Swin Transformer

Team #9

Sujy Han

Jaehyup Seong

Al Jamil, Mouhammed

2022.05.27

• Index

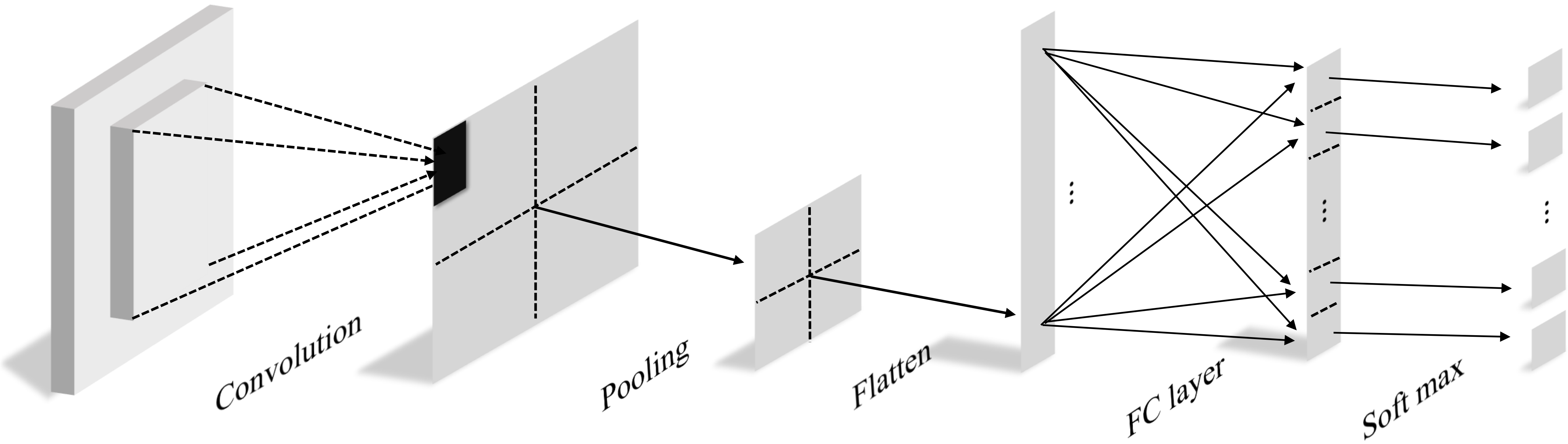
- ➢ CNN and ViT
  - CNN architecture
  - Inductive bias
  - Vision Transformer

- ➢ DeiT
  - DeiT and CNN
  - What is DeiT?
  - DeiT and other Transformers

- ➢ Swin Transformers
  - ➢ Problem statements
  - ➢ Proposed methods
  - ➢ Experiments
  - ➢ Adaptation

Index

• CNN Architecture



- ➢ The convolution operation derives the **translation equivariance**
- ➢ Pooling layer and Softmax function derive the **translation invariance**
- ➢ We call these 2 properties the **inductive bias** of CNN architecture

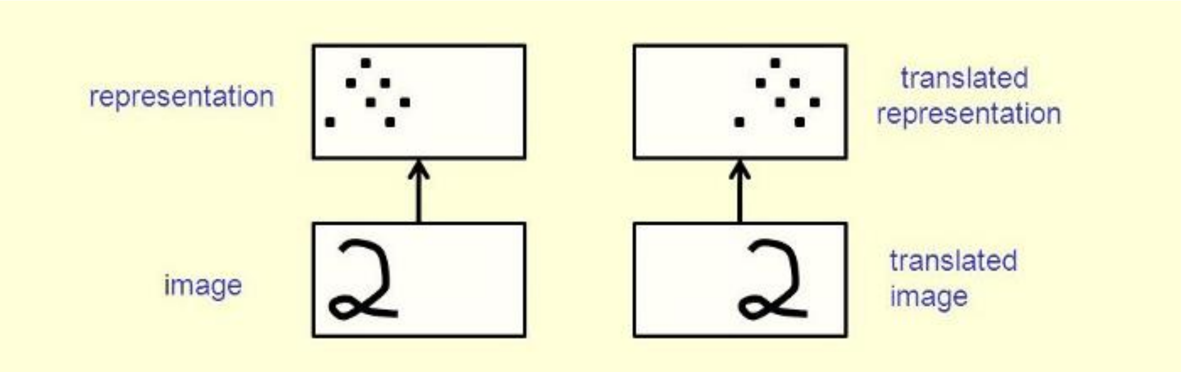
• Inductive bias of CNN architecture

❖ **Transition Invariance**

➢ Although the location of the target is changed, the output of CNN architecture remains

❖ **Transition Equivariance**

➢ Although the location of the target is changed, the parameter of kernel is same



• Inductive bias of CNN architecture

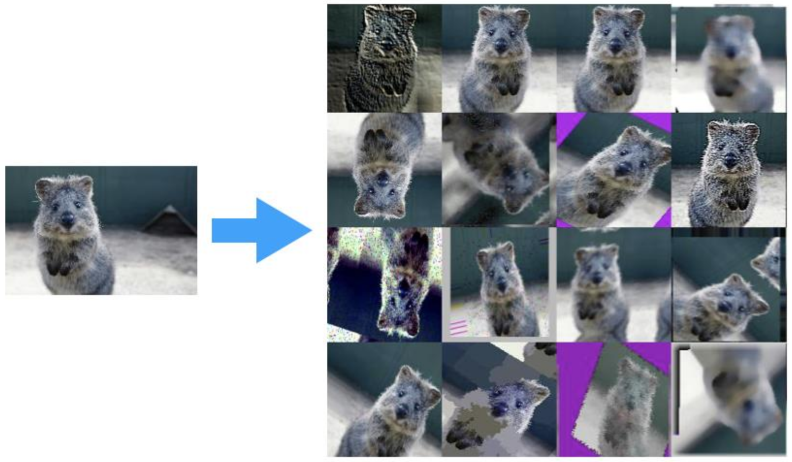
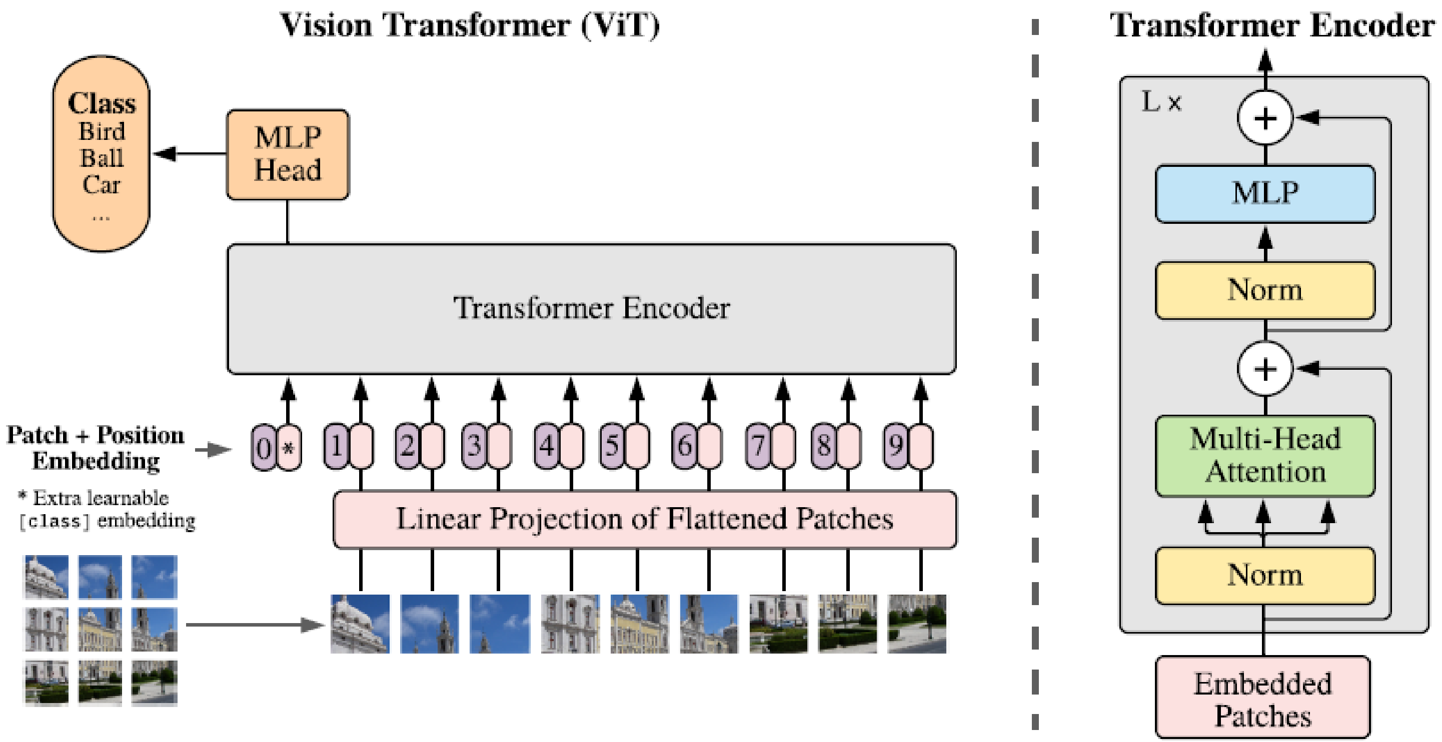


Figure credit: https://github.com/aleju/imgaug

- ❖ **Strong aspects of pre-defined properties**
  - ➢ Well predict most unseen images with pre-defined prediction
  - ➢ Good performance even with the low number of training data

- ❖ **Weak aspects of pre-defined properties**
  - ➢ Loss of spatial features due to translation invariance
    - • This can be overcome by data augmentation, but it demands a lot of cost overhead
  - ➢ Poorly predict some images, out of the pre-defined induction (Poorly train about global information)

• Vision Transformer (ViT)



- ➢ Idea from the transformer, which used for NLP
- ➢ ViT has no convolution operation and the pooling layer
- ➢ Thus, ViT has no **inductive bias** of translation equivariance and translation invariance

• Vision Transformer (ViT)

How can we deal without inductive bias?

- ❖ **Huge amount of training data (e.g., JFT-300M)**
  - ➢ To predict the image without the pre-define induction, generalized properties of the image are required
  - ➢ Generalized properties of images can be derived with a huge amount of data

- ❖ **Without convolution operation and pooling layer,**

  Attention is all you need

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

• Vision Transformer (ViT)

Image: $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$

$N = HW/P^2$

1. To make flatten patches

Patch: $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$

$\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$

2. Linear projection

Linearly projected patch: $\mathbf{x}_p\mathbf{E} \in \mathbb{R}^{N \times D}$

$\mathbf{x}_{class} \in \mathbb{R}^{D}$

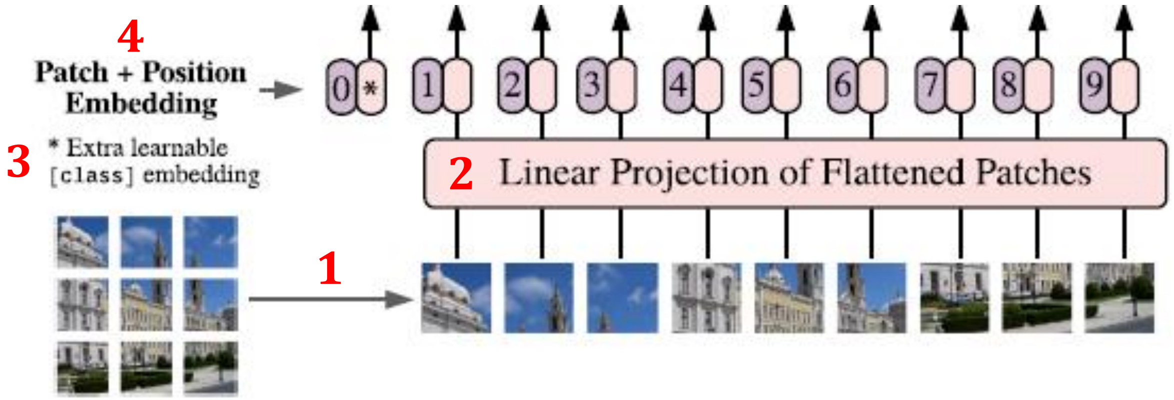
3. Add the class token

Add Class token: $\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}]$

$\mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$

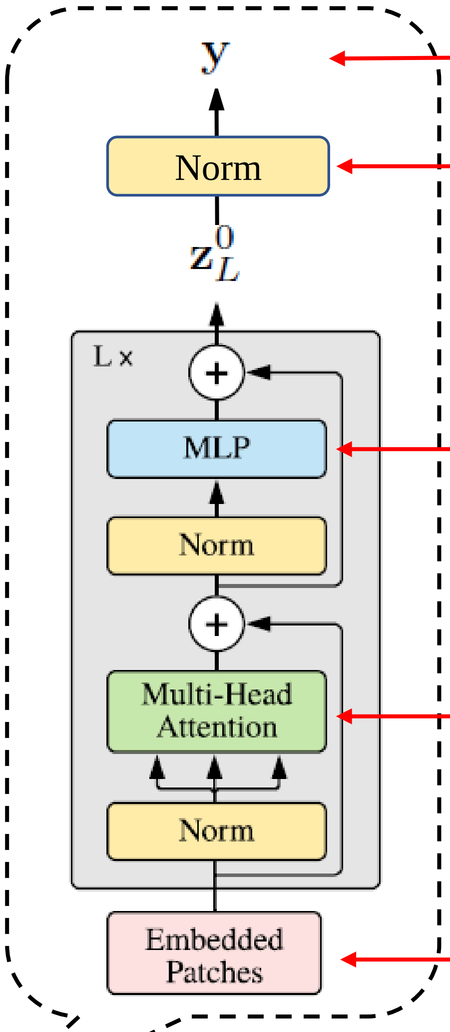
4. Add the positioning vector

Embed positioning vector: $\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos},$



$$E_{(pos,\,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$E_{(pos,\,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

• Vision Transformer (ViT)



The input of MLP head, $\mathbf{y} \in \mathbb{R}^D$

$$\mathbf{y} = \mathrm{LN}(\mathbf{z}_L^0)$$

$$\mathbf{z}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell$$

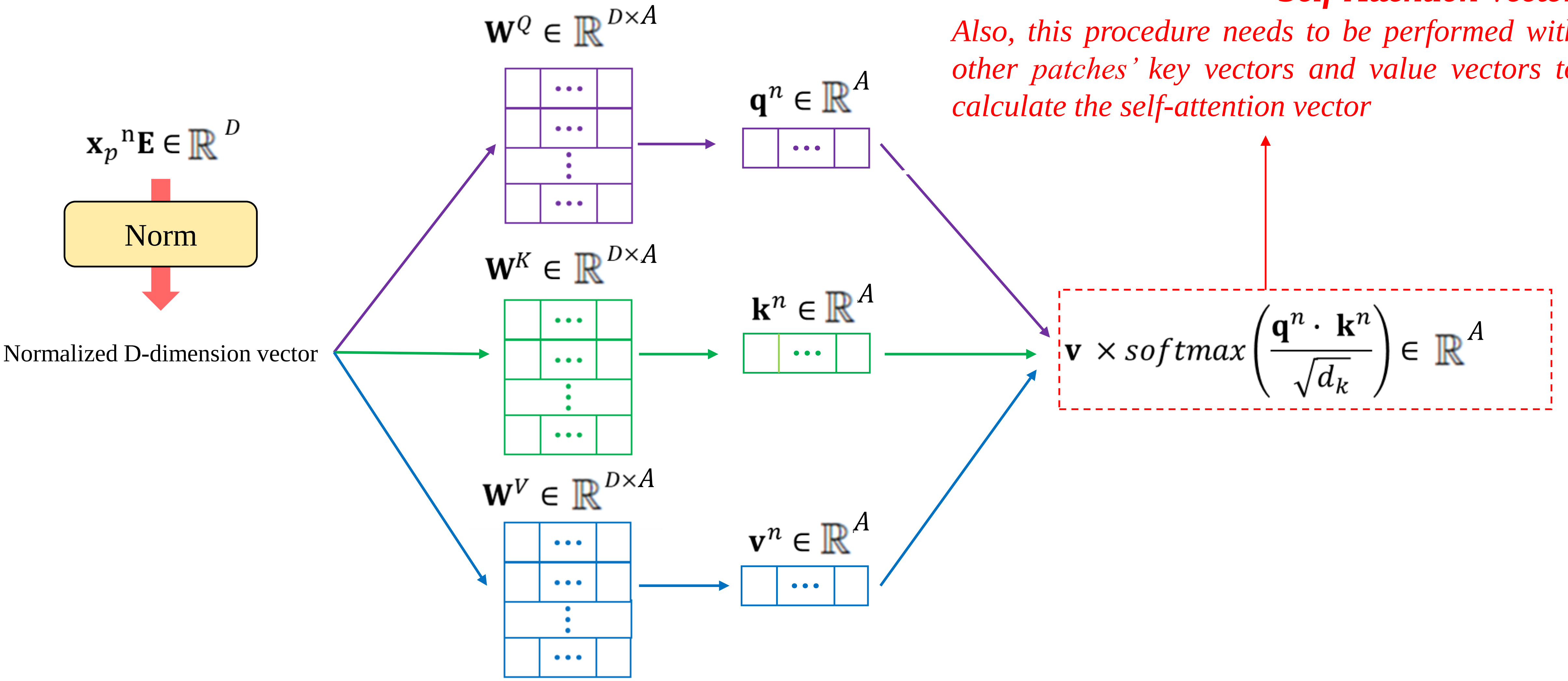
$$\mathbf{z}'_\ell = \mathrm{MSA}(\underbrace{\mathrm{LN}(\mathbf{z}_{\ell-1})}_{\textit{Linear Normalization}}) + \mathbf{z}_{\ell-1},$$

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}$$

Transformer Encoder

• Vision Transformer (ViT)

Single Self-Attention



Self-Attention Vector

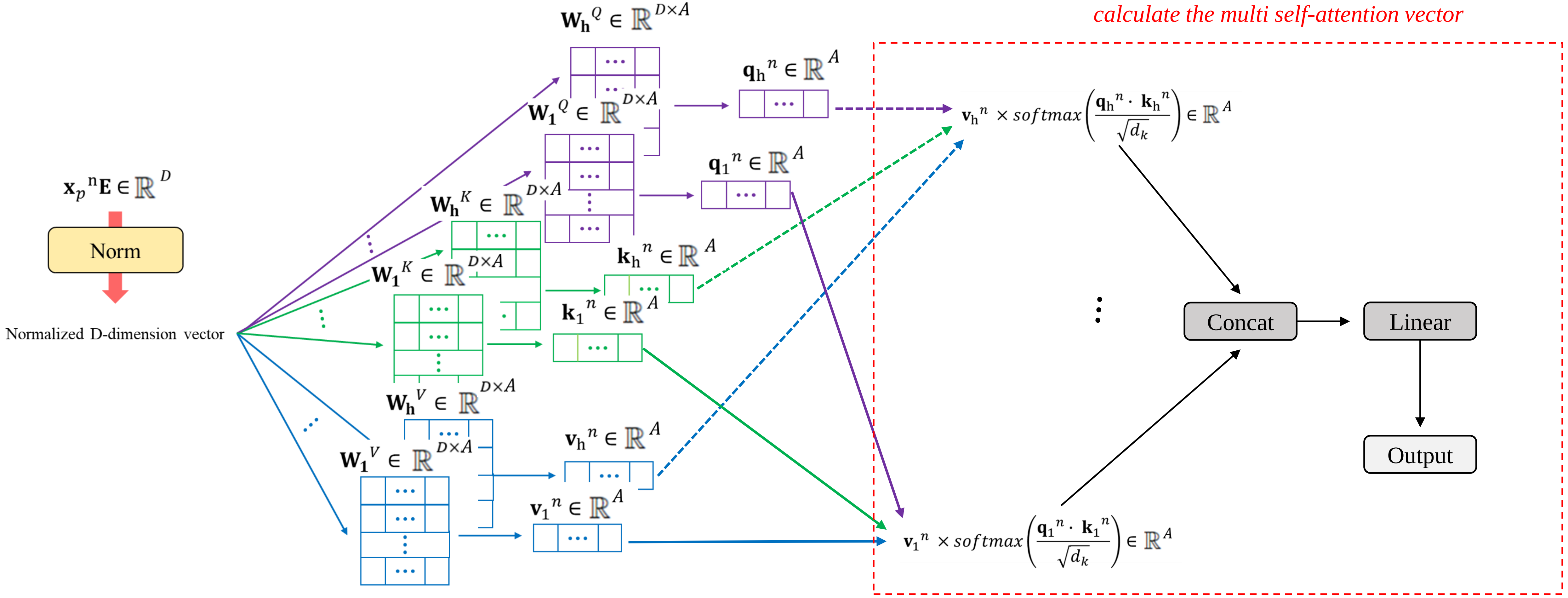
Also, this procedure needs to be performed with other patches' key vectors and value vectors to calculate the self-attention vector

• Vision Transformer (ViT)

Multi Self-Attention (Multi-head attention)

Multi self-Attention Vector

Also, this procedure needs to be performed with other patches' key vectors and value vectors to calculate the multi self-attention vector



• Vision Transformer (ViT)

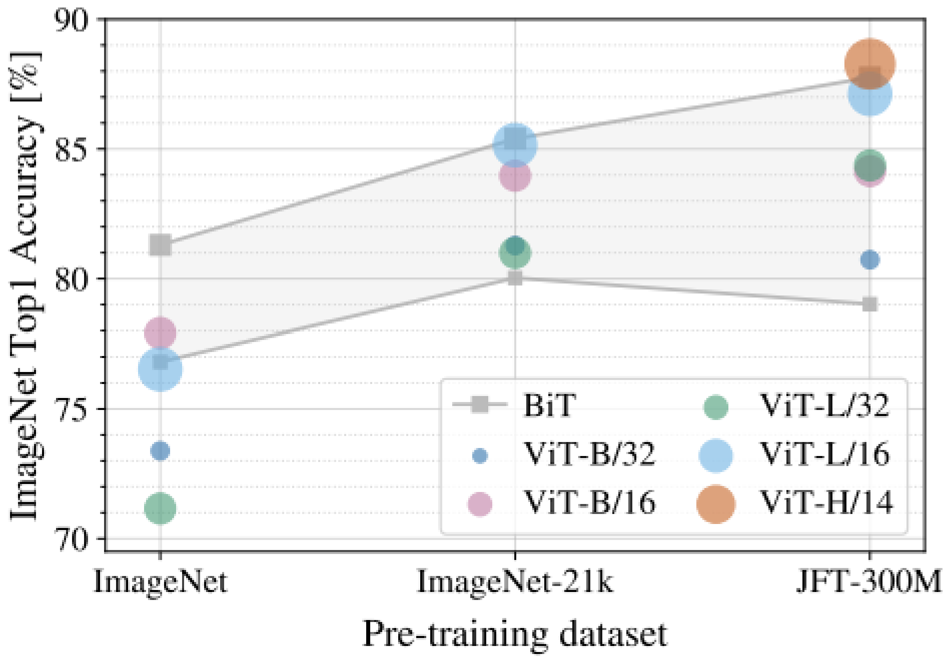
Pre-trained with JFT-300M dataset

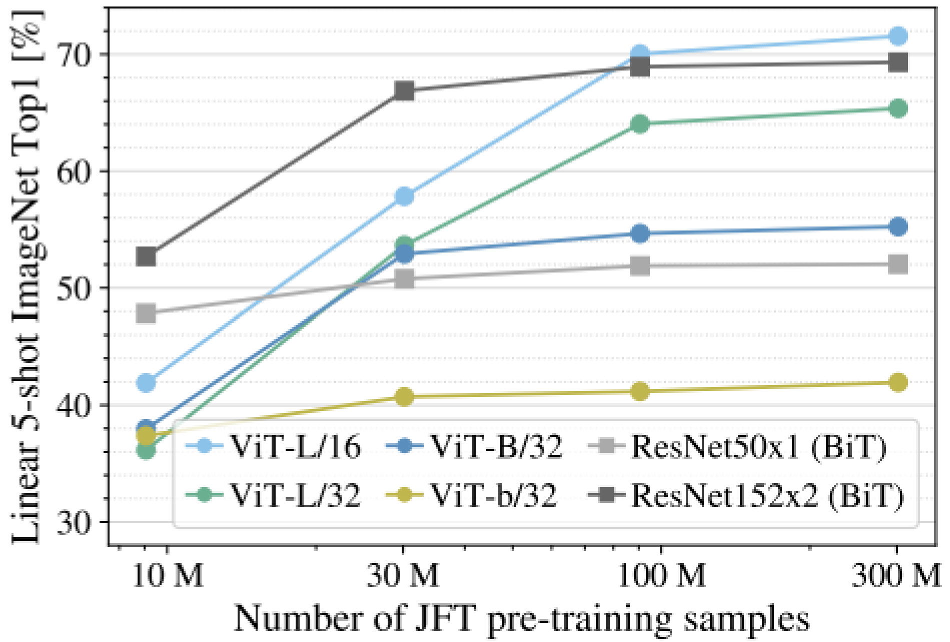
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15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

- Although the number of patches increases when the size of the image gets bigger, the amount of GPU memory does not change much.
- Therefore, ViT can use the large batch size, which enables to use of high GPU utilization.
- High GPU utilization accelerates the training speed.

• Vision Transformer (ViT)

Importance of the size of pre-trained data

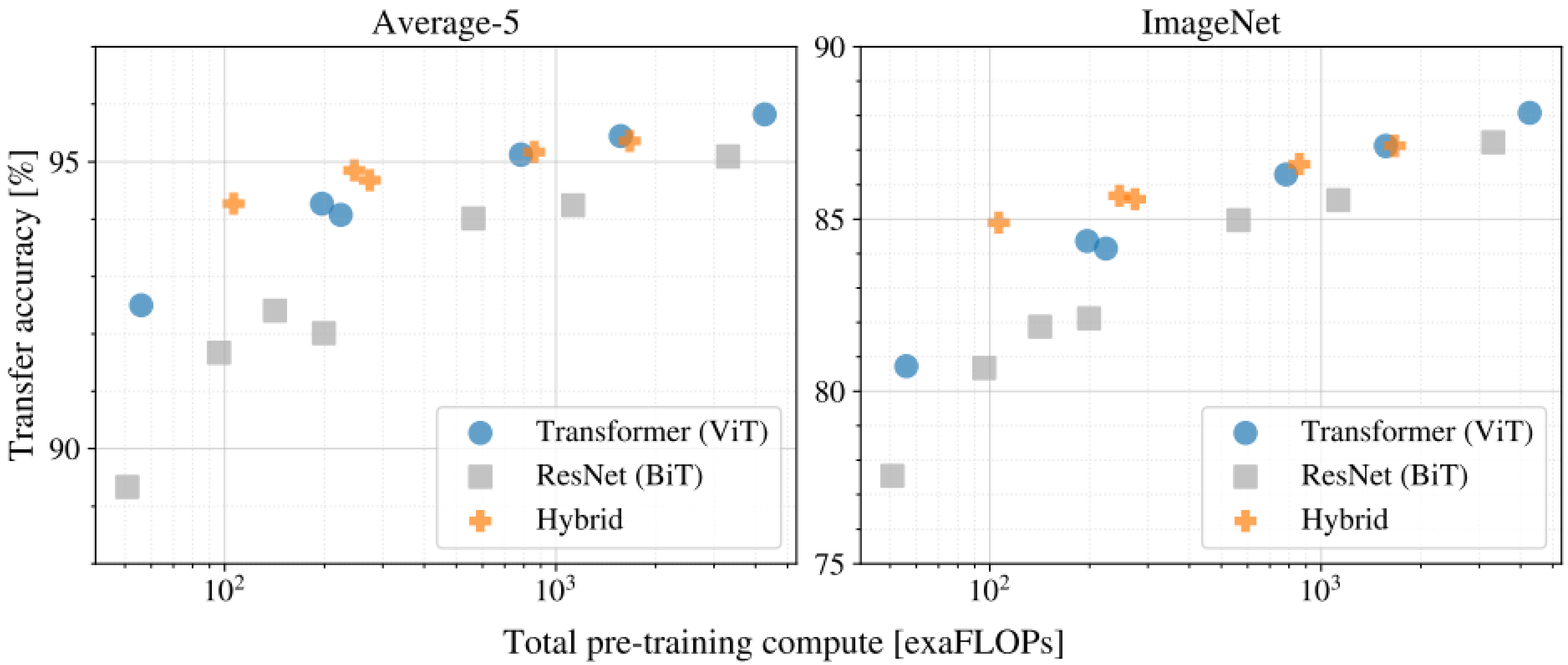




- ➢ Large ViT models perform worse than BiT ResNets when pre-trained on smaller datasets.
- ➢ Large ViT models perform better than ResNets when pre-trained by larger datasets.

• Vision Transformer (ViT)

Hybrid ViT



- ➢ Patches can be substituted for features of the image, which are formed from the feature maps of the CNN
- ➢ The input sequence is obtained by simply flattening the spatial dimension of the feature map
- ➢ Highly performs than casual ViT, when training cost is limited

• Index

- CNN and ViT
  - CNN architecture
  - Inductive bias
  - Vision Transformer

- DeiT
  - DeiT and CNN
  - What is DeiT?
  - DeiT and other Transformers

- Swin Transformers
  - Problem statements
  - Proposed methods
  - Experiments
  - Adaptation

DeiT


Training data-efficient image transformers
& distillation through attention
Facebook's New AI
Technique DeiT

• DeiT and ConvNets

- ➢is a class of artificial neural network (**ANN**), most commonly applied to analyze visual imagery

- ➢Convolutional neural networks have been the main design paradigm for image understanding tasks
- ➢They can be found at the core of everything from Facebook's photo tagging to self-driving cars.

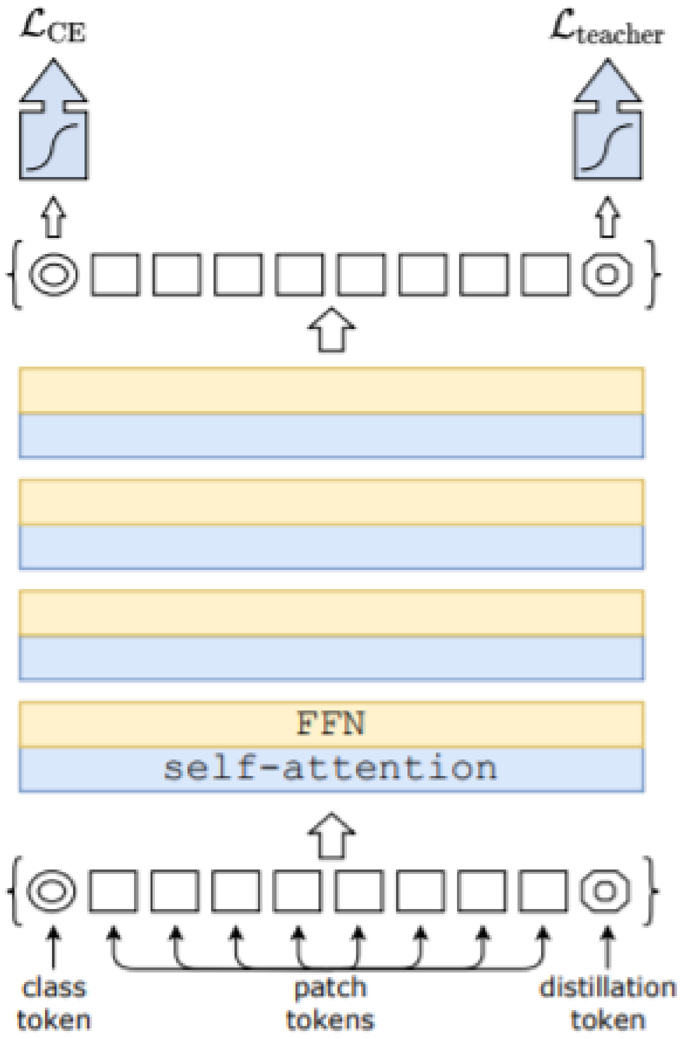
• What is DeiT?

- DeiT: Data efficient image transformers
- DeiTs are a special kind of convolution free Transformers which involves a teacher-student strategy that relies on a distillation token ensuring that the student learns from the teacher through attention.

- In DeiT new distillation procedure based on distillation tokens is implemented

• DeiT

DeiT Architecture



• DeiT

Difference between DeiT and other Transformers

- Normal vision transformer (ViT) needs a large volume of curated data in order to be effective.

- While using the DeiT approach the trasformers can be trained on a single compute r in less than 3 days (53 hours of pre-training, and optionally 20 hours of fine-tuni ng) with a top-1 accuracy of 83.1% on ImageNet with no external data (according to the result of the research paper expirement)

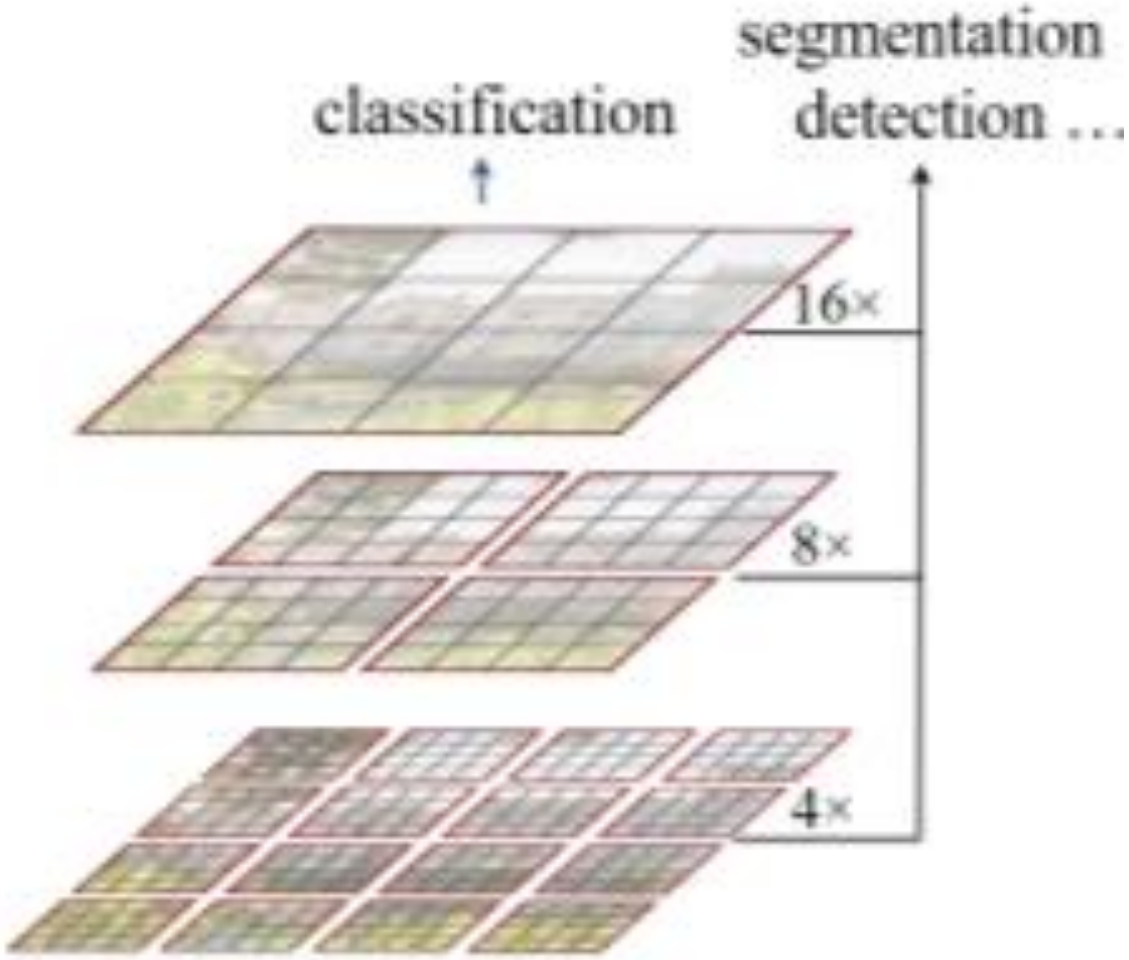
• Index

- CNN and ViT
  - CNN architecture
  - Inductive bias
  - Vision Transformer

- DeiT
  - DeiT and CNN
  - What is DeiT?
  - DeiT and other Transformers

- Swin Transformers
  - Problem statements
  - Proposed methods
  - Experiments
  - Adaptation

- **Swin Transformers**



• Swin Transformers

What is Swin Transformers

➢Swin Transformers are basically Hierarchical Vision Transformer using Shifted Wi ndows.

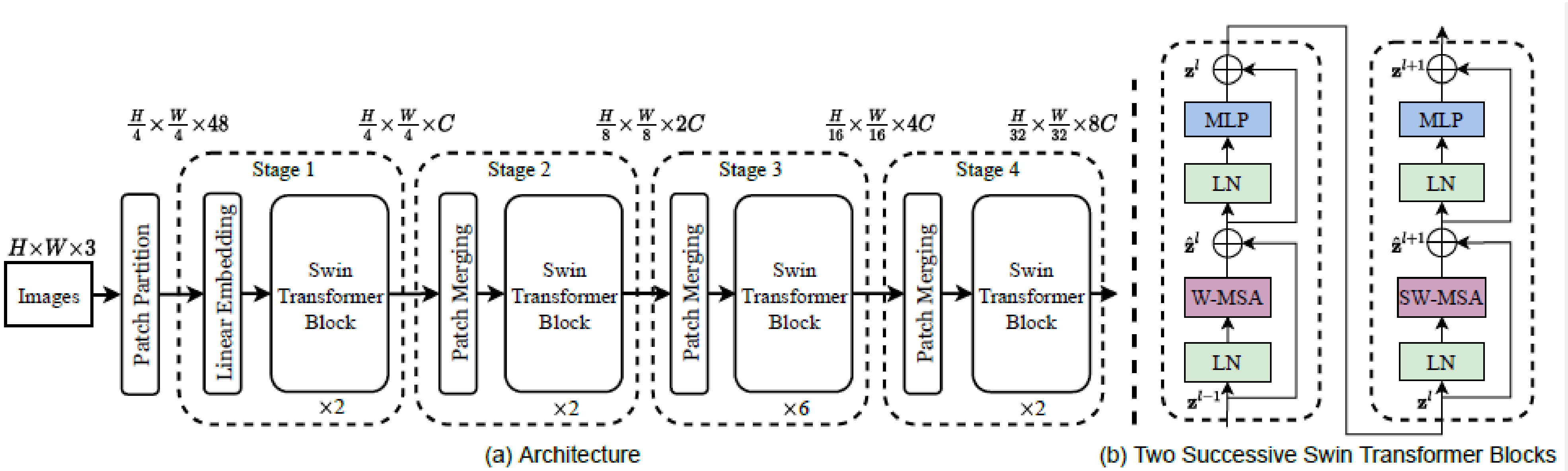


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

• Swin Transformers

Problem Statement

- Computer vision has long been dominated by convolutional neural networks (CNNs)

- Due to tremendous success in the NLP (Neuro-linguistic programming), it was adapted to computer vision such as image classification and joint vision-language modeling.

- Scaling Problem: Visual elements can vary substantially in scale, unlike the word tokens.

- The computational complexity of its self-attention is quadratic to image size.

• Swin Transformers

Overall Architecture

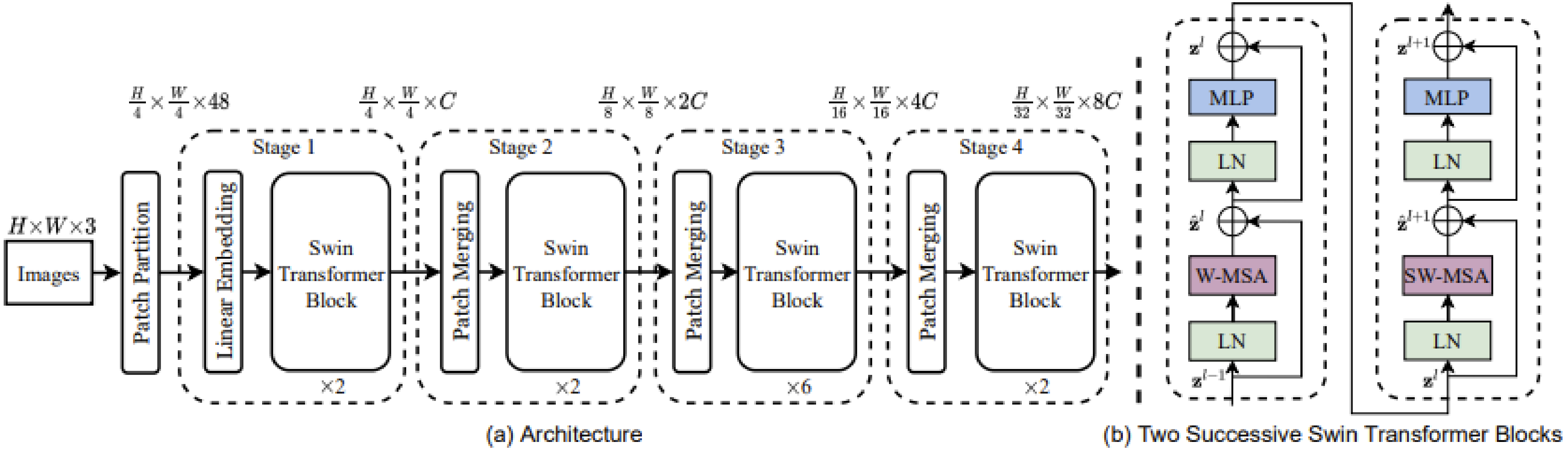


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

- ➢ Figure 3 illustrates the tiny version of Swin Transformer.
- ➢ It consists of Patch Partition, Patch Merging and Swin Transformer Block.

• Swin Transformers

Overall Architecture : Patch Partition

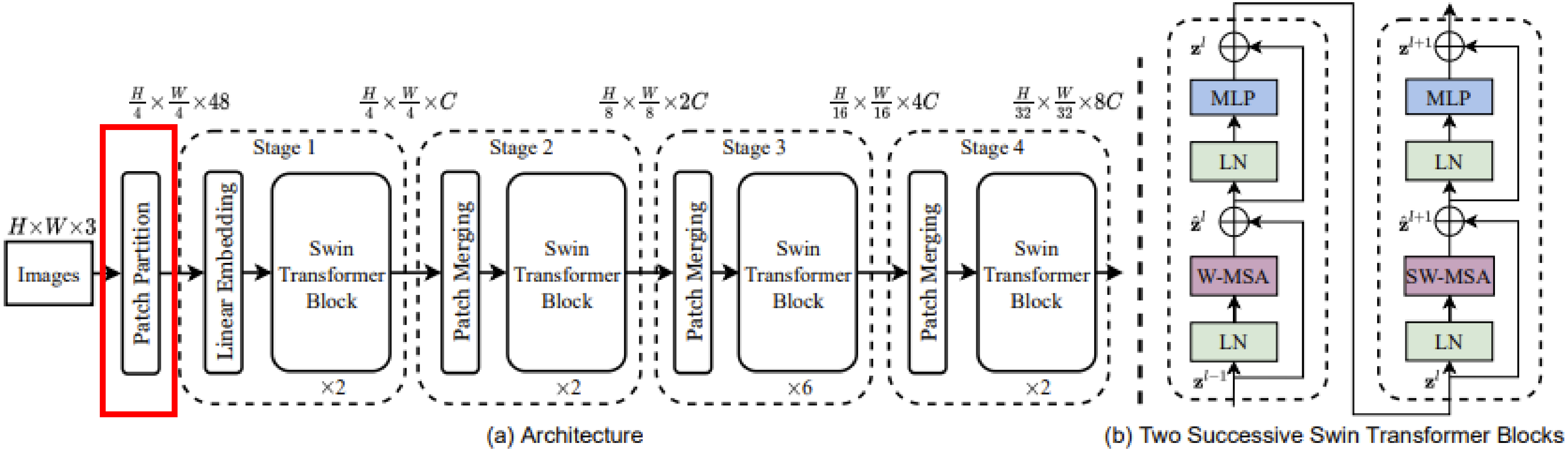


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

- ➢ An input RGB image (HxWx3) is split **into non-overlapping patches**.
- ➢ Each patch is treated as a **token**.

• Swin Transformers

Overall Architecture : Stage 1

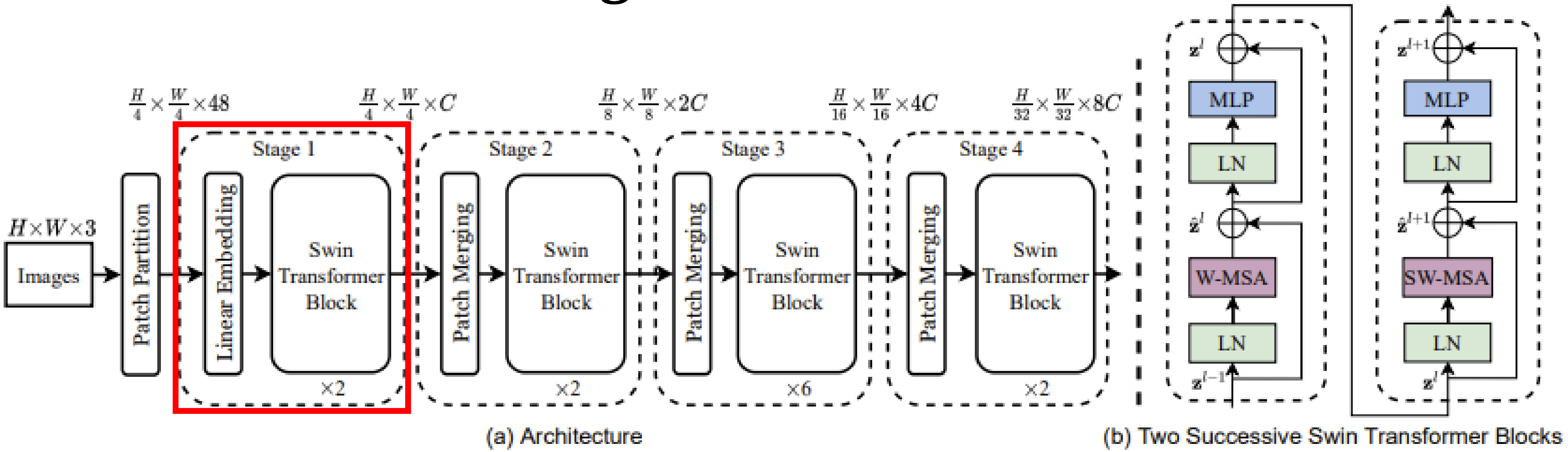


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

- **Linear embedding layer** is applied on an input tensor.
- Instead of general MSA (Multi-head Self Attention), **W-MSA and SW-M**SA are used.

• Swin Transformers

Overall Architecture : Stage 2~4

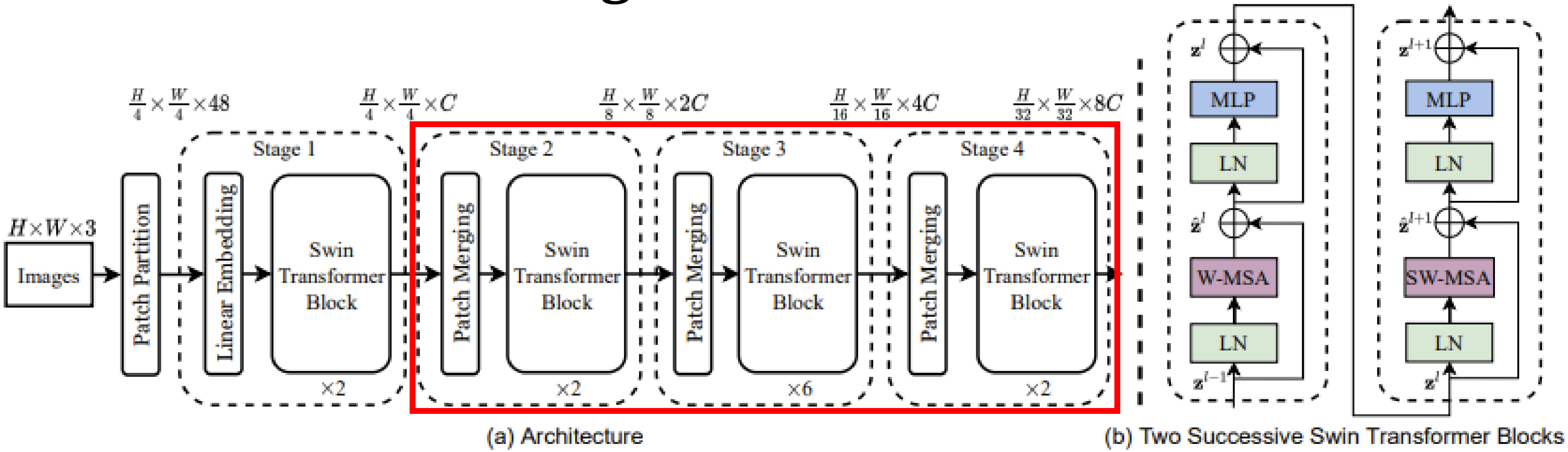


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

- **The number of tokens is reduced** by patch merging layers.
- The first patch merging layer **concatenates** the features of each group of **2x2 neighboring patches**.
- **Swin Transformer blocks** are applied after the feature transformation.

- These stages produce **a hierarchical representation**, which helps comparative abilities in computer vision.

• Swin Transformers

Overall Architecture : Swin Transformer Block

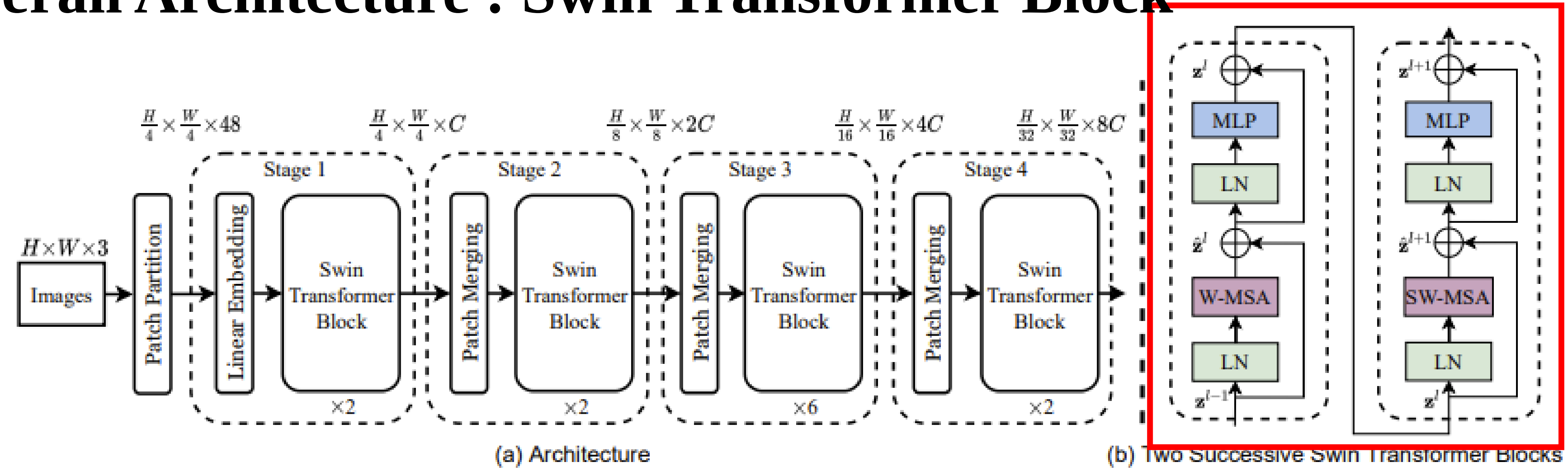


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

- Swin Transformer is built by **replacing the standard MSA module** in a Transformer block **by a module based on shifted windows**.
- A Swin Transformer block consists of a **shifted window based MSA module**, followed by a 2-layer MLP with GELU non-linearity in between.

• Swin Transformers

Shifted Window based Self-Attention

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \qquad (1)$$

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

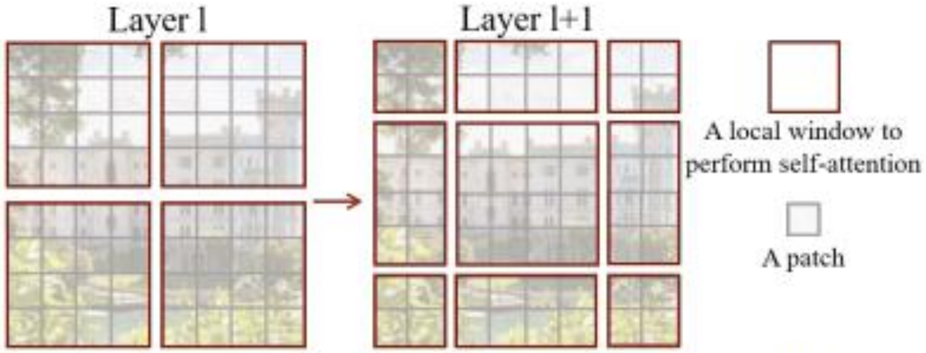


Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

- **Global self-attention** computes the relationships between a token and all other tokens, which leads to **quadratic complexity** w.r.t. the number of tokens.
- **Self-attention within local-windows** is proposed for efficient modeling.
- (1) is quadratic to patch number hw while (2) is **linear** when M is fixed.

- **Shifted window partitioning** approach is to introduce **cross-window connections**.

• Swin Transformers

Efficient batch computation for shifted configuration

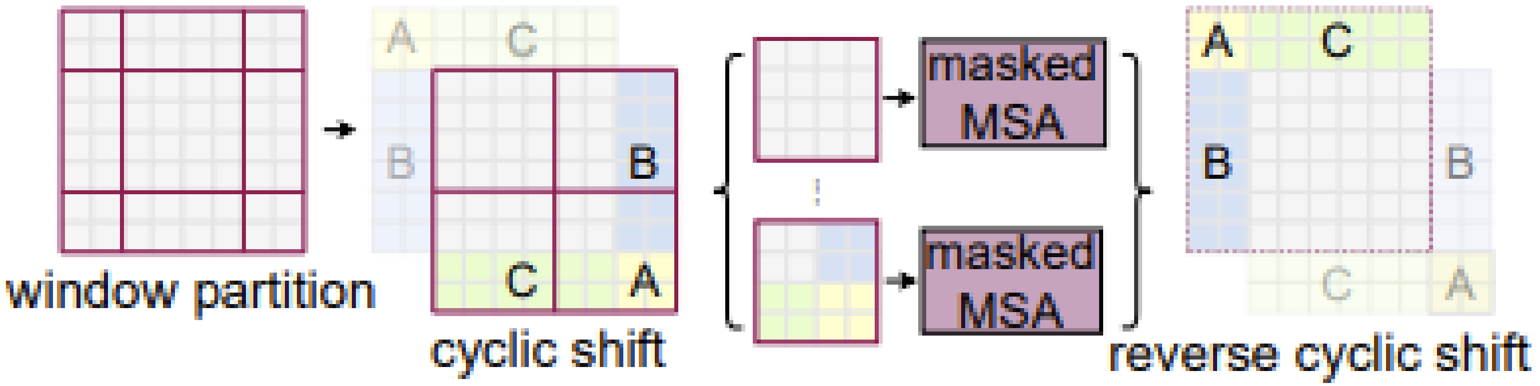


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

- ➢ In the shifted configuration, some windows will be **smaller than M x M**
- ➢ For more efficient computation, **cyclic-shifting** toward the top-left direction is proposed.
- ➢ **A masking mechanism** is employed to limit self-attention computation to within each sub-window.

• Swin Transformers

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

➢ C is the channel number of the hidden layers in the first stage.
➢ Swin-B is the base model, which has the similar model size and computation complexity to ViT-B and DeiT-B.
➢ Swin-T, Swin-S, Swin-L are versions of about 0.25x, 0.5x and 2x the model size and computational complexity, respectively.

• Swin Transformers

Image Classification on ImageNet-1K / 22K

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	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 [34]	384^2	388M	204.6G	-	84.4
R-152x4 [34]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [62] and a V100 GPU, following [57].

• **Swin Transformers**

Object Detection on COCO

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

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

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

(c) System-level Comparison

Method	mini-val		test-dev		#param.	FLOPs
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}		
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
DetectoRS* [42]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [23]	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.

• Swin Transformers

Semantic Segmentation on ADE20K

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
DNL [65]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [67]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [63]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [67]	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 [73]	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.

• Swin Transformers

Contributions

- Swin Transformer presents a new vision Transformer.
- It produces a hierarchical feature representation.
- Its shifted window based self-attention has linear computational complexity with respect to input image size.

• Swin Transformers

SwinIR : Image Restoration Using Swin Transformer

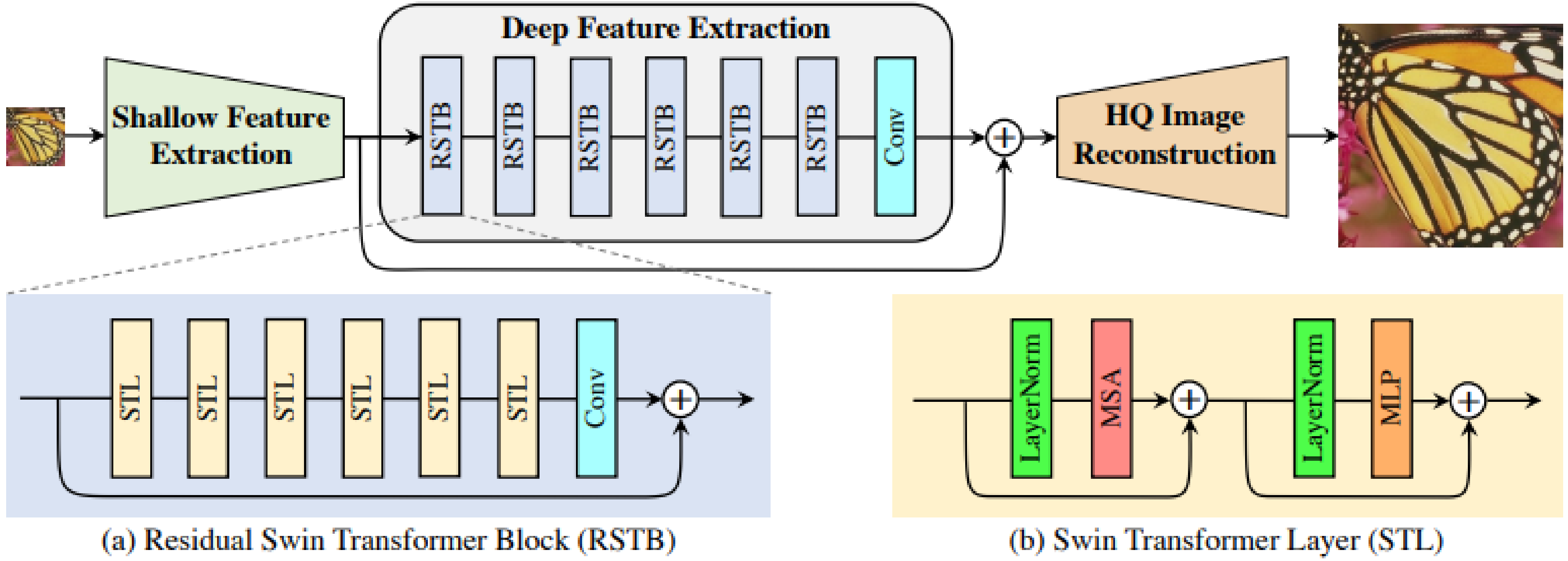


Figure 2: The architecture of the proposed SwinIR for image restoration.

- ➢ An **image restoration model based on Swin Transformer**
- ➢ SwinIR consists of three modules : shallow feature extraction, deep feature extraction and high-quality image reconstruction modules.

• Swin Transformers

SwinIR : Image Restoration Using Swin Transformer

Figure 4: Visual comparison of **bicubic image SR** (×4) methods. Compared images are derived from [9]. Best viewed by zooming.

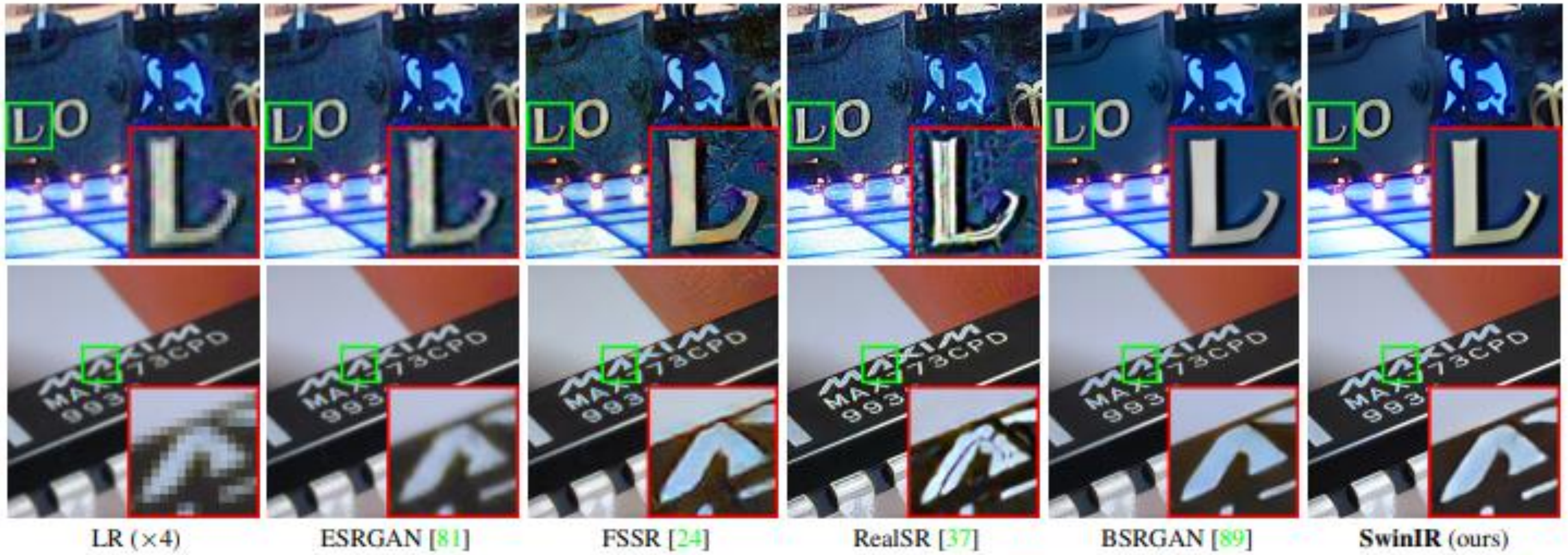


Figure 5: Visual comparison of **real-world image SR** (×4) methods on RealSRSet [89]. Compared images are derived from [89].

• Swin Transformers

SwinIR : Image Restoration Using Swin Transformer

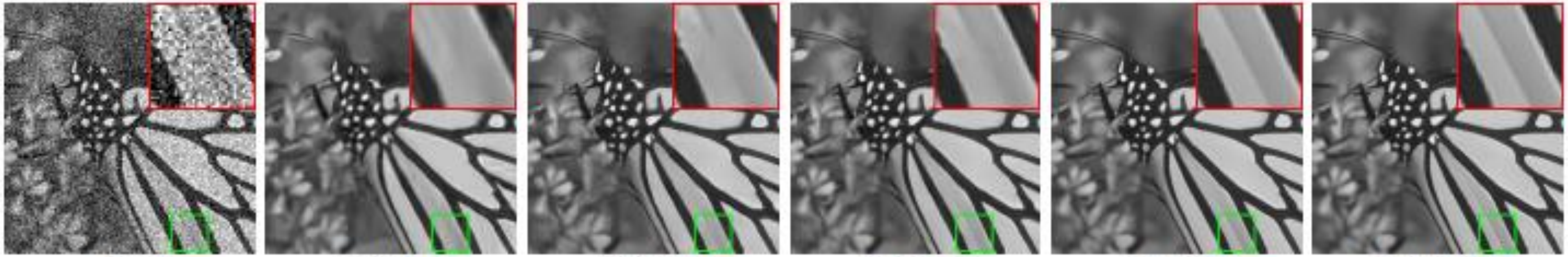


Figure 6: Visual comparison of **grayscale image denoising** (noise level 50) methods on image "*Monarch*" from Set12 [90]. Compared images are derived from [88].

Figure 7: Visual comparison of **color image denoising** (noise level 50) methods on image "*163085*" from CBSD68 [59]. Compared images are derived from [88].