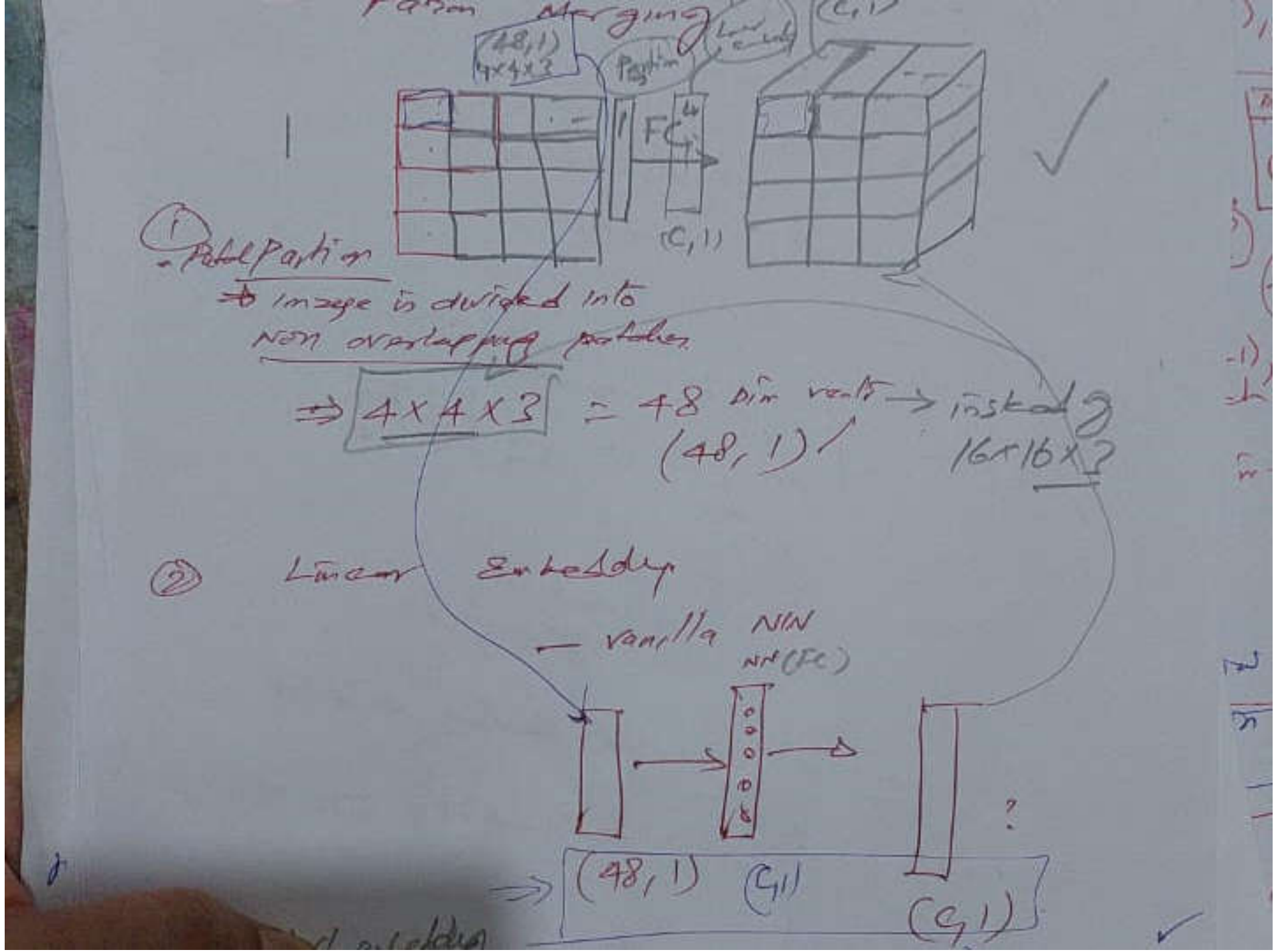
Swin Transformer

window-based MSA




$\frac{H}{4} \times \frac{W}{4} \times 48$
$\frac{H}{4} \times \frac{W}{4} \times C$
$\frac{H}{8} \times \frac{W}{8} \times 2C$
$\frac{H}{16} \times \frac{W}{16} \times 4C$
Stage 1
Stage 2
Stage 3
Stage 4
$\times W \times 3$
nages
Patch Partition
Linear Embedding
Swin Transformer Block
×2
Patch Merging
Swin Transformer Block
×2
Patch Merging
Swin Transformer Block
×6
Patch Merging
(a) Architecture


$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
4x4 kernel
stride 4
kernels
$\frac{H}{4} \times \frac{W}{4} \times C$
FC
TAXI



① Patch Partition

→ Image is divided into non overlapping patches

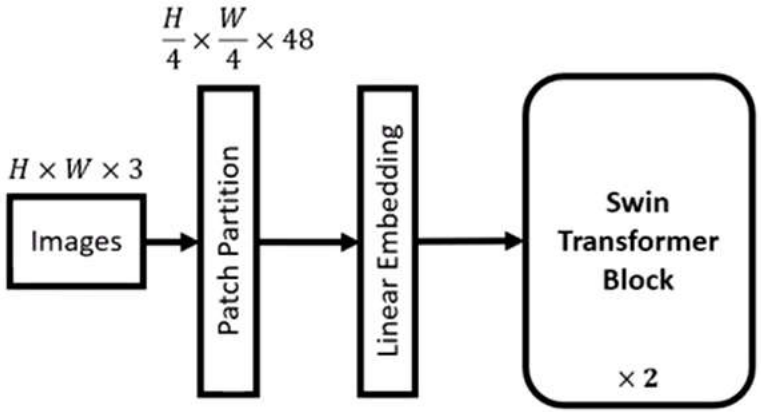
⇒ 4×4×3 = 48 dim vector → (48,1) → instead 16×16×3

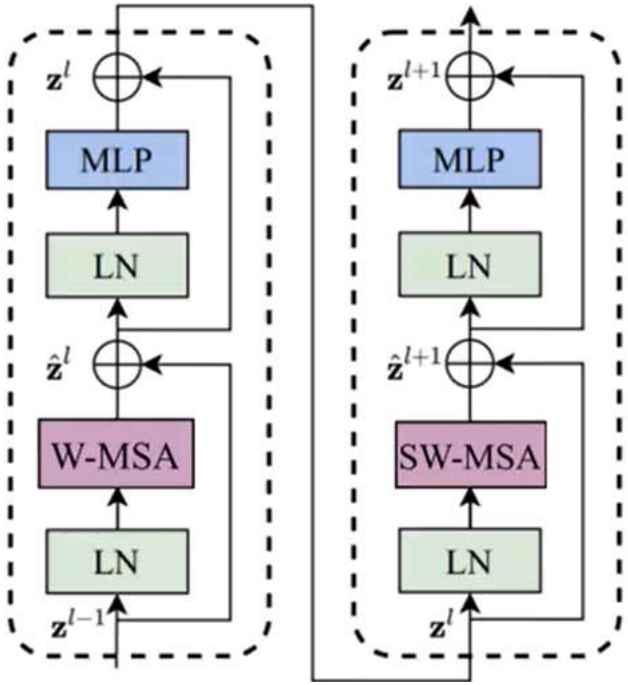
② Linear Embedding

— vanilla NN

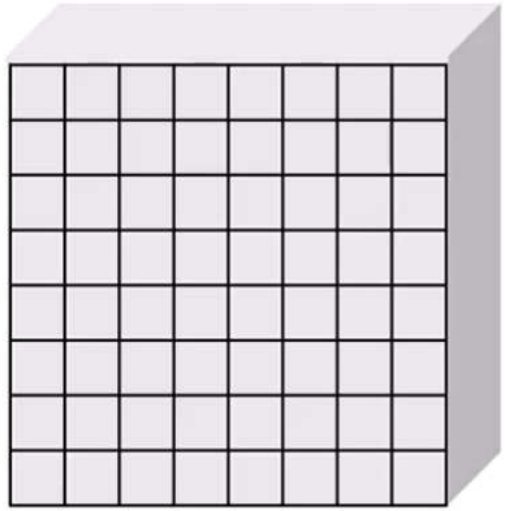
NN (FC)

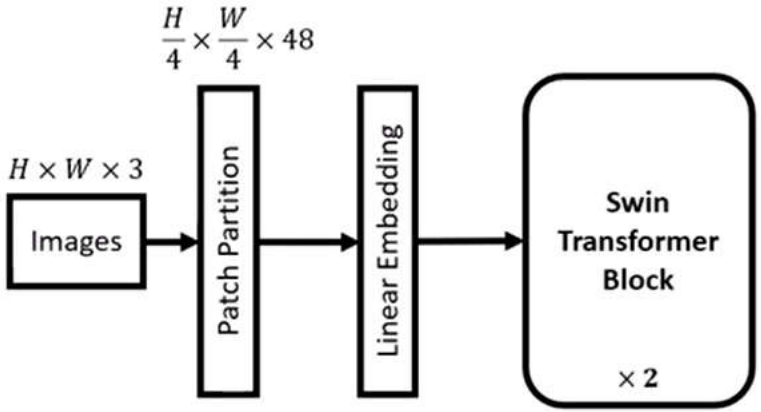
(48,1) (C,1) (C,1)

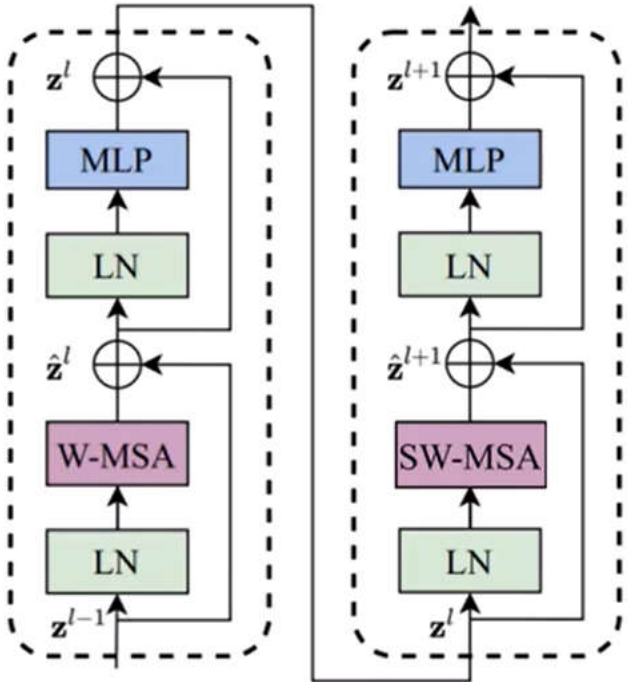




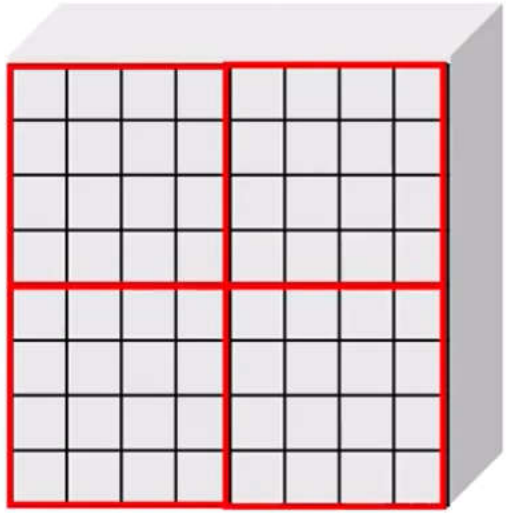
$$\frac{H}{4} \times \frac{W}{4} \times C$$







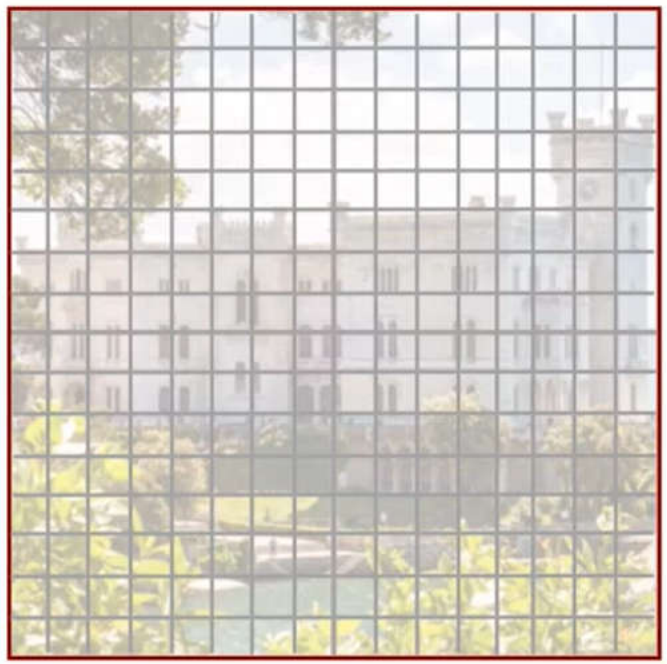
$\frac{H}{4} \times \frac{W}{4} \times C$



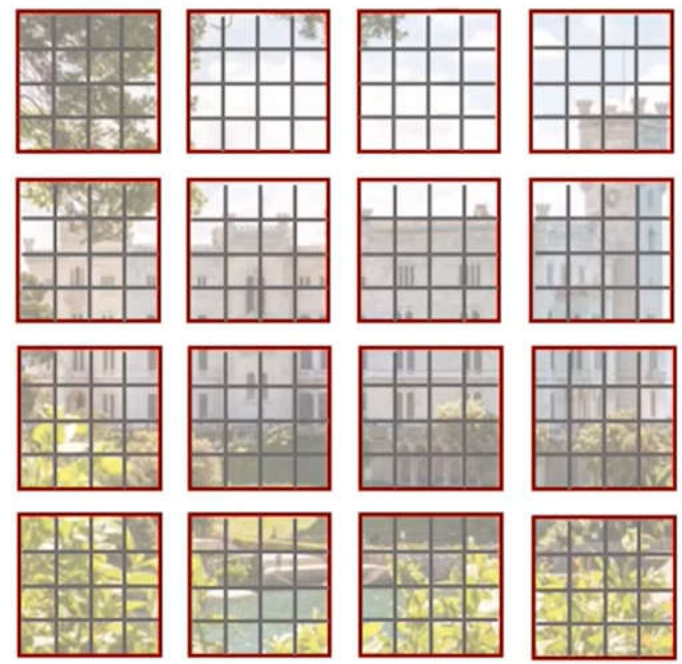
W-MSA

Locality by non-overlapped windows

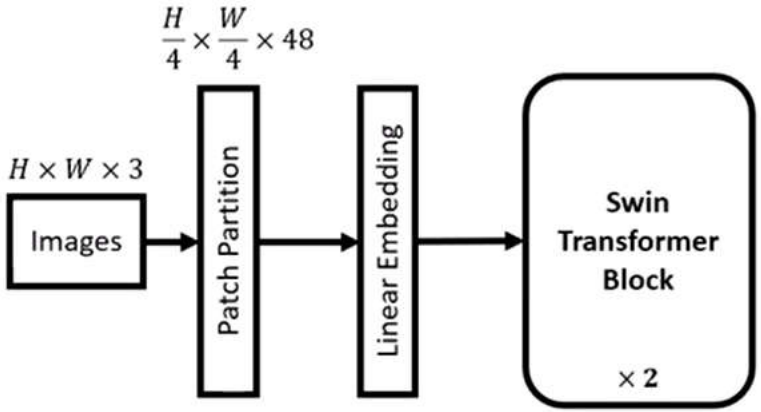
- Proves beneficial in modeling the high correlation in visual signals (Yann LeCun)

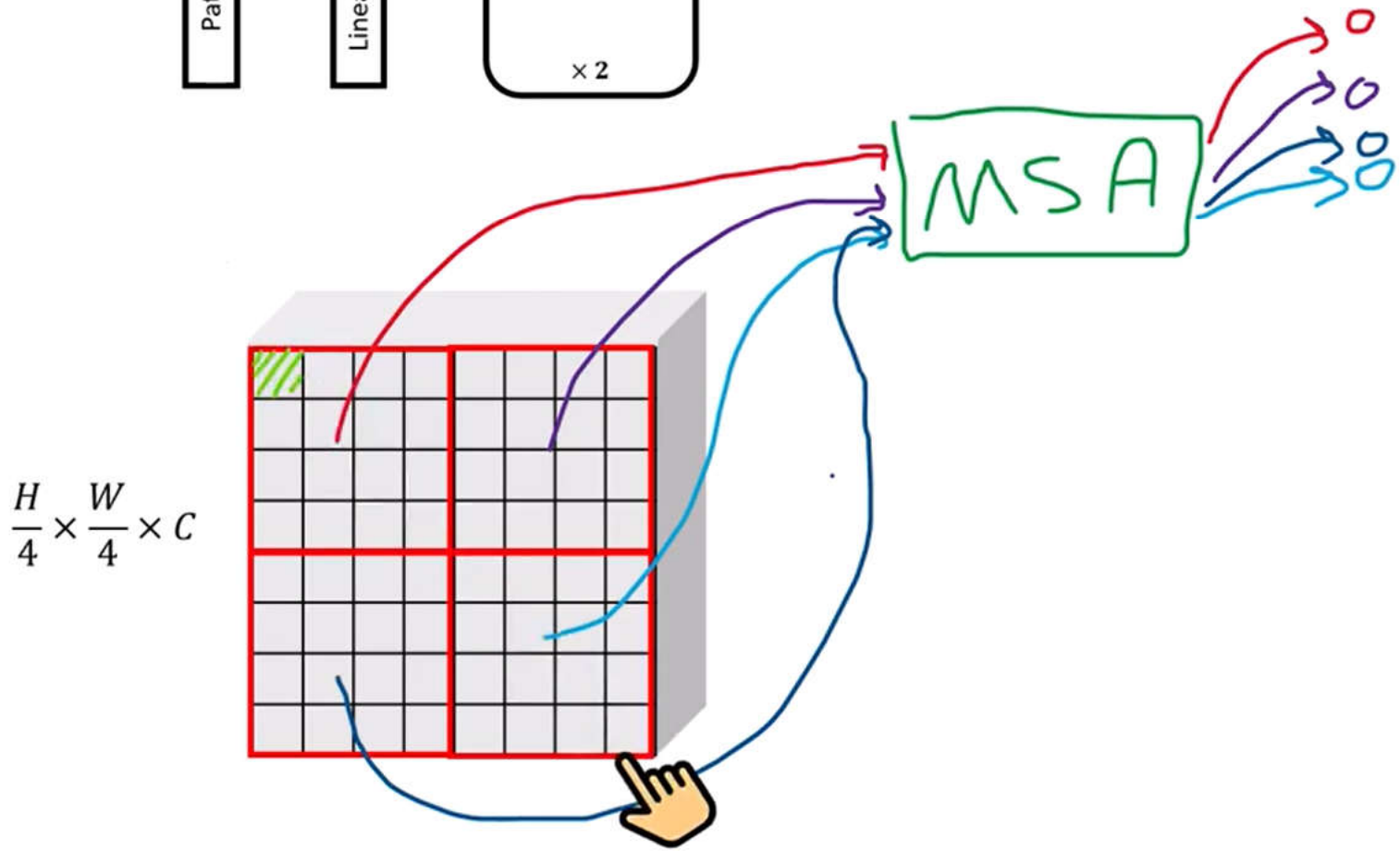


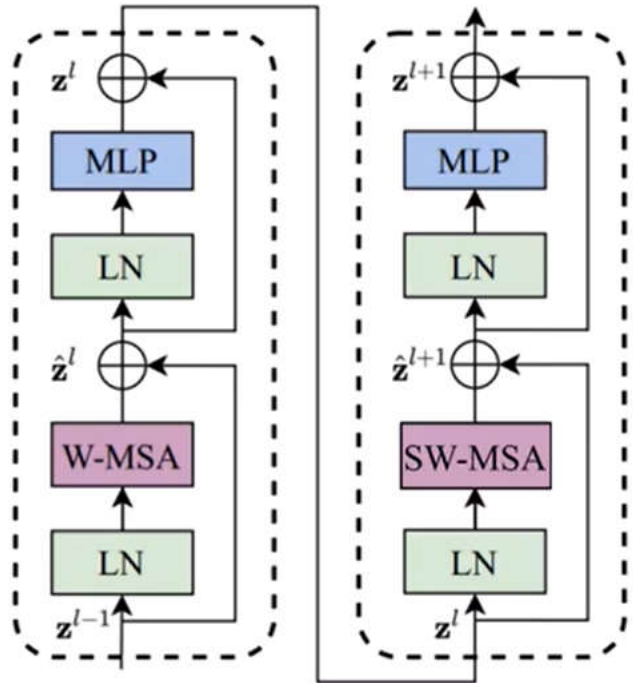
ViT: 256^2=65536 (Global)



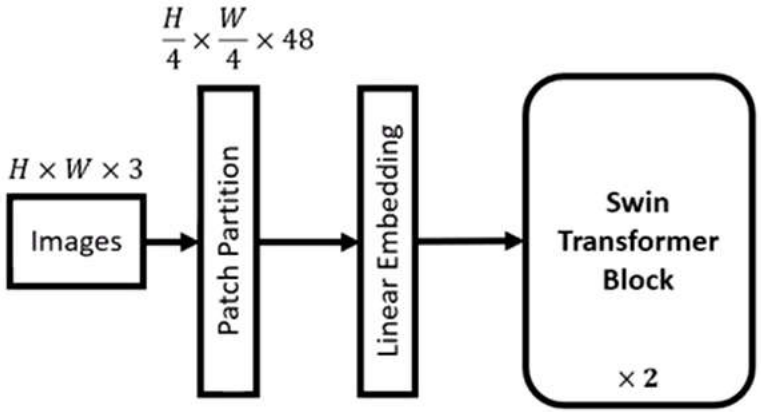
Swin Transformer: 16x16^2=4096 (Local)

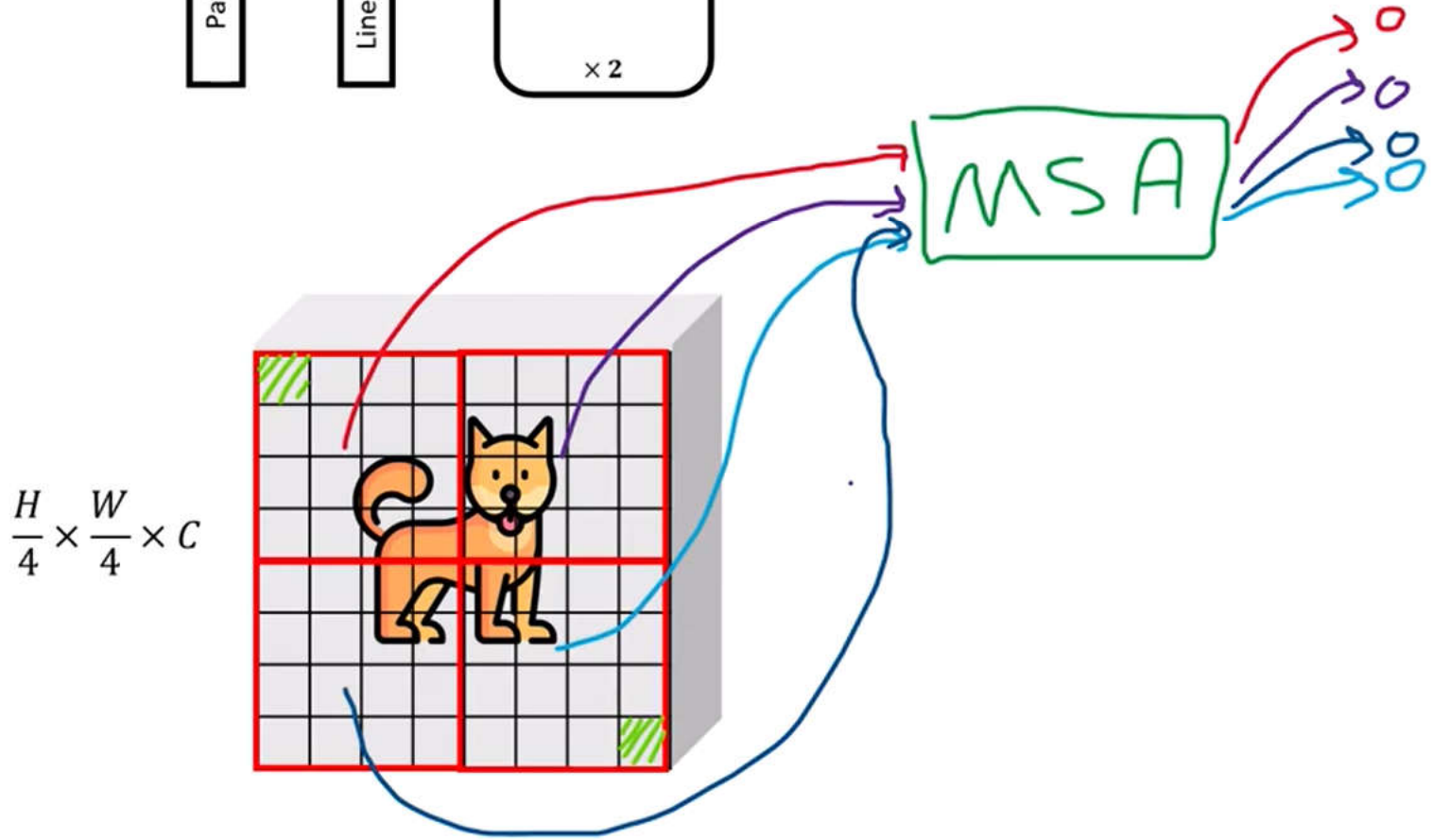

$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
$\times 2$

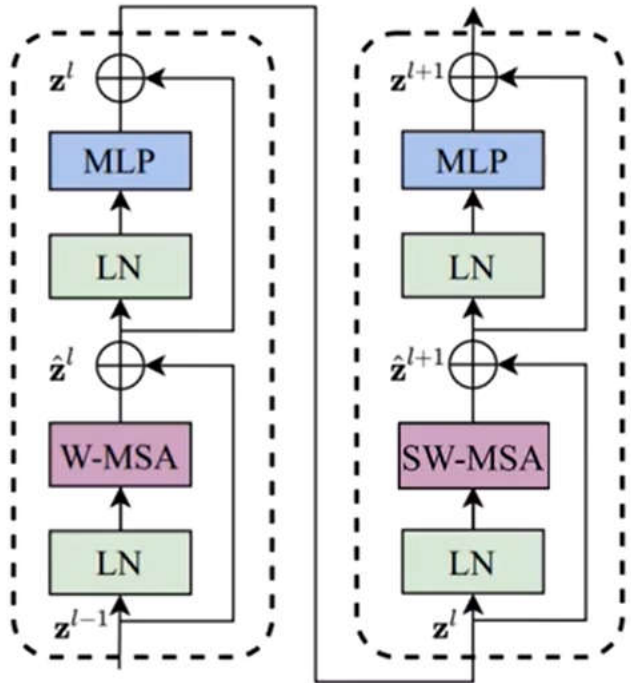

MSA
$\frac{H}{4} \times \frac{W}{4} \times C$


$\mathbf{z}^l$
MLP
LN
$\hat{\mathbf{z}}^l$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^l$

W-MSA

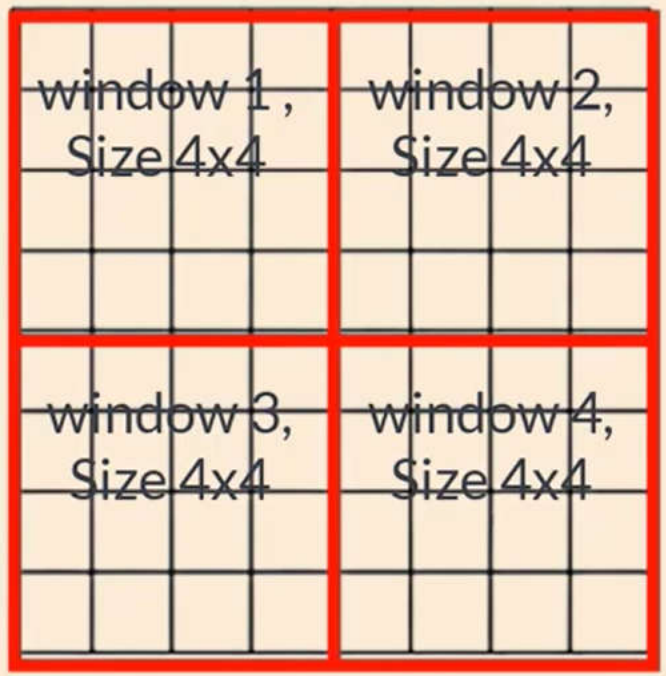

$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
$\times 2$


MSA
$\frac{H}{4} \times \frac{W}{4} \times C$

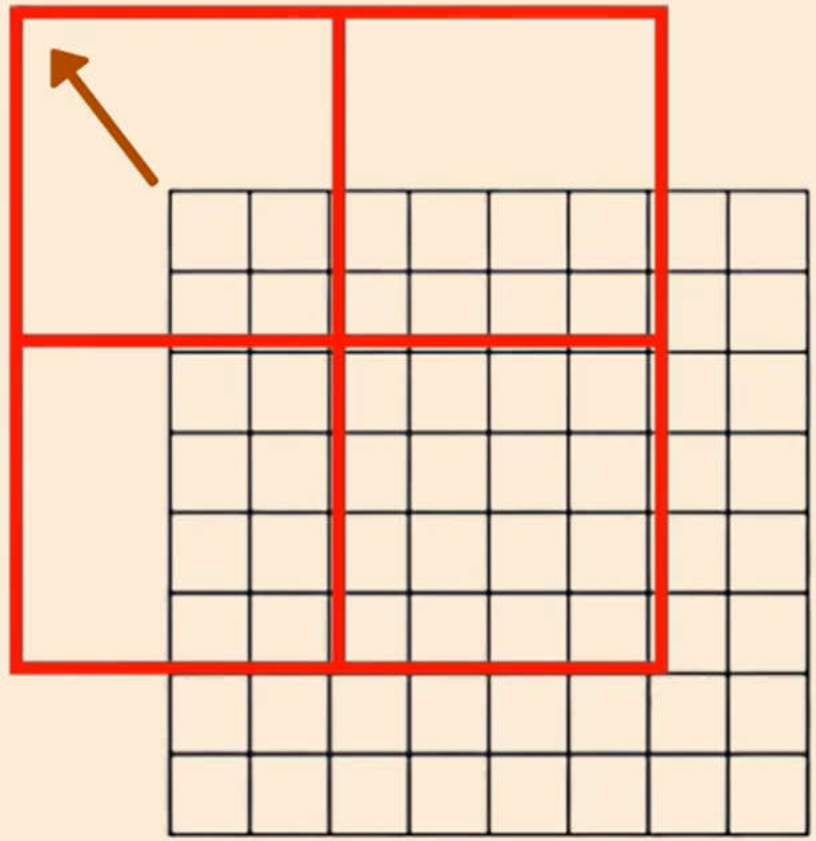

$\mathbf{z}^l$
MLP
LN
$\hat{\mathbf{z}}^l$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^l$

W-MSA

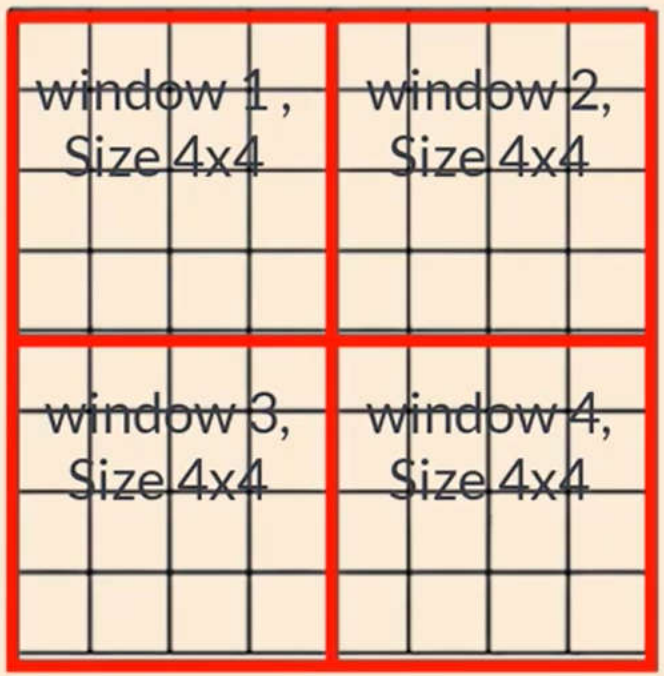
Shifted Window self attenion:



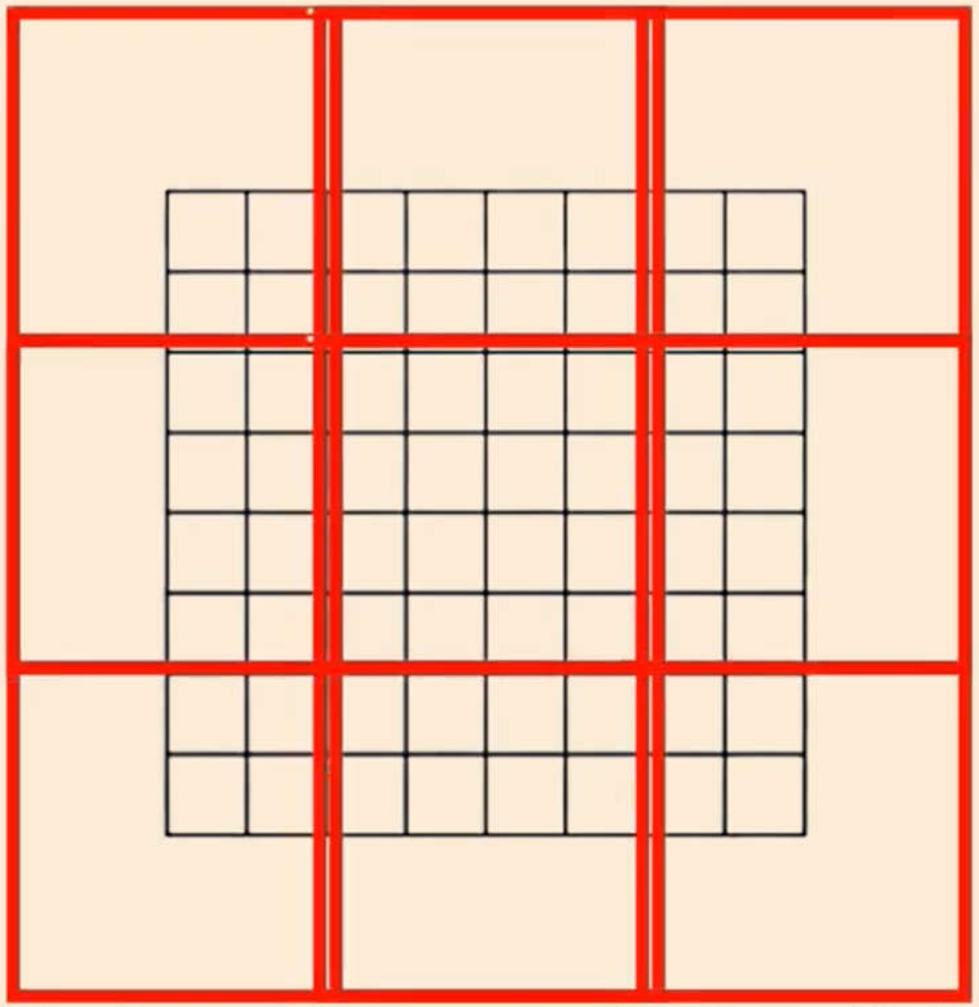
8*8 feature map



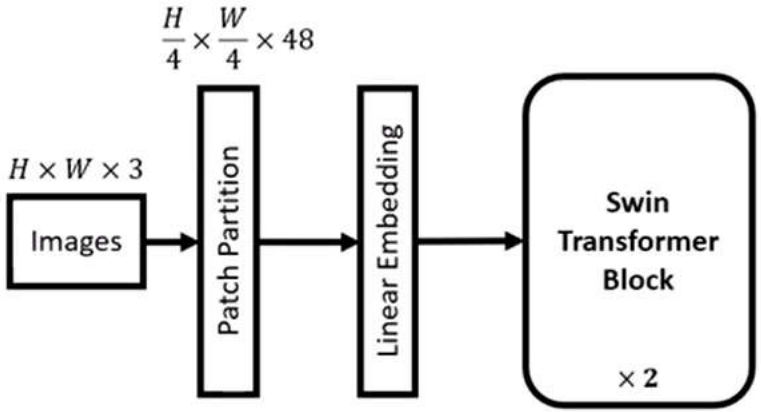
Shifted Window self attenion:

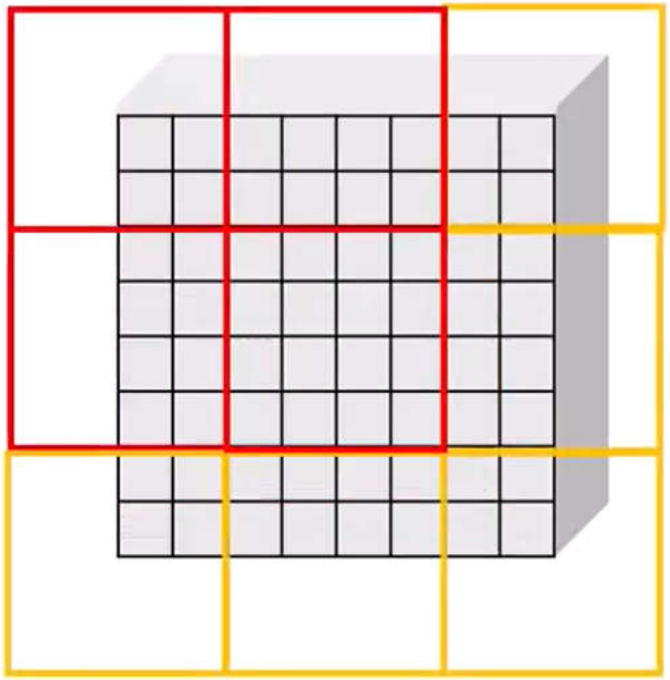


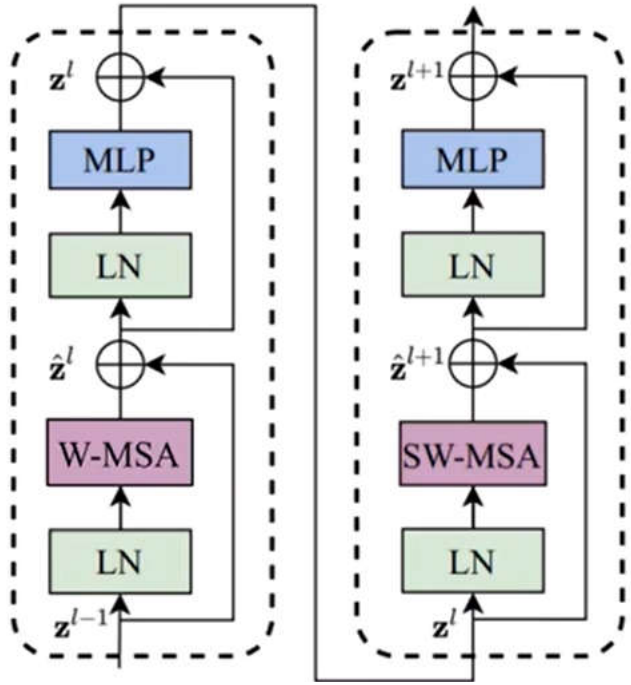
8*8 feature map



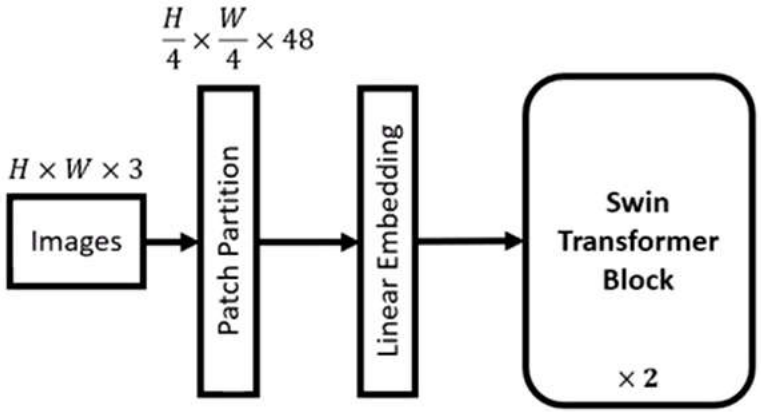
displacing the windows by ([M/2],[M/2]) pixels from the regularly partitioned windows. So the shift size is **2** in this example since the window size *M=4*

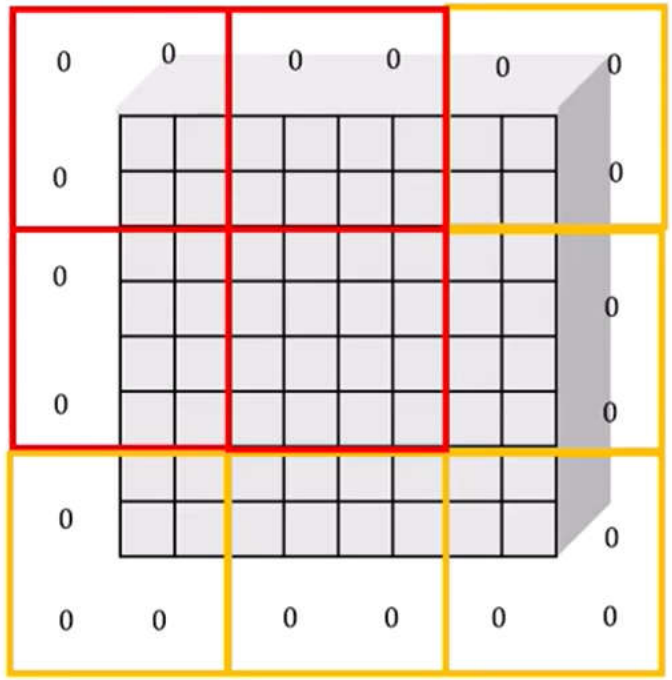

$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
$\times 2$

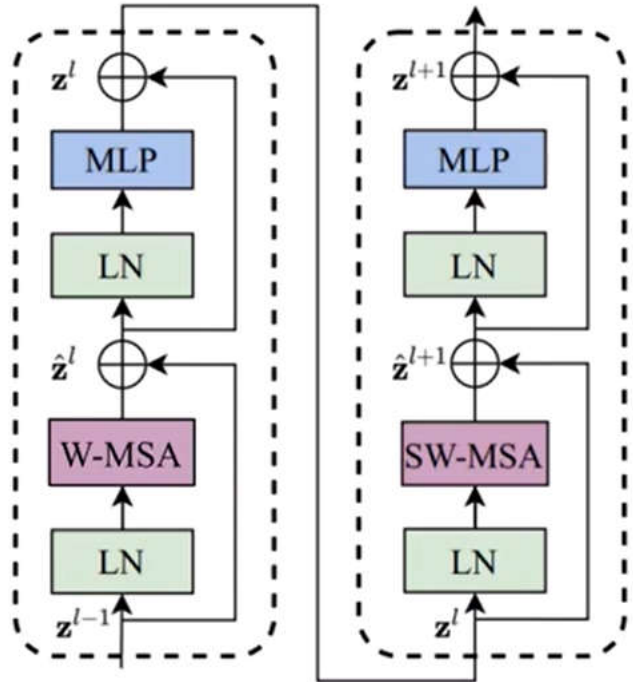



$\mathbf{z}^{l}$
MLP
LN
$\hat{\mathbf{z}}^{l}$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^{l}$

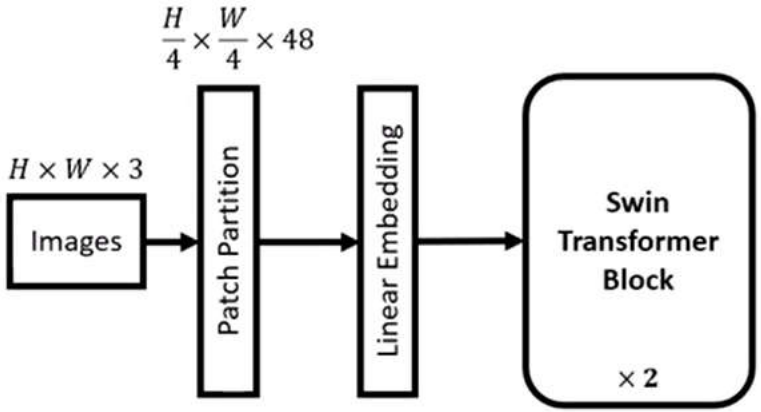
SW-MSA

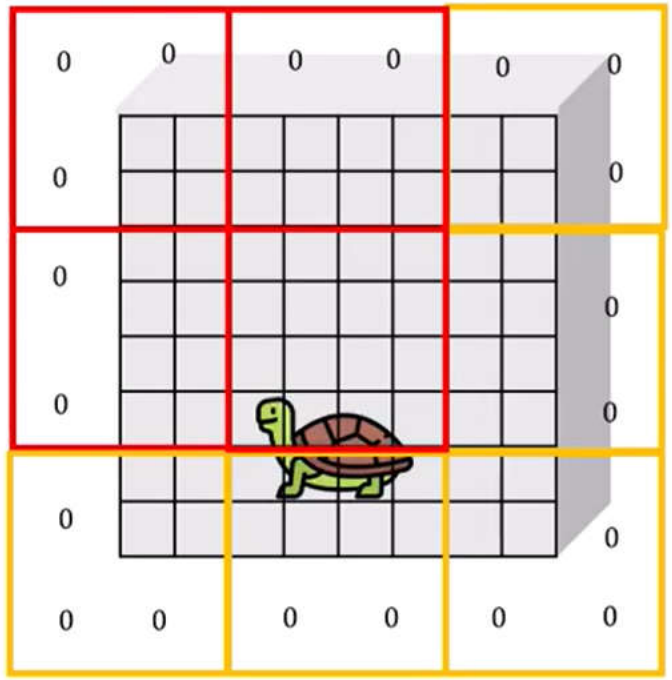

$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
$\times 2$

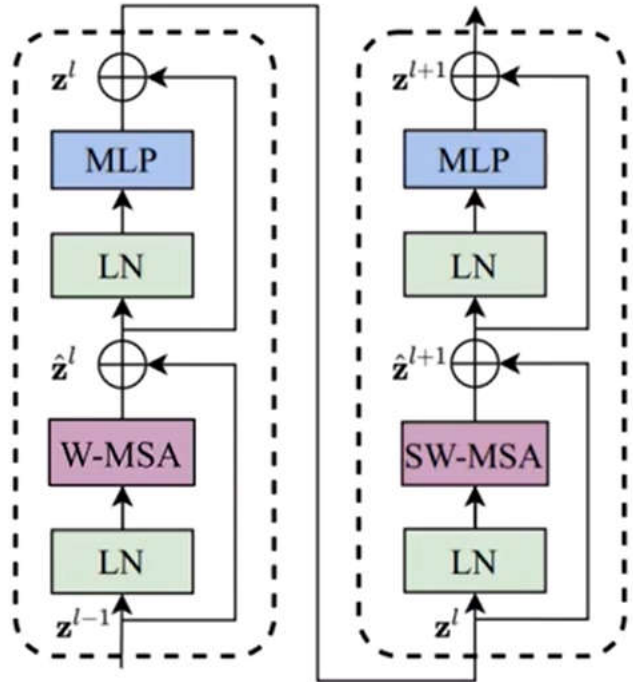

0


$\mathbf{z}^l$
MLP
LN
$\hat{\mathbf{z}}^l$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^l$

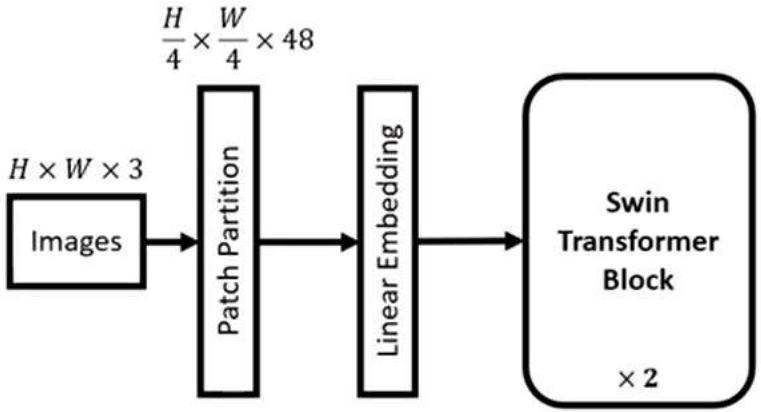
SW-MSA

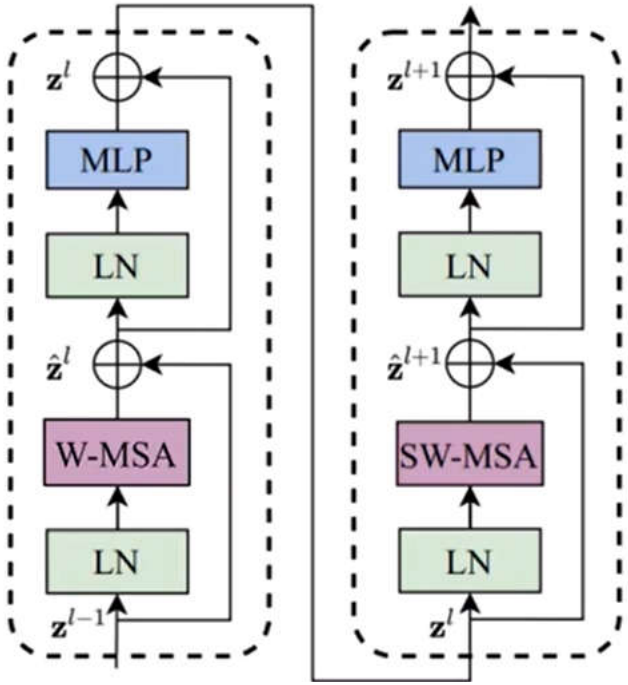

$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
$\times 2$

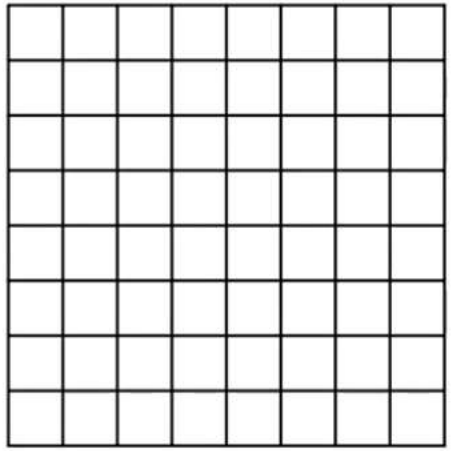

0


$\mathbf{z}^l$
MLP
LN
$\hat{\mathbf{z}}^l$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^l$

SW-MSA

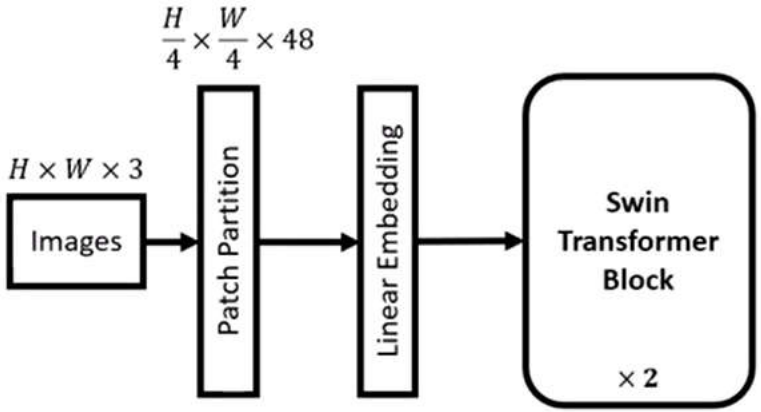


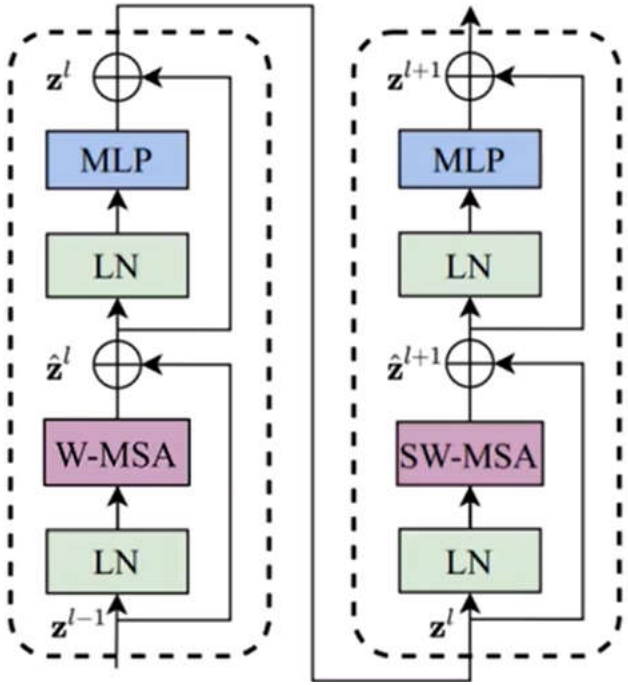


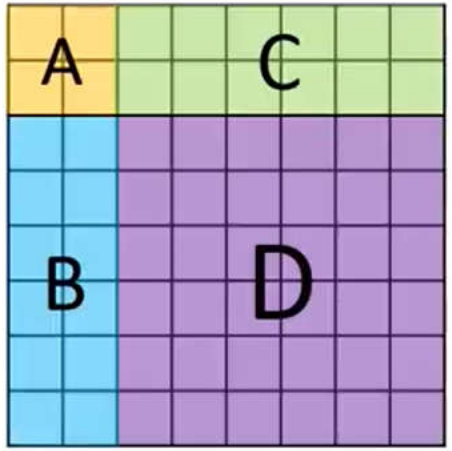


Original Patches

SW-MSA

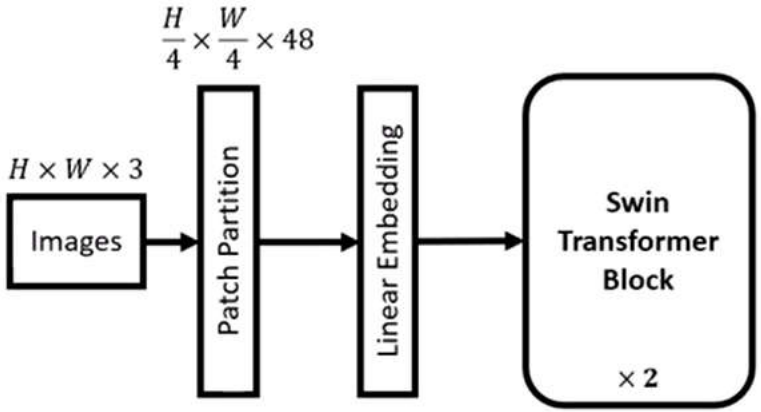


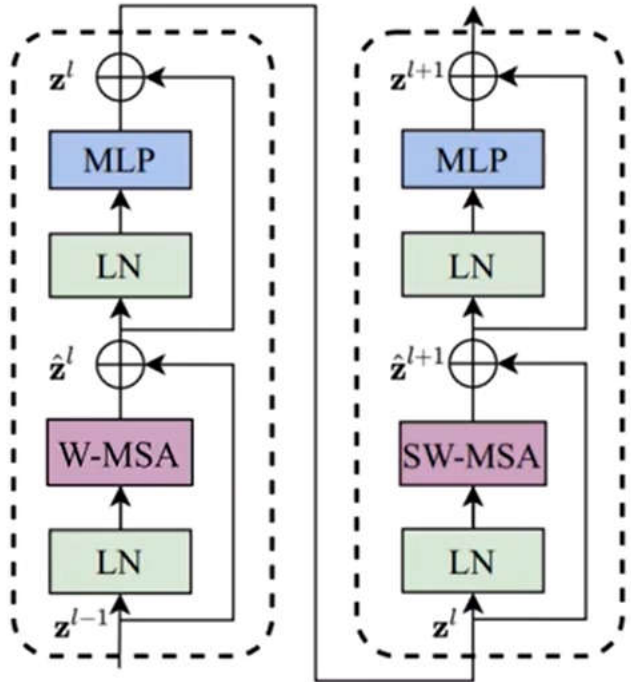


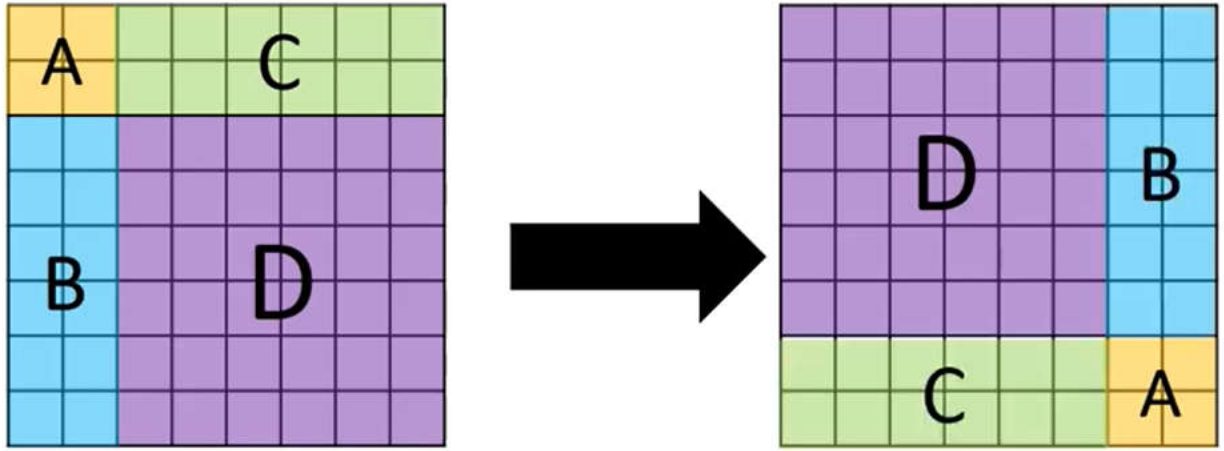


Original Patches

SW-MSA



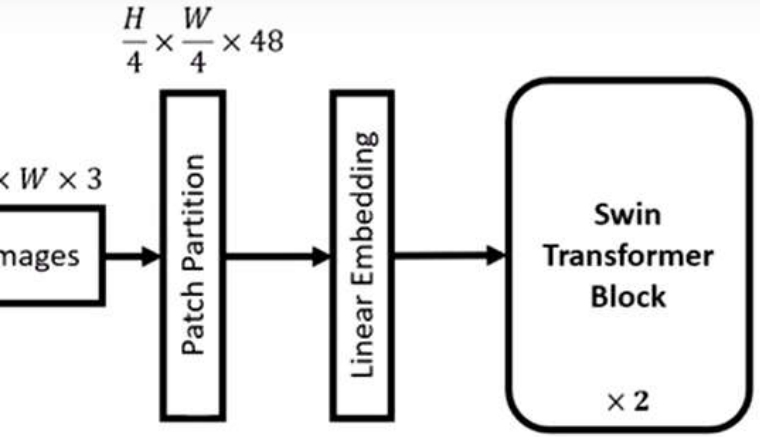




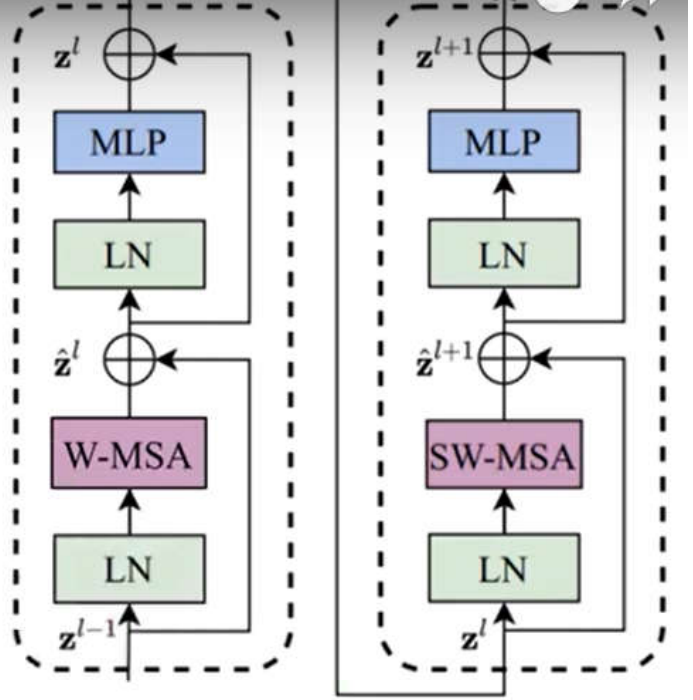
Original Patches

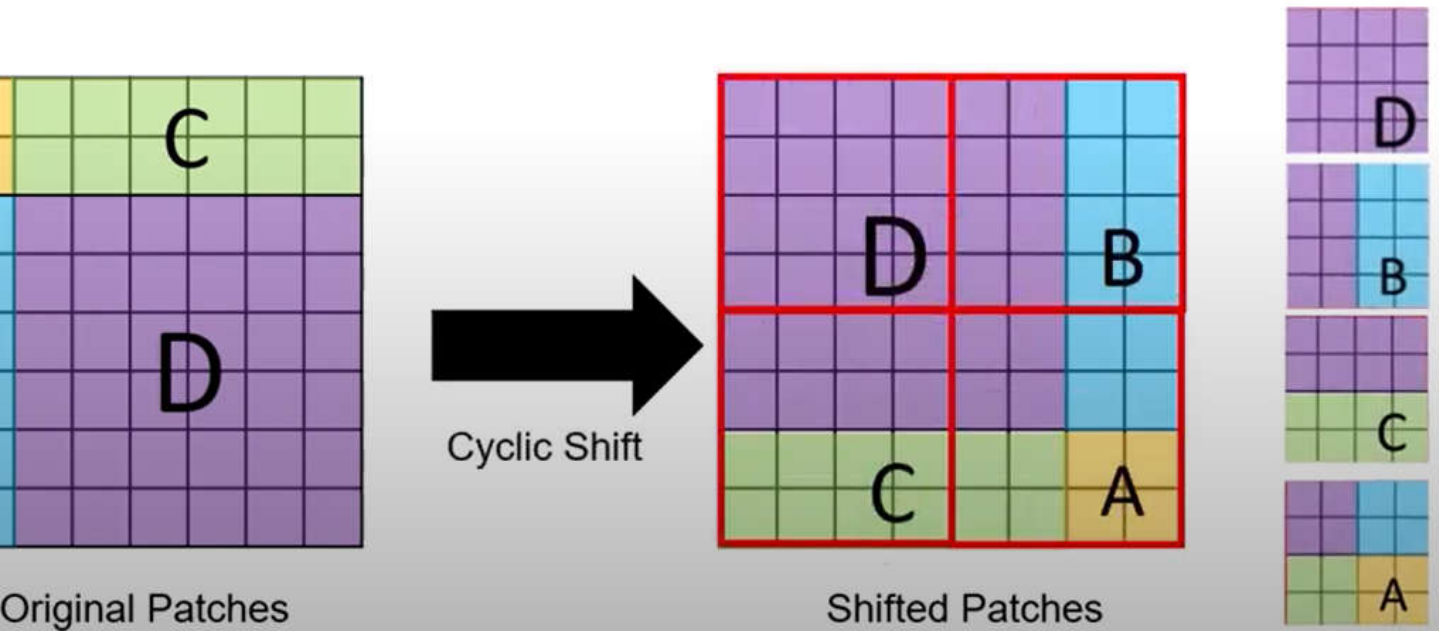
Shifted Patches

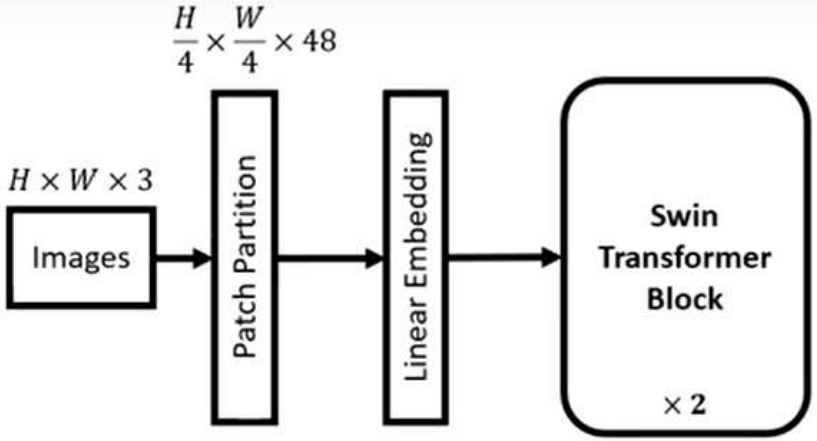
SW-MSA


$\frac{H}{4} \times \frac{W}{4} \times 48$
$\times W \times 3$
mages
Patch Partition
Linear Embedding
Swin Transformer Block
$\times 2$

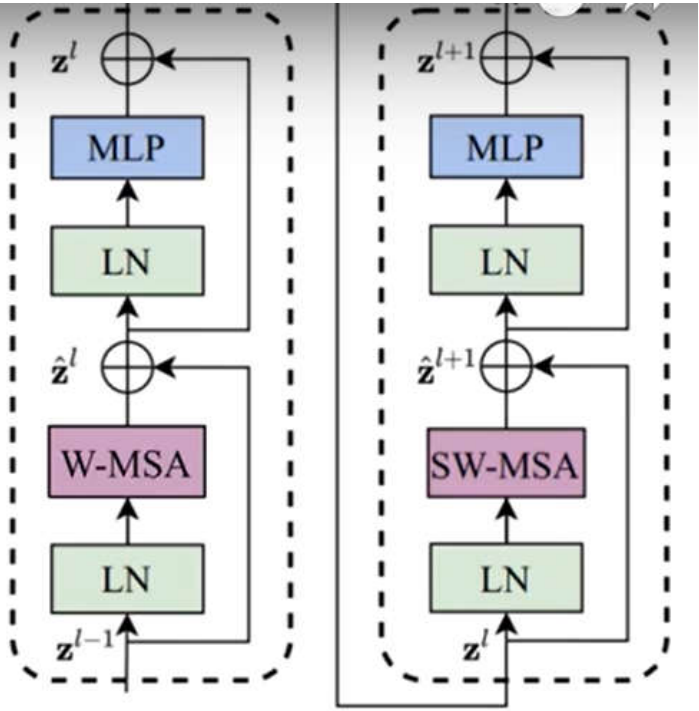

SW-MSA

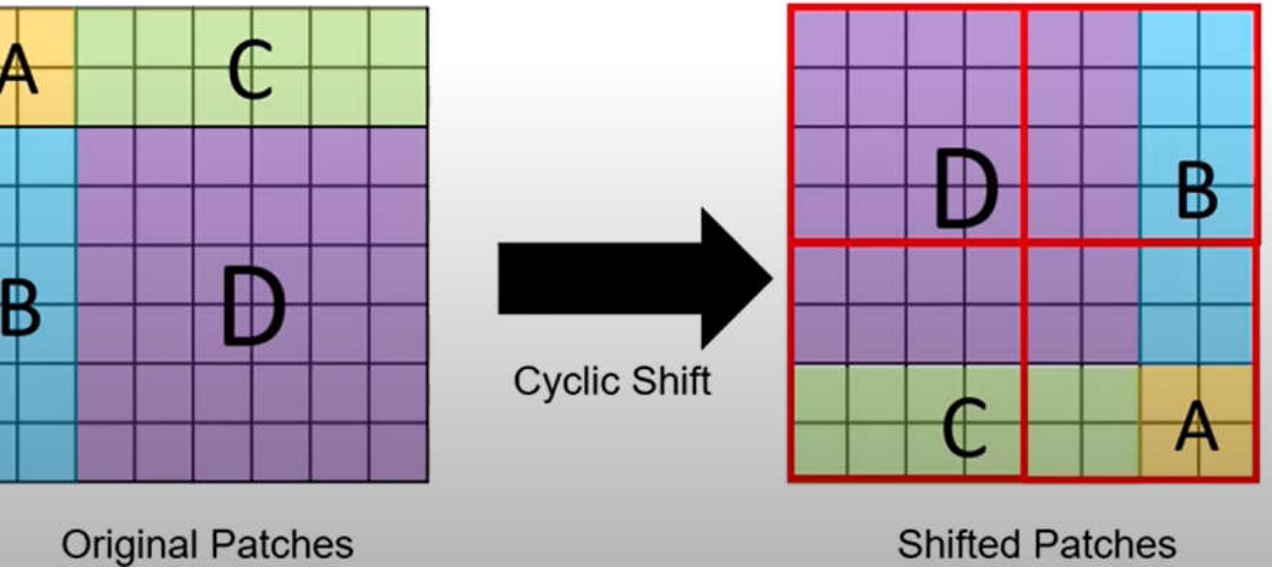

$\mathbf{z}^{l}$
MLP
LN
$\hat{\mathbf{z}}^{l}$
W-MSA
LN
$\mathbf{z}^{l-1}$
$\mathbf{z}^{l+1}$
MLP
LN
$\hat{\mathbf{z}}^{l+1}$
SW-MSA
LN
$\mathbf{z}^{l}$

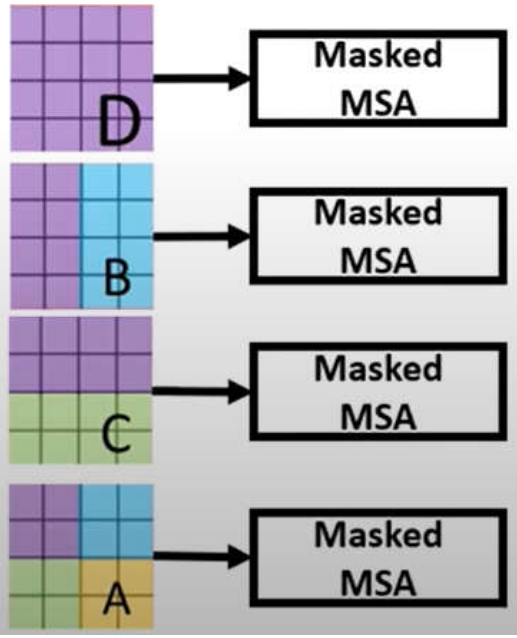

C
D
Cyclic Shift
D
B
C
A
D
B
C
A
Original Patches
Shifted Patches

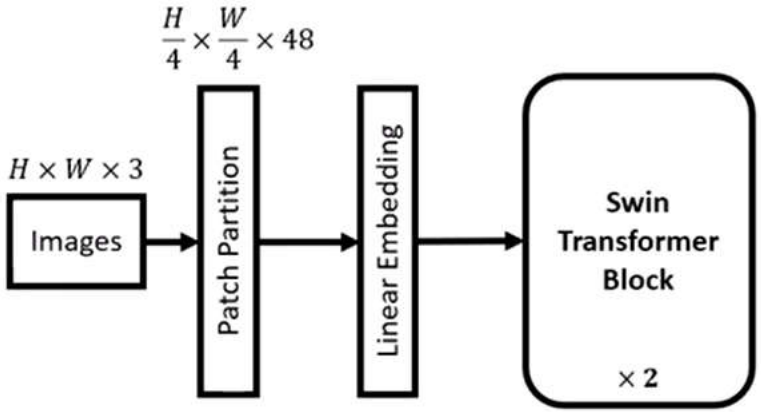

$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
× 2

SW-MSA

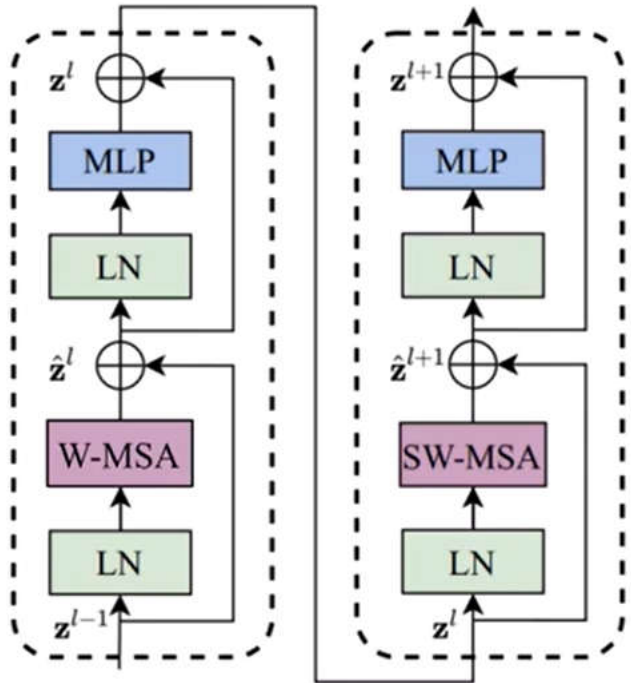

z^l
MLP
LN
$\hat{z}^l$
W-MSA
LN
z^{l-1}
z^{l+1}
MLP
LN
$\hat{z}^{l+1}$
SW-MSA
LN
z^l

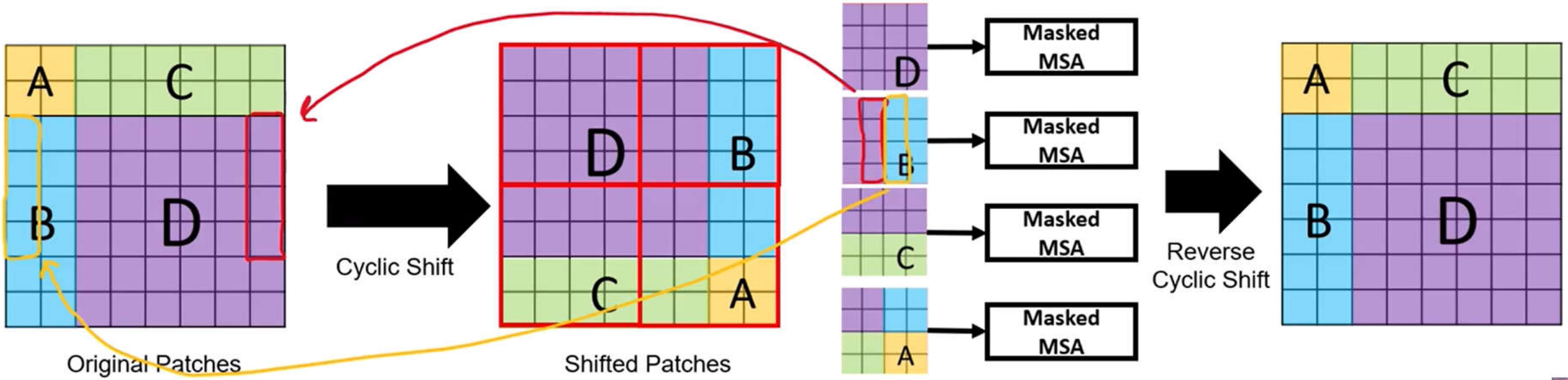

A
C
B
D
Cyclic Shift
D
B
C
A
Original Patches
Shifted Patches


D
B
C
A
Masked MSA
Masked MSA
Masked MSA
Masked MSA


$\frac{H}{4} \times \frac{W}{4} \times 48$
$H \times W \times 3$
Images
Patch Partition
Linear Embedding
Swin Transformer Block
× 2

SW-MSA


z^l
MLP
LN
$\hat{z}^l$
W-MSA
LN
z^{l-1}
z^{l+1}
MLP
LN
$\hat{z}^{l+1}$
SW-MSA
LN
z^l


A
C
B
D
Cyclic Shift
D
B
C
A
Original Patches
Shifted Patches
D
B
C
A
Masked MSA
Masked MSA
Masked MSA
Masked MSA
Reverse Cyclic Shift
A
C
B
D

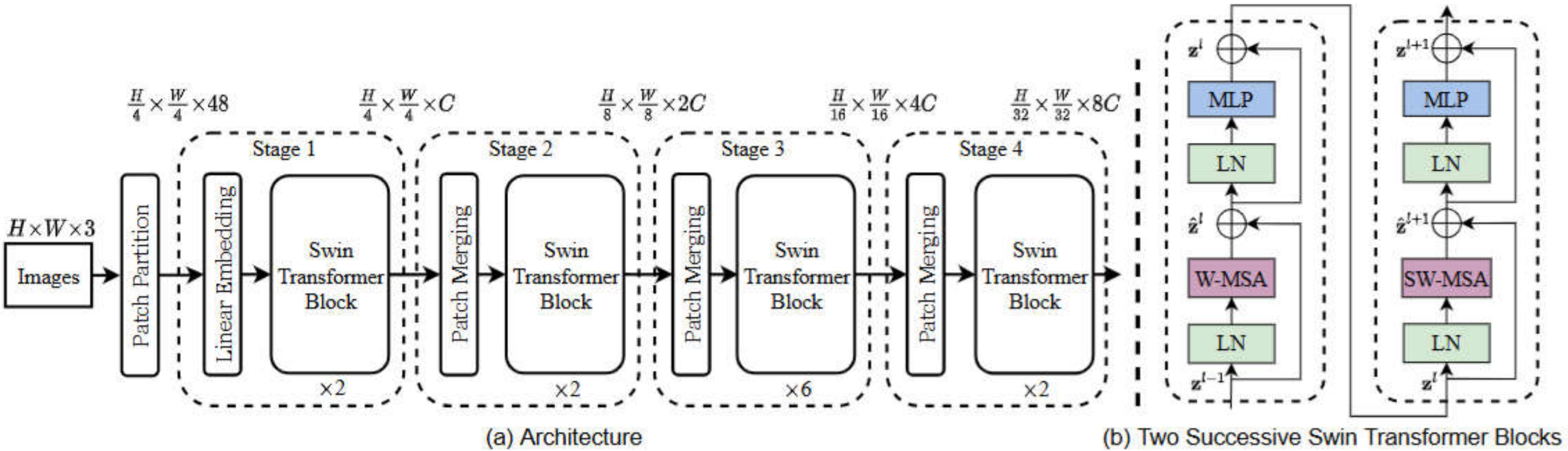
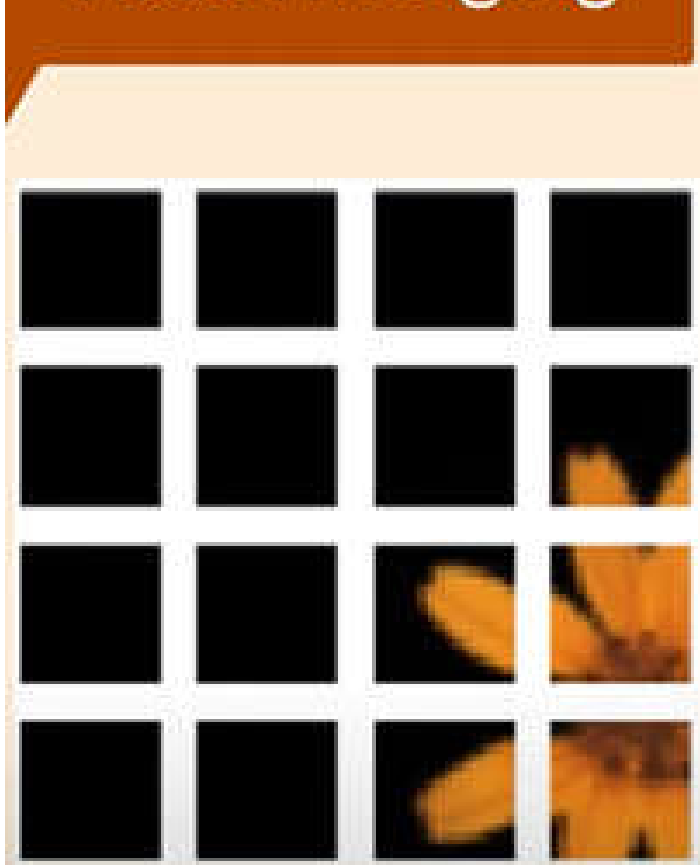
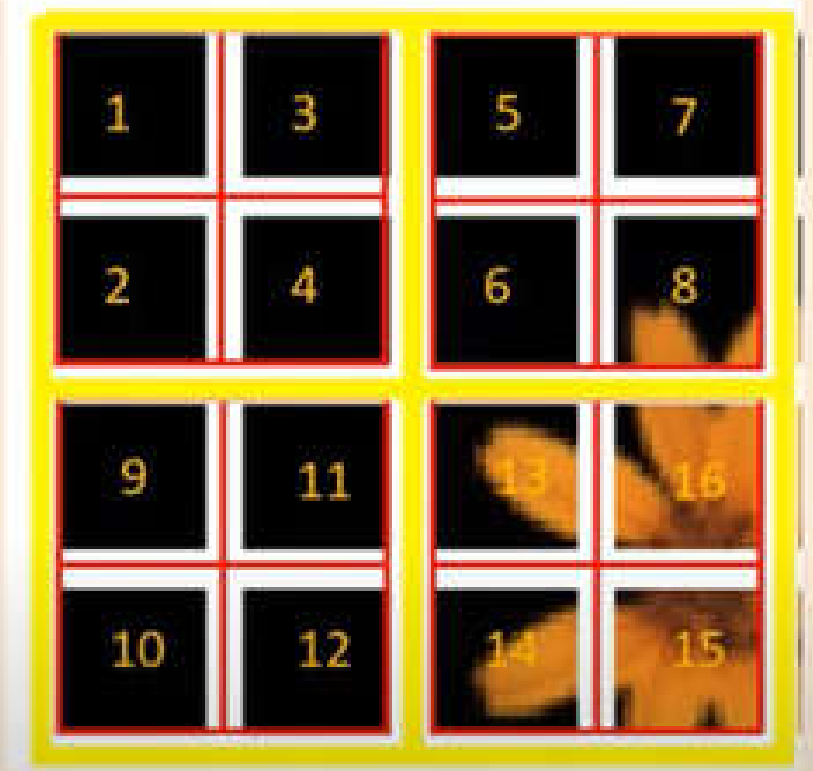
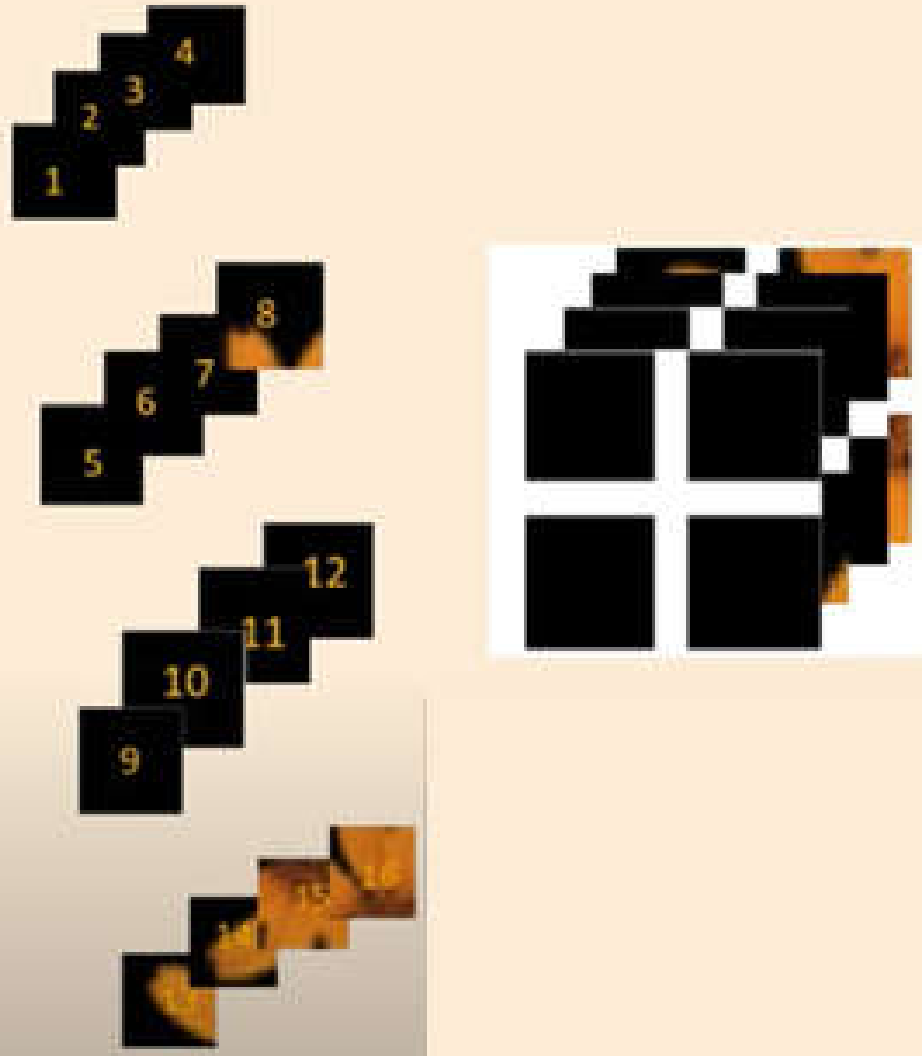


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Patch merging

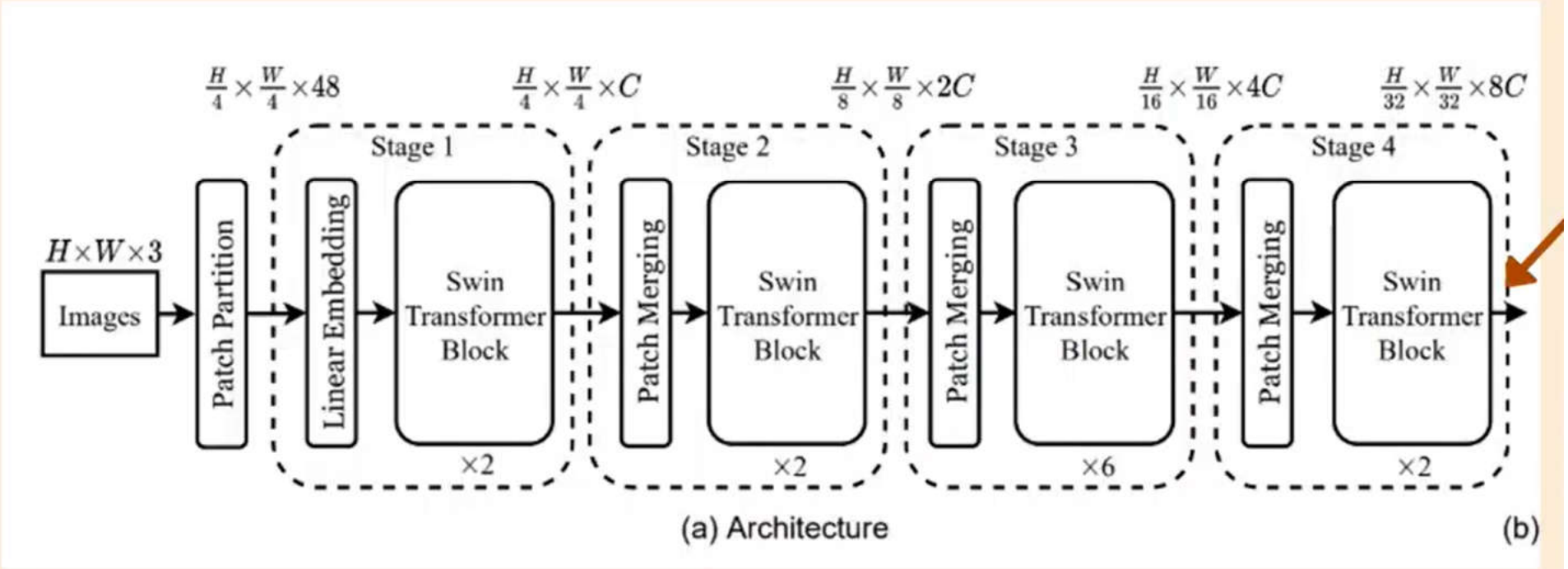




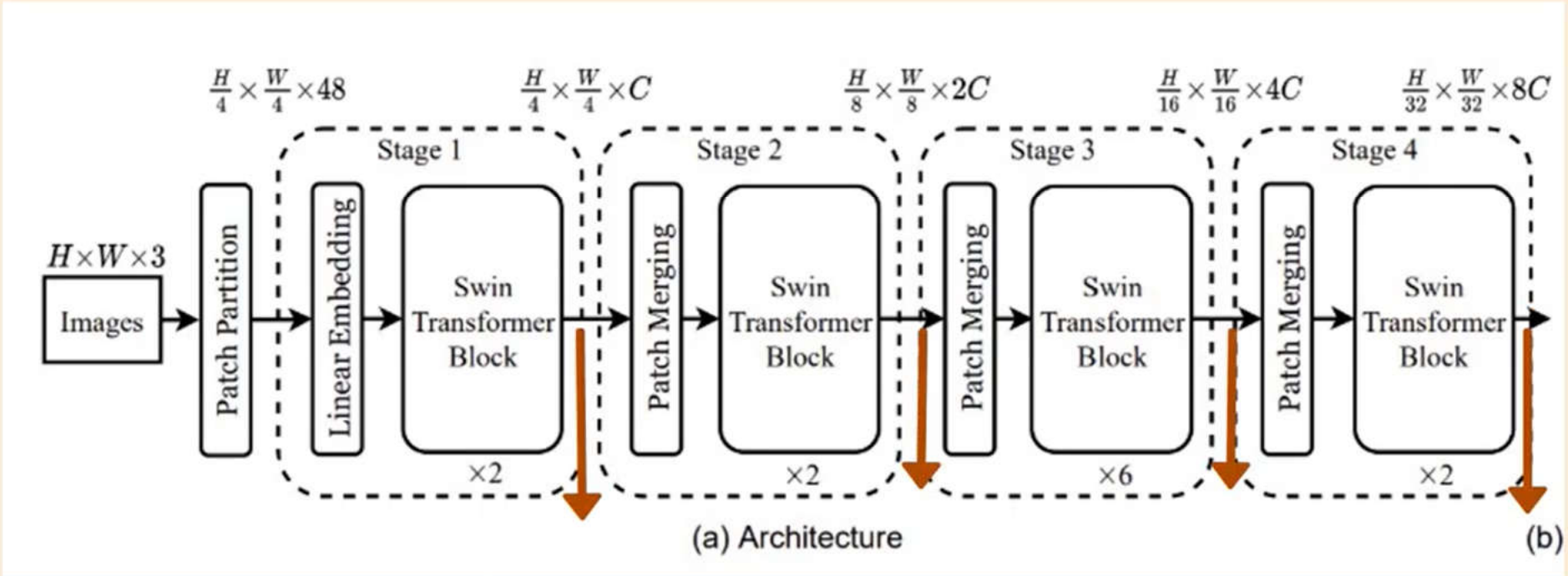


Split image into 2*2 grid

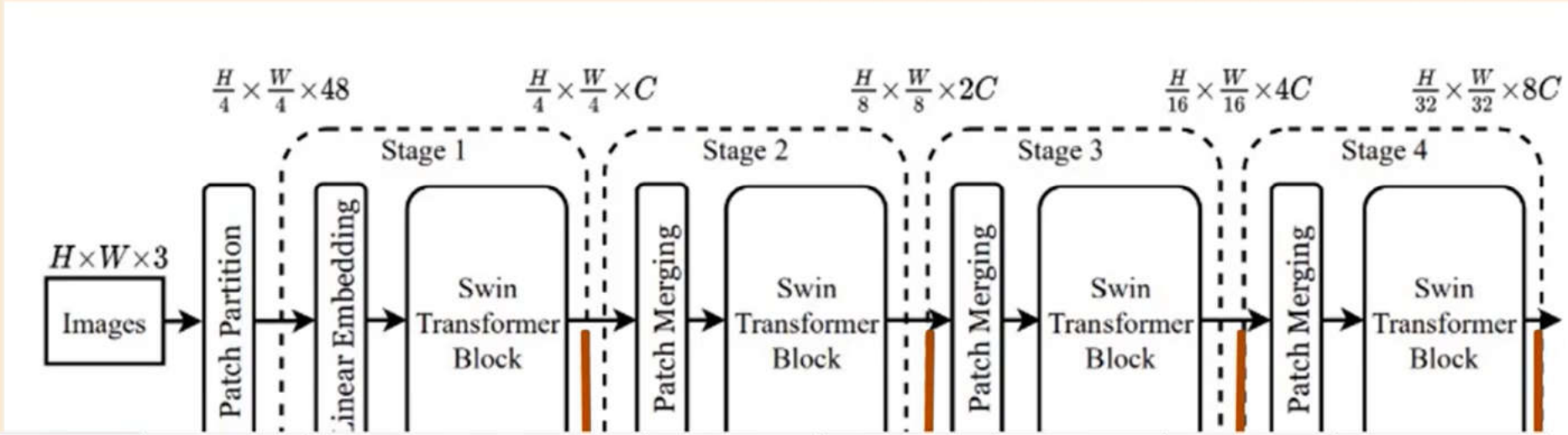
Task Specific Head



Task Specific Head



Task Specific Head



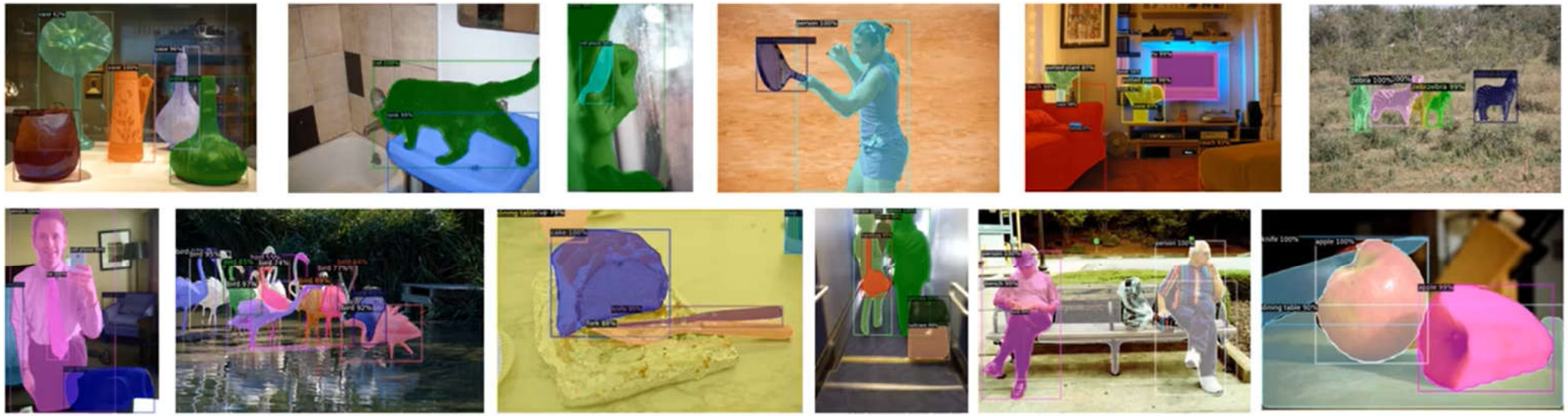
SWIN Variant	Window size, M	C, Num of hidden layers in Stage1	Blocks in stage 1	Blocks in stage 2	Blocks in stage 3	Blocks in stage 4	Comparable Models
SWIN-T	7	96	2	2	6	2	DeiT-S / ResNet50
SWIN-S	7	96	2	2	18	2	ResNet101
SWIN-B	7	128	2	2	18	2	-
SWIN-L	7	192	2	2	18	2	-

Application: ImageNet-1K classification

- Swin Transformer surpasses ViT-L/16 with the same pre-training data

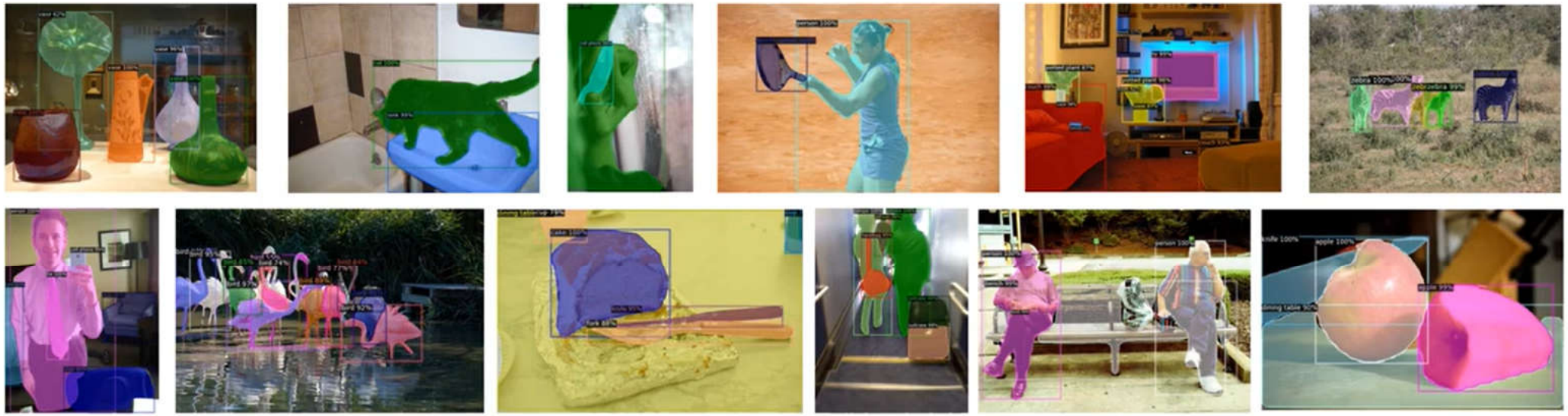
Model	data	params	FPS	ImageNet-1K (top-1)
EfficientNet-L2	JFT-300M	480M	--	85.5
ViT-L/16	ImageNet-14M	307M	27	85.15
ViT-H/14	JFT-300M	632M	10	88.55
ViT-G/14	JFT-3B	3B	-	90.45
Swin (Huge)	**ImageNet-14M**	**391M**	**25**	**88.65 (+3.5)**

Application: object detection



- COCO object detection: #1 #2 #3 for single model (60.6 mAP)
  - Significantly surpass all previous CNN models (+3.5 mAP)
- COCO instance segmentation: #1 for single model (52.4 mAP)
  - Significantly surpass all previous CNN models (+3.3 mAP)

Application: object detection



- COCO object detection: #1 #2 #3 for single model (60.6 mAP)
  - Significantly surpass all previous CNN models (+3.5 mAP)
- COCO instance segmentation: #1 for single model (52.4 mAP)
  - Significantly surpass all previous CNN models (+3.3 mAP)

Application: object detection

- Performs consistently better than CNN on various object detectors and various model sizes (+3~4.5 mAP)

(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	+4.2
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	+3.7
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	+3.5
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	+3.4

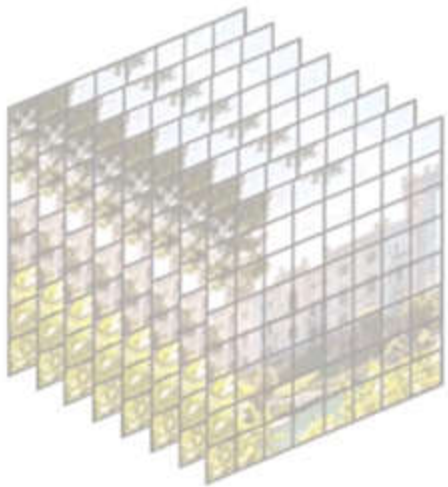
(b) Various backbones w. Cascade Mask R-CNN

	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	+4.2
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	+3.7
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	+3.6

Application: semantic segmentation

- **ADE20K semantic segmentation: #1 for single model (53.9 mIoU)**
  - The largest and most difficult semantic segmentation benchmark
    - 20,000 training images, 150 categories
  - Significantly surpass all previous CNN models (+5.5 mIoU vs. the previous best CNN model)

Application: video recognition



3D tokens: T'×H'×W' = 8×8×8
Window size: P×M×M = 4×4×4

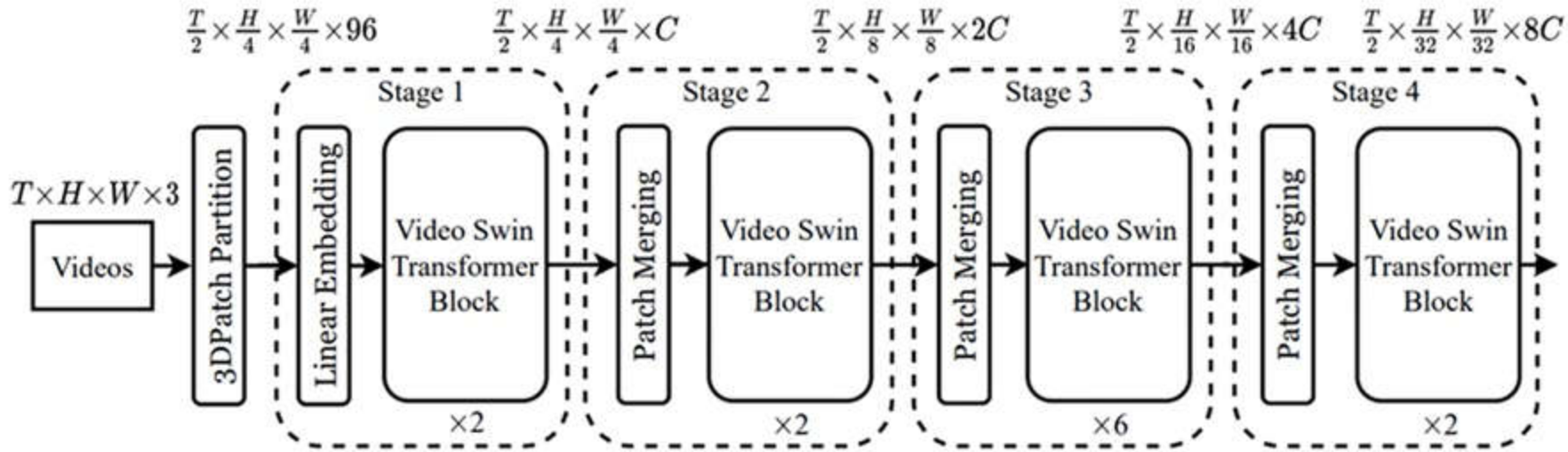


Figure 2: Overall architecture of Video Swin Transformer (tiny version, referred to as Swin-T).

Application: video recognition

- Swin Transformer achieves SOTA on major video benchmarks with 20x less pre-training data and 3x smaller model size

Table 1: Comparison to state-of-the-art on Kinetics-400. "384↑" signifies that the model uses a larger spatial resolution of 384×384. "Views" indicates # temporal clip × # spatial crop. The magnitudes are Giga (10^9) and Mega (10^6) for FLOPs and Param respectively.

Method	Pretrain	Top-1	Top-5	Views	FLOPs	Param
R(2+1)D [37]	-	72.0	90.0	10 × 1	75	61.8
I3D [6]	ImageNet-1K	72.1	90.3	-	108	25.0
NL I3D-101 [40]	ImageNet-1K	77.7	93.3	10 × 3	359	61.8
ip-CSN-152 [36]	-	77.8	92.8	10 × 3	109	32.8
CorrNet-101 [39]	-	79.2	-	10 × 3	224	-
SlowFast R101+NL [13]	-	79.8	93.9	10 × 3	234	59.9
X3D-XXL [12]	-	80.4	94.6	10 × 3	144	20.3
MViT-B, 32×3 [10]	-	80.2	94.4	1 × 5	170	36.6
MViT-B, 64×3 [10]	-	81.2	95.1	3 × 3	455	36.6
TimeSformer-L [3]	ImageNet-21K	80.7	94.7	1 × 3	2380	121.4
ViT-B-VTN [29]	ImageNet-21K	78.6	93.7	1 × 1	4218	11.04
ViViT-L/16x2 [1]	ImageNet-21K	80.6	94.7	4 × 3	1446	310.8
ViViT-L/16x2 320 [1]	ImageNet-21K	81.3	94.7	4 × 3	3992	310.8
ip-CSN-152 [36]	IG-65M	82.5	95.3	10 × 3	109	32.8
ViViT-L/16x2 [1]	JFT-300M	82.8	95.5	4 × 3	1446	310.8
ViViT-L/16x2 320 [1]	JFT-300M	83.5	95.5	4 × 3	3992	310.8
ViViT-H/16x2 [1]	JFT-300M	84.8	95.8	4 × 3	8316	647.5
Swin-T	ImageNet-1K	78.8	93.6	4 × 3	88	28.2
Swin-S	ImageNet-1K	80.6	94.5	4 × 3	166	49.8
Swin-B	ImageNet-1K	80.6	94.6	4 × 3	282	88.1
Swin-B	ImageNet-21K	82.7	95.5	4 × 3	282	88.1
Swin-L	ImageNet-21K	83.1	95.9	4 × 3	604	197.0
Swin-L (384↑)	ImageNet-21K	**84.9**	**96.6**	10 × 5	2107	200.0

+3.6% using the same pre-training data

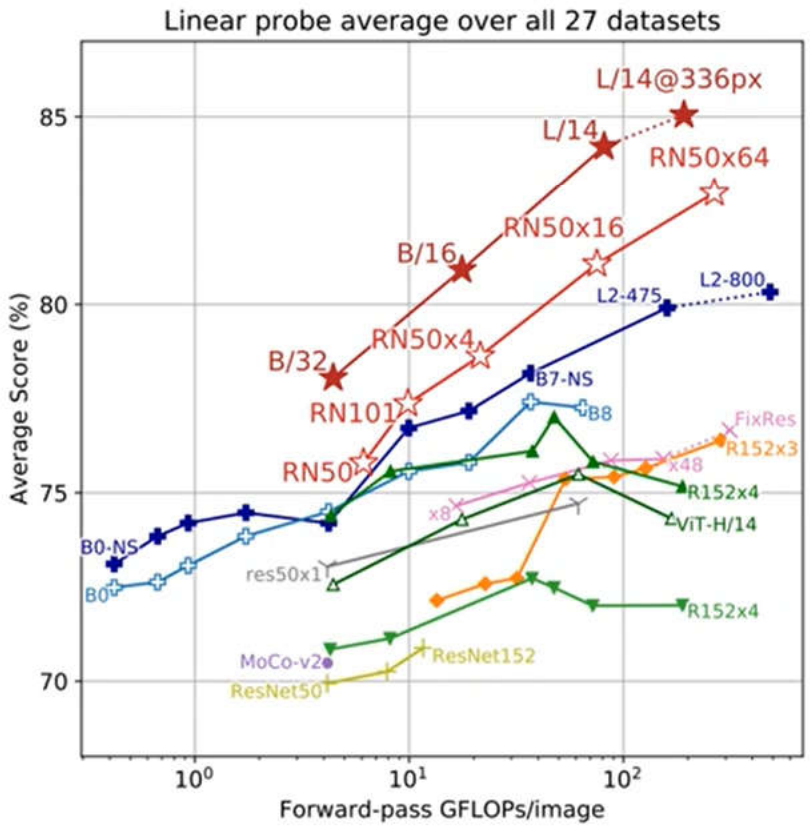
Table 2: Comparison to state-of-the-art on Kinetics-600.

Method	Pretrain	Top-1	Top-5	Views	FLOPs	Param
SlowFast R101+NL [13]	-	81.8	95.1	10 × 3	234	59.9
X3D-XL [12]	-	81.9	95.5	10 × 3	48	11.0
MViT-B-24, 32×3 [9]	-	83.8	96.3	5 × 1	236	52.9
TimeSformer-HR [3]	ImageNet-21K	82.4	96	1 × 3	1703	121.4
ViViT-L/16x2 320 [1]	ImageNet-21K	83.0	95.7	4 × 3	3992	310.8
ViViT-H/16x2 [9]	JFT-300M	85.8	96.5	4 × 3	8316	647.5
Swin-B	ImageNet-21K	83.8	96.4	4 × 3	282	88.1
Swin-L (384↑)	ImageNet-21K	**85.9**	**97.1**	4 × 3	2107	200.0

+2.9% using the same pre-training data

Reason IV to use Transformer in computer vision

- Better connect vision and language: unified modeling

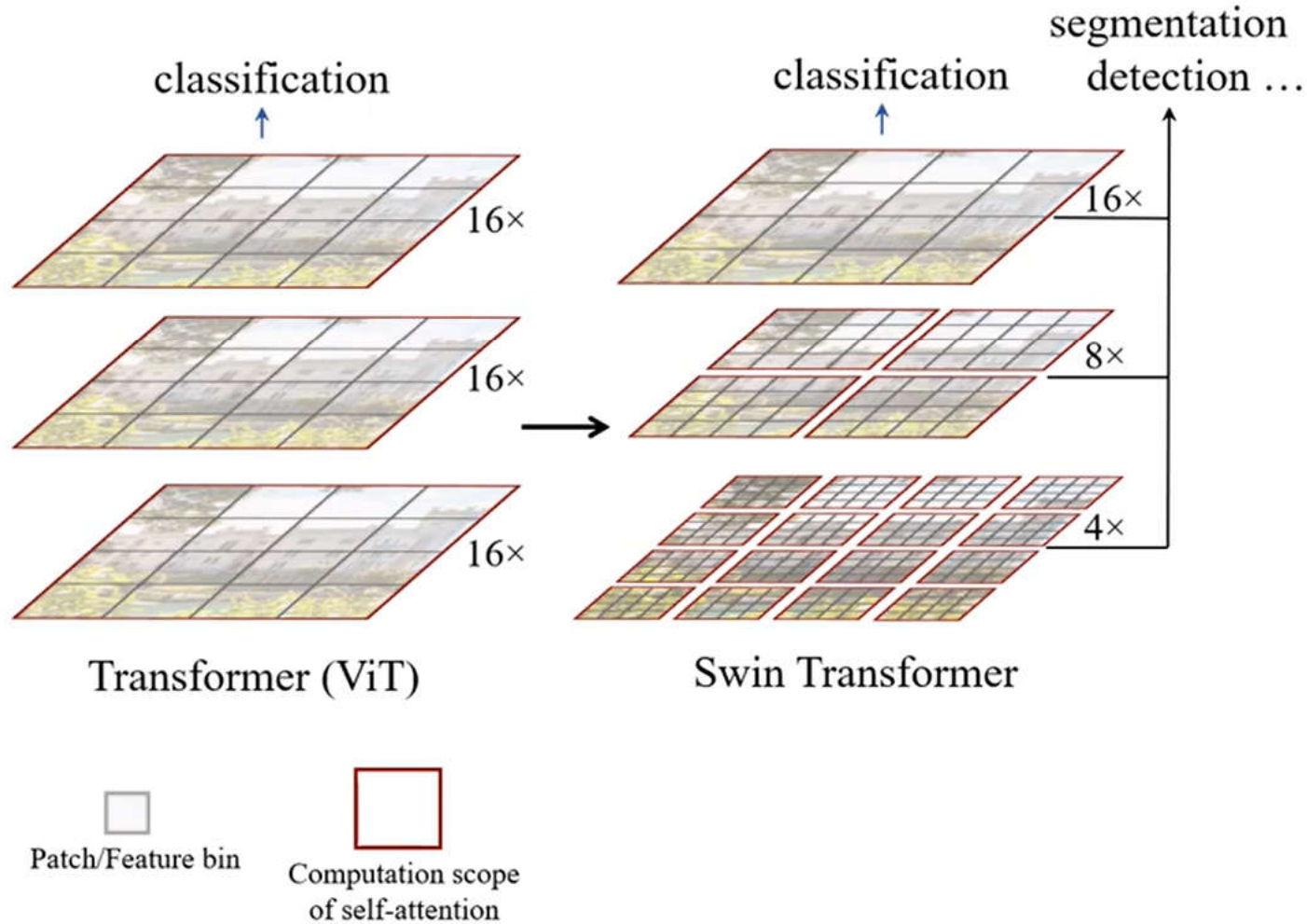


CLIP

https://openai.com/blog/clip/

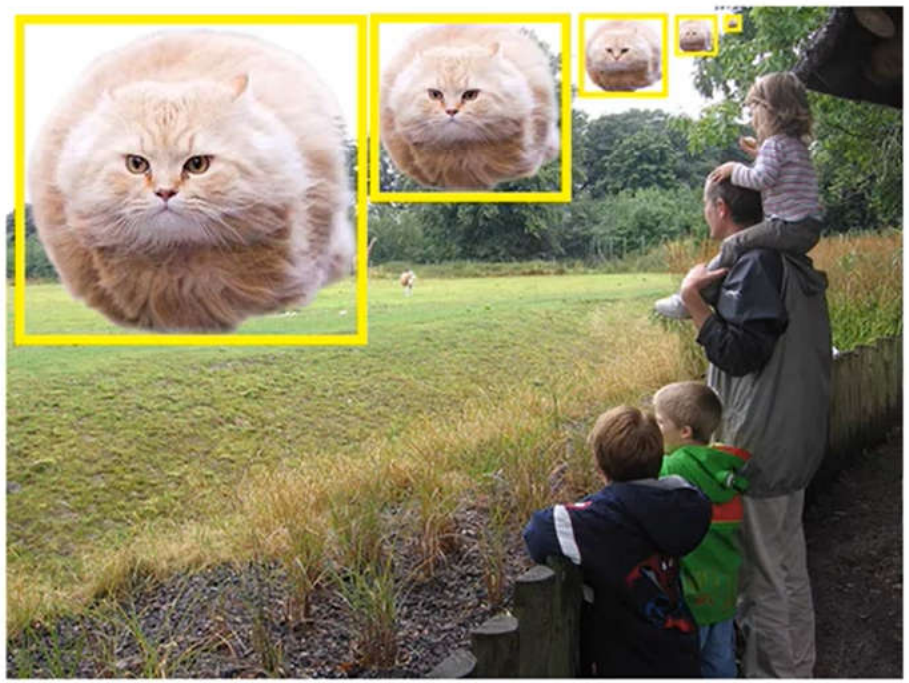
Swin Transformer =

- Transformer
  - Strong modeling power
- + good priors for visual modeling
  - Hierarchy
  - Locality
  - Translational invariance

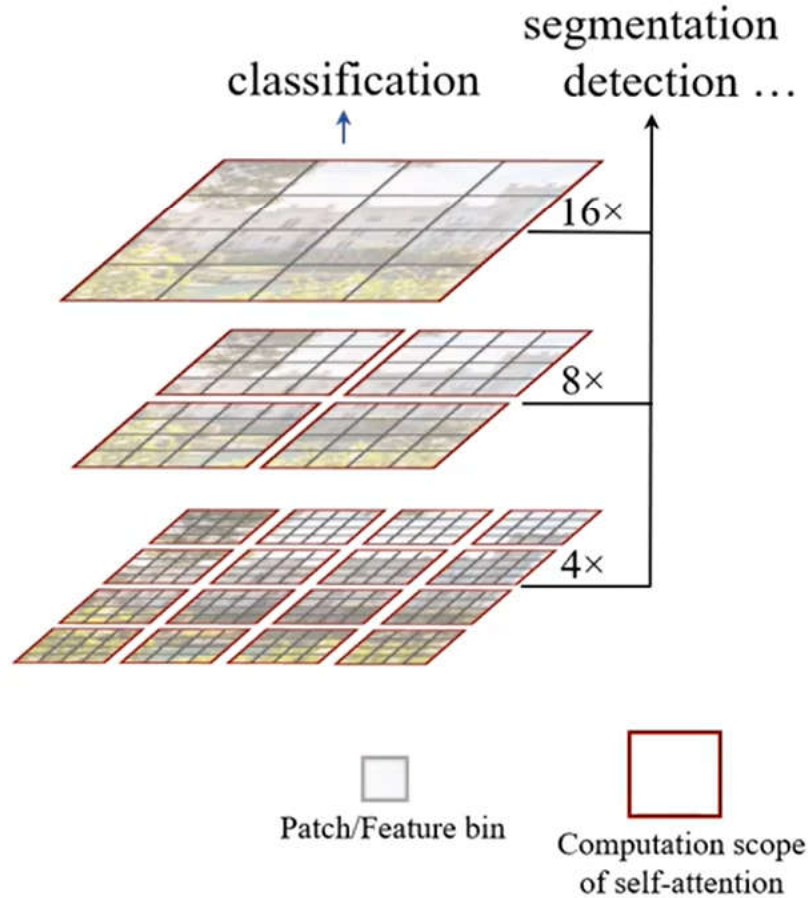


Hierarchy

- Processing objects of different scales

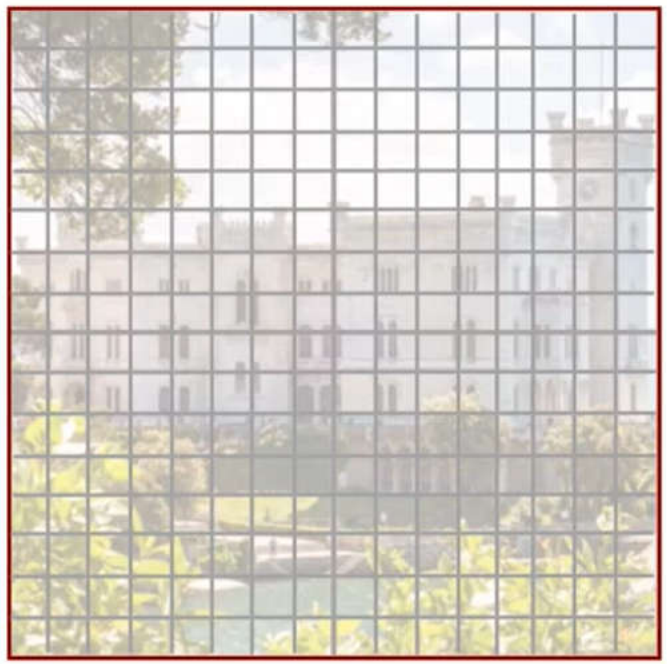


Left figure credit by Ross Girshick

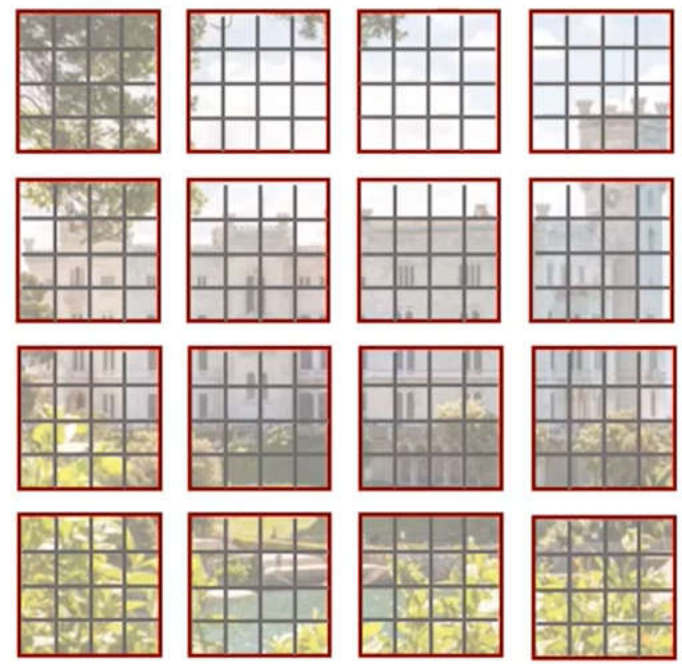


Locality by non-overlapped windows

- Proves beneficial in modeling the high correlation in visual signals (Yann LeCun)



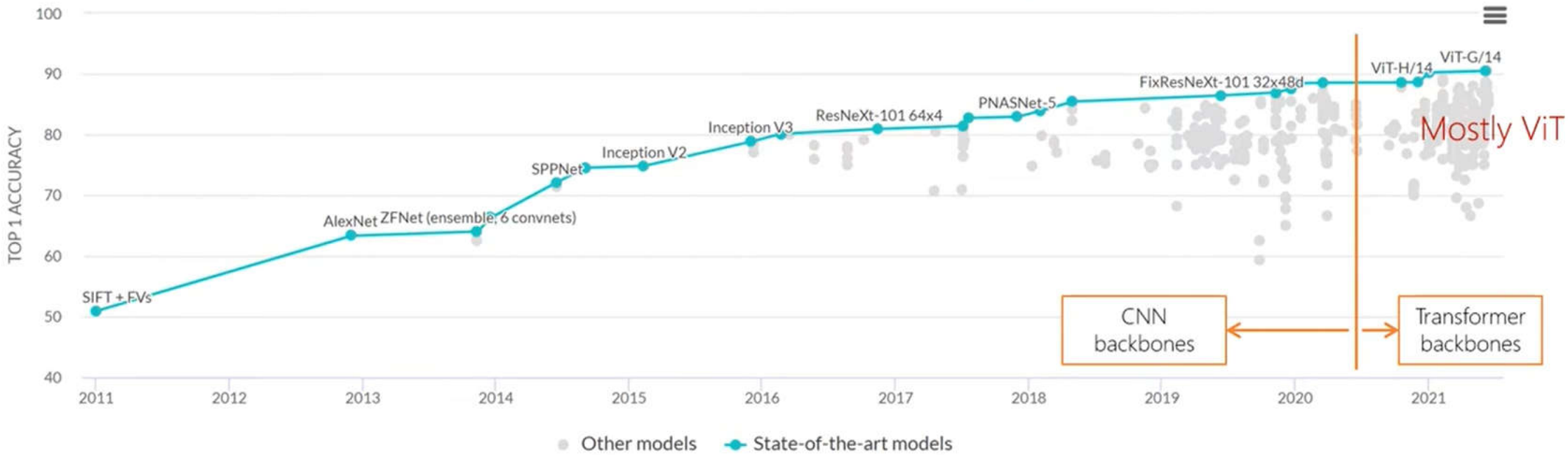
ViT: 256^2=65536 (Global)



Swin Transformer: 16x16^2=4096 (Local)

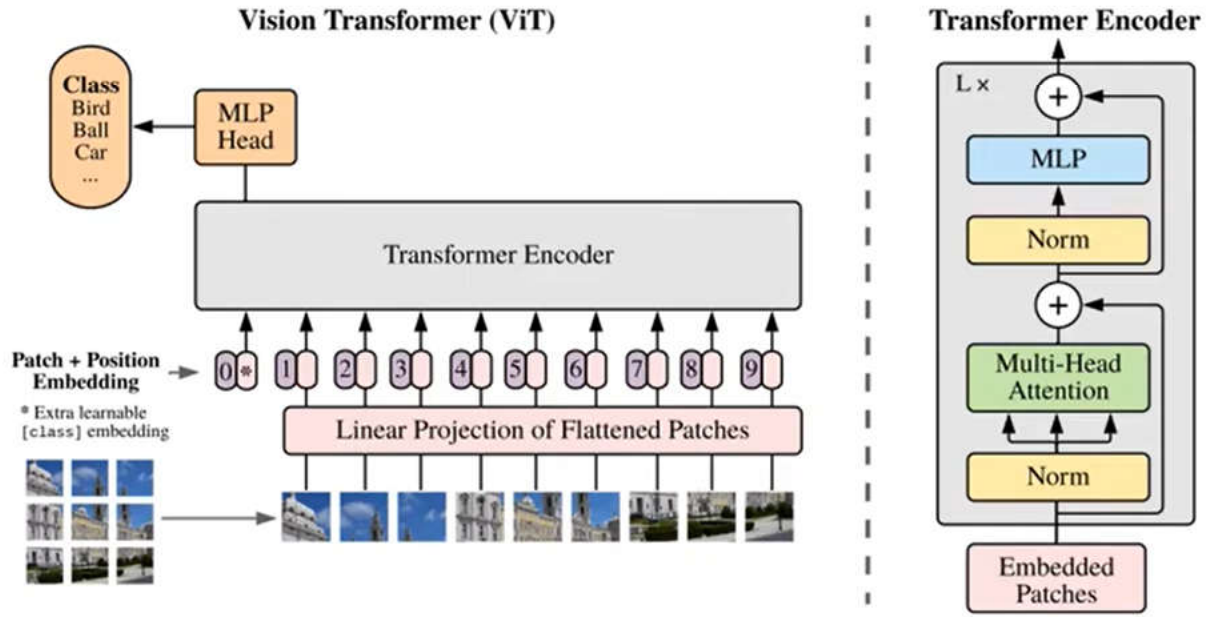
An answer: will also refresh & dominate CV

ImageNet-1K image classification



Vision Transformer (ViT, 10/2020)

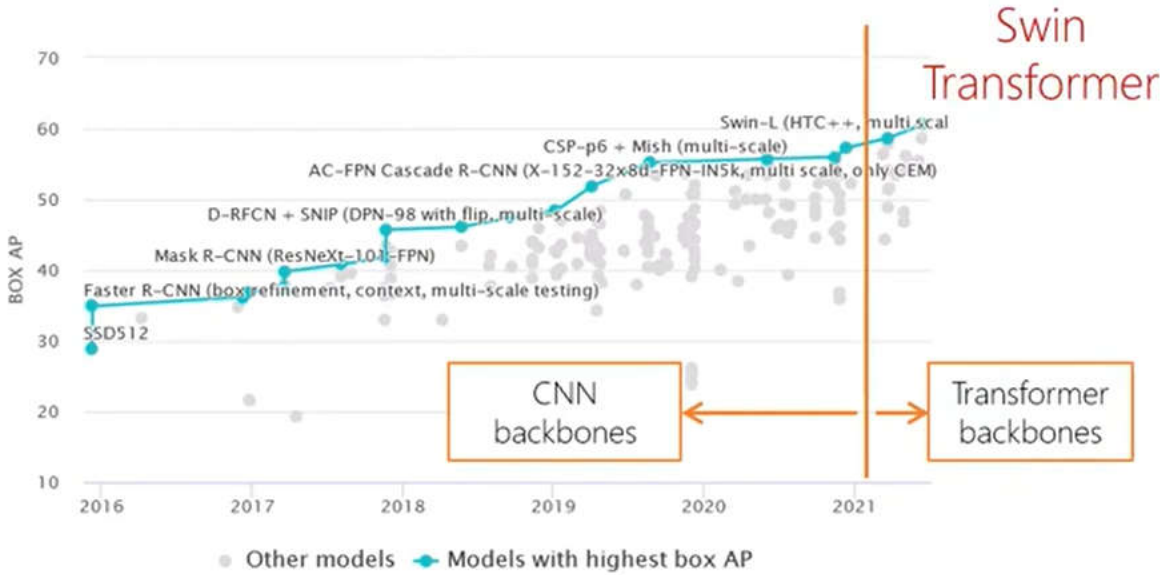
- SOTA performance on Image classification



Alexey Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR' 2021

An answer: will also refresh & dominate CV

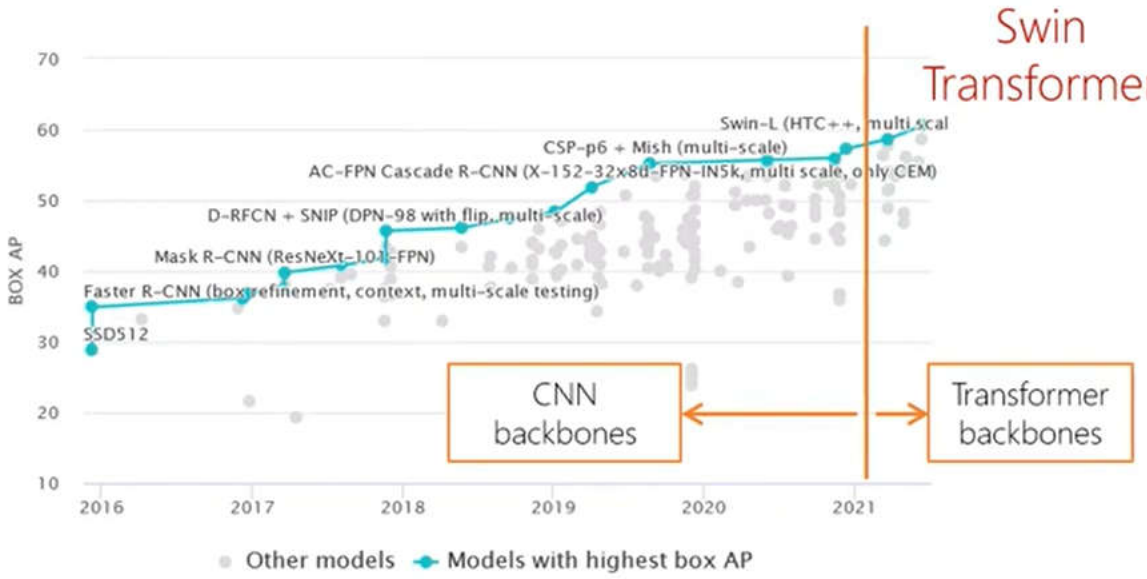
COCO object detection

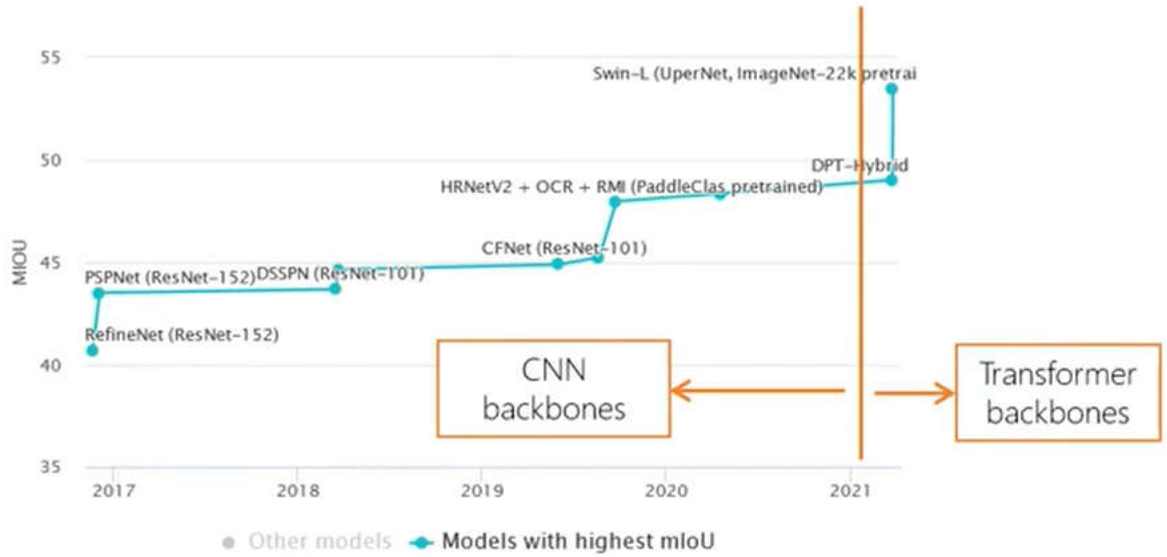


An answer: will also refresh & dominate CV

COCO object detection

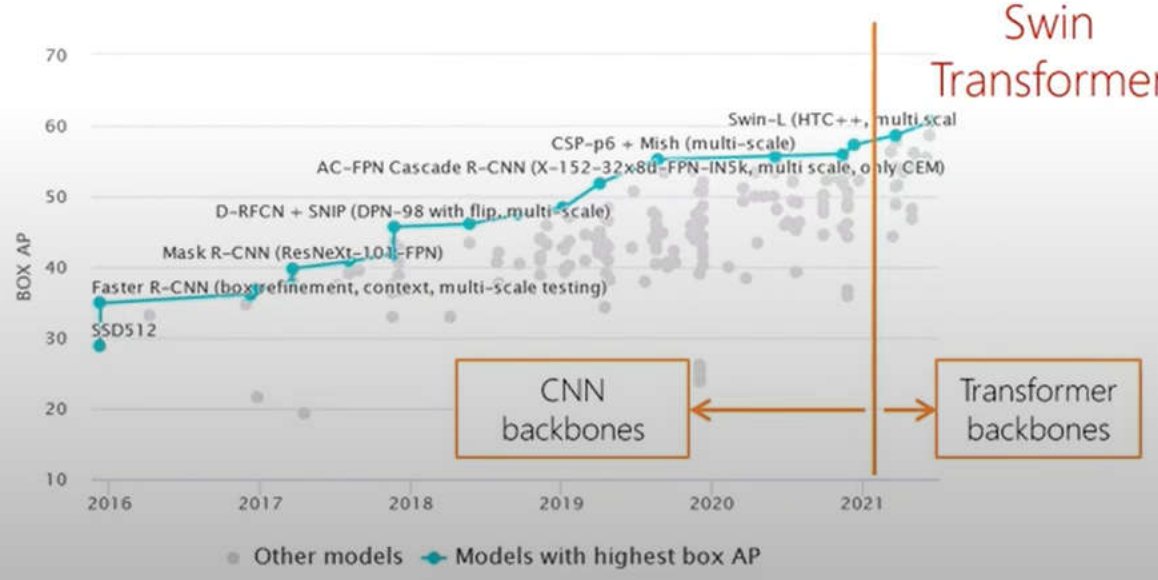
ADE20K semantic segmentation



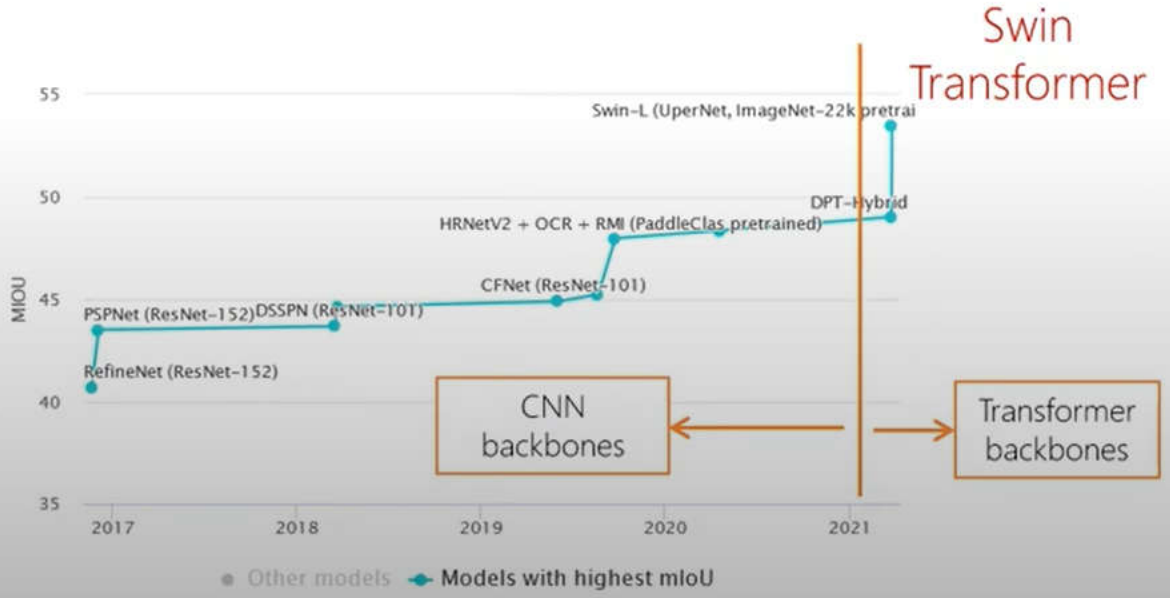


An answer: will also refresh & dominate CV

COCO object detection



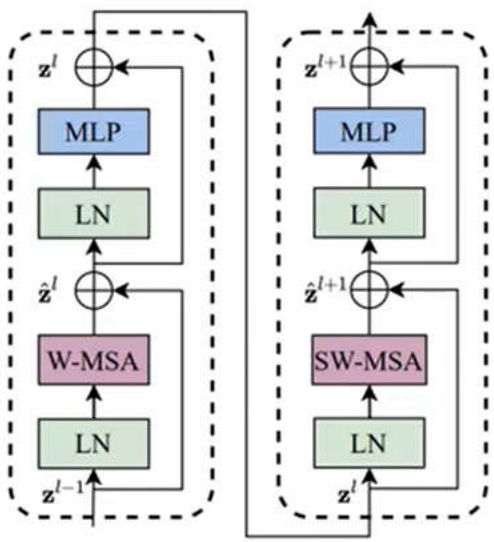
ADE20K semantic segmentation



Swin Transformer (03/2021)

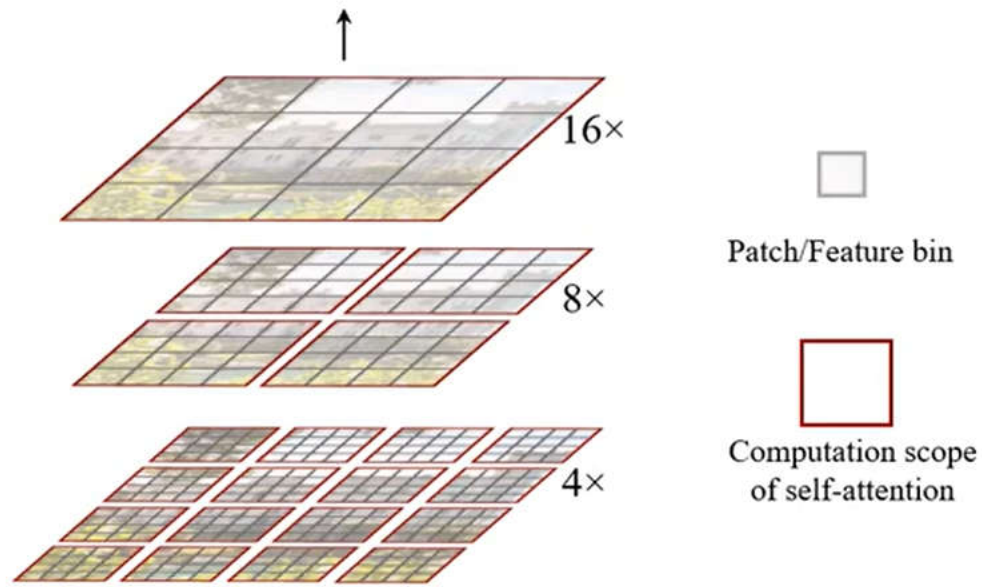
- SOTA performance on object detection and semantic segmentation

Transformer
(strong modeling power)



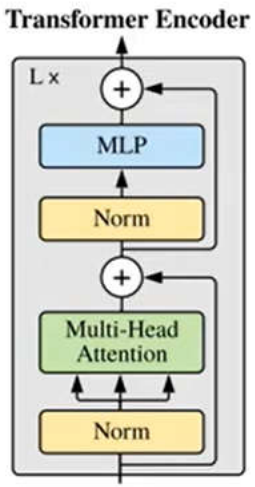
+

good priors for visual signals
(hierarchy / locality / translation invariance)



Ze Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. Arxiv 2021

4 years unleash the power of Transformer in CV



Reason I: General modeling capability

Reason II: Complement convolution

2017.06

2017.11

2019.4

Reason III: Strong modeling power

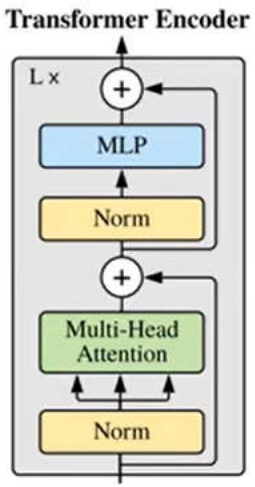
2021.1

Reason IV: Better connect vision and language

Reason V: Scalability

2021.6

4 years unleash the power of Transformer in CV



Reason I: General modeling capability

Reason II: Complement convolution

2017.06

2017.11

2019.4

Reason III: Strong modeling power

2021.1

Reason IV: Better connect vision and language

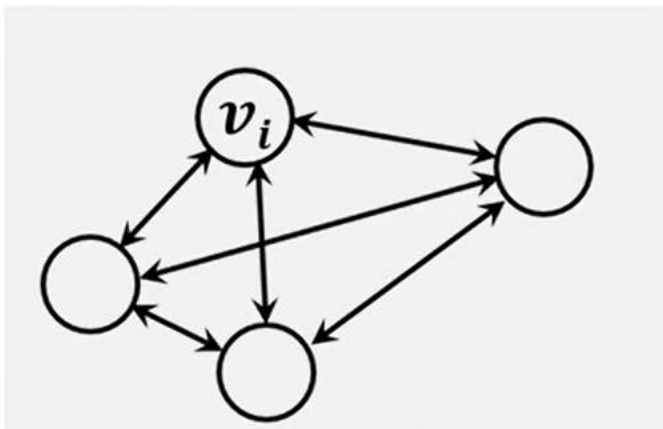
Reason V: Scalability

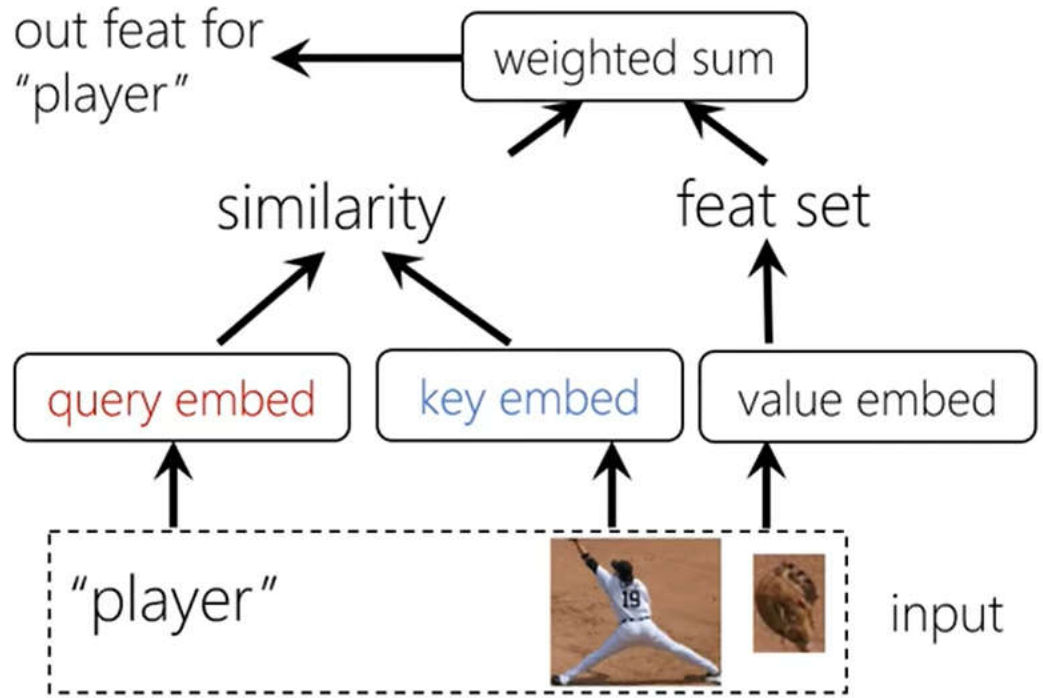
2021.6

Swin Transformer: a general-purpose backbone

Reason I to use Transformer in computer vision

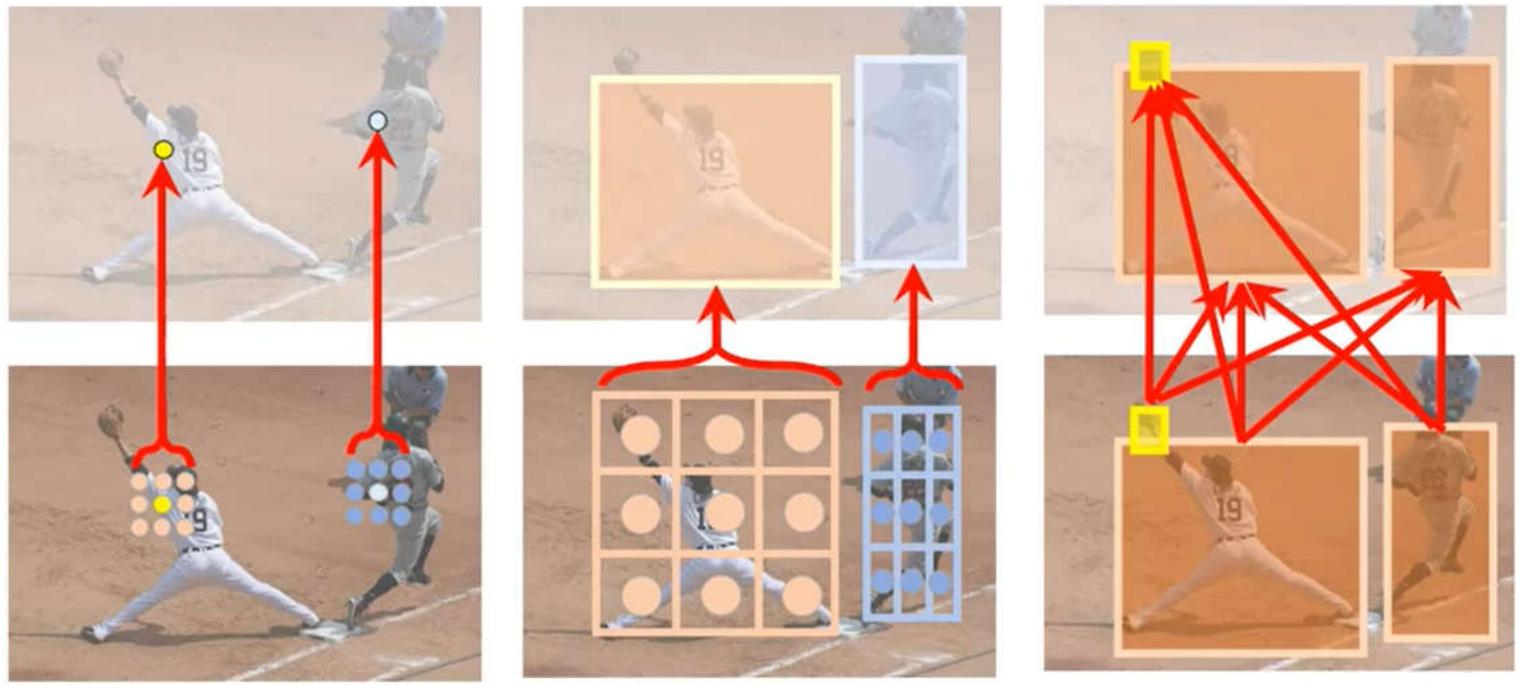
- General modeling capability
  - All concepts (concrete or abstract) and their relationships can be modeled by a graph
  - Modeling arbitrary relationship via verification, which is hard by CNN





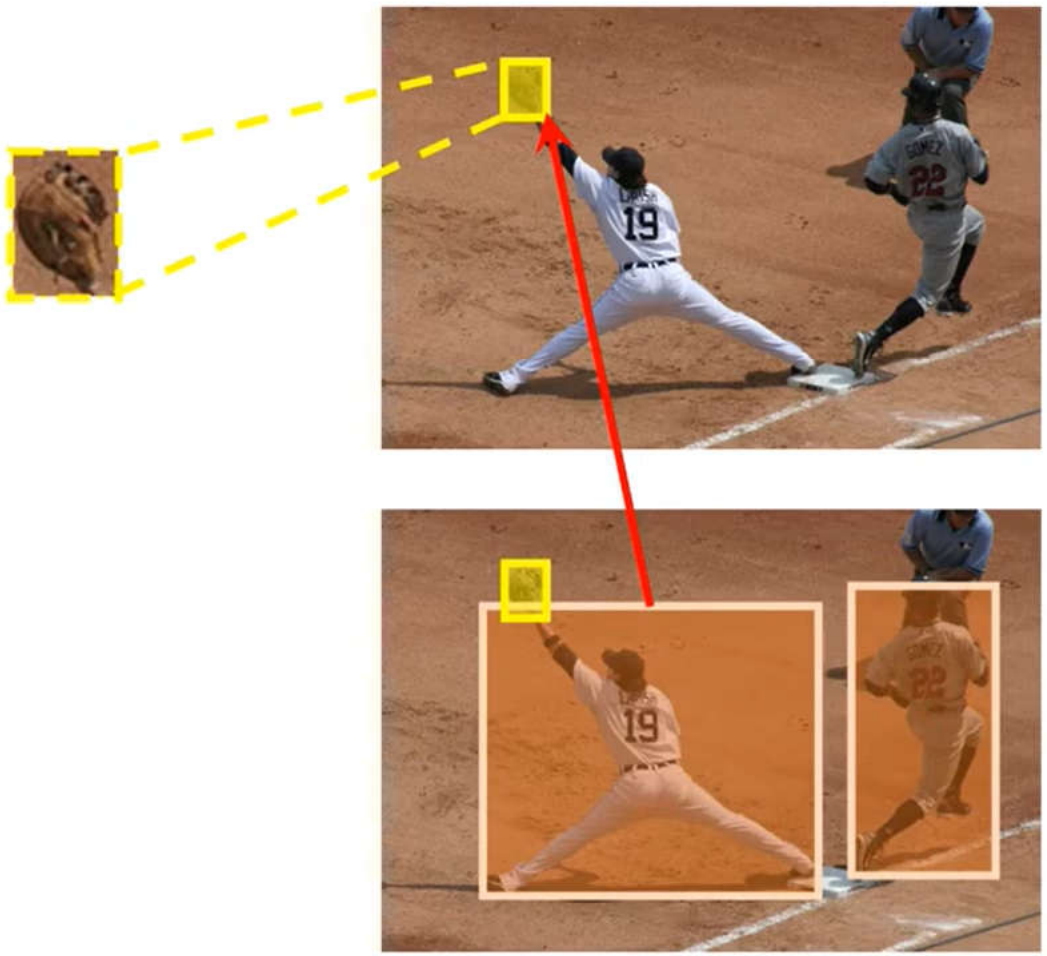
Reason I to use Transformer in computer vision

- General modeling capability
  - Can model all of pixel-to-pixel, object-to-pixel, object-to-object relationships



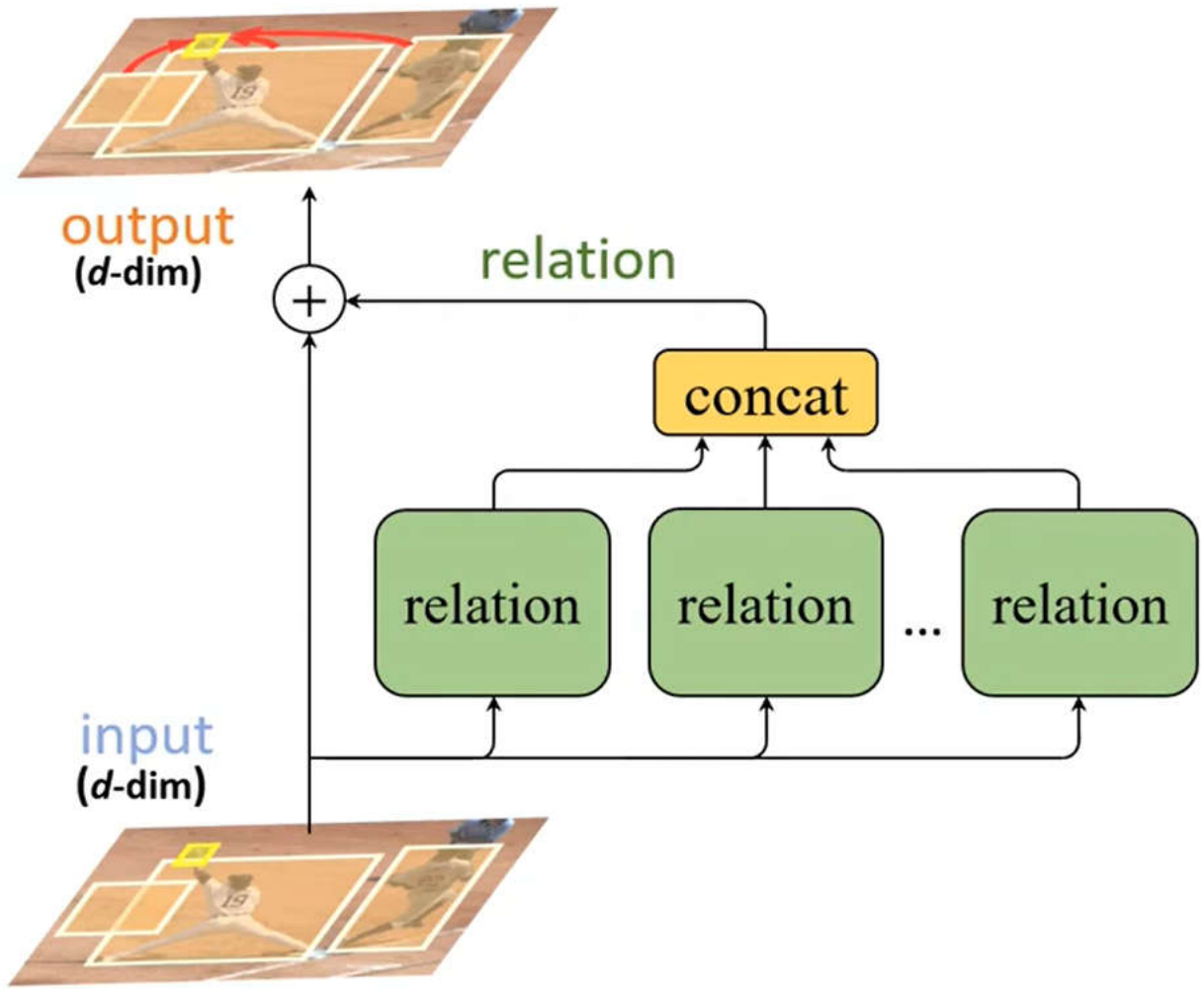
pixel-to-pixel　　object-to-pixel　　object-to-object

Relation Networks for Object Detection (CVPR'2018)



It is much easier to detect the ***glove*** if we know there is a ***baseball player***.

Relation Networks for Object Detection (CVPR'2018)



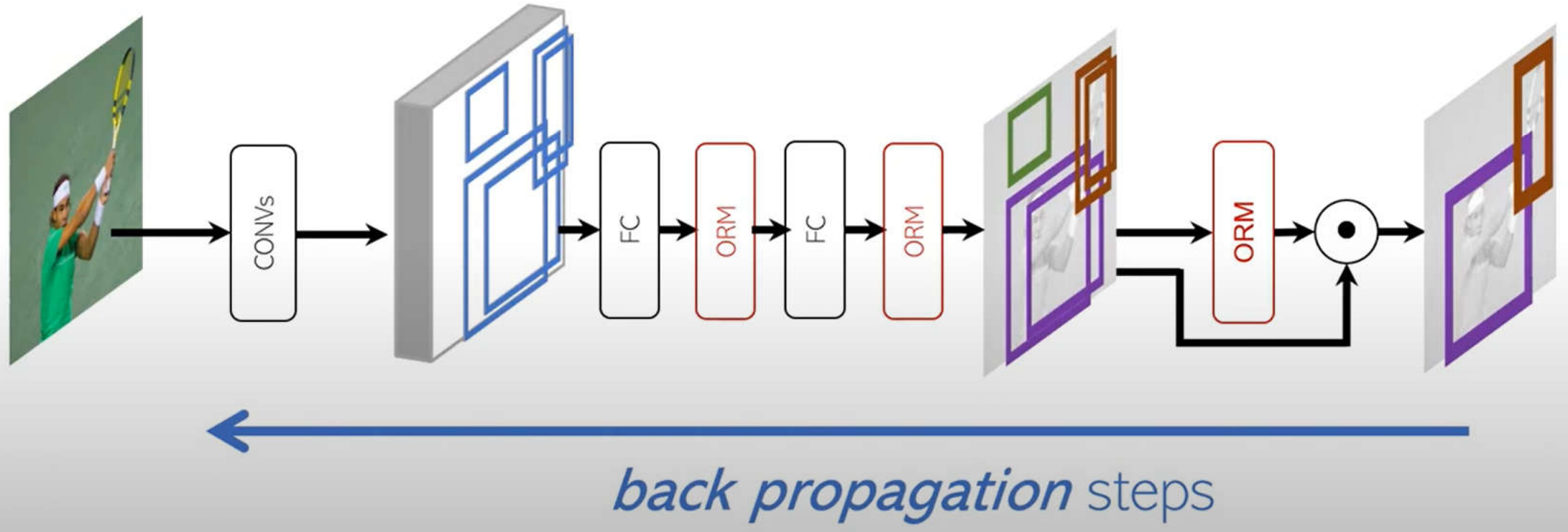
Key Feature

✓Relative position

Han Hu et al. *Relation Networks for Object Detection*. CVPR 2018

Relation Networks for Object Detection (CVPR'2018)

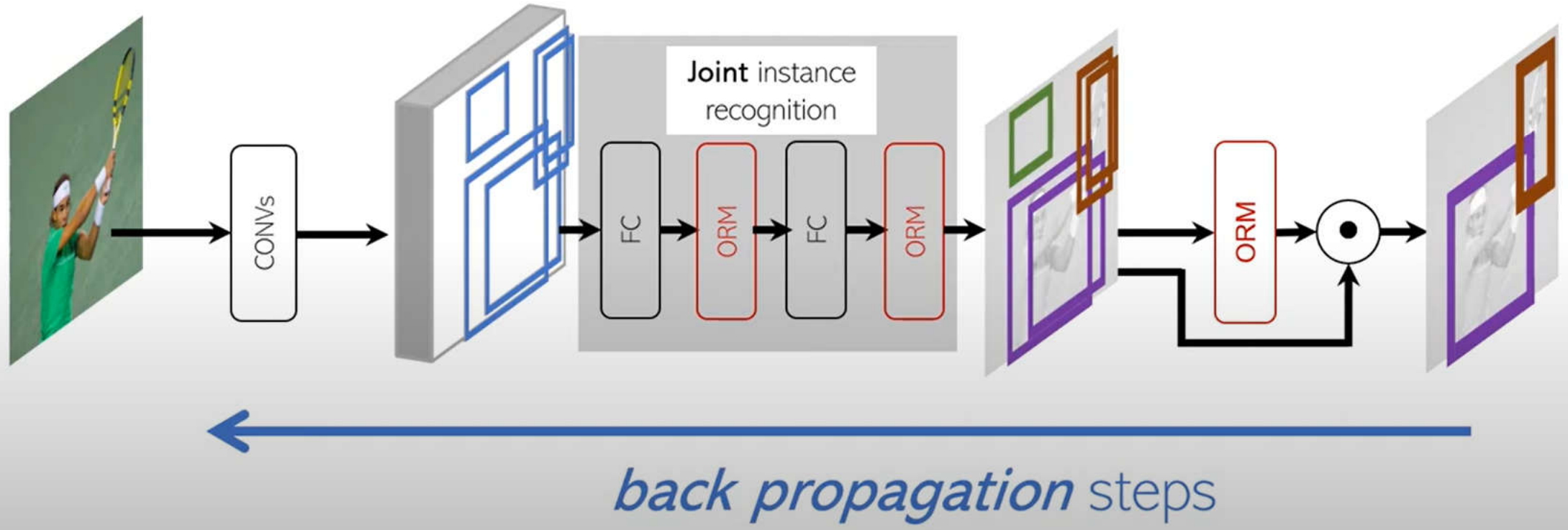
- The first fully end-to-end object detector



Han Hu et al. *Relation Networks for Object Detection*. CVPR 2018

Relation Networks for Object Detection (CVPR'2018)

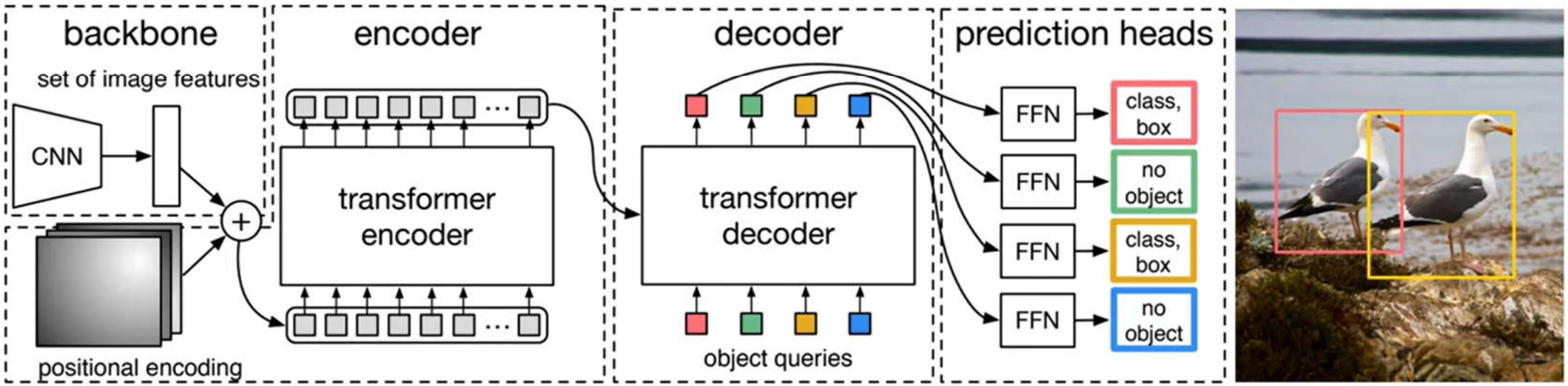
- The first fully end-to-end object detector



Han Hu et al. *Relation Networks for Object Detection*. CVPR 2018

DeTR (ECCV'2020)

- Another end-to-end object detector



Nicolas Carion et al. End-to-End Object Detection with Transformers. ECCV 2020

Reason II to use Transformer in computer vision

- Complement convolution

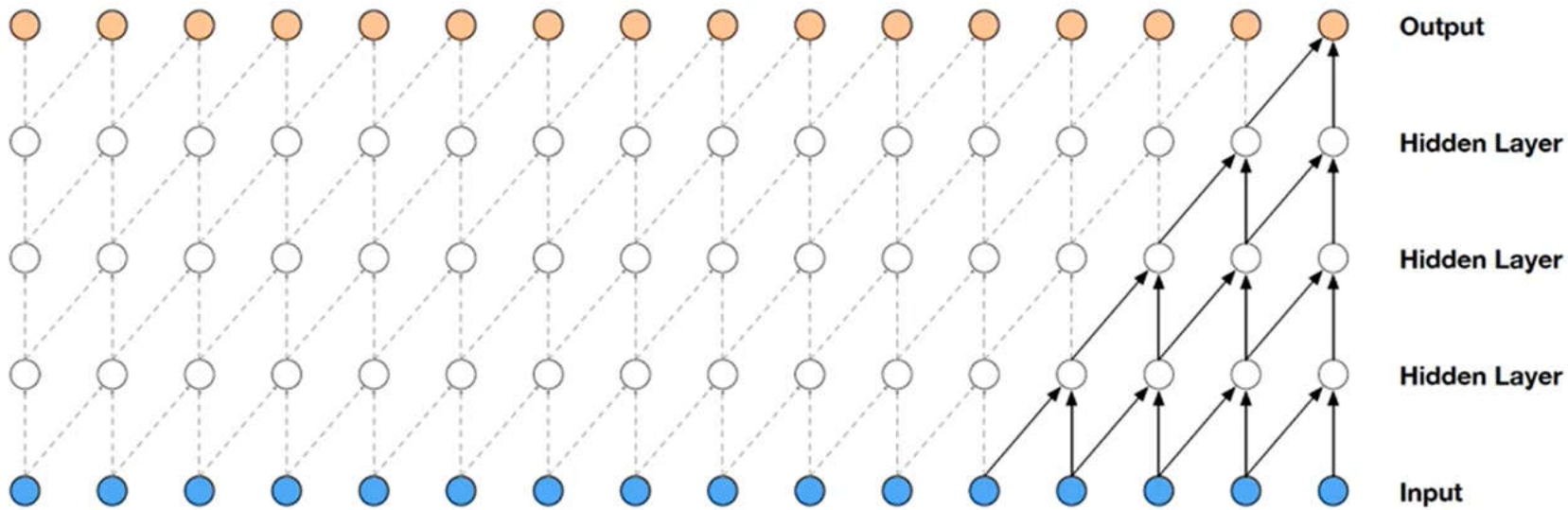


Figure credit: Van Den Oord et al.

Reason II to use Transformer in computer vision

- Complement convolution
  - "Convolution is too local!"
  - Global (Transformer) vs. local (conv.)

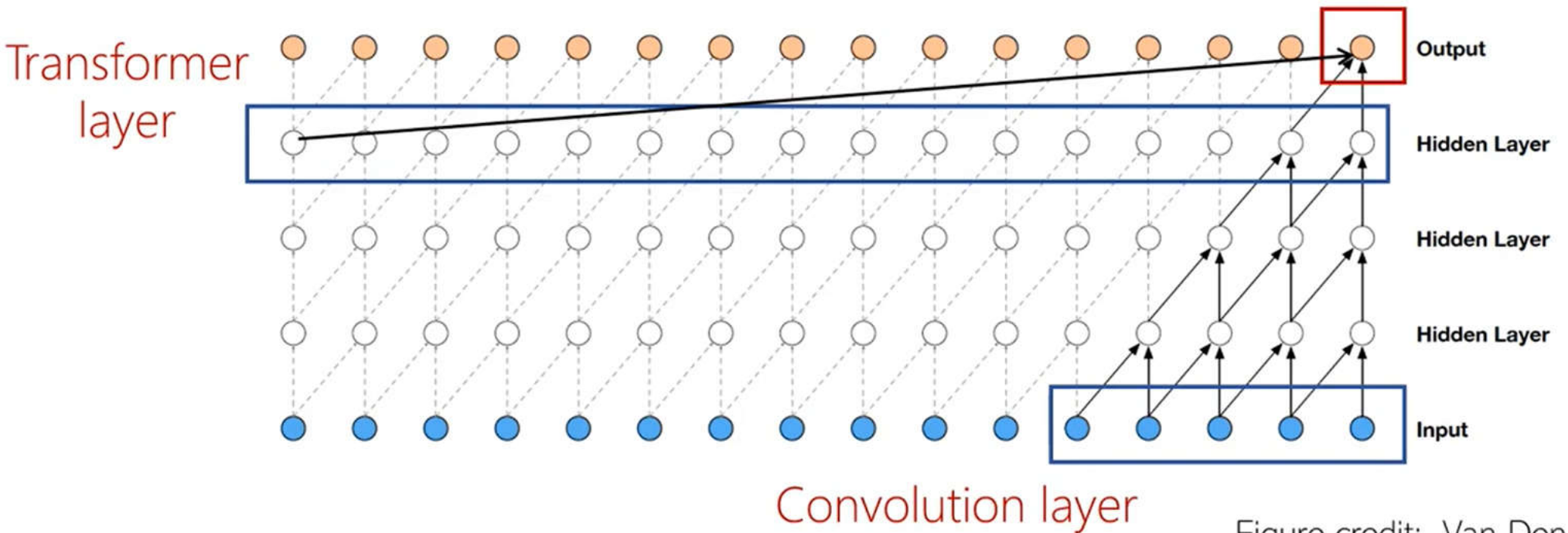
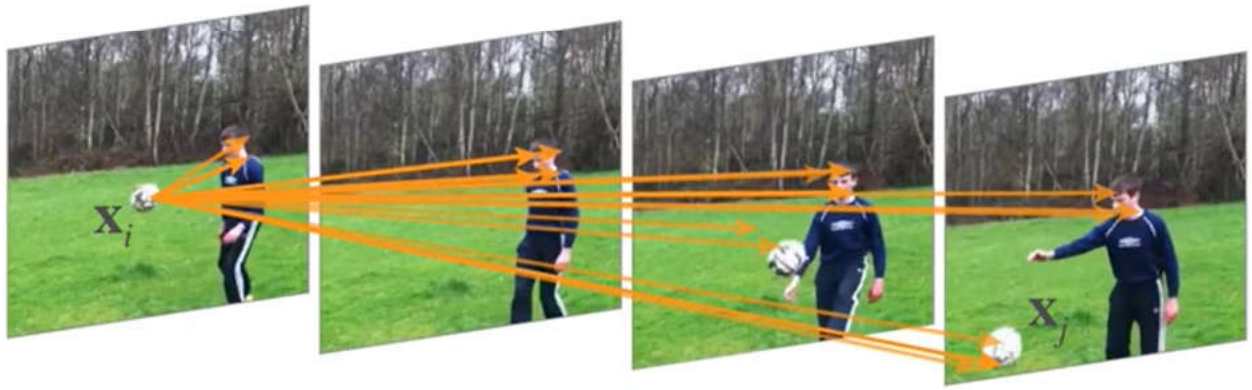
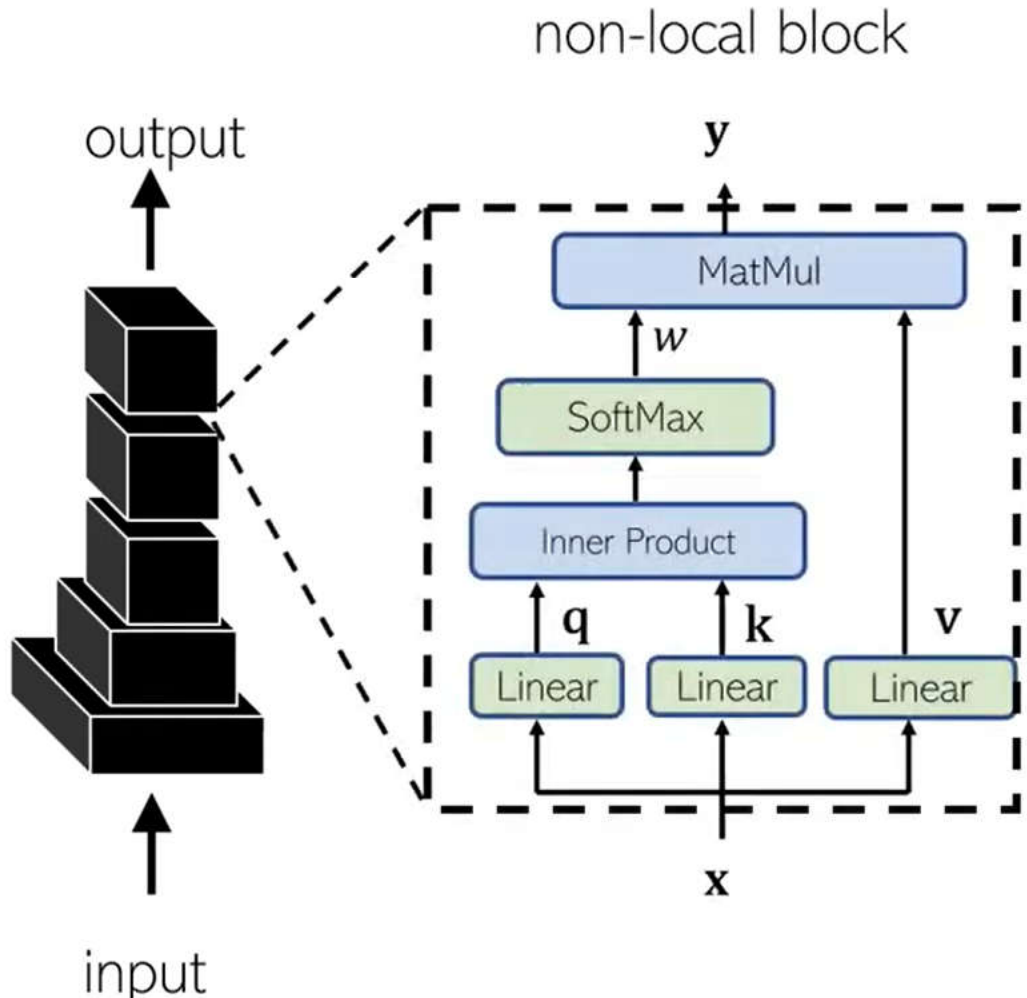


Figure credit: Van Den Oord et al.

Non-local networks (CVPR'2018)





The Degeneration Problem of NLNet

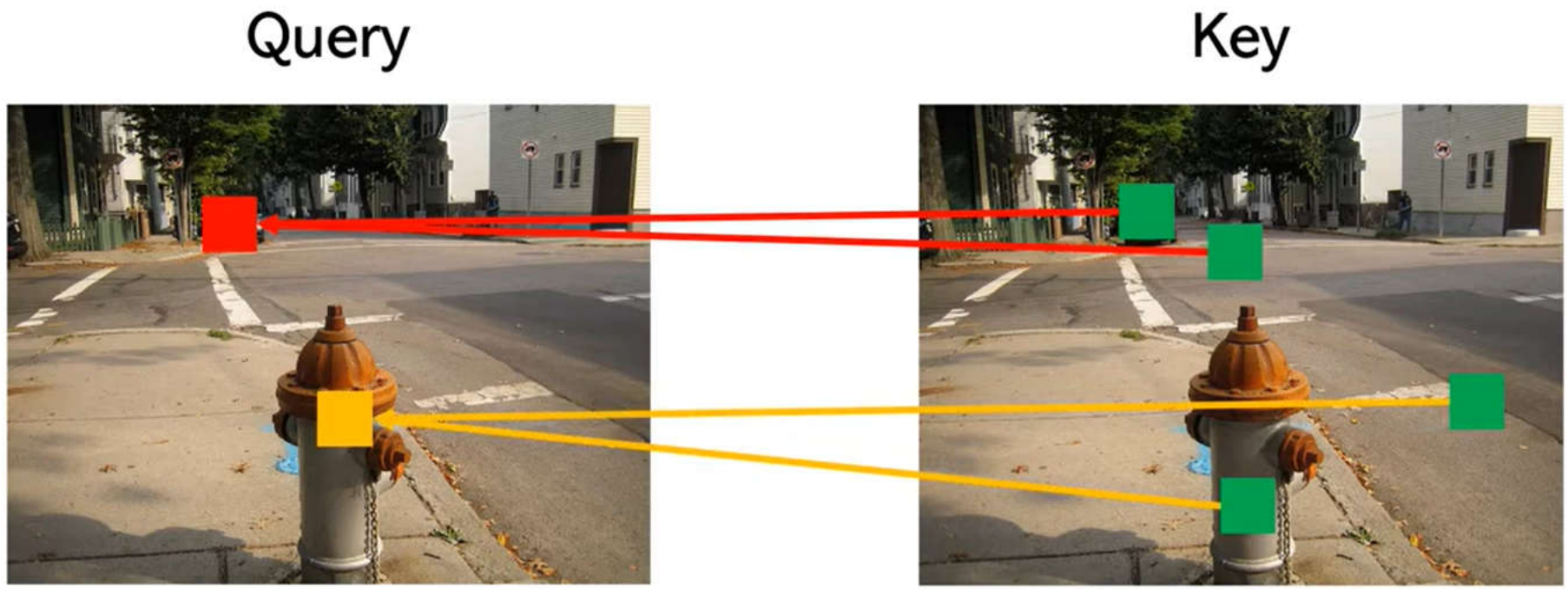
- Expectation of Ideally Learnt Relation
  - Different queries affected by **different** key

Query

Key

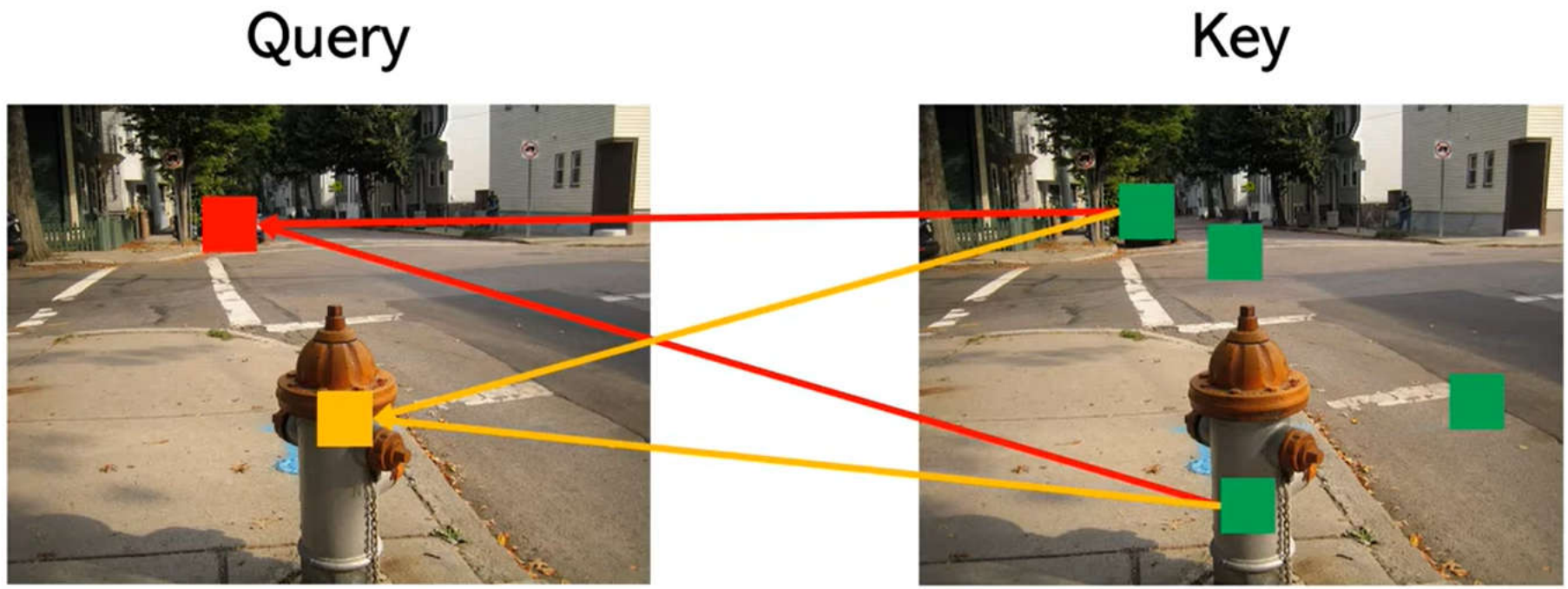
The Degeneration Problem of NLNet

- Expectation of Ideally Learnt Relation
  - Different queries affected by **different** key



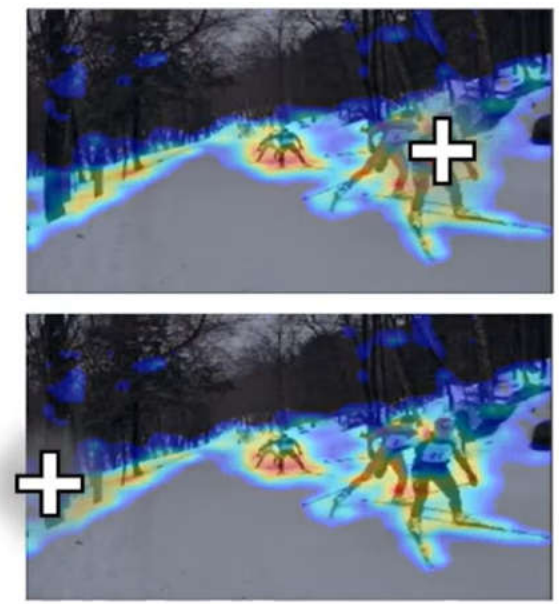
The Degeneration Problem of NLNet

- What does the Self-Attention Learn?
  - Different queries affected by the **same** keys

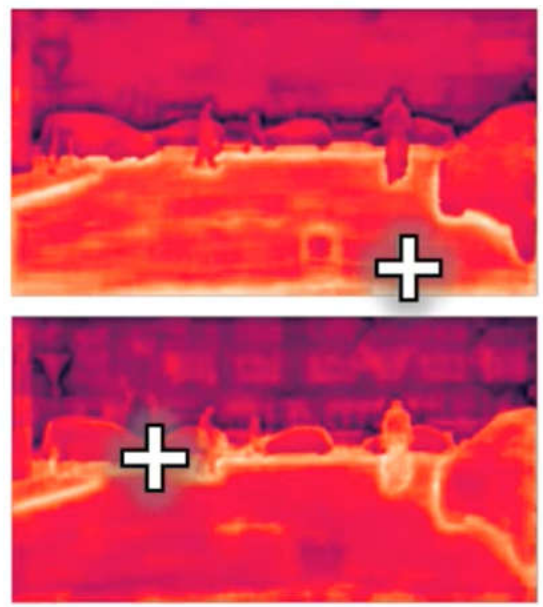


Visualizations on Real Tasks

- ✚ indicates the query point



Object Detection



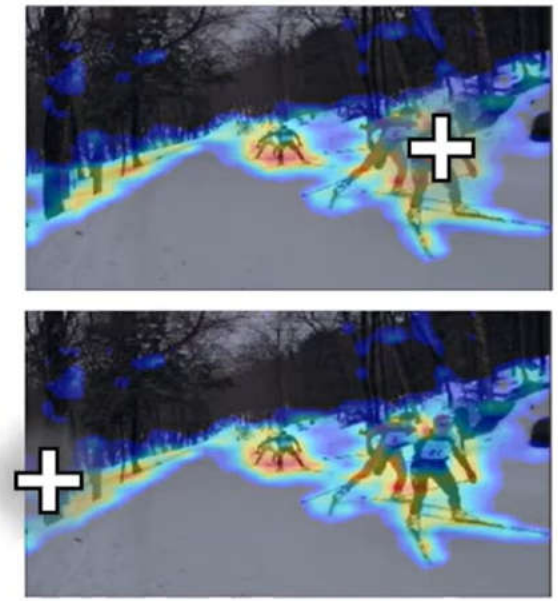
Semantic Segmentation

[GCNet, ICCVW'2019]

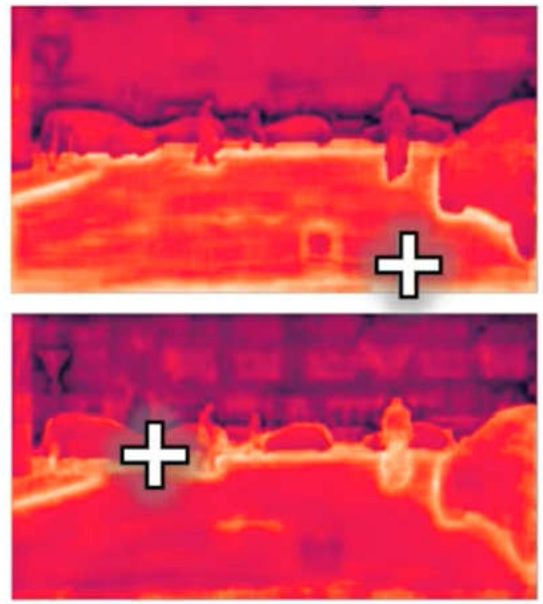
https://arxiv.org/pdf/1904.11492.pdf

Visualizations on Real Tasks

- ✚ indicates the query point
- The activation map for different queries are similar
- The self-attention model degenerates to a unary model



Object Detection



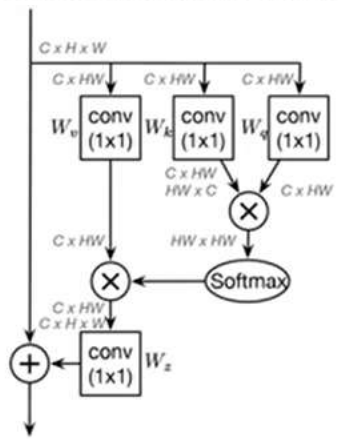
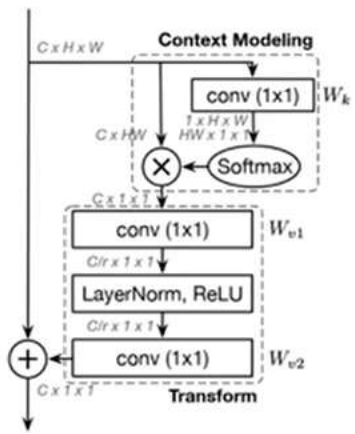
Semantic Segmentation

[GCNet, ICCVW'2019]

https://arxiv.org/pdf/1904.11492.pdf

GCNet (ICCVW'2019, PAMI'2021)

- Find the degeneration issue in computer vision
- Explicitly leverage degenerated formulation for better efficiency

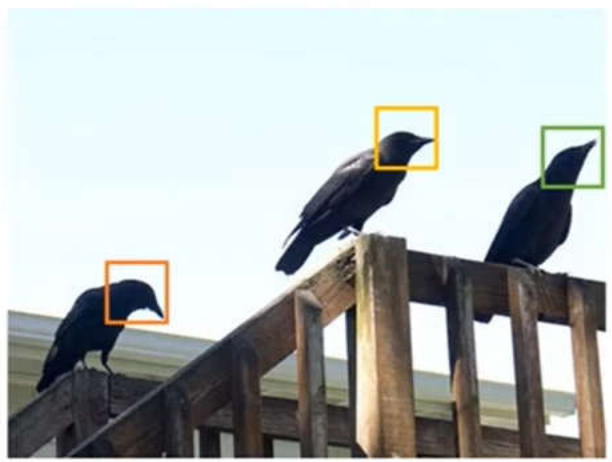
	Non-Local Block	Global Context Block	Reduction ratio
			
FLOPs	9.3G	4.0M	2,300x
model size	2.1M	0.1M	20x
accuracy (mAP)	38.0	38.1	unchanged

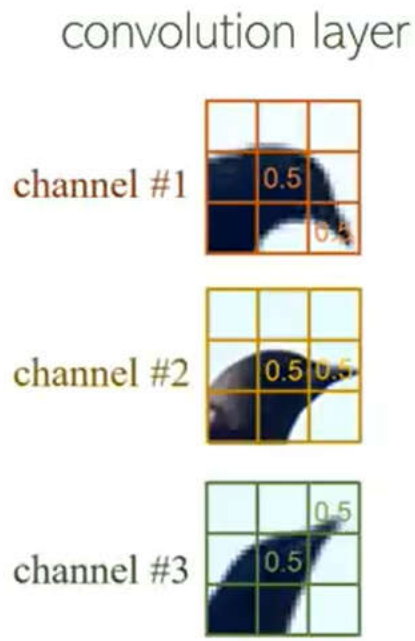
Reason III to use Transformer in computer vision

- Powerful due to adaptive computation

Reason III to use Transformer in computer vision

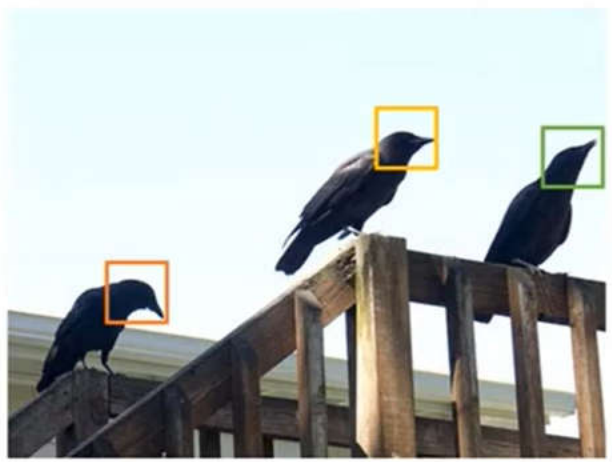
- Powerful due to <u>adaptive computation</u>
  - "Convolution is exponentially inefficient!"

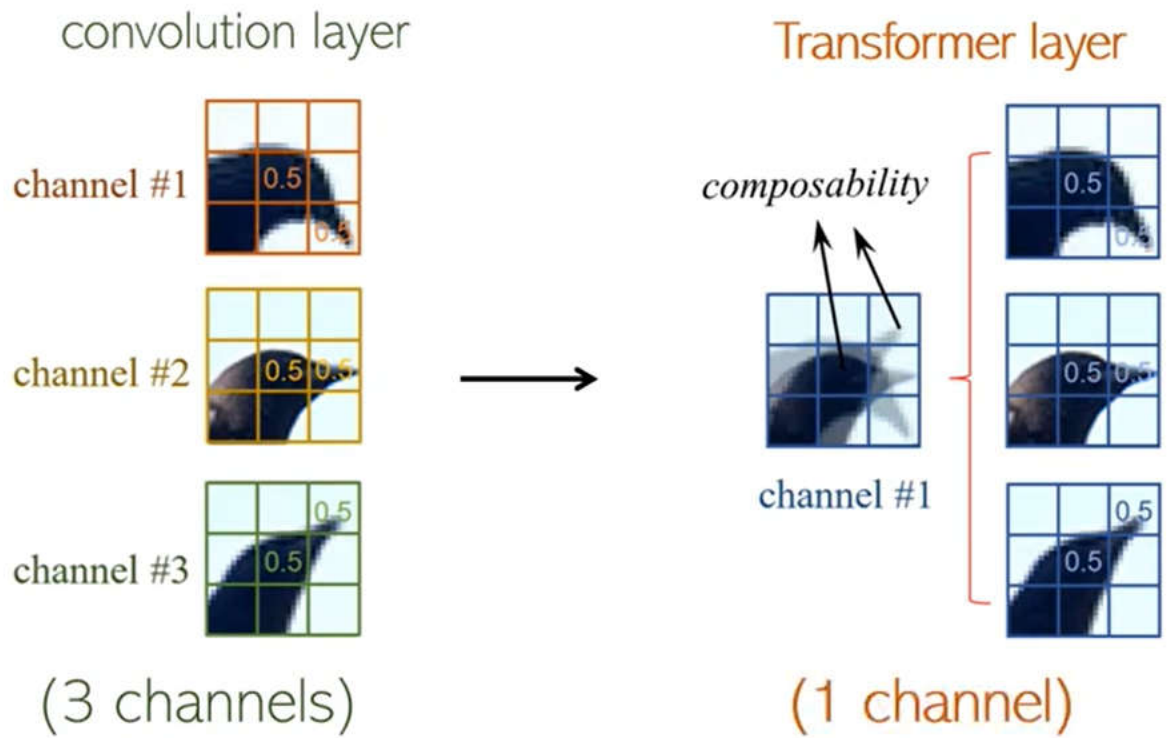




Reason III to use Transformer in computer vision

- Powerful due to <u>adaptive computation</u>
  - "Convolution is exponentially inefficient!"





Local relation networks (2019.4)

- Transformer as backbones

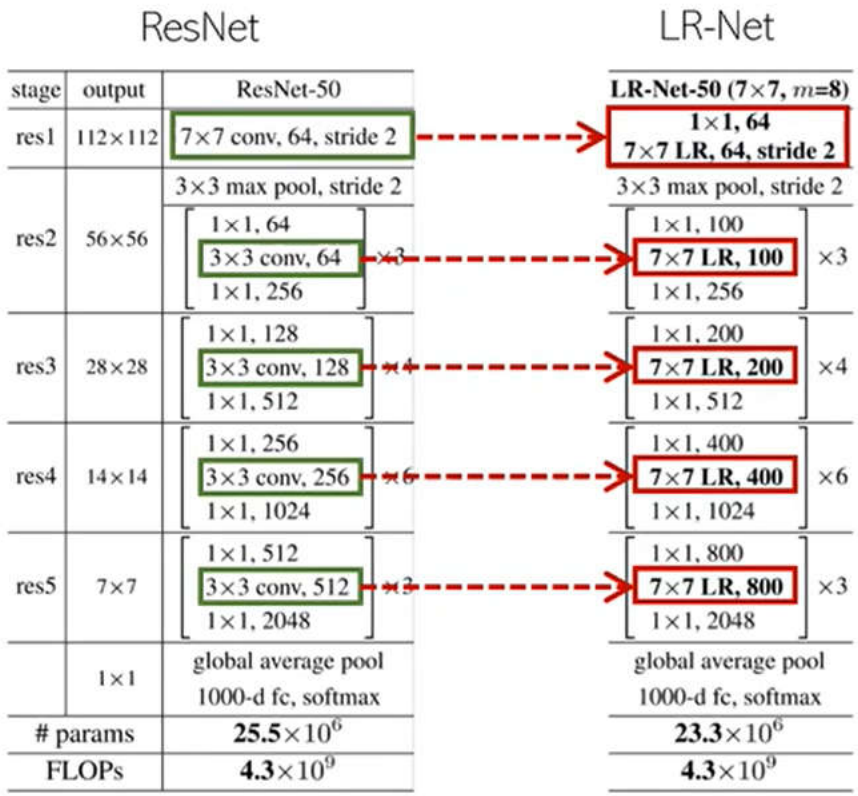
ResNet

stage	output	ResNet-50
res1	112×112	7×7 conv, 64, stride 2
res2	56×56	3×3 max pool, stride 2 [1×1, 64 3×3 conv, 64 1×1, 256] ×3
res3	28×28	[1×1, 128 3×3 conv, 128 1×1, 512] ×4
res4	14×14	[1×1, 256 3×3 conv, 256 1×1, 1024] ×6
res5	7×7	[1×1, 512 3×3 conv, 512 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax
# params		$\mathbf{25.5}\times10^6$
FLOPs		$\mathbf{4.3}\times10^9$

Han Hu et al. *Local Relation Networks for Visual Recognition. ICCV 2019*

Local relation networks (2019.4)

- Transformer as backbones



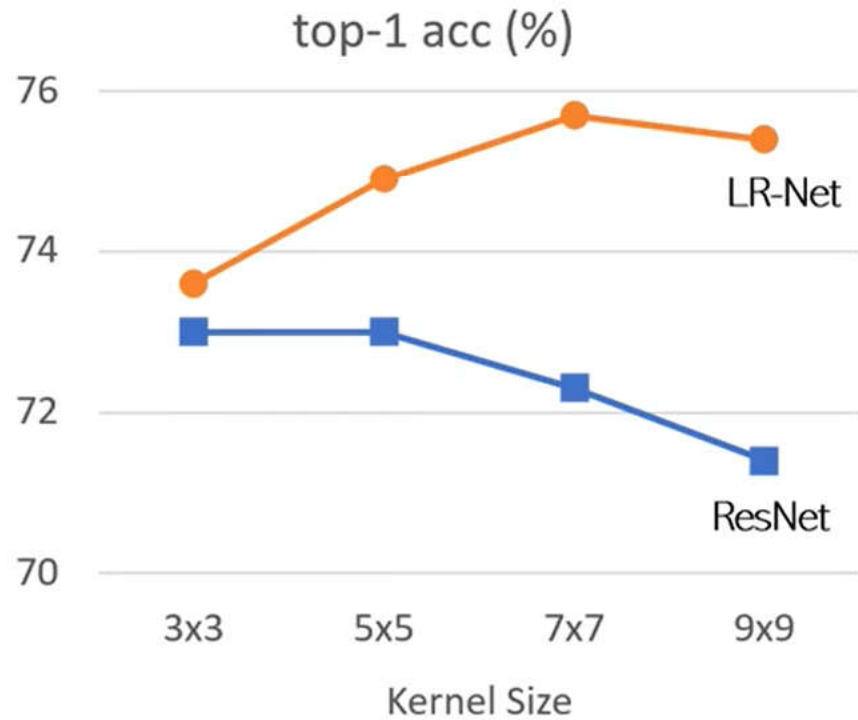
stage	output	ResNet-50	LR-Net-50 (7×7, m=8)
res1	112×112	7×7 conv, 64, stride 2	**1×1, 64** **7×7 LR, 64, stride 2**
res2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		[1×1, 64 3×3 conv, 64 1×1, 256] ×3	[1×1, 100 **7×7 LR, 100** 1×1, 256] ×3
res3	28×28	[1×1, 128 3×3 conv, 128 1×1, 512] ×4	[1×1, 200 **7×7 LR, 200** 1×1, 512] ×4
res4	14×14	[1×1, 256 3×3 conv, 256 1×1, 1024] ×6	[1×1, 400 **7×7 LR, 400** 1×1, 1024] ×6
res5	7×7	[1×1, 512 3×3 conv, 512 1×1, 2048] ×3	[1×1, 800 **7×7 LR, 800** 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params		**25.5**×10^6	**23.3**×10^6
FLOPs		**4.3**×10^9	**4.3**×10^9

Han Hu et al. *Local Relation Networks for Visual Recognition. ICCV 2019*

Local relation networks (2019.4)

- Transformer as backbones

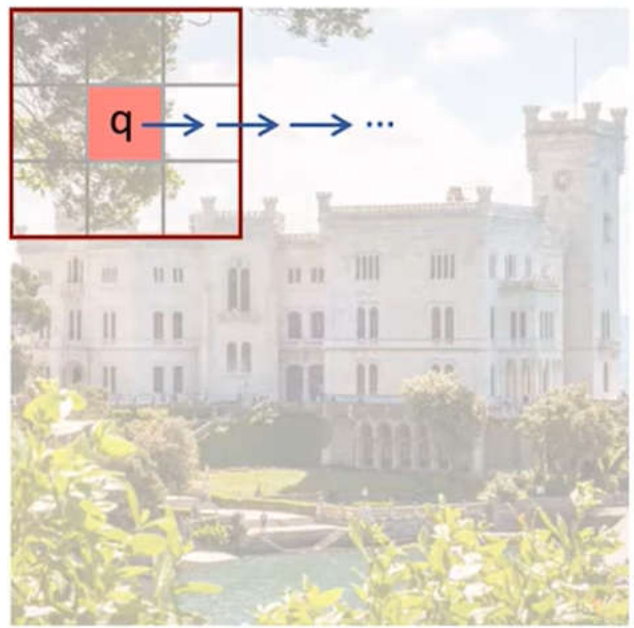
		ResNet	LR-Net
stage	output	ResNet-50	**LR-Net-50 (7×7, m=8)**
res1	112×112	7×7 conv, 64, stride 2	**1×1, 64** **7×7 LR, 64, stride 2**
res2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		1×1, 64 3×3 conv, 64 1×1, 256 ×3	1×1, 100 **7×7 LR, 100** 1×1, 256 ×3
res3	28×28	1×1, 128 3×3 conv, 128 1×1, 512 ×4	1×1, 200 **7×7 LR, 200** 1×1, 512 ×4
res4	14×14	1×1, 256 3×3 conv, 256 1×1, 1024 ×6	1×1, 400 **7×7 LR, 400** 1×1, 1024 ×6
res5	7×7	1×1, 512 3×3 conv, 512 1×1, 2048 ×3	1×1, 800 **7×7 LR, 800** 1×1, 2048 ×3
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params		**25.5**×10^6	**23.3**×10^6
FLOPs		**4.3**×10^9	**4.3**×10^9



Han Hu et al. *Local Relation Networks for Visual Recognition. ICCV 2019*

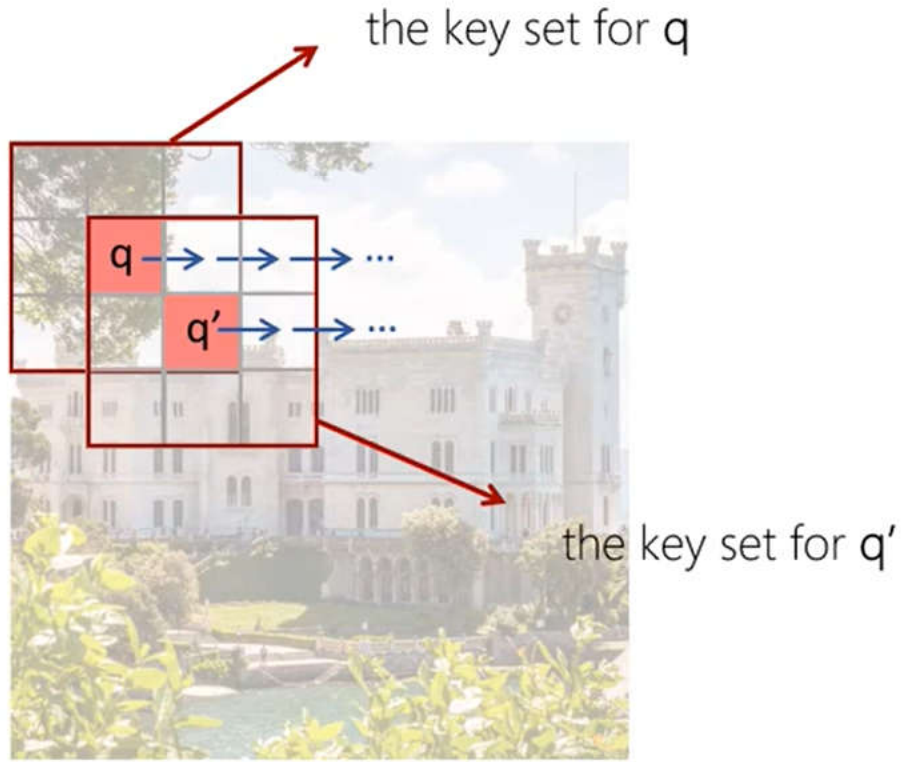
But … slow in real computation

- Because different queries use different key sets



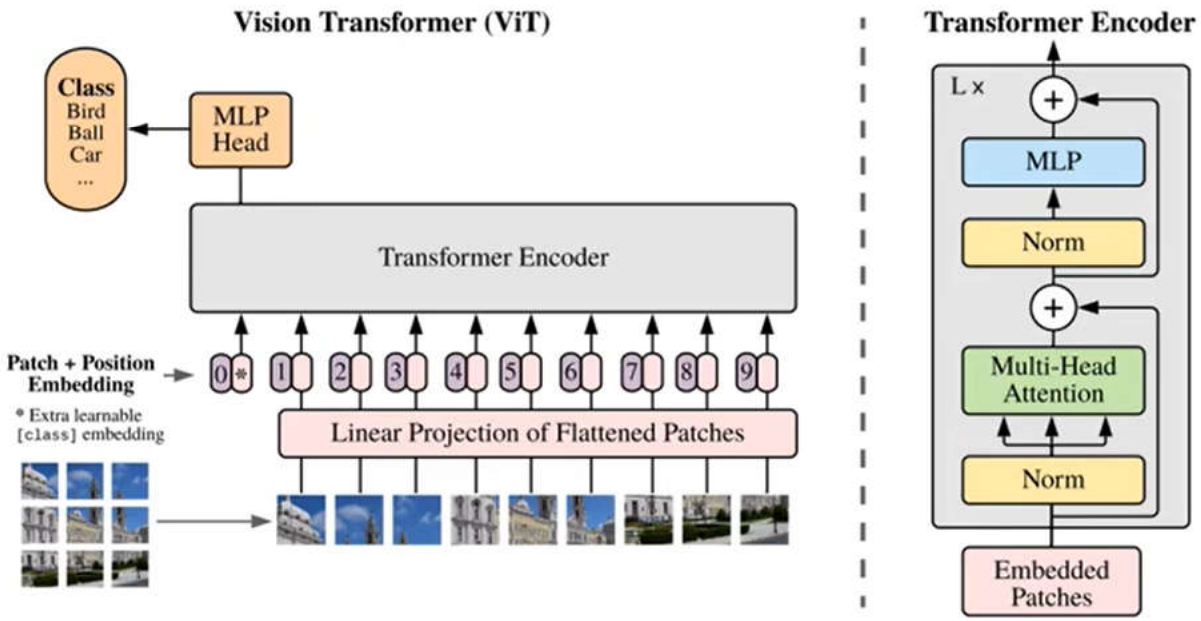
But … slow in real computation

- Because different queries use different key sets



Vision Transformer (ViT)

- by Google Brain (2020.10)



Computation speed

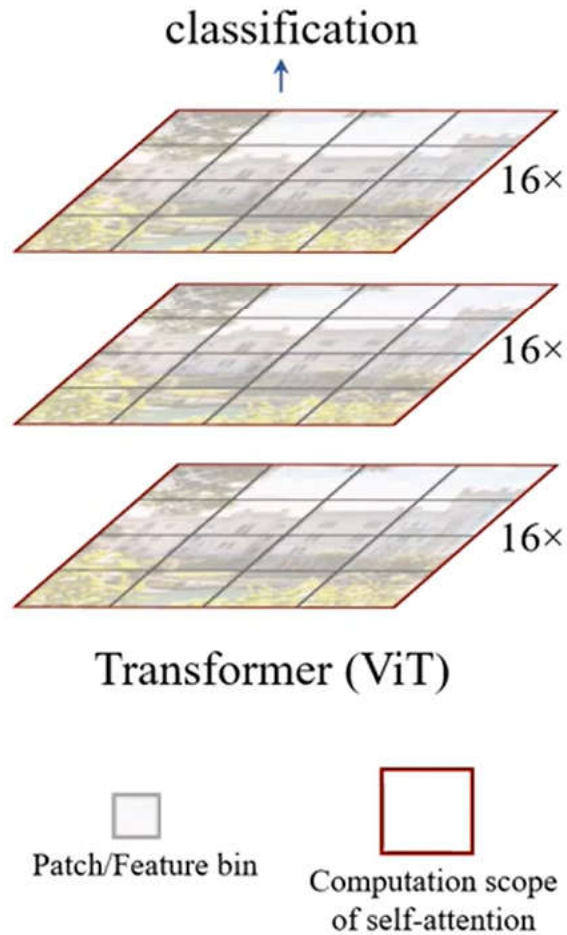
Model	FLOPs	Speed (TPU)
R50	4.3G	~2100 im/s
ViT-B/32	4.3G	~3000 im/s

+50% speed-up

Alexey Dosovitskiy et al. an Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR' 2021

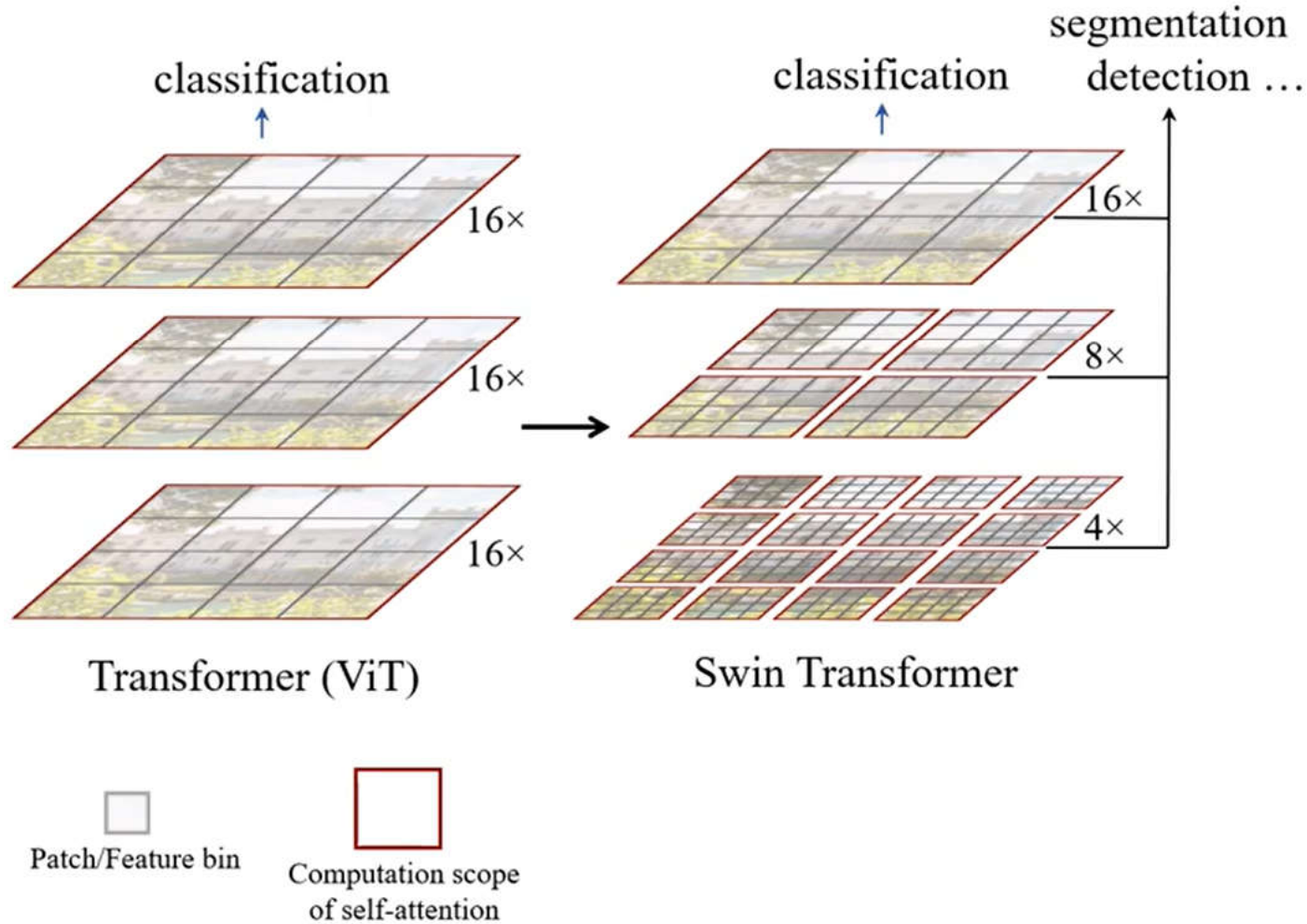
Swin Transformer =

- Transformer
  - Strong modeling power
- + good priors for visual modeling
  - Hierarchy
  - Locality
  - Translational invariance



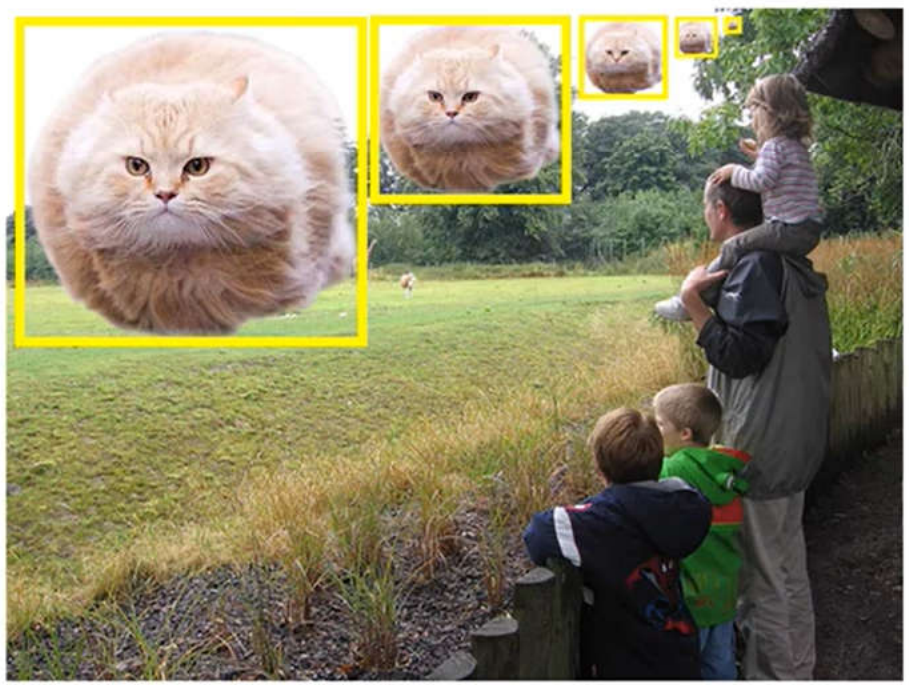
Swin Transformer =

- Transformer
  - Strong modeling power
- + good priors for visual modeling
  - Hierarchy
  - Locality
  - Translational invariance

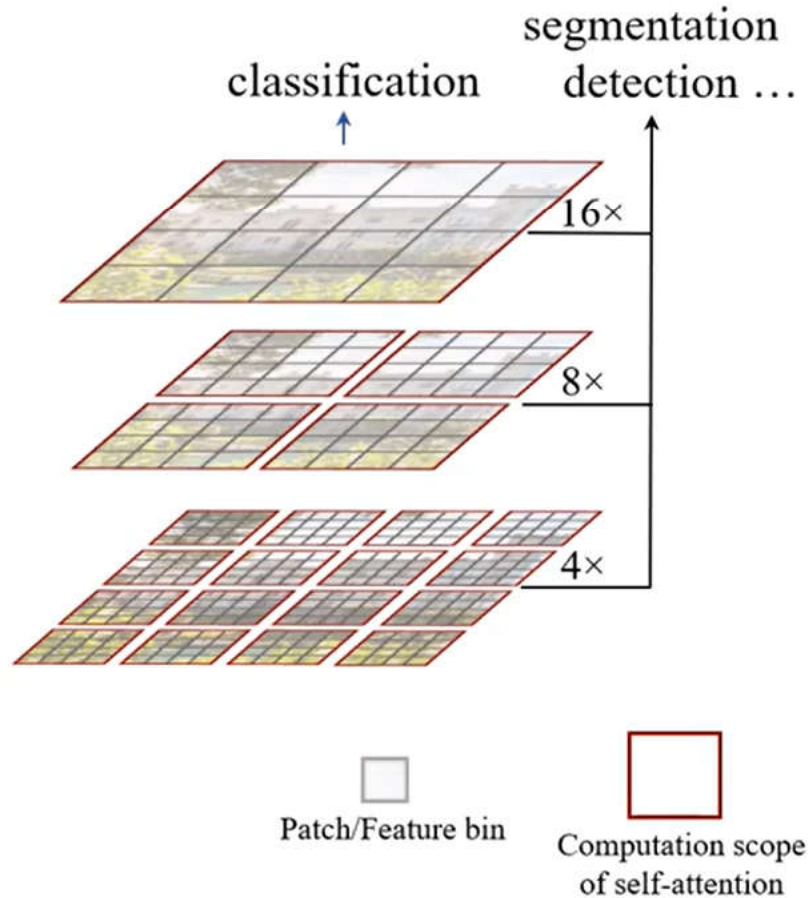


Hierarchy

- Processing objects of different scales

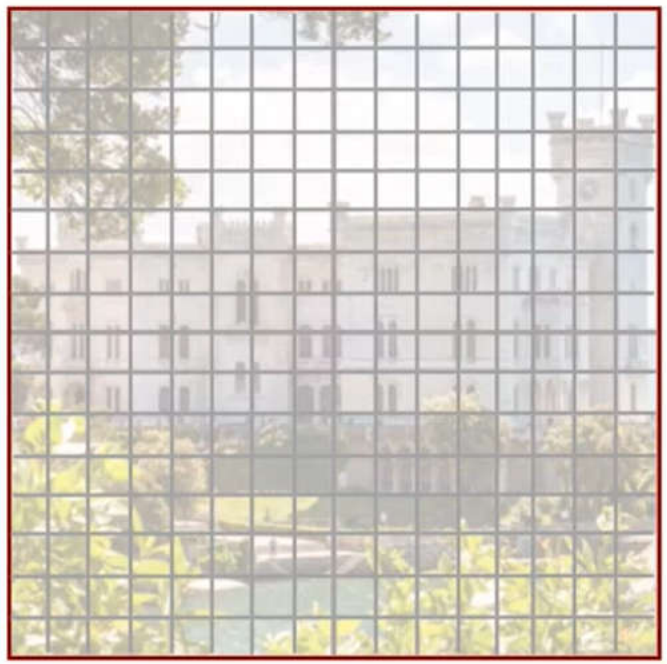


Left figure credit by Ross Girshick

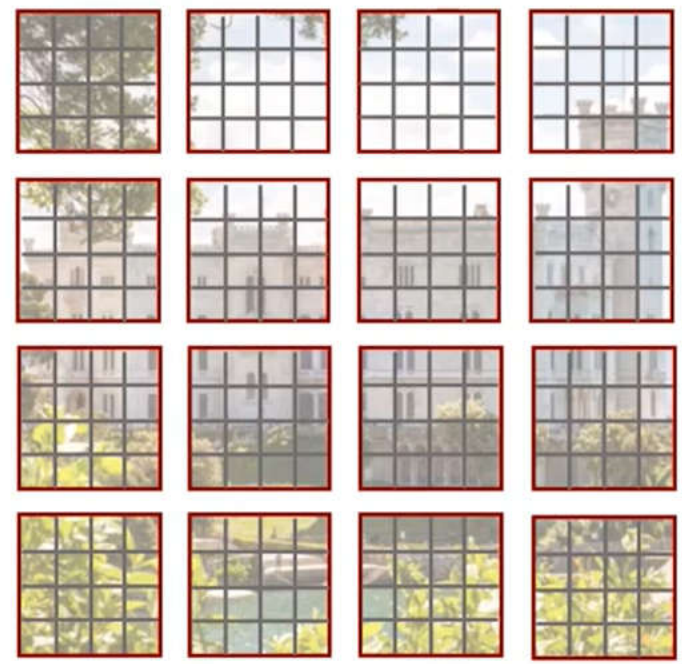


Locality by non-overlapped windows

- Proves beneficial in modeling the high correlation in visual signals (Yann LeCun)



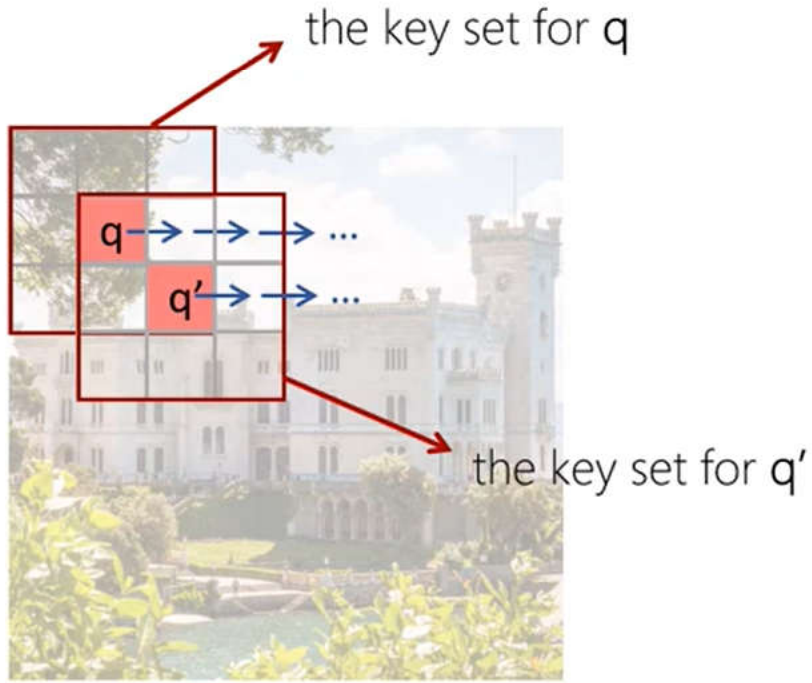
ViT: 256^2=65536 (Global)



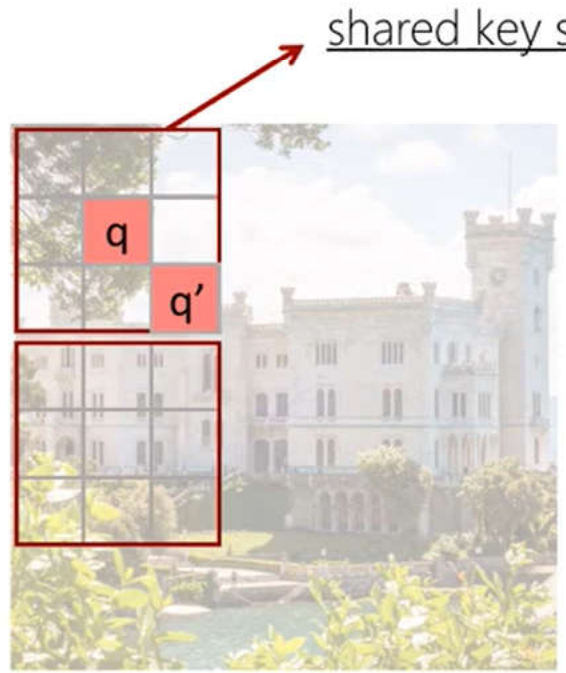
Swin Transformer: 16x16^2=4096 (Local)

Locality by non-overlapped windows

- Compared to sliding window (LR-Net)
  - Shared key set enables friendly memory access and is thus good for speed (larger than 3x)



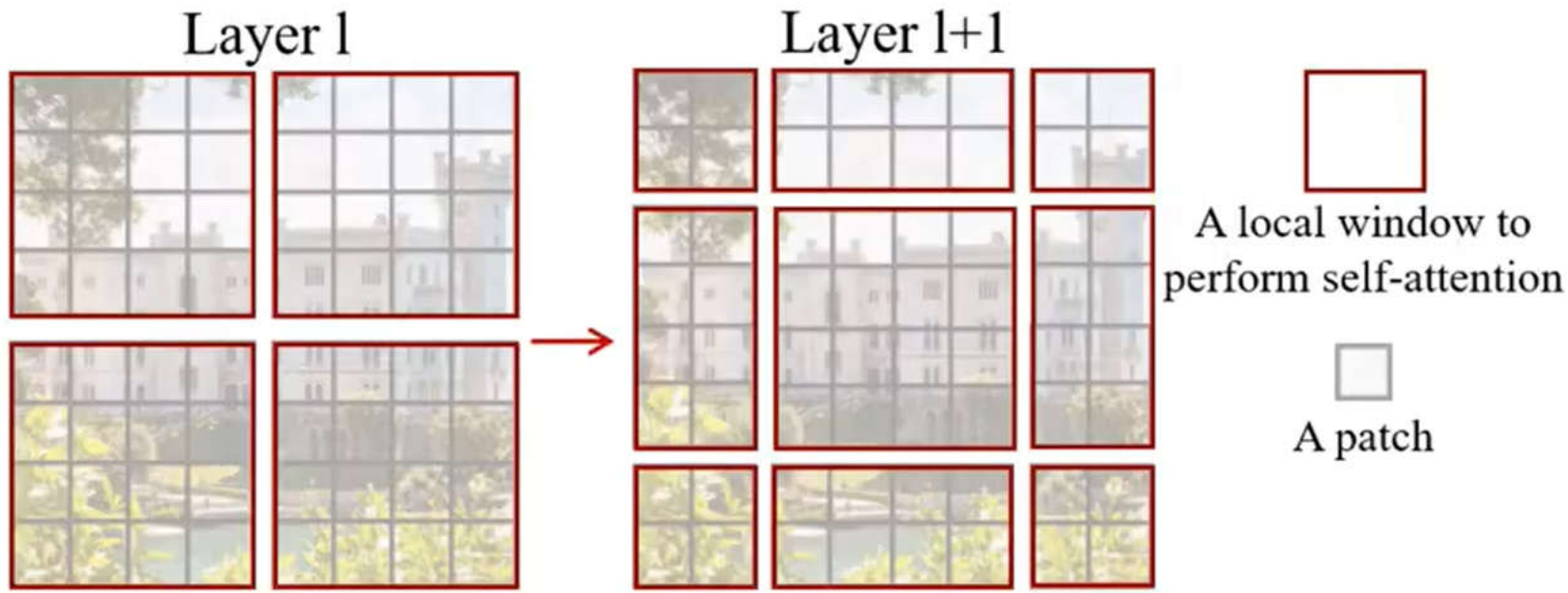
sliding window (LR-Net)



Non-overlapped window (Swin Transformer)

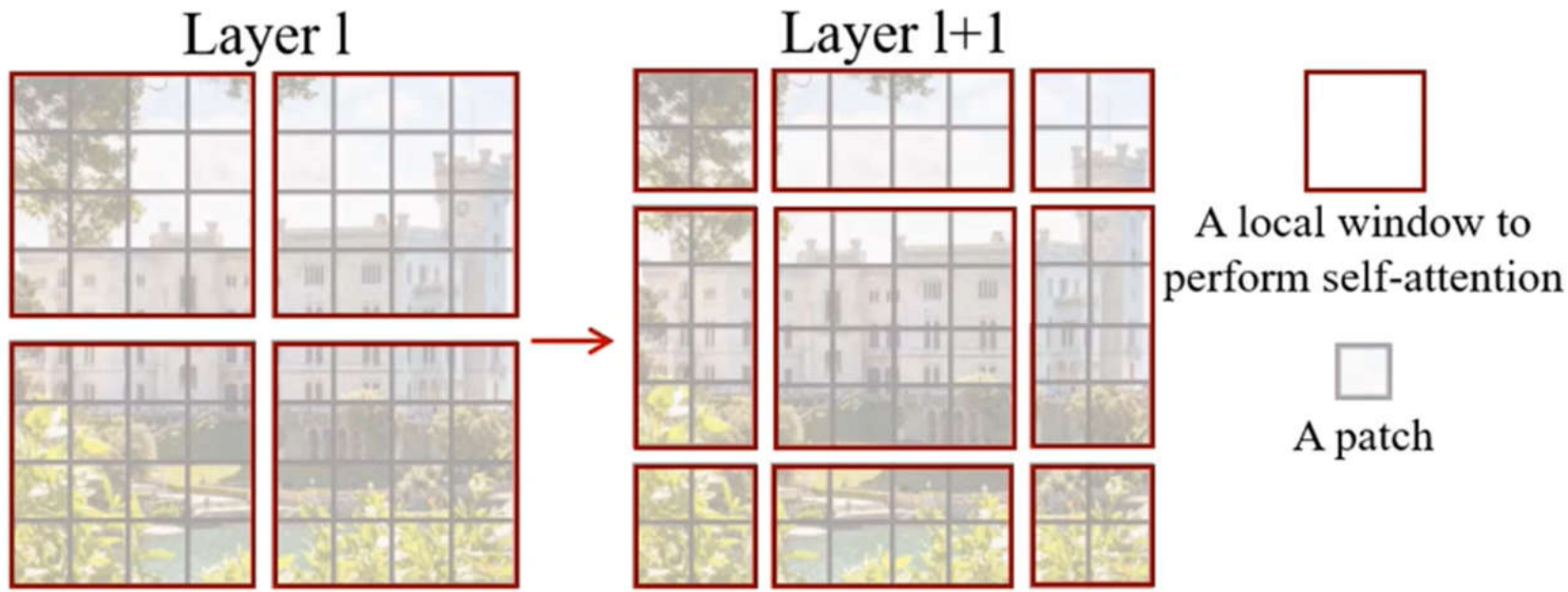
Shifted non-overlapped windows

- Enable cross-window connection
  - Non-overlapped windows will result in no connection between windows
  - Performs as effective or even slightly better than the sliding window approach, due to regularization effects



Shifted non-overlapped windows

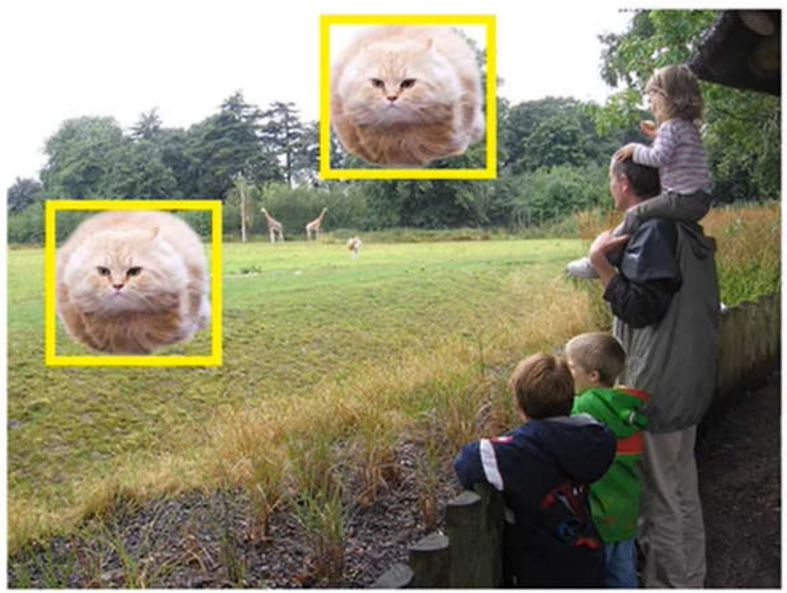
- Enable cross-window connection
  - Non-overlapped windows will result in no connection between windows
  - Performs as effective or even slightly better than the sliding window approach, due to regularization effects



Translational semi-invariance

- Relative position bias plays a more important role in vision than in NLP

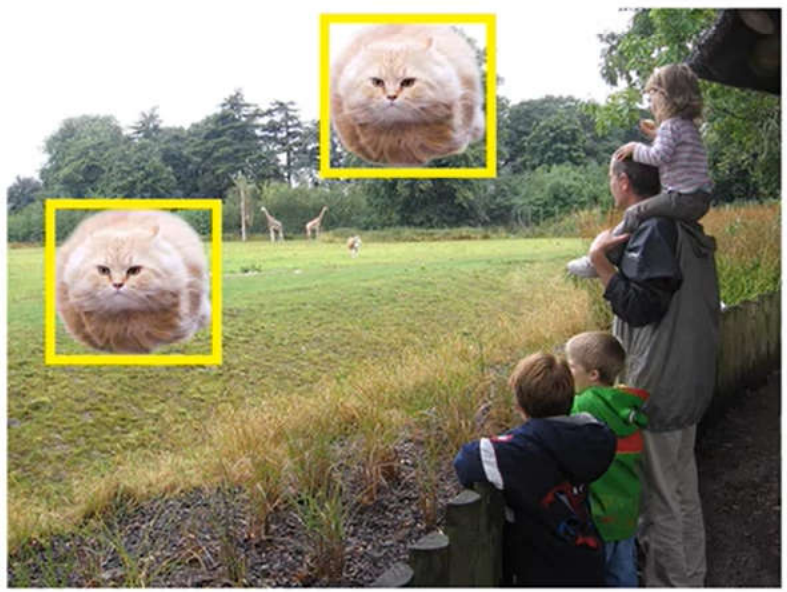
$$\textbf{Attention}(Q, K, V) = \textbf{SoftMax}(QK^T/\sqrt{d} + B)V,$$

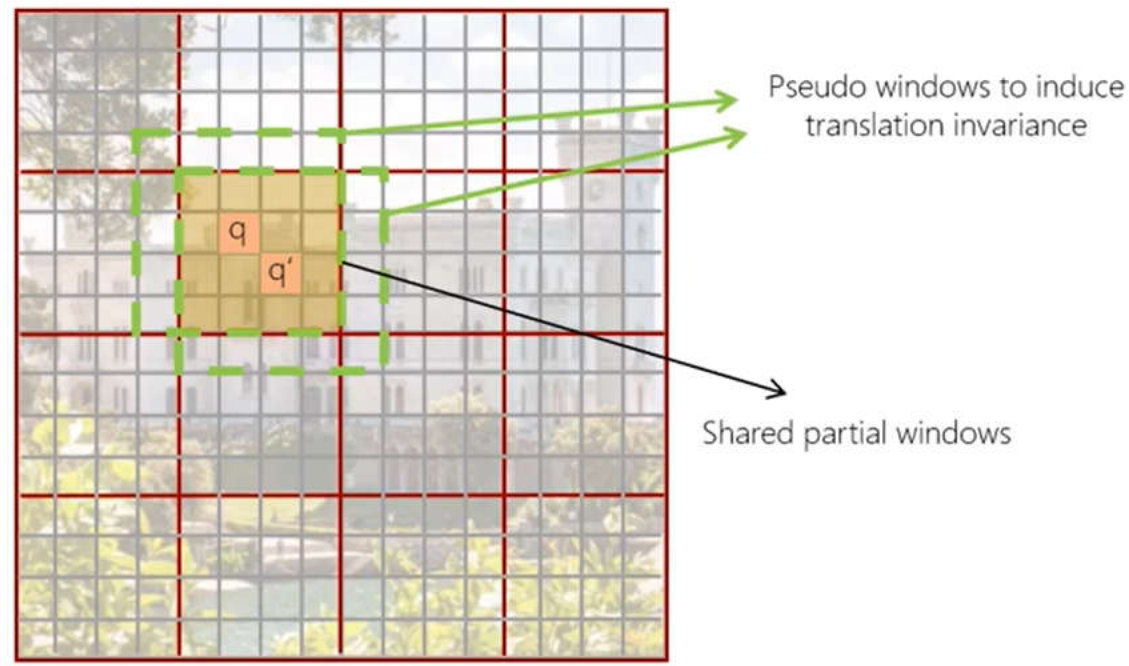


Translational semi-invariance

- Relative position bias plays a more important role in vision than in NLP

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

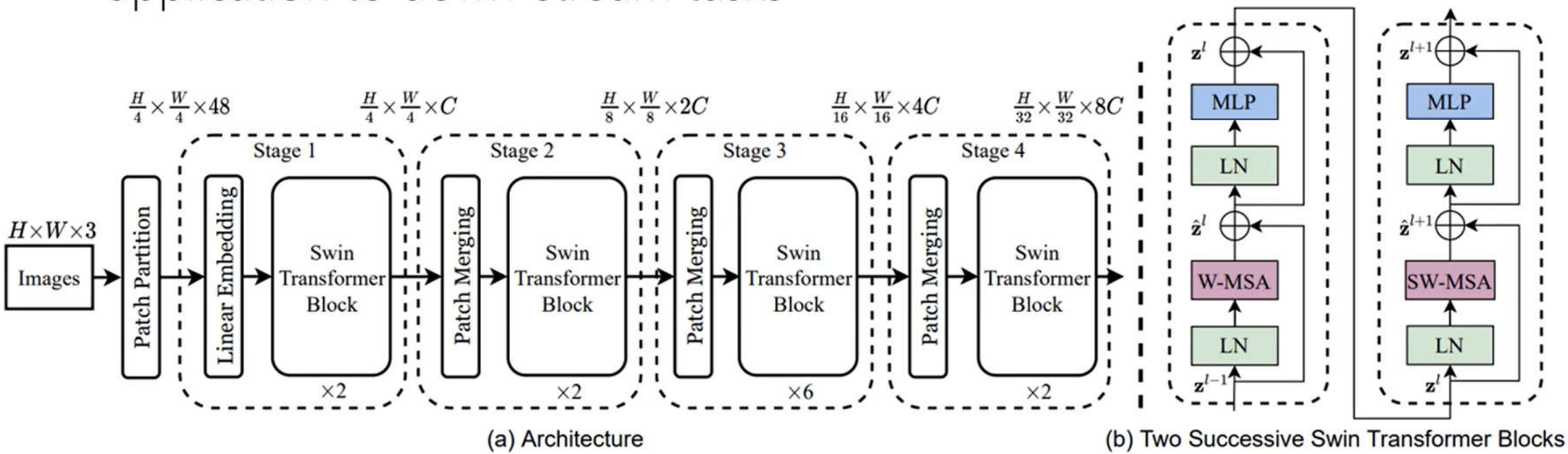




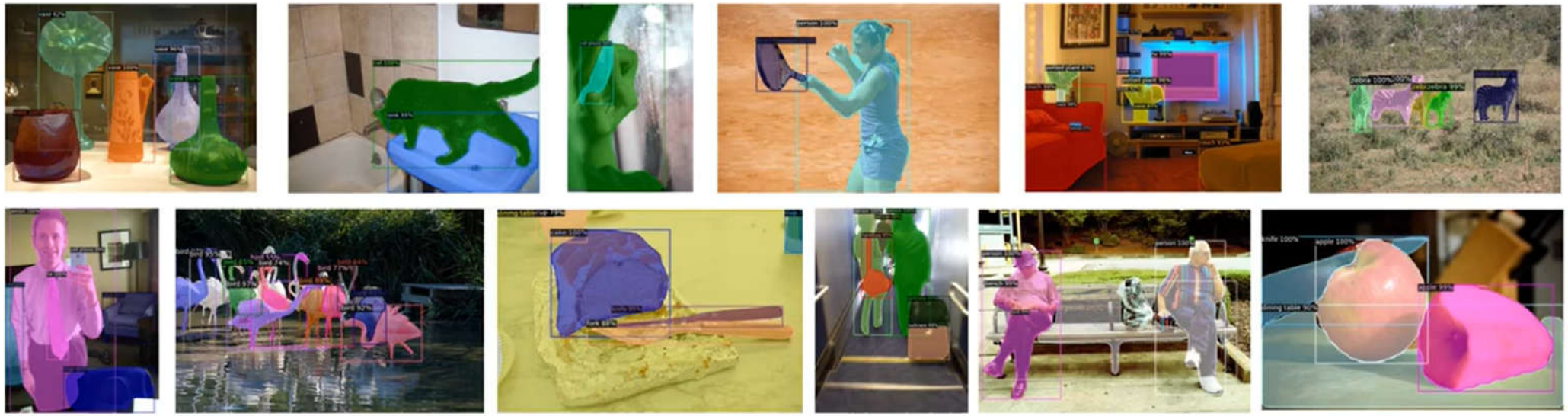
semi-invariance is as effective as full-invariance in our experiments

Architecture instantiations

- Resolution of each stage is set similar as ResNet, to facilitate application to down-stream tasks

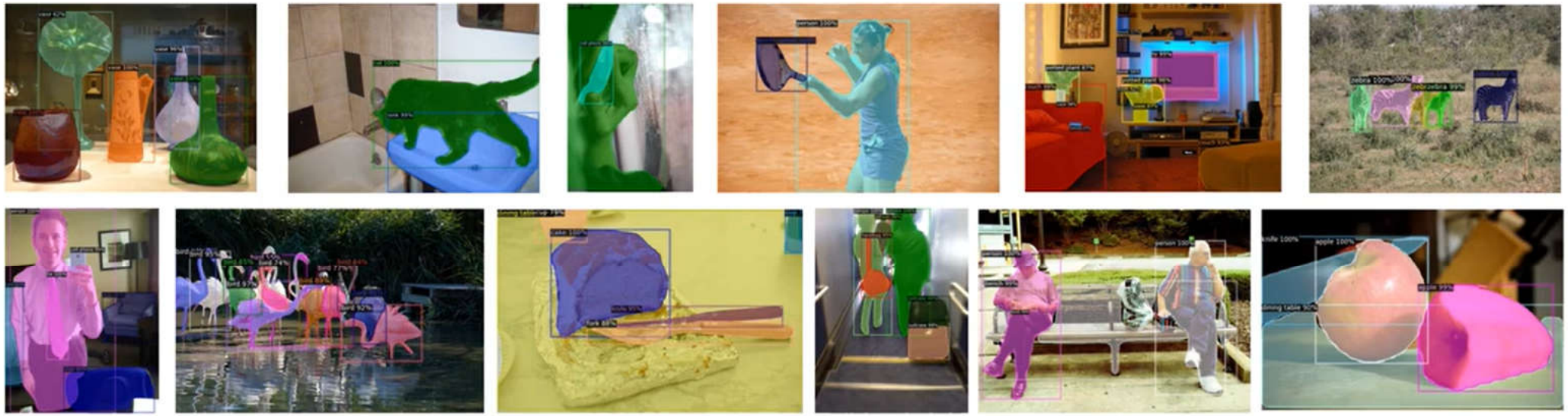


Application: object detection



- COCO object detection: #1 #2 #3 for single model (60.6 mAP)
  - Significantly surpass all previous CNN models (+3.5 mAP)
- COCO instance segmentation: #1 for single model (52.4 mAP)
  - Significantly surpass all previous CNN models (+3.3 mAP)

Application: object detection



- COCO object detection: #1 #2 #3 for single model (60.6 mAP)
  - Significantly surpass all previous CNN models (+3.5 mAP)
- COCO instance segmentation: #1 for single model (52.4 mAP)
  - Significantly surpass all previous CNN models (+3.3 mAP)

Application: object detection

- Performs consistently better than CNN on various object detectors and various model sizes (+3~4.5 mAP)

(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	+4.2
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	+3.7
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	+3.5
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	+3.4

(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	+4.2
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	+3.7
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	+3.6

Application: semantic segmentation

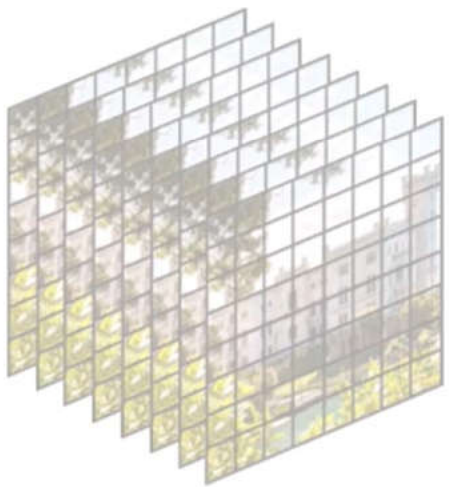
- **ADE20K semantic segmentation: #1 for single model (53.9 mIoU)**
  - The largest and most difficult semantic segmentation benchmark
    - 20,000 training images, 150 categories
  - Significantly surpass all previous CNN models (+5.5 mIoU vs. the previous best CNN model)

Application: ImageNet-1K classification

- Swin Transformer surpasses ViT-L/16 with the same pre-training data

Model	data	params	FPS	ImageNet-1K (top-1)
EfficientNet-L2	JFT-300M	480M	--	85.5
ViT-L/16	ImageNet-14M	307M	27	85.15
ViT-H/14	JFT-300M	632M	10	88.55
ViT-G/14	JFT-3B	3B	-	90.45
Swin (Huge)	**ImageNet-14M**	**391M**	**25**	**88.65 (+3.5)**

Application: video recognition (coming soon)



3D tokens: T'×H'×W' = 8×8×8
Window size: P×M×M = 4×4×4

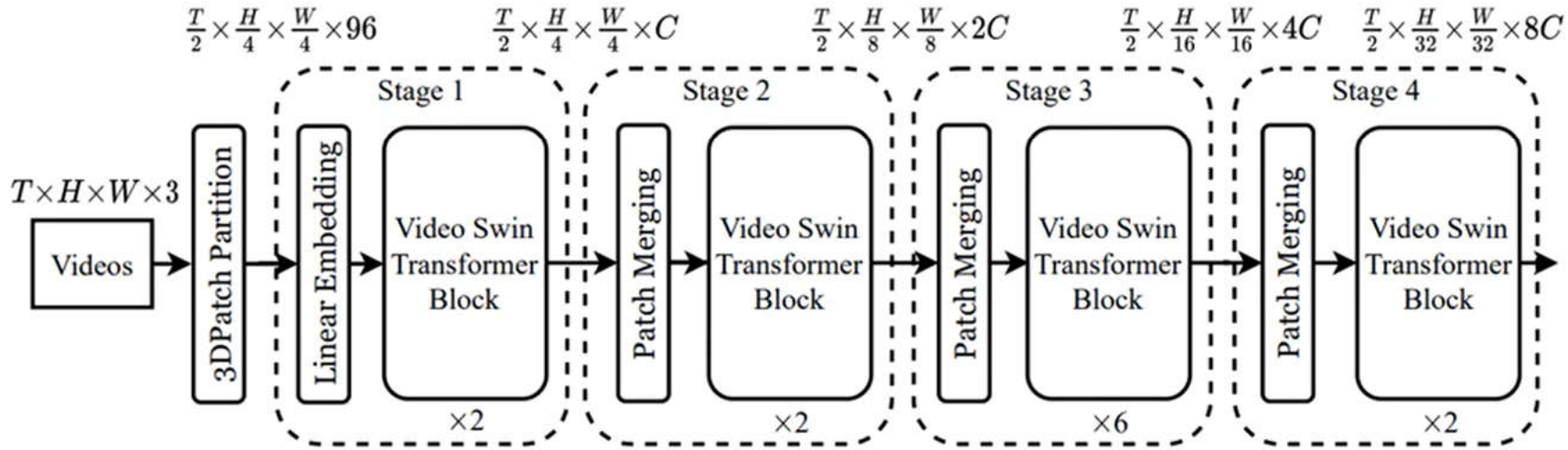


Figure 2: Overall architecture of Video Swin Transformer (tiny version, referred to as Swin-T).

Application: video recognition

- Swin Transformer achieves SOTA on major video benchmarks with 20x less pre-training data and 3x smaller model size

Table 1: Comparison to state-of-the-art on Kinetics-400. "384↑" signifies that the model uses a larger spatial resolution of 384×384. "Views" indicates # temporal clip × # spatial crop. The magnitudes are Giga (10^9) and Mega (10^6) for FLOPs and Param respectively.

Method	Pretrain	Top-1	Top-5	Views	FLOPs	Param
R(2+1)D [37]	-	72.0	90.0	10 × 1	75	61.8
I3D [6]	ImageNet-1K	72.1	90.3	-	108	25.0
NL I3D-101 [40]	ImageNet-1K	77.7	93.3	10 × 3	359	61.8
ip-CSN-152 [36]	-	77.8	92.8	10 × 3	109	32.8
CorrNet-101 [39]	-	79.2	-	10 × 3	224	-
SlowFast R101+NL [13]	-	79.8	93.9	10 × 3	234	59.9
X3D-XXL [12]	-	80.4	94.6	10 × 3	144	20.3
MViT-B, 32×3 [10]	-	80.2	94.4	1 × 5	170	36.6
MViT-B, 64×3 [10]	-	81.2	95.1	3 × 3	455	36.6
TimeSformer-L [3]	ImageNet-21K	80.7	94.7	1 × 3	2380	121.4
ViT-B-VTN [29]	ImageNet-21K	78.6	93.7	1 × 1	4218	11.04
ViViT-L/16x2 [1]	ImageNet-21K	80.6	94.7	4 × 3	1446	310.8
ViViT-L/16x2 320 [1]	ImageNet-21K	81.3	94.7	4 × 3	3992	310.8
ip-CSN-152 [36]	IG-65M	82.5	95.3	10 × 3	109	32.8
ViViT-L/16x2 [1]	JFT-300M	82.8	95.5	4 × 3	1446	310.8
ViViT-L/16x2 320 [1]	JFT-300M	83.5	95.5	4 × 3	3992	310.8
ViViT-H/16x2 [1]	JFT-300M	84.8	95.8	4 × 3	8316	647.5
Swin-T	ImageNet-1K	78.8	93.6	4 × 3	88	28.2
Swin-S	ImageNet-1K	80.6	94.5	4 × 3	166	49.8
Swin-B	ImageNet-1K	80.6	94.6	4 × 3	282	88.1
Swin-B	ImageNet-21K	82.7	95.5	4 × 3	282	88.1
Swin-L	ImageNet-21K	83.1	95.9	4 × 3	604	197.0
Swin-L (384↑)	ImageNet-21K	**84.9**	**96.6**	10 × 5	2107	200.0

+3.6% using the same pre-training data

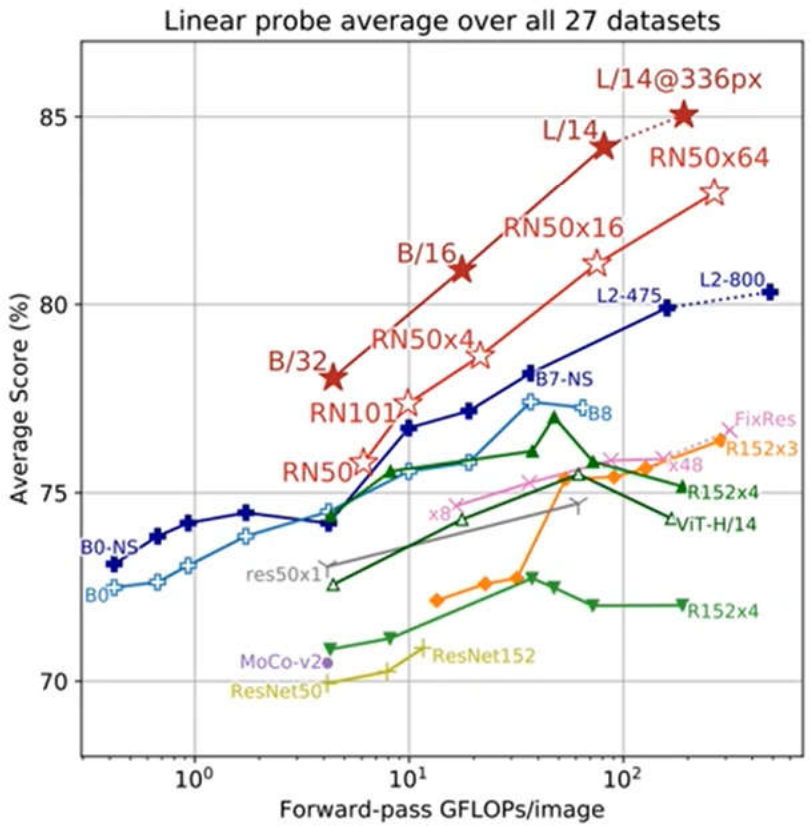
Table 2: Comparison to state-of-the-art on Kinetics-600.

Method	Pretrain	Top-1	Top-5	Views	FLOPs	Param
SlowFast R101+NL [13]	-	81.8	95.1	10 × 3	234	59.9
X3D-XL [12]	-	81.9	95.5	10 × 3	48	11.0
MViT-B-24, 32×3 [9]	-	83.8	96.3	5 × 1	236	52.9
TimeSformer-HR [3]	ImageNet-21K	82.4	96	1 × 3	1703	121.4
ViViT-L/16x2 320 [1]	ImageNet-21K	83.0	95.7	4 × 3	3992	310.8
ViViT-H/16x2 [9]	JFT-300M	85.8	96.5	4 × 3	8316	647.5
Swin-B	ImageNet-21K	83.8	96.4	4 × 3	282	88.1
Swin-L (384↑)	ImageNet-21K	**85.9**	**97.1**	4 × 3	2107	200.0

+2.9% using the same pre-training data

Reason IV to use Transformer in computer vision

- Better connect vision and language: unified modeling

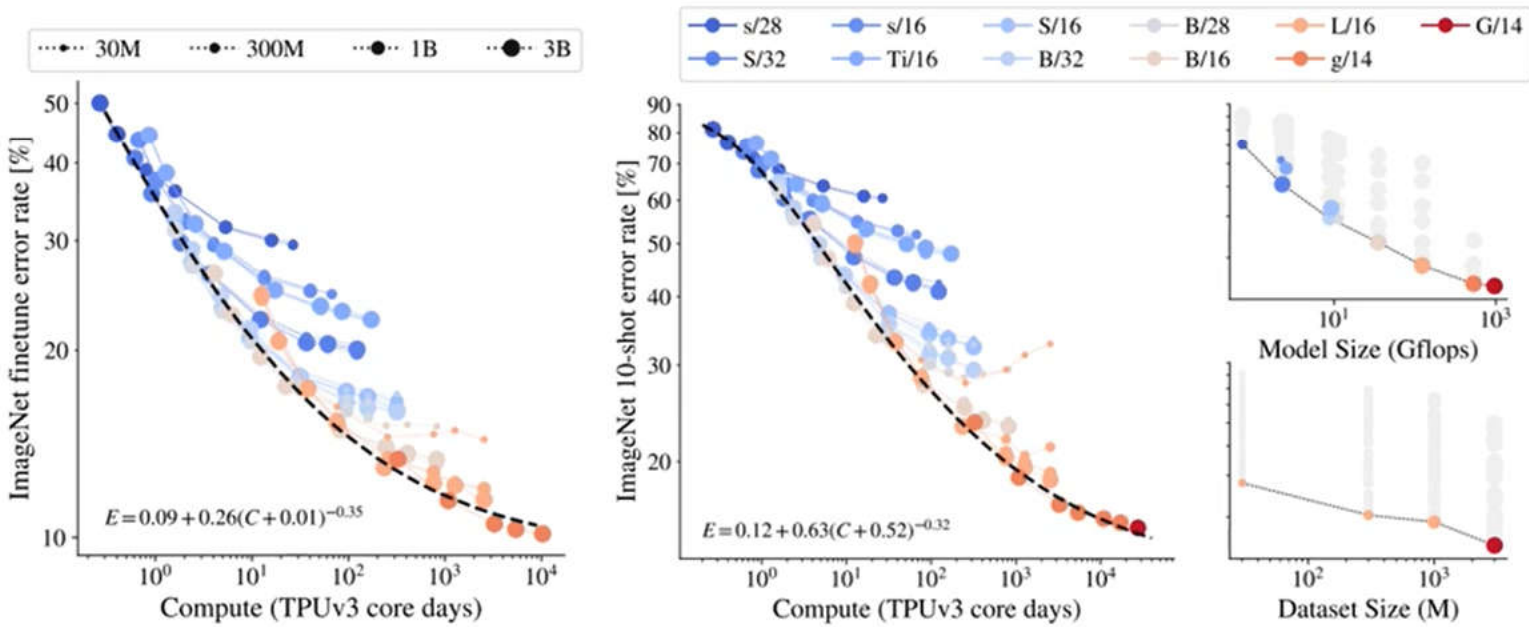


CLIP

https://openai.com/blog/clip/

Reason V to use Transformer in computer vision

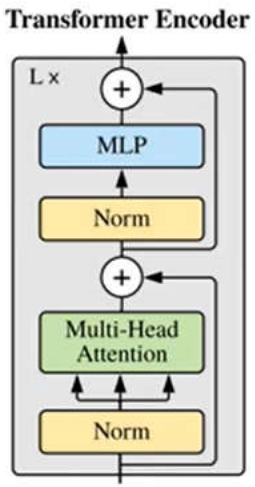
- Scalable to models and data



- ViT G/14
  - 1000 G Flops
  - 2B parameters
  - 3B images

Scaling Vision Transformers: https://arxiv.org/pdf/2106.04560.pdf

Summary: 4 years unleash the power of Transformer in CV

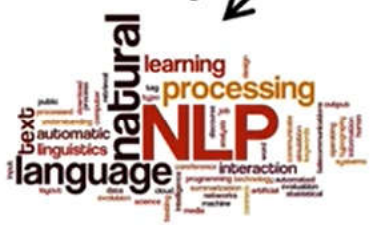


RelationNet (CVPR'2018)
LearnRegionFeat (ECCV'2018)
STRN (ICCV'2020)
MEGA (CVPR'2020)
DeTr (ECCV'2020)
RelationNet++ (NeruIPS'2020)

Transformer (2017.6)

refresh & dominate

refresh & dominate



ViT-G (Arxiv'2021)

Reason I: General modeling capability

2019.4

2021.1

Reason V: Scalability

2017.06

2017.11

Reason II: Complement convolution

NLNet (CVPR'2018)
GCNet (ICCVW'2019)
DNL (ECCV'2020)

Reason III: Strong modeling power

LRNet (ICCV'2019)
ViT (ICLR'2021)
Swin Transformer (Arxiv'2021)

Reason IV: Better connect vision and language

CLIP (ICML'2021)

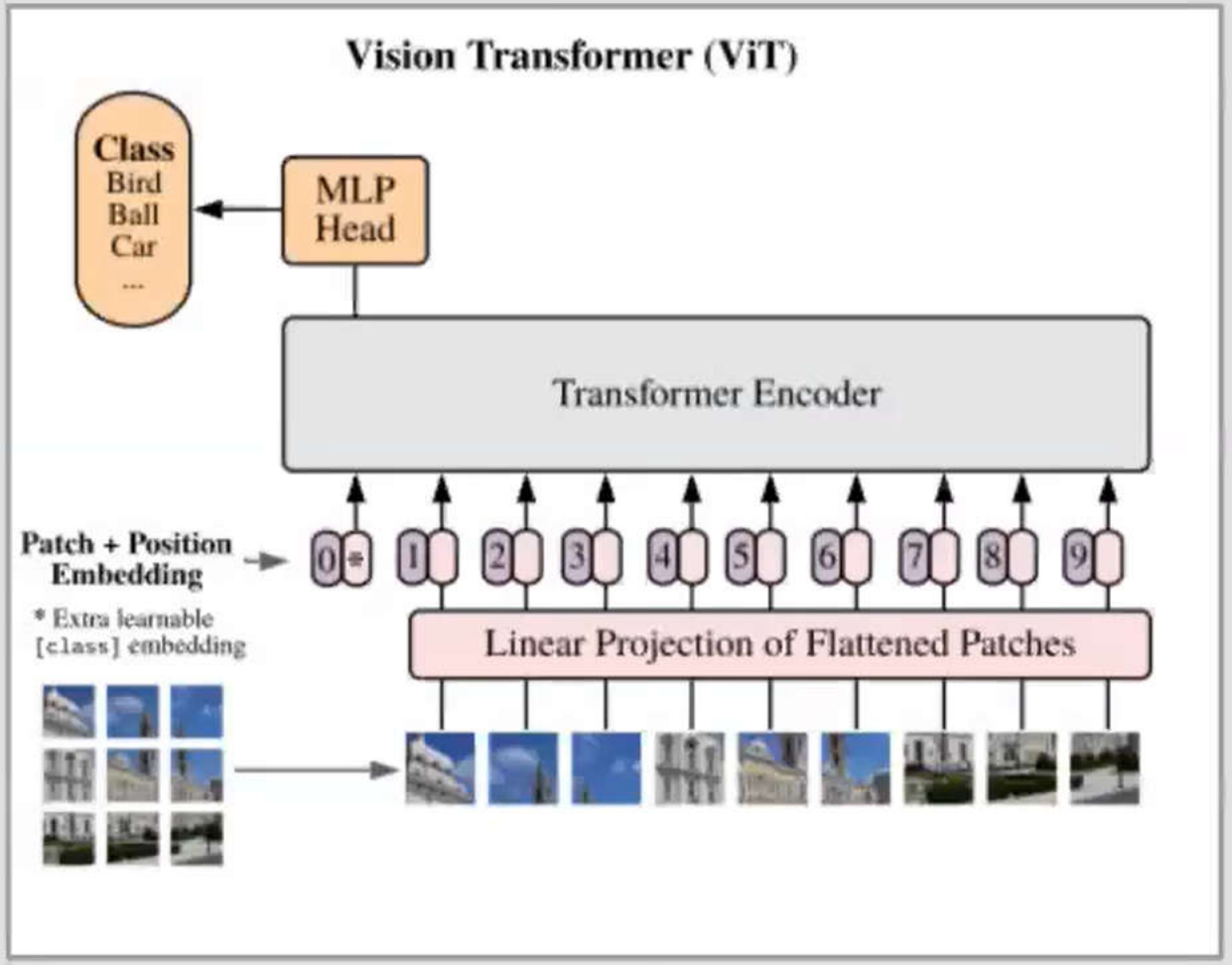
2021.6

sehar

VISION TRANSFORMER

- The **Vision Transformer**, or **ViT**, is a model for image classification that has Transformer-like architecture.

- An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder.

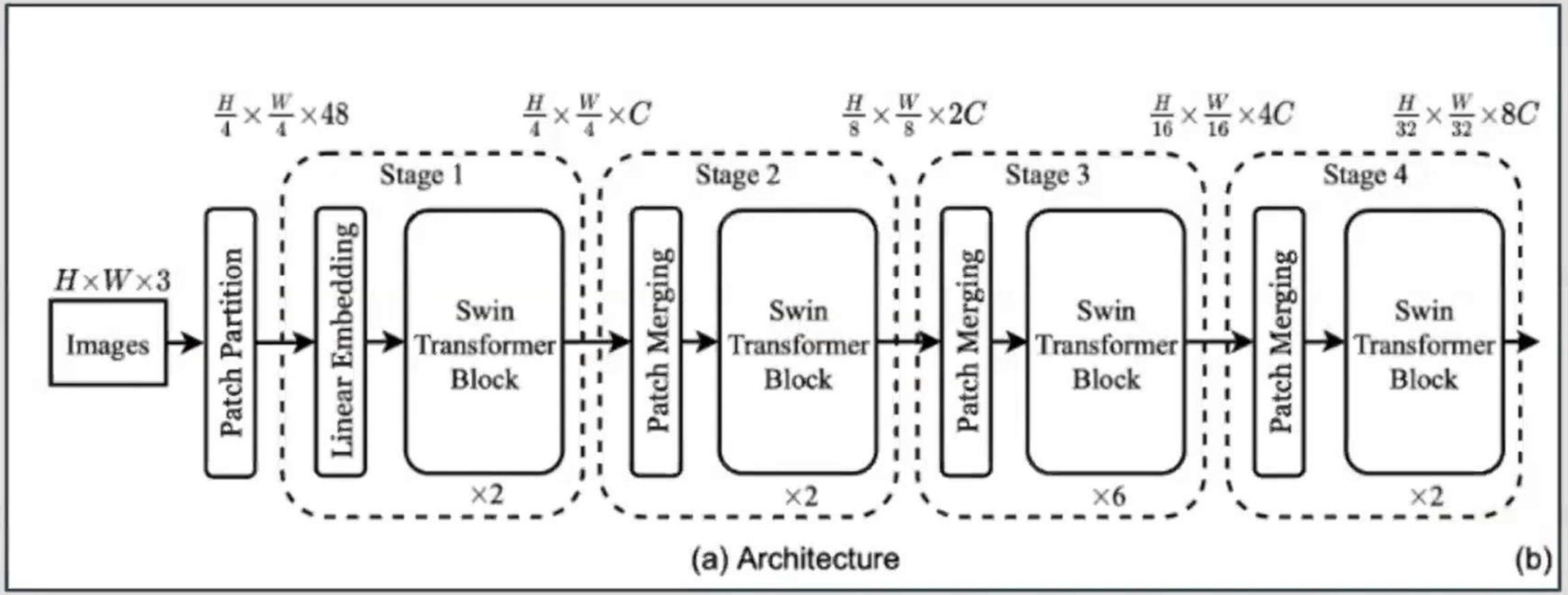


https://arxiv.org/abs/2010.11929

MOTIVATION

- General purpose backbone of a transformer
- Hierarchical feature representation and has linear computational complexity with respect to input image size.
- Shifted window based self-attention is shown to be effective and efficient on vision problems.

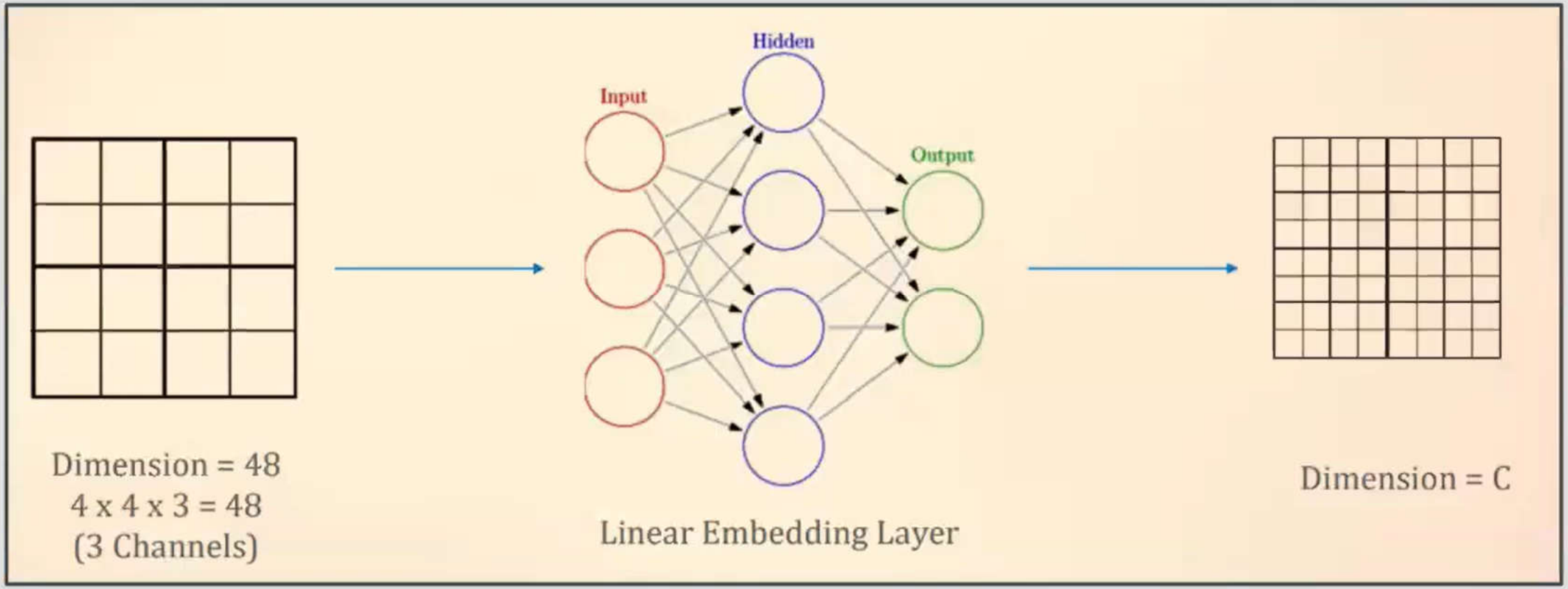
SWIN TRANSFORMER



https://arxiv.org/abs/2103.14030

Composed of stages of patch merging and successive transformer blocks

PATCH PARTITION & LINEAR EMBEDDING



https://www.crcv.ucf.edu/wp-content/uploads/2018/11/lecture-8-Swin-Transformer-Hierarchical-Vision-Transfromer-using-Shifted-Windows-slide.pdf

Where C is the channel numbers of hidden layers in the Stage 1