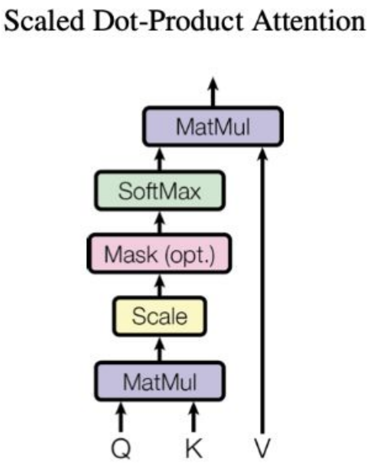
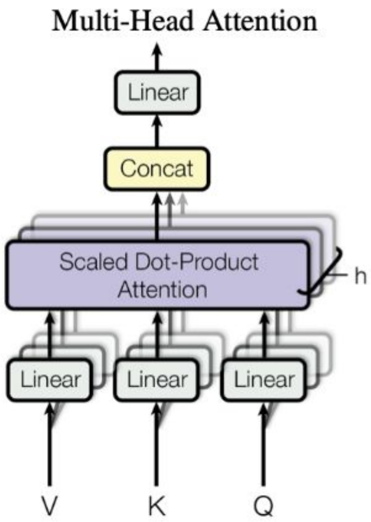
Transformer for Vision

Deep Learning

Aziz Temirkhanov
Lambda, HSE

Transformer

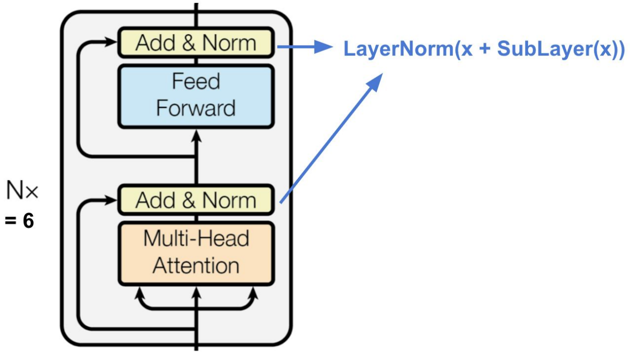


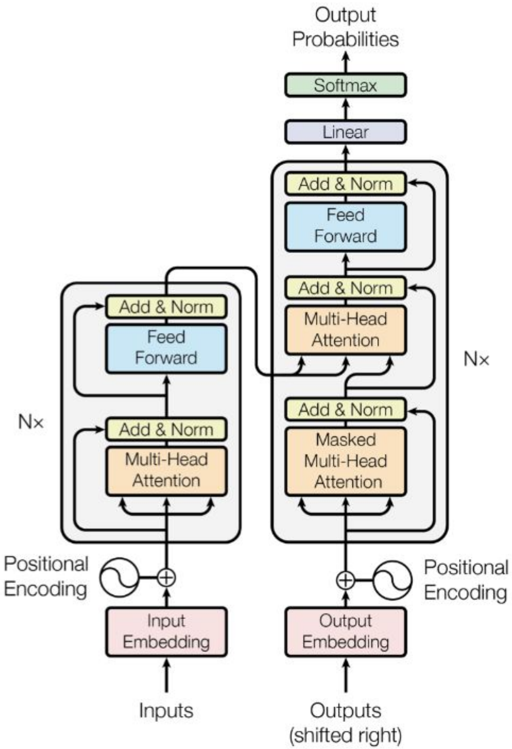


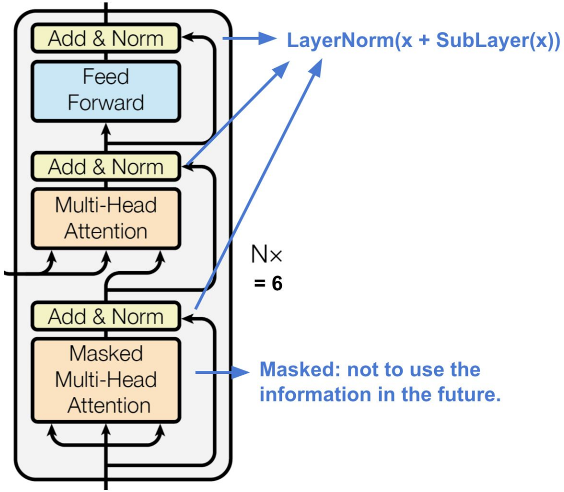
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_\mathrm{h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

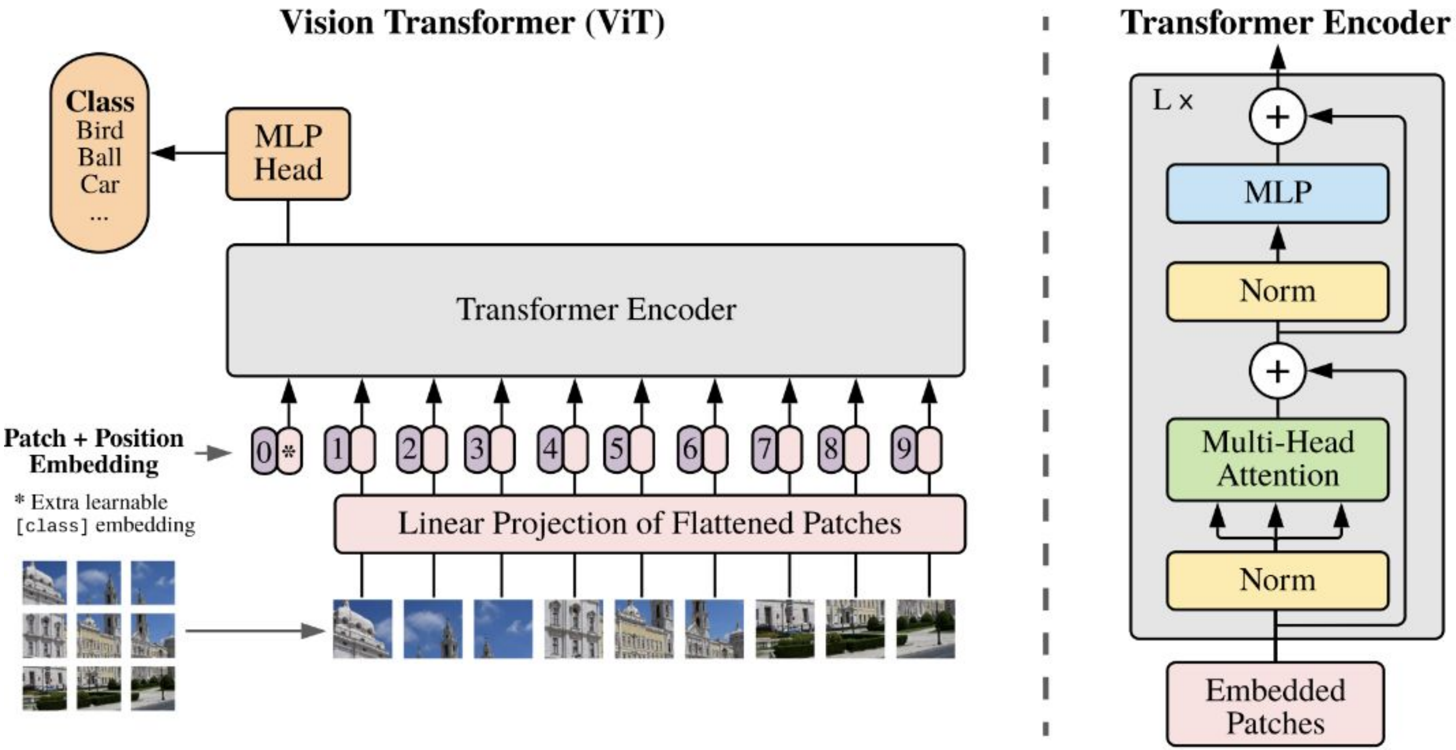
Transformer







ViT

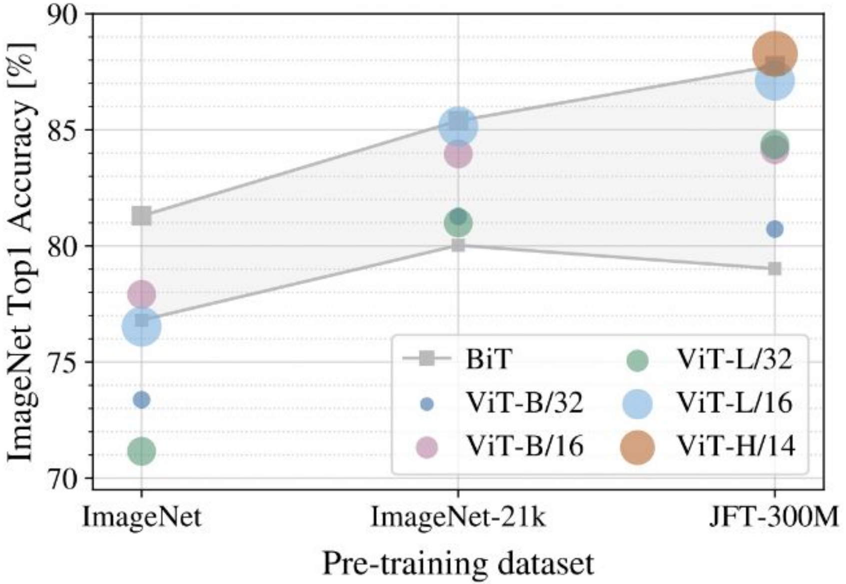

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L ×
+
MLP
Norm
+
Multi-Head Attention
Norm
Embedded Patches

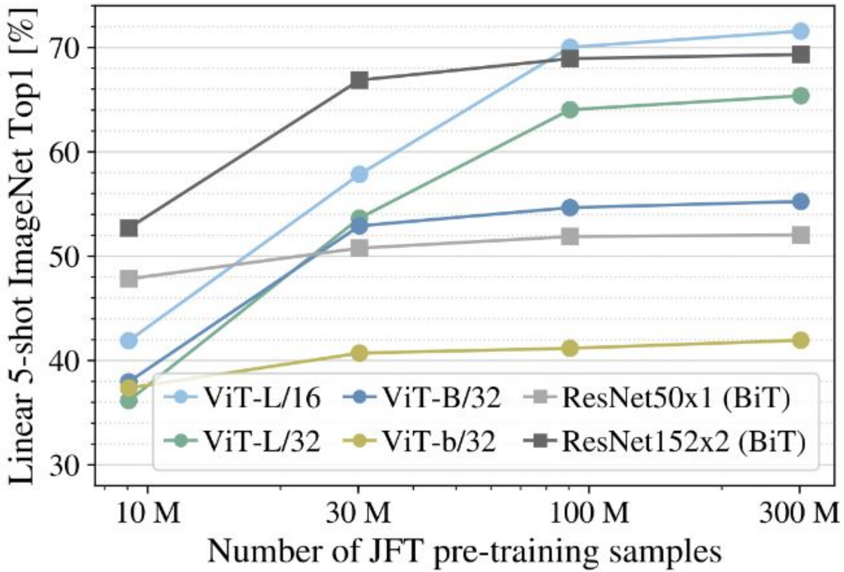
ViT training

- Large amount of data
- Data Augmentation and Model Regularization
  - Random Augment
  - Label Smoothing
  - Stochastic Depth
  - CutMix and MixUp
  - Erasing
- AdamW (or Adam) instead of SGD
- Large weight decay value like 0.1 (recall that for CNN this value is usually around 10e4-10e3)
- Warmup lr scheduler

ViT

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M



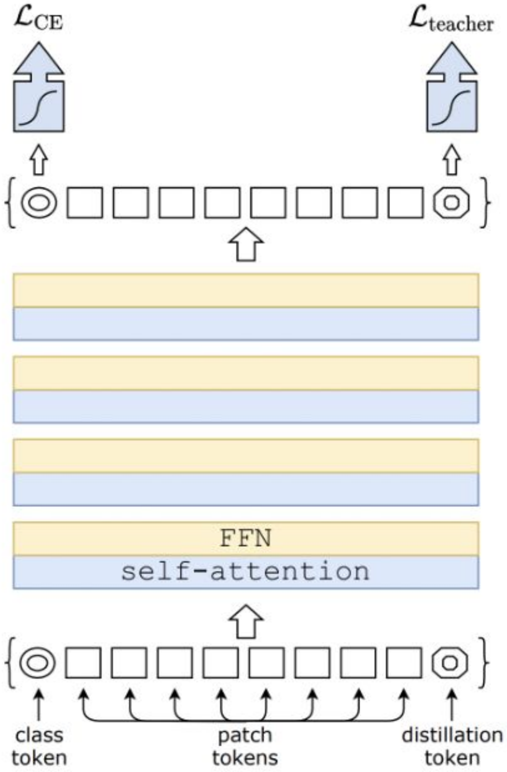


DeiT

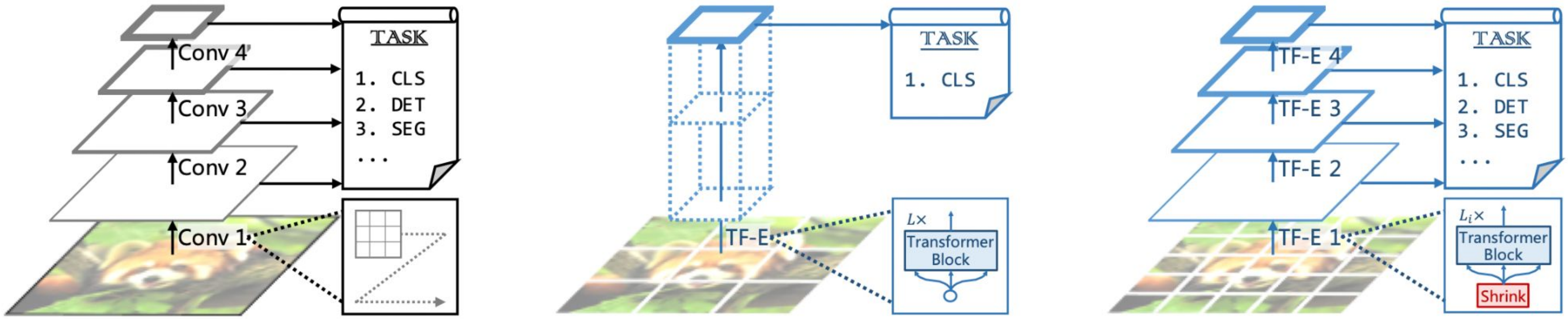
$$\mathcal{L} = (1-\lambda)\mathcal{L}_{CE}(y_{pred}, y_{true}) + \lambda\tau^2 D_{KL}(y_{pred}/\tau, y_{teacher}/\tau)$$

$$\mathcal{L} = \frac{1}{2}\mathcal{L}_{CE}(y_{pred}, y_{true}) + \frac{1}{2}\mathcal{L}_{CE}(y_{pred}, y_{teacher})$$

method ↓	Supervision label	Supervision teacher	ImageNet top-1 (%) Ti 224	S 224	B 224	B↑384
DeiT– no distillation	✓	✗	72.2	79.8	81.8	83.1
DeiT– usual distillation	✗	soft	72.2	79.8	81.8	83.2
DeiT– hard distillation	✗	hard	74.3	80.9	83.0	84.0
DeiT⚗: class embedding	✓	hard	73.9	80.9	83.0	84.2
DeiT⚗: distil. embedding	✓	hard	74.6	81.1	83.1	84.4
DeiT⚗: class+distillation	✓	hard	74.5	81.2	83.4	84.5



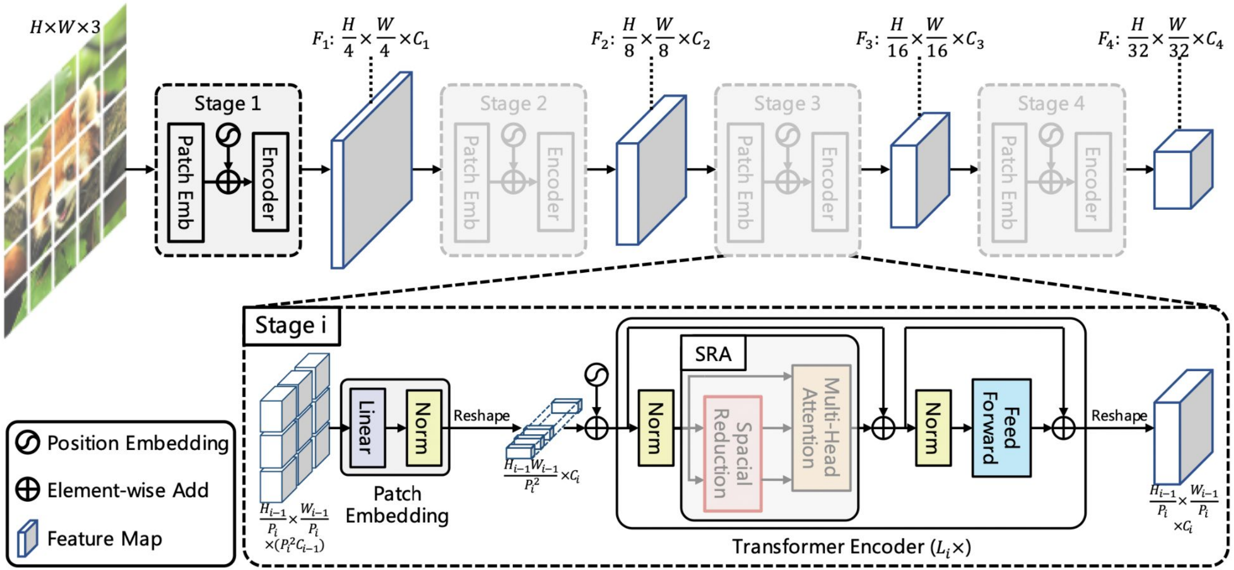
PVT



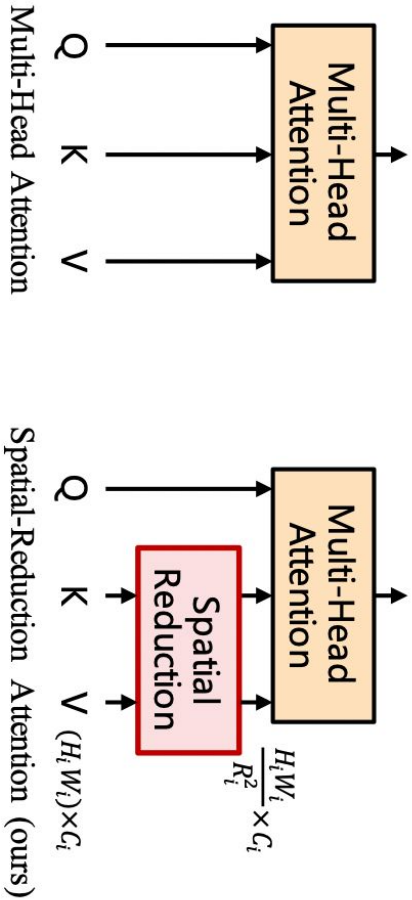
(a) CNNs: VGG [54], ResNet [22], *etc.* (b) Vision Transformer [13] (c) Pyramid Vision Transformer (ours)

- FPN introduced a novel approach into object detection
- Deeper layers is responsible for larger features
- Whereas first layers focusing on little details and smaller objects

PVT

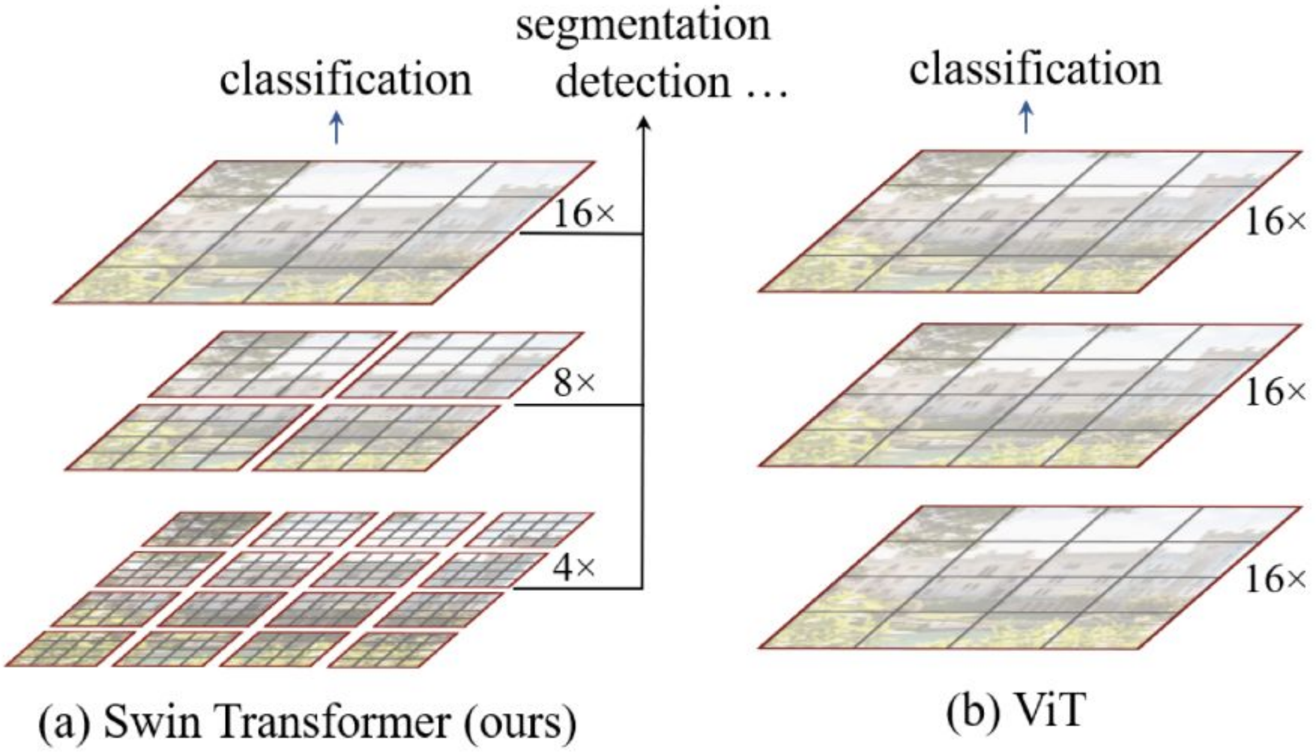


$H\times W\times 3$
$F_1: \frac{H}{4}\times\frac{W}{4}\times C_1$
$F_2: \frac{H}{8}\times\frac{W}{8}\times C_2$
$F_3: \frac{H}{16}\times\frac{W}{16}\times C_3$
$F_4: \frac{H}{32}\times\frac{W}{32}\times C_4$
Stage 1
Stage 2
Stage 3
Stage 4
Patch Emb
Encoder
Stage i
Linear
Norm
Patch Embedding
Reshape
SRA
Norm
Spacial Reduction
Multi-Head Attention
Norm
Feed Forward
Reshape
Transformer Encoder ($L_i\times$)
$\frac{H_{i-1}}{P_i}\times\frac{W_{i-1}}{P_i}\times(P_i^2C_{i-1})$
$\frac{H_{i-1}W_{i-1}}{P_i^2}\times C_i$
$\frac{H_{i-1}}{P_i}\times\frac{W_{i-1}}{P_i}\times C_i$
Position Embedding
Element-wise Add
Feature Map



Multi-Head Attention
Q
K
V
Multi-Head Attention
Spatial-Reduction Attention (ours)
Spatial Reduction
$(H_iW_i)\times C_i$
$\frac{H_iW_i}{R_i^2}\times C_i$

SWIN


segmentation
detection ...
classification
16×
8×
4×
classification
16×
16×
16×
(a) Swin Transformer (ours)
(b) ViT

SWIN

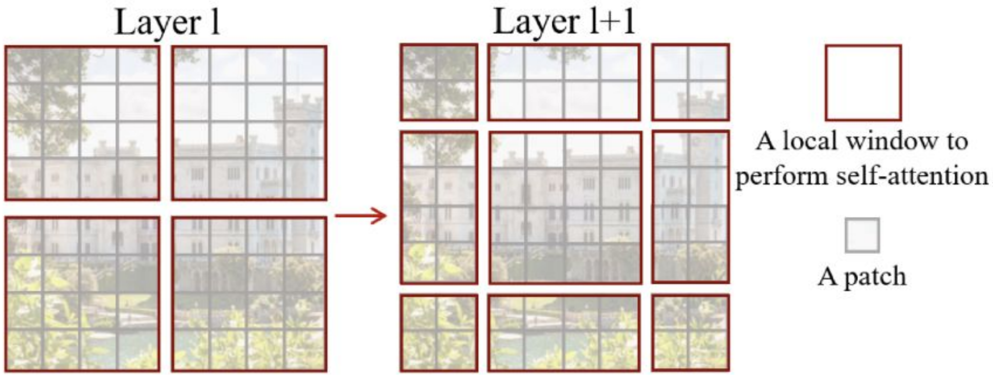


Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l + 1$ (right), the window partitioning 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.

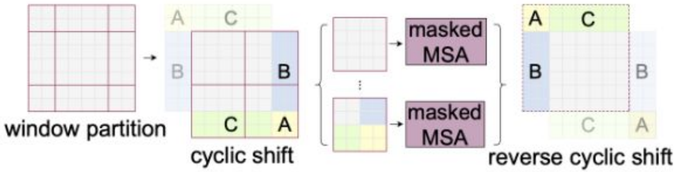


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

SWIN

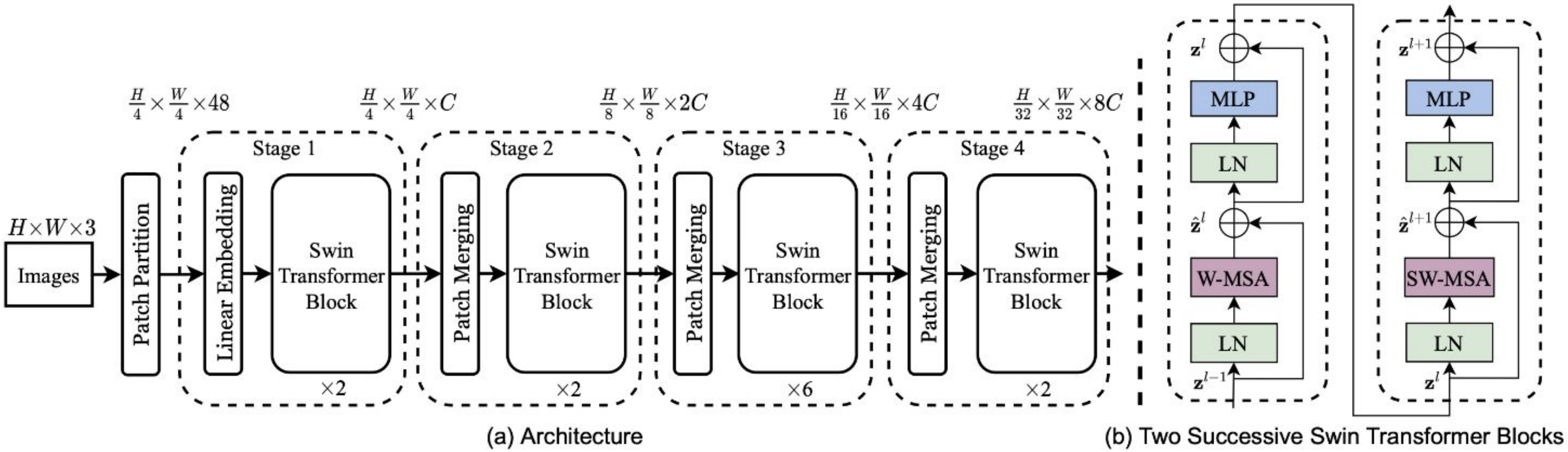
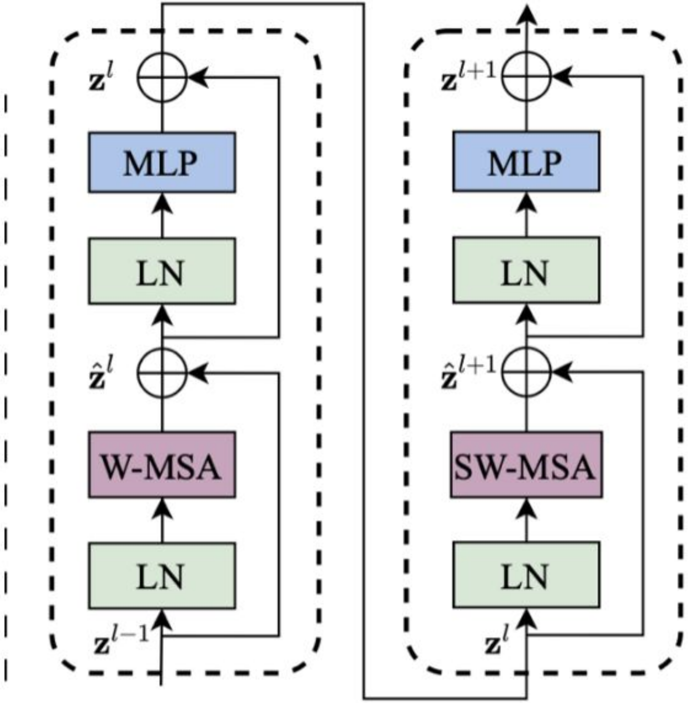


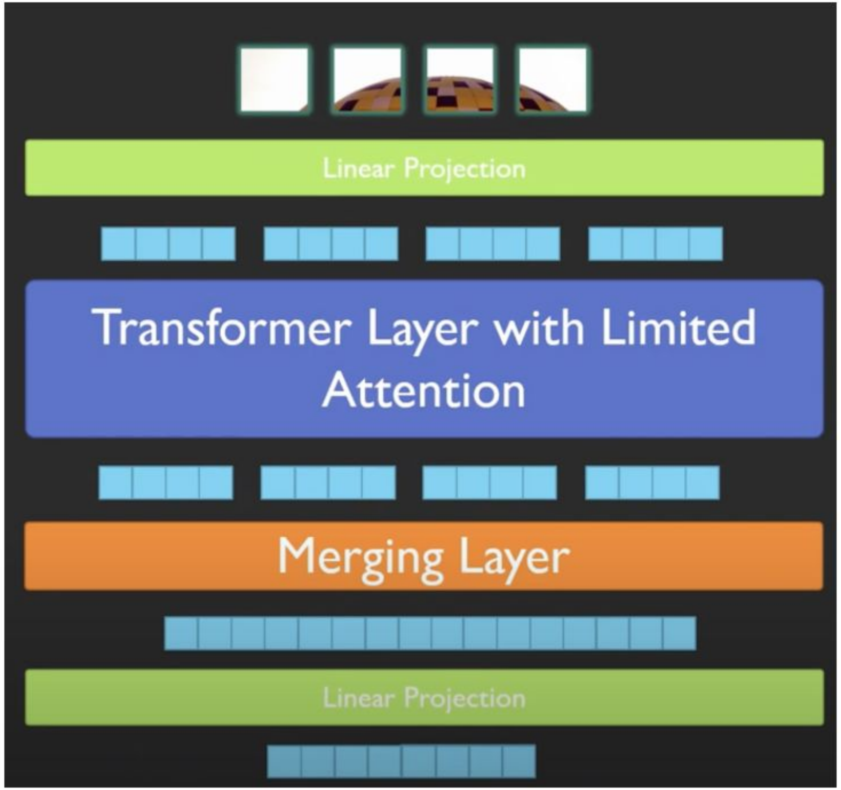
Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

SWIN



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

SWIN


Linear Projection
Transformer Layer with Limited Attention
Merging Layer
Linear Projection

SWIN

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

SWIN

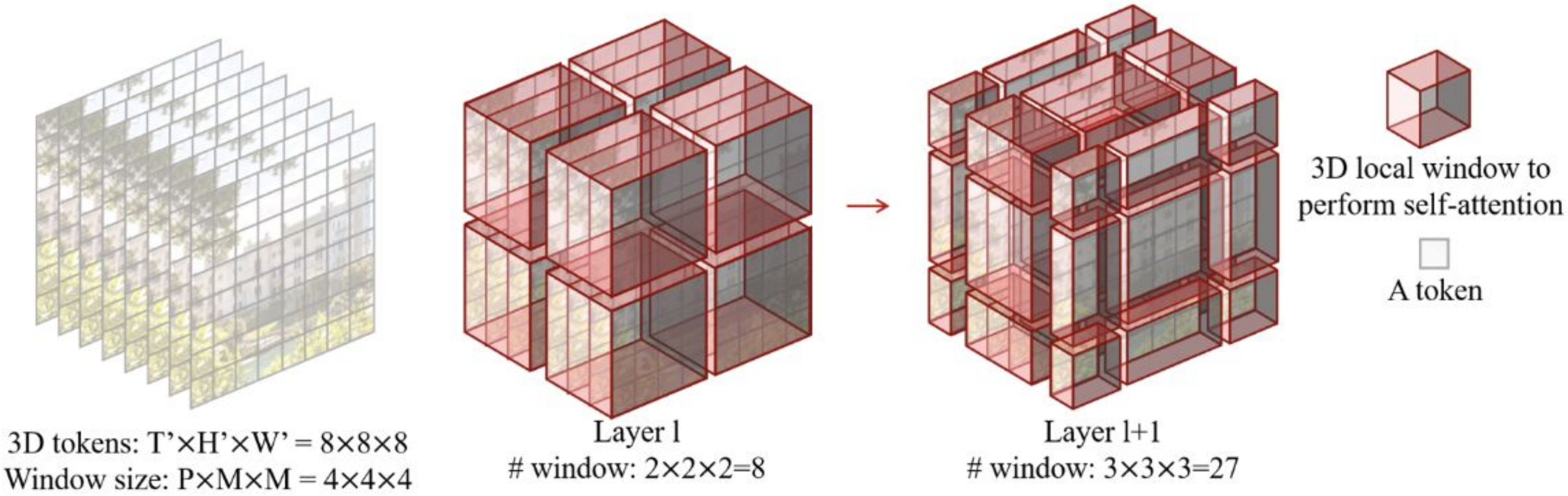
COCO object detection

(a) Various frameworks							
Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	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-50	44.5	63.4	48.2	106M	166G	21.0
R-CNN	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

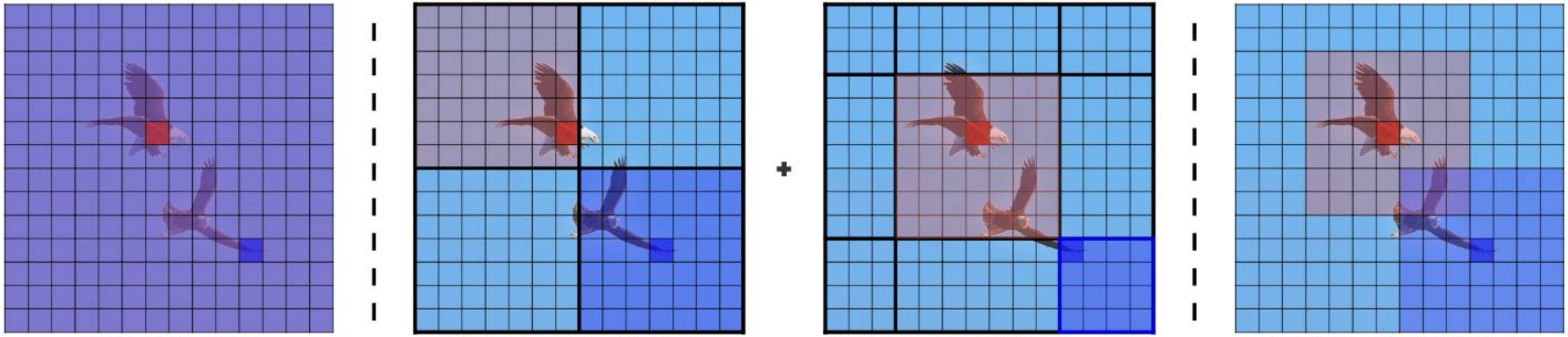
ADE20K semantic segmentation

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

SWIN for Video

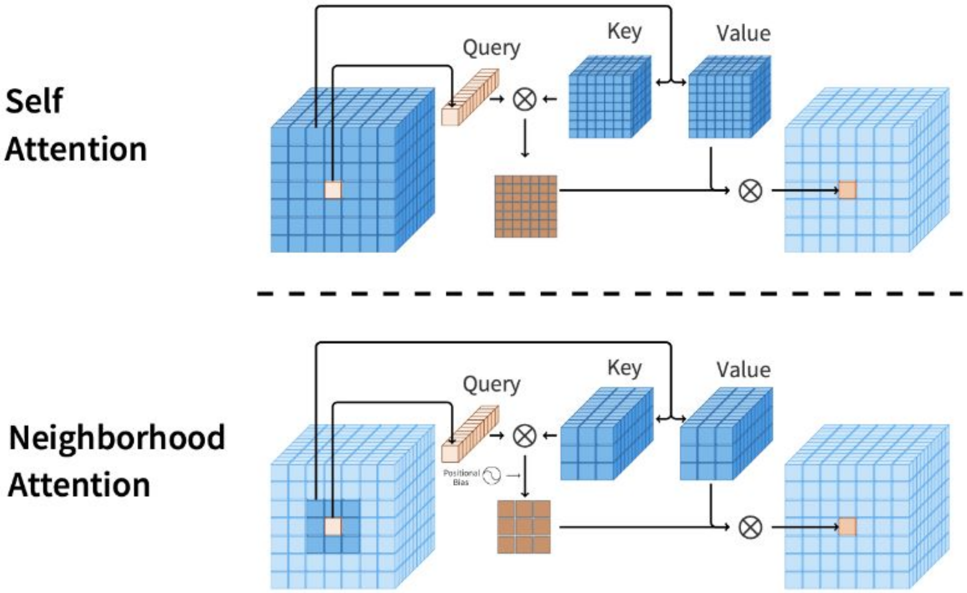


NAT



Self Attention (ViT) | Window Self Attention (Swin) | Shifted Window Self Attention (Swin) | Neighborhood Attention (NAT)

NAT



- k nearest neighboring key projections

$$\mathbf{A}_i^k = \begin{bmatrix} Q_i K_{\rho_1(i)}^T + B_{(i,\rho_1(i))} \\ Q_i K_{\rho_2(i)}^T + B_{(i,\rho_2(i))} \\ \vdots \\ Q_i K_{\rho_k(i)}^T + B_{(i,\rho_k(i))} \end{bmatrix}, \qquad (2)$$

$$\mathbf{V}_i^k = \begin{bmatrix} V_{\rho_1(i)}^T & V_{\rho_2(i)}^T & \cdots & V_{\rho_k(i)}^T \end{bmatrix}^T. \qquad (3)$$

$$\mathrm{NA}_k(i) = softmax\left(\frac{\mathbf{A}_i^k}{\sqrt{d}}\right)\mathbf{V}_i^k, \qquad (4)$$

NAT

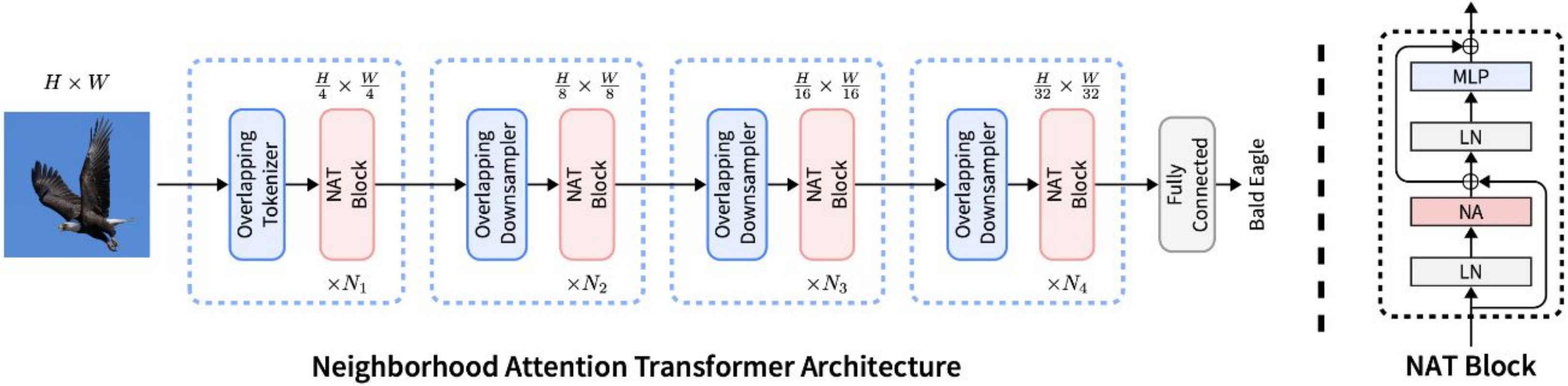
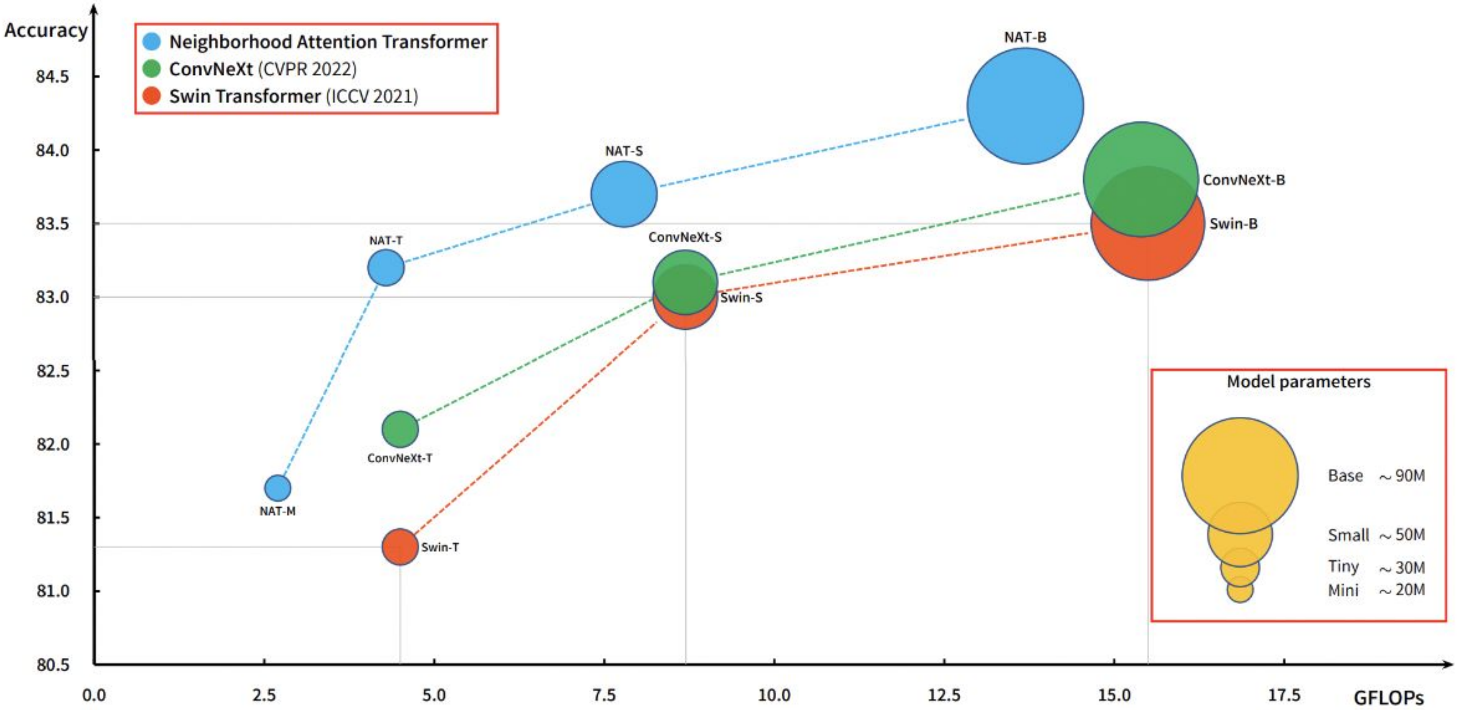


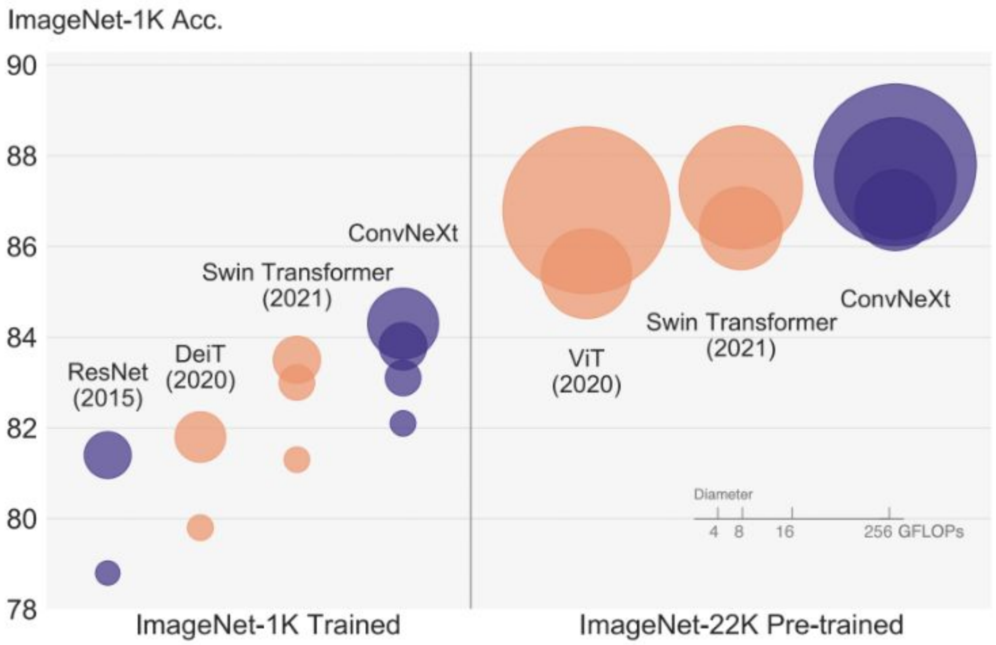
Figure 5. An overview of our model, NAT, with its hierarchical design. The model starts off with a convolutional downsampler, then moves on to 4 sequential levels, each consisting of multiple NAT Blocks, which are transformer-like encoder layers. Each layer is comprised of a multi-headed neighborhood attention (NA), a multi-layered perceptron (MLP), Layer Norm (LN) before each module, and skip connections. Between the levels, feature maps are downsampled to half their spatial size, while their depth is doubled. This allows for easier transfer to downstream tasks through feature pyramids.

NAT

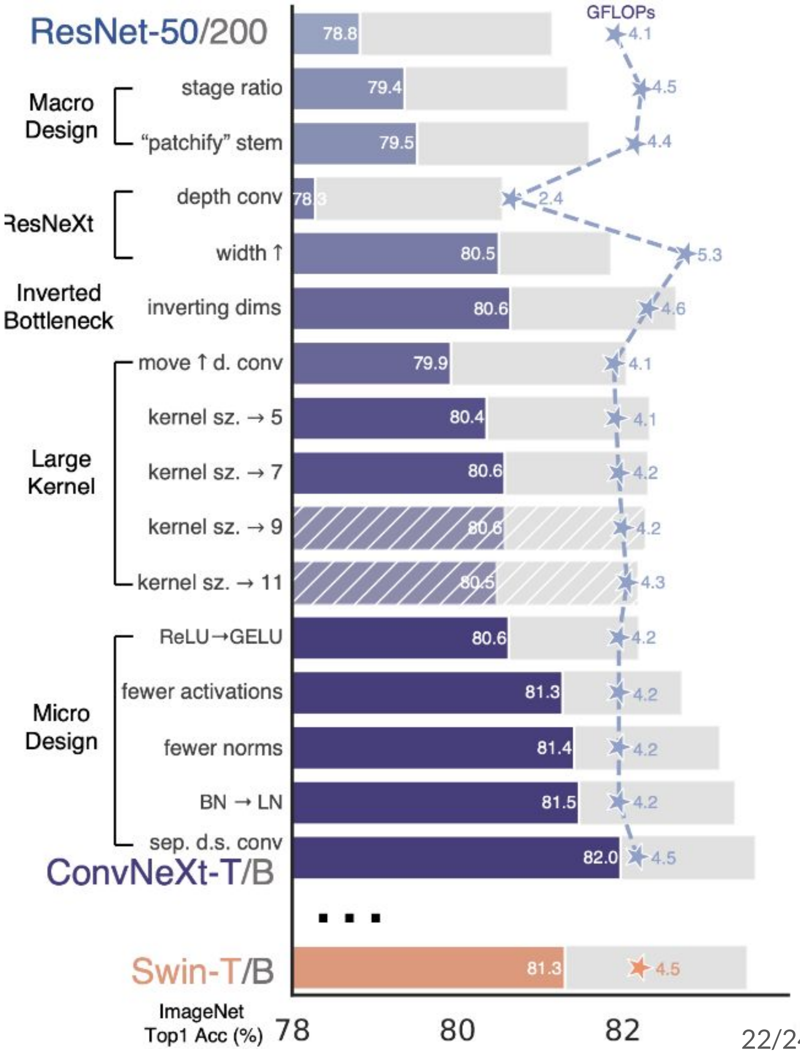


Accuracy
Neighborhood Attention Transformer
ConvNeXt (CVPR 2022)
Swin Transformer (ICCV 2021)
84.5
84.0
83.5
83.0
82.5
82.0
81.5
81.0
80.5
NAT-B
NAT-S
NAT-T
NAT-M
ConvNeXt-B
Swin-B
ConvNeXt-S
Swin-S
ConvNeXt-T
Swin-T
Model parameters
Base ~90M
Small ~50M
Tiny ~30M
Mini ~20M
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
GFLOPs

ConvNext



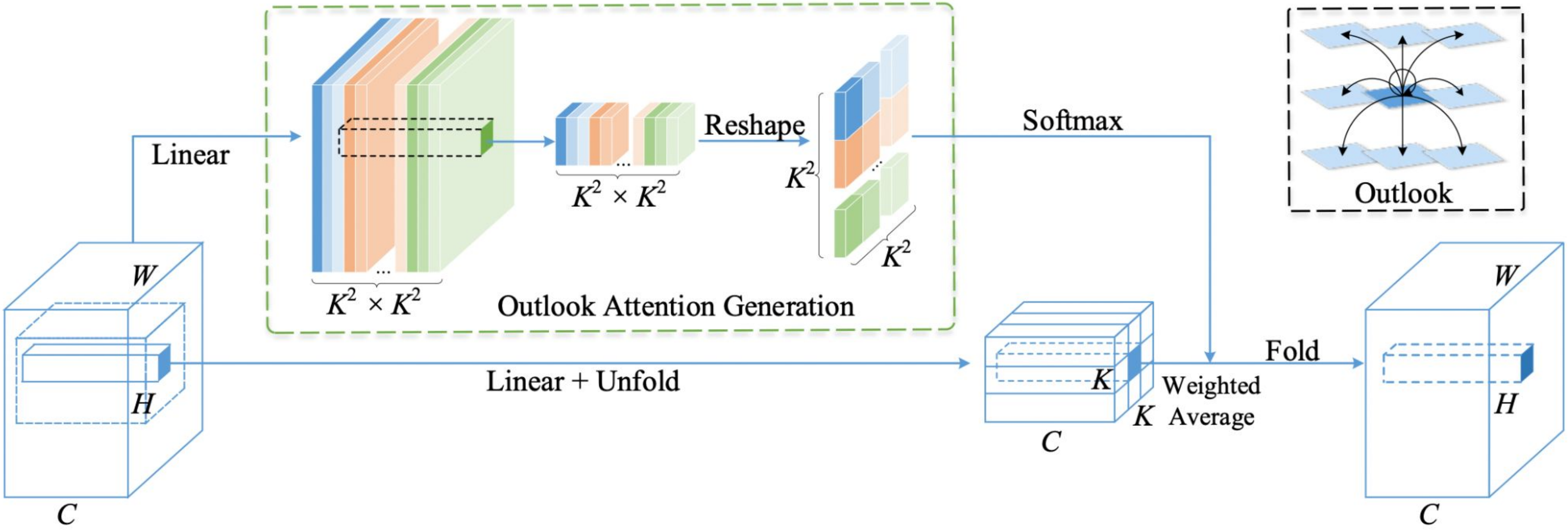
ImageNet-1K Acc.
90
88
86
84
82
80
78
ResNet (2015)
DeiT (2020)
Swin Transformer (2021)
ConvNeXt
ViT (2020)
Swin Transformer (2021)
ConvNeXt
Diameter
4 8 16 256 GFLOPs
ImageNet-1K Trained
ImageNet-22K Pre-trained



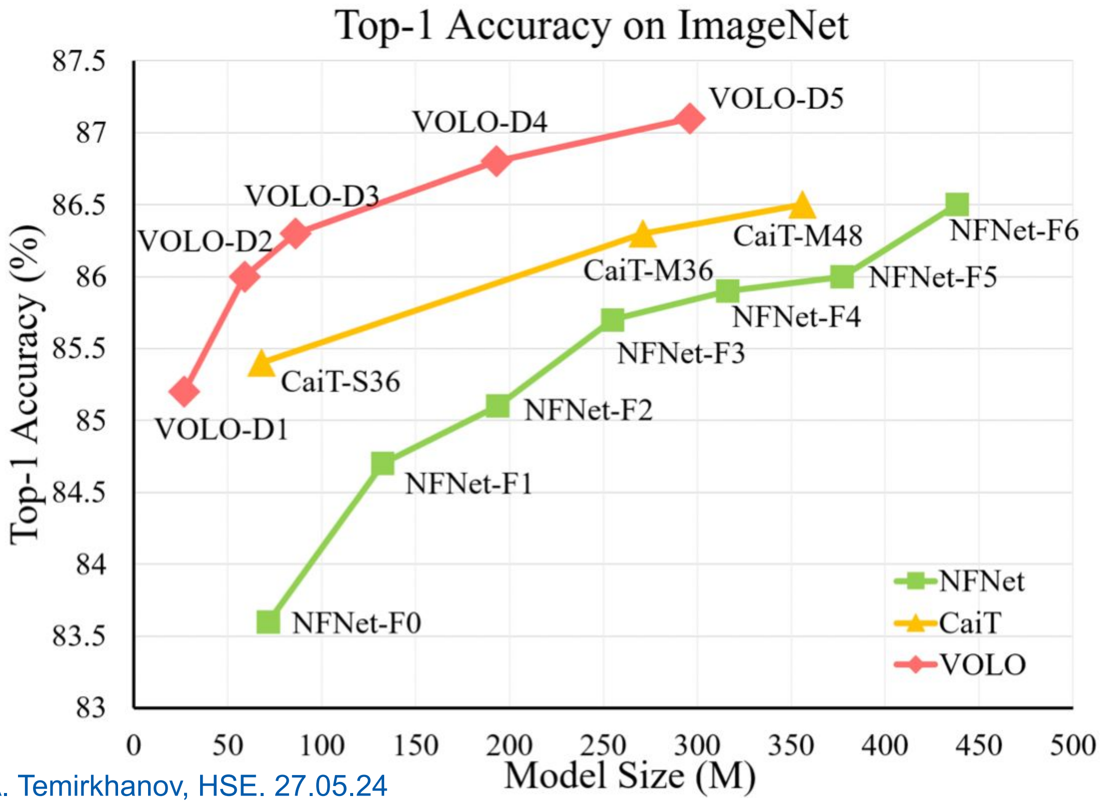
ResNet-50/200
GFLOPs
78.8
4.1
Macro Design
stage ratio
79.4
4.5
"patchify" stem
79.5
4.4
ResNeXt
depth conv
78.3
2.4
width ↑
80.5
5.3
Inverted Bottleneck
inverting dims
80.6
4.6
Large Kernel
move ↑ d. conv
79.9
4.1
kernel sz. → 5
80.4
4.1
kernel sz. → 7
80.6
4.2
kernel sz. → 9
80.6
4.2
kernel sz. → 11
80.5
4.3
Micro Design
ReLU→GELU
80.6
4.2
fewer activations
81.3
4.2
fewer norms
81.4
4.2
BN → LN
81.5
4.2
sep. d.s. conv
82.0
4.5
ConvNeXt-T/B
...
Swin-T/B
81.3
4.5
ImageNet Top1 Acc (%)
78
80
82

VOLO

$$\hat{X} = \mathrm{OutlookAttn}(\mathrm{LN}(X)) + X$$
$$Z = \mathrm{MLP}(\mathrm{LN}(\hat{X})) + \hat{X}$$



VOLO



Method	Backbone	Pretrained	mIoU
DenseASPP [66]	DenseNet [28]	ImgNet-1k	80.6
DeepLabv3+ [6]	Xception-65 [8]	ImgNet-1k	79.1
DPC [5]	Xception-71 [8]	ImgNet-1k	80.8
DANet [17]	ResNet-101	ImgNet-1k	81.5
CCNet [31]	ResNet-101	ImgNet-1k	81.3
Strip Pooling [24]	ResNet-101	ImgNet-1k	81.9
SETR [75]	ViT-L [14]	ImgNet-22k	82.1
PatchDiverse [18]	Swin-L [37]	ImgNet-22k	83.6
SpineNet-S143+ [42]	SpineNet	ImgNet-1k	83.0
SegFormer-B5 [64]	SegFormer	ImgNet-1k	84.0
VOLO-D1	VOLO	ImgNet-1k	83.1
VOLO-D4	VOLO	ImgNet-1k	**84.3**

Method	Backbone	Pretrained	mIoU	Pixel
PSPNet [74]	ResNet-269	ImgNet-1k	44.9	81.7
UperNet [62]	ResNet-101	ImgNet-1k	44.9	-
Strip Pooling [24]	ResNet-101	ImgNet-1k	45.6	82.1
DeepLabV3+ [6]	ResNeSt200	ImgNet-1k	48.4	-
SETR [75]	ViT-Large	ImgNet-22k	50.3	83.5
SegFormer-B5 [64]	SegFormer	ImgNet-1k	51.8	-
Swin-B [37]	Swin-B	ImgNet-22k	51.6	-
Seg-L-Mask/16 [46]	ViT-Large	ImgNet-22k	53.2	-
Swin-L [37]	Swin-L	ImgNet-22k	53.5	-
VOLO-D1	VOLO	ImgNet-1k	50.5	83.3
VOLO-D3	VOLO	ImgNet-1k	52.9	84.6
VOLO-D5	VOLO	ImgNet-1k	**54.3**	**85.0**