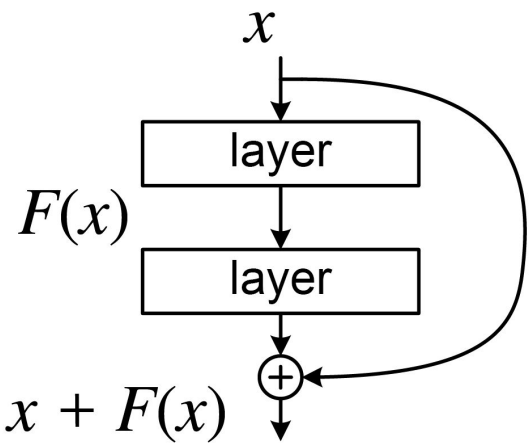
LSDL Lecture 03
Modern vision architectures

Ildus Sadrtdinov, 30.09.2024

ResNet



Key features:

- Residual connections
- Batch Normalization
- Global Average Pooling

Successors:

- ResNeXt
- WideResNet
- DenseNet
- RegNet
- …..

He et al., Deep Residual Learning for Image Recognition, 2015, https://arxiv.org/abs/1512.03385

Big Transfer

Datasets:

- ImageNet-1K (1.28M, public)
- ImageNet-21K (14.2M, public)
- JFT-300M (300M, proprietary)

Pre-training

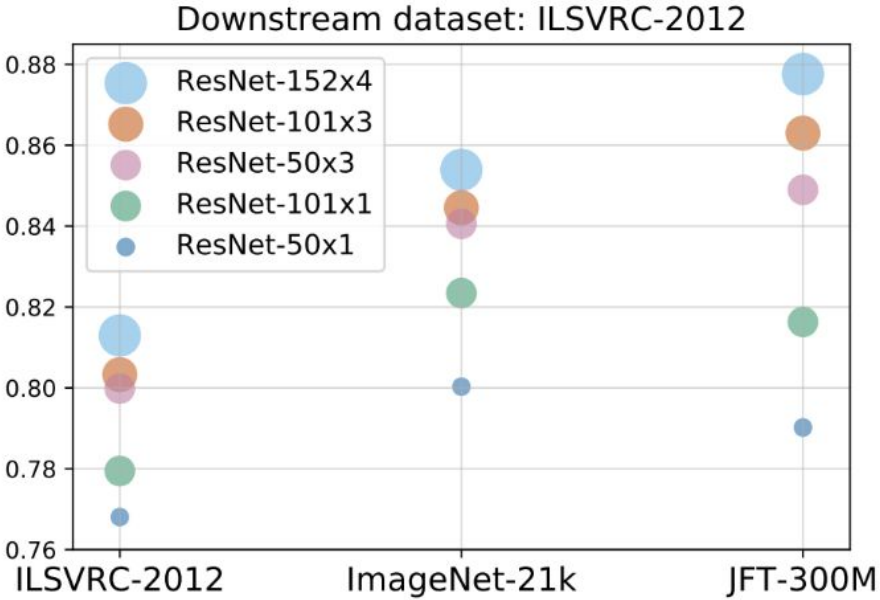
- ResNet as a basic backbone
- BatchNorm → GroupNorm
- Weight Standartization
- Weight decay
- No MixUp required

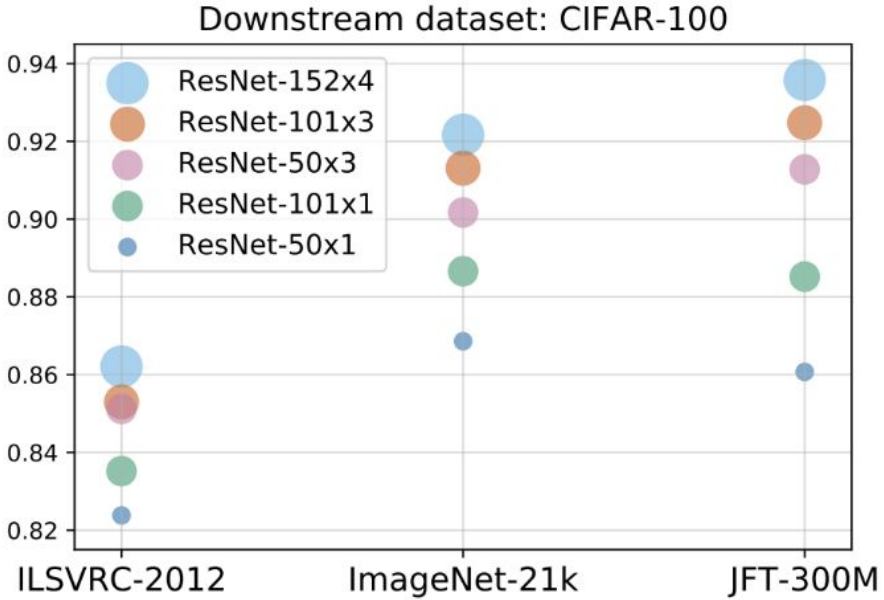
Fine-tuning

- Resize to a square, crop a smaller square, horizontal flips
- No weight decay, dropout
- MixUp is sometimes useful

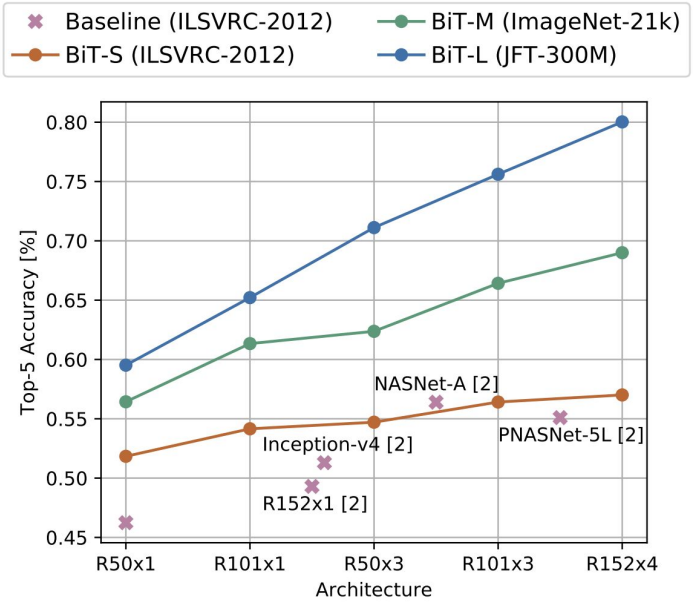
Kolesnikov et al., Big Transfer (BiT): General Visual Representation Learning, 2019, https://arxiv.org/pdf/1912.11370

Big Transfer

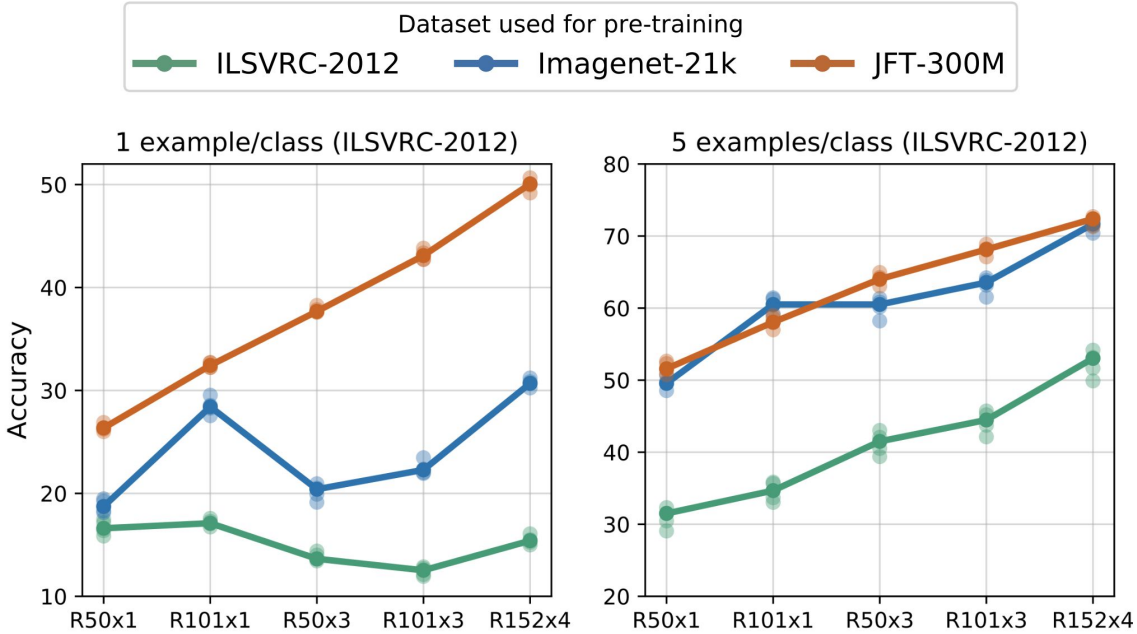




Big Transfer



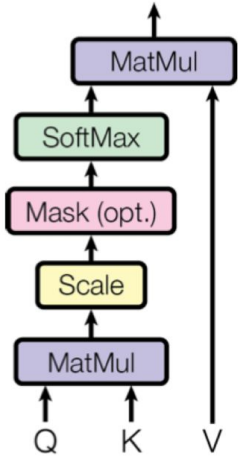
ObjectNet transfer



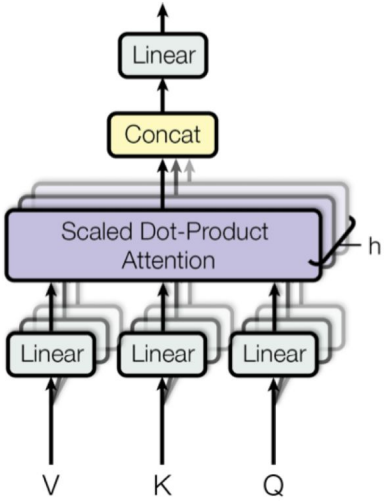
Few-shot linear probe

Transformer

Scaled Dot-Product Attention



Multi-Head Attention



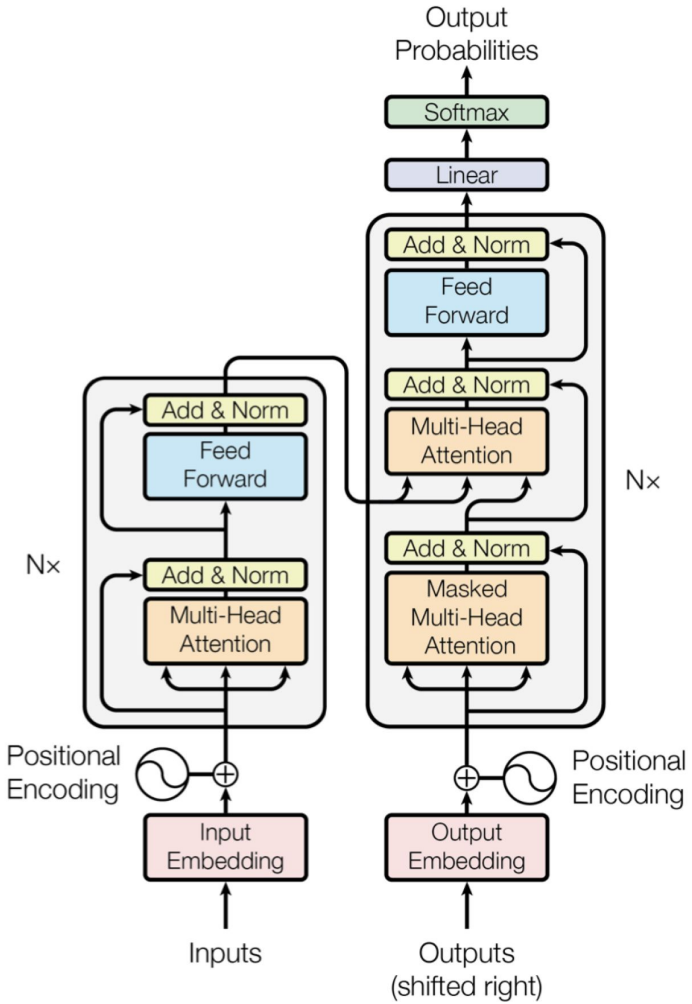
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

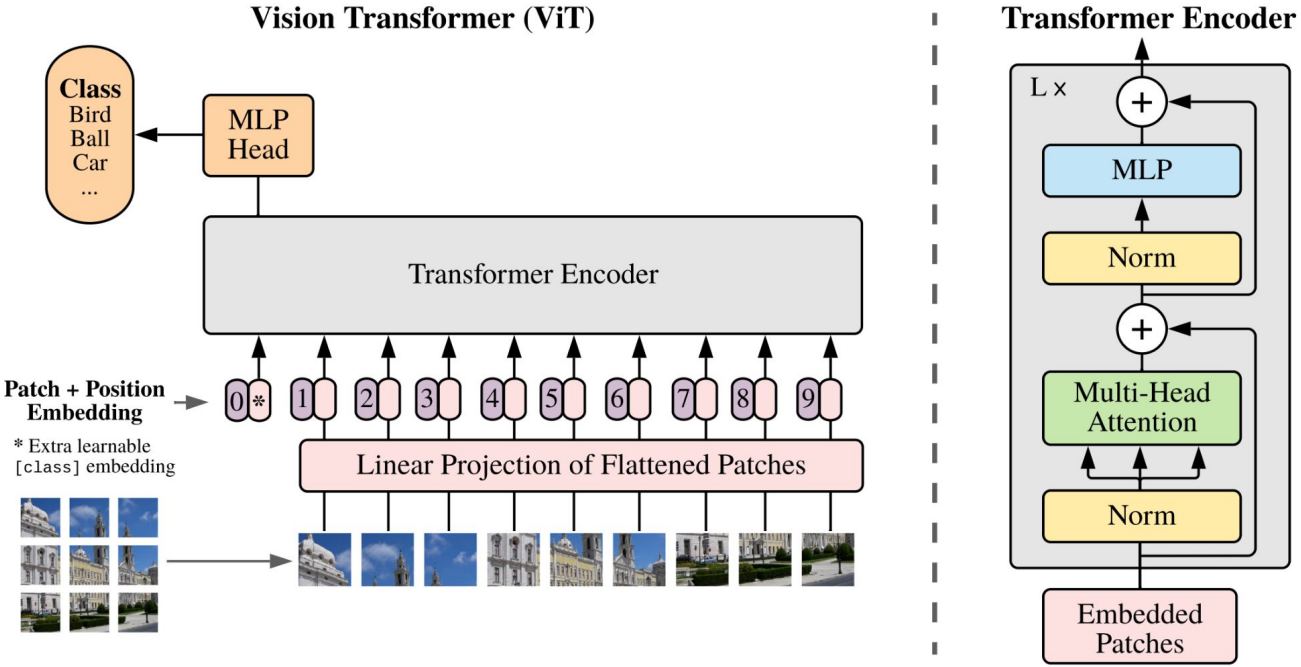
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Vaswani et al., Attention Is All You Need, 2017, https://arxiv.org/abs/1706.03762

Transformer



Vision Transformer



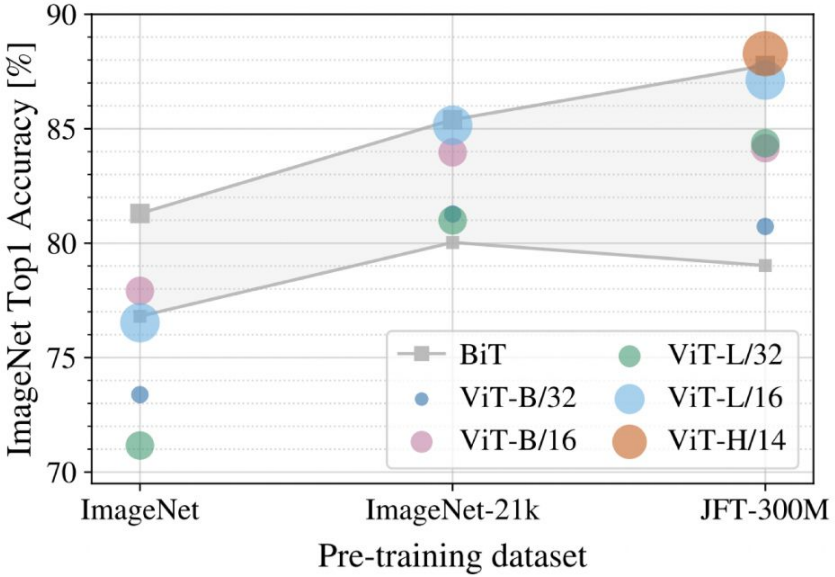
Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2020, https://arxiv.org/abs/2010.11929

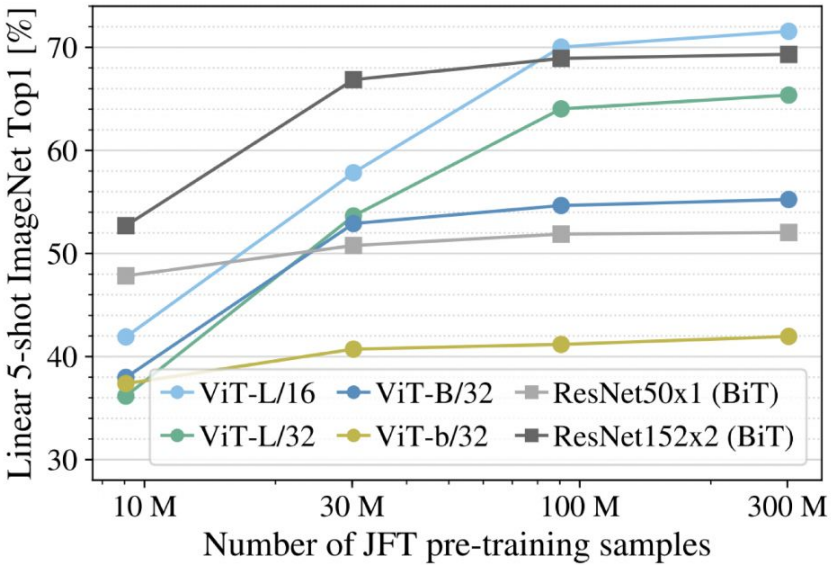
How to train your ViT

- Use checkpoints that were pre-trained on more upstream data
- Applying data augmentation and model regularization
- To fine-tune select the model with the best validation performance on pretrain data
- Use warmup and a high weight decay (of 0.1)!
- Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$

Vision Transformer

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



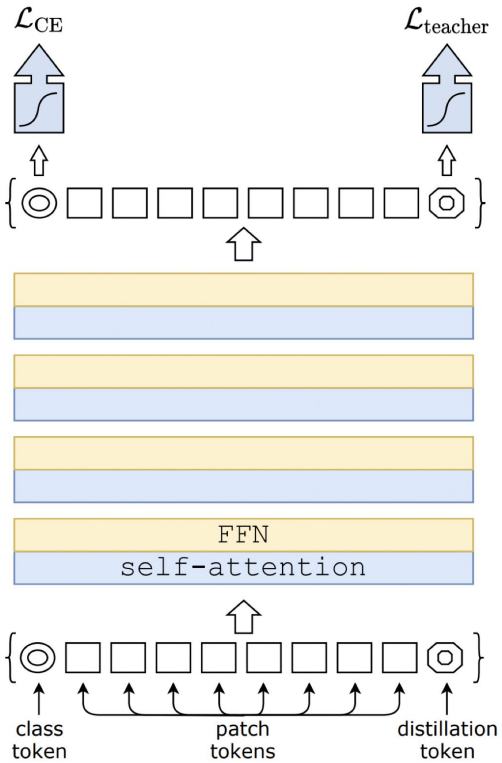


DeiT

$$\mathcal{L}_{\text{global}} = (1-\lambda)\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \lambda\tau^2 \text{KL}(\psi(Z_s/\tau), \psi(Z_t/\tau))$$

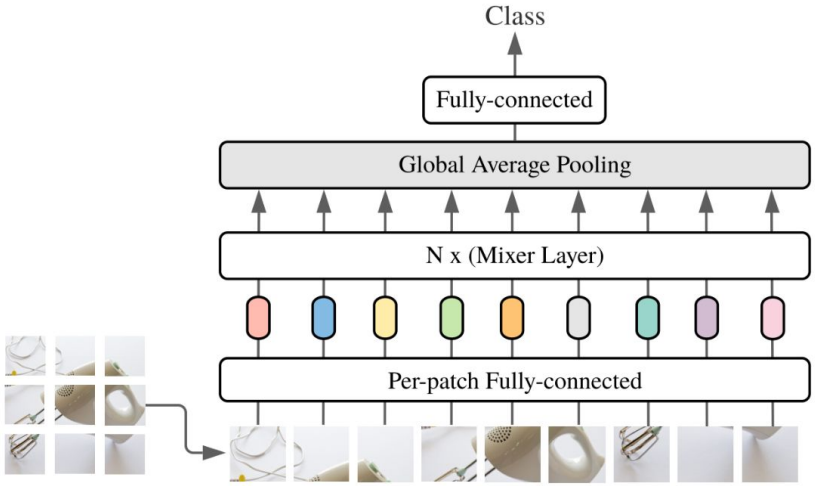
$$\mathcal{L}_{\text{global}}^{\text{hardDistill}} = \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y_t)$$

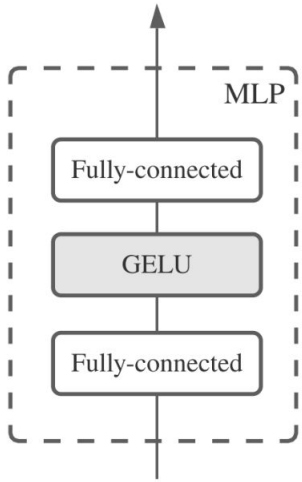
method ↓	Supervision label	Supervision teacher	ImageNet top-1 (%) Ti 224	S 224	B 224	B↑384
DeiT– no distillation	✓	✗	72.2	79.8	81.8	83.1
DeiT– usual distillation	✗	soft	72.2	79.8	81.8	83.2
DeiT– hard distillation	✗	hard	74.3	80.9	83.0	84.0
DeiT⚗: class embedding	✓	hard	73.9	80.9	83.0	84.2
DeiT⚗: distil.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

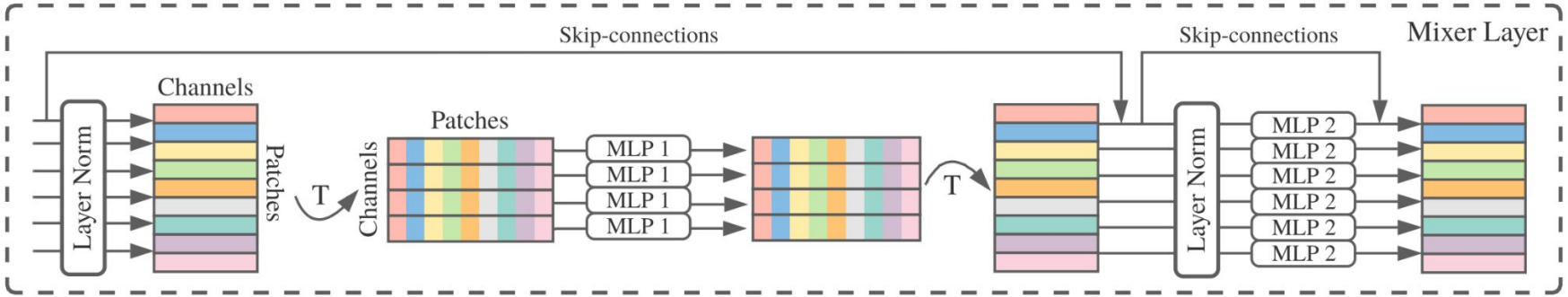


Touvron et al., Training data-efficient image transformers & distillation through attention, 2020, https://arxiv.org/abs/2012.12877

MLP Mixer







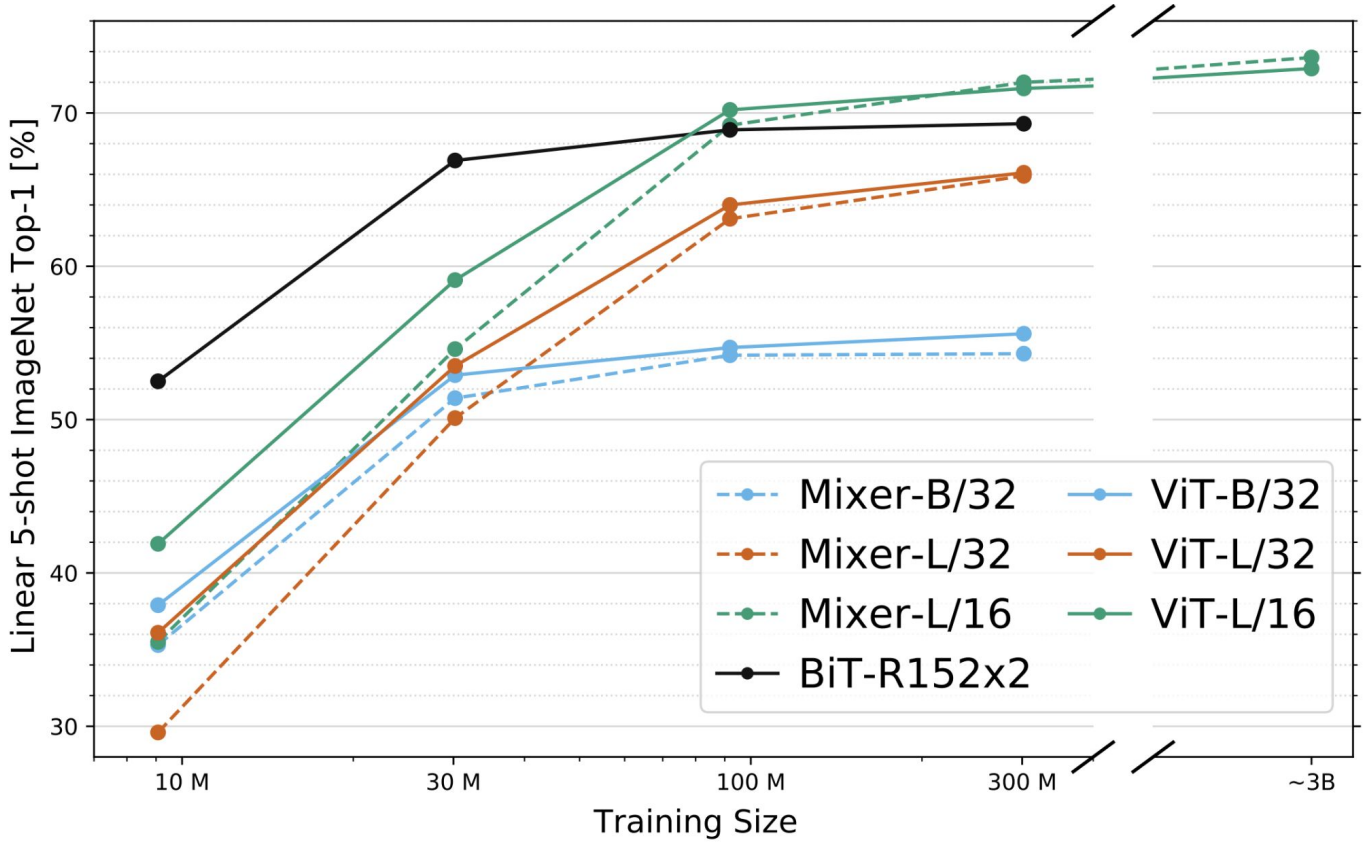
Hassani et al., MLP-Mixer: An all-MLP Architecture for Vision, 2021, https://arxiv.org/pdf/2105.01601

MLP Mixer: results

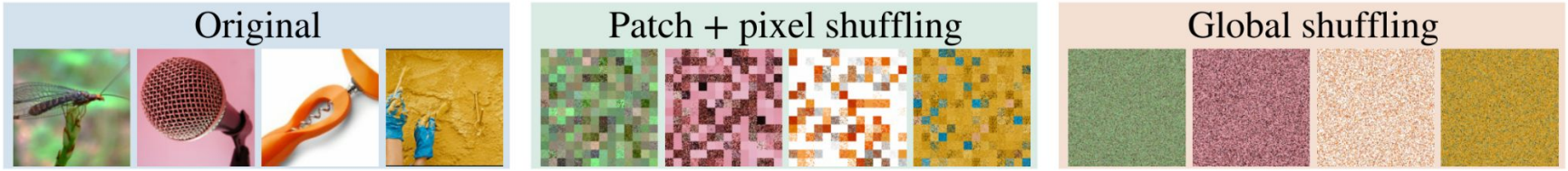
	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (public)						
● HaloNet [51]*	85.8	—	—	—	120	0.10k
● Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
● ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
● BiT-R152x4 [22]	85.39	—	94.04	70.64	26	0.94k
Pre-trained on JFT-300M (proprietary)						
● NFNet-F4+ [7]	89.2	—	—	—	46	1.86k
● Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
● BiT-R152x4 [22]	87.54	90.54	95.33	76.29	26	9.90k
● ViT-H/14 [14]	88.55	90.72	95.97	77.63	15	2.30k

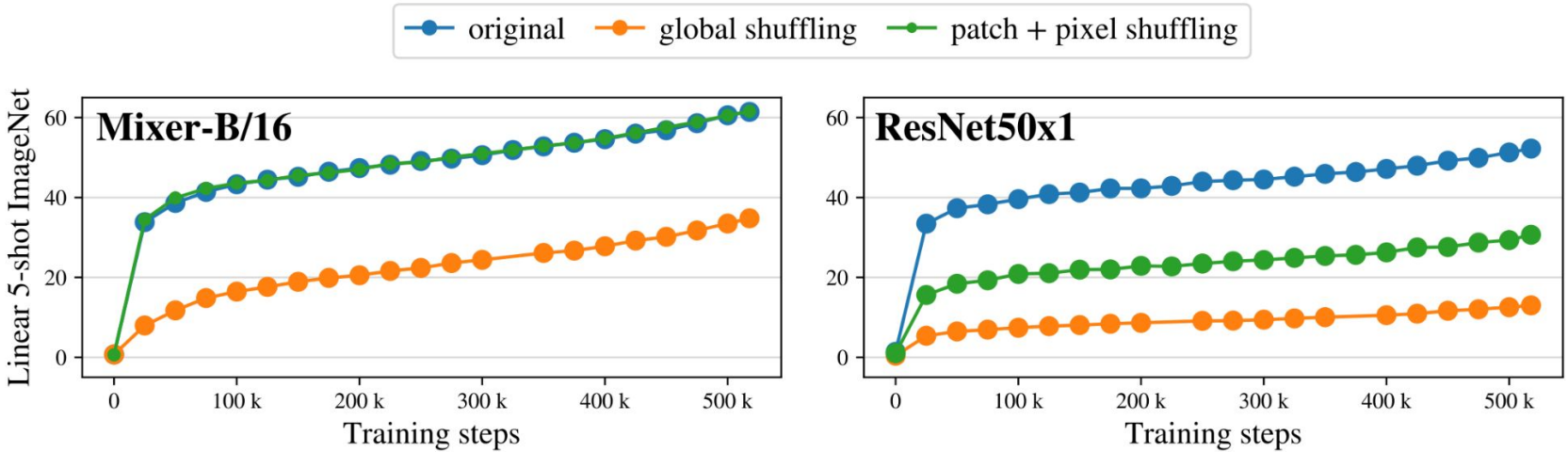
● MLP-based Mixer ● convolution-based ● attention-based

MLP Mixer: dataset size

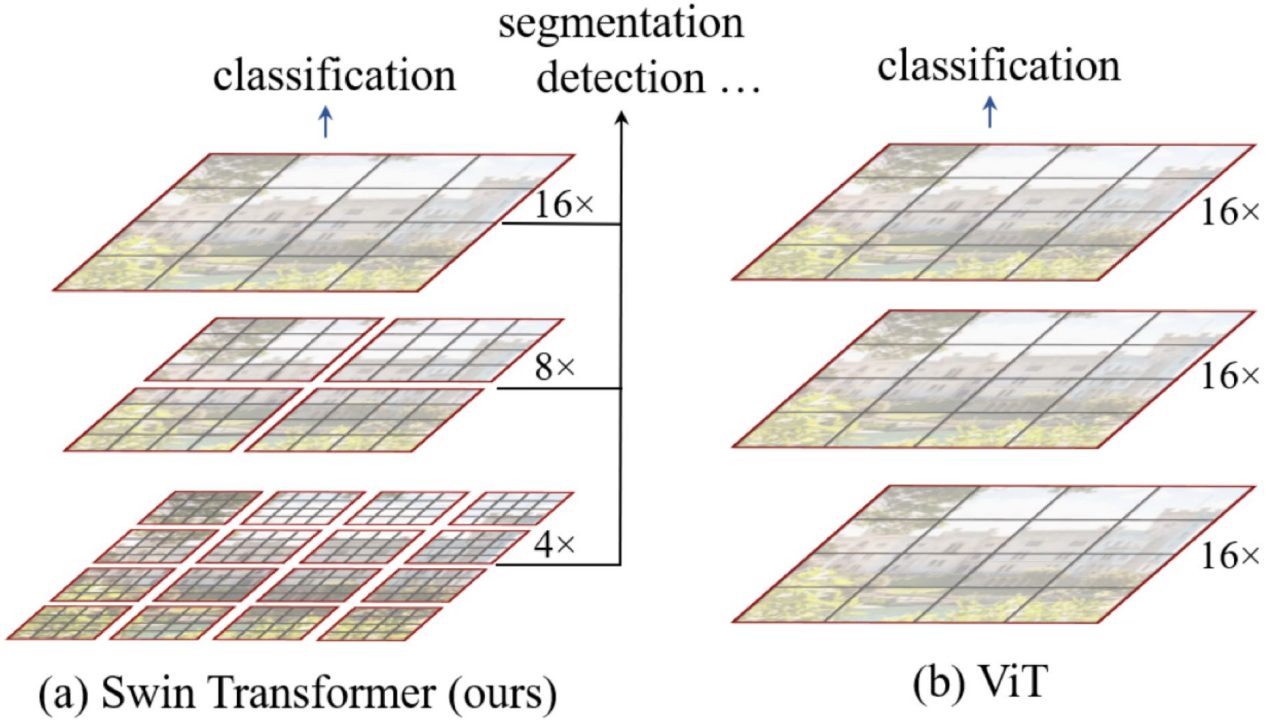


MLP Mixer: inductive bias





Swin: hierarchical structure



Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, 2021, https://arxiv.org/pdf/2103.14030

Swin: shifted windows

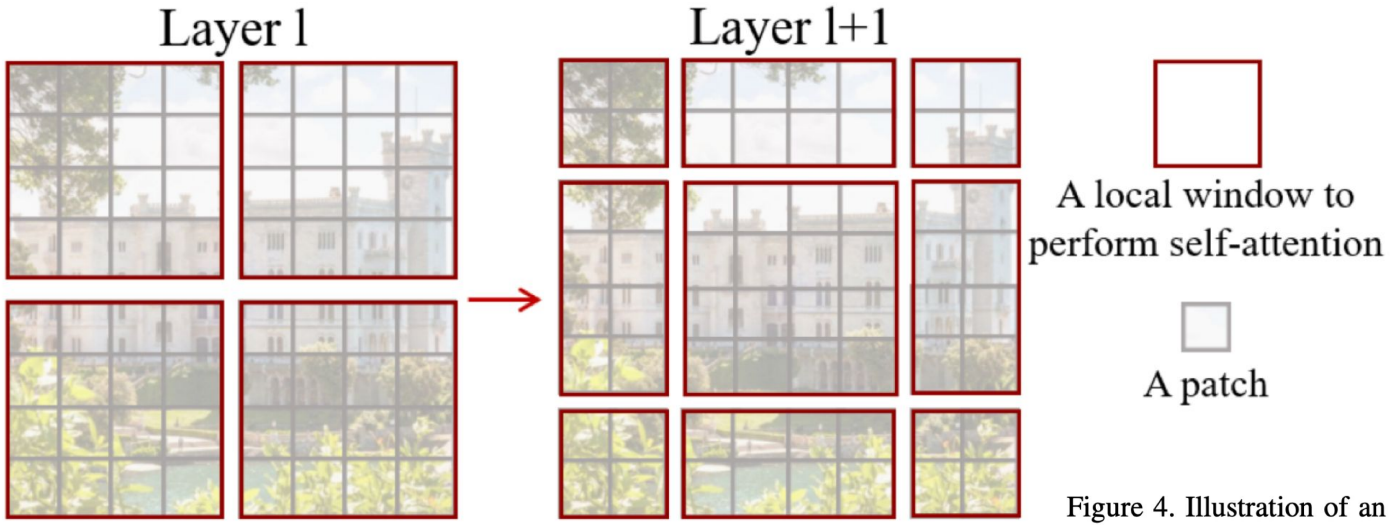
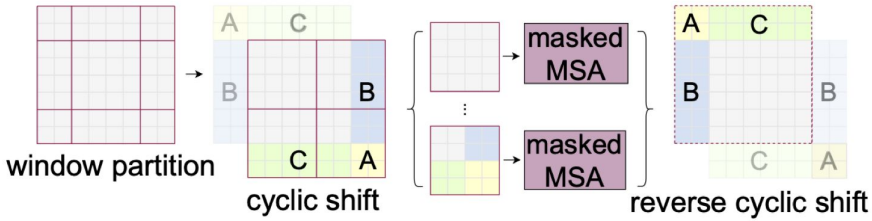
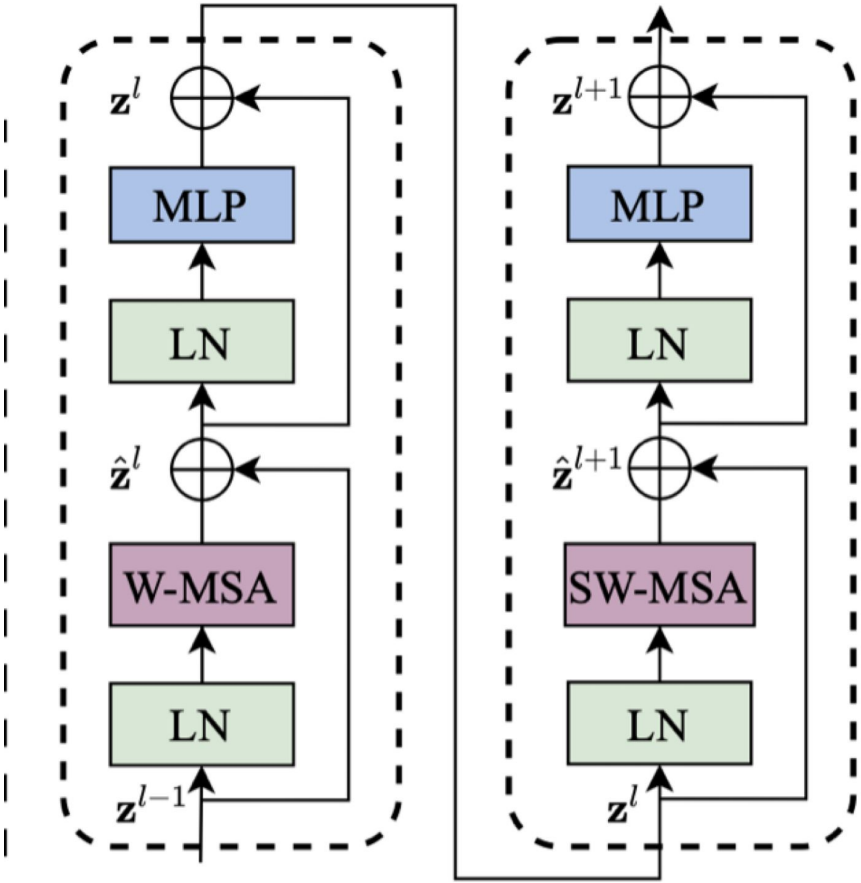


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

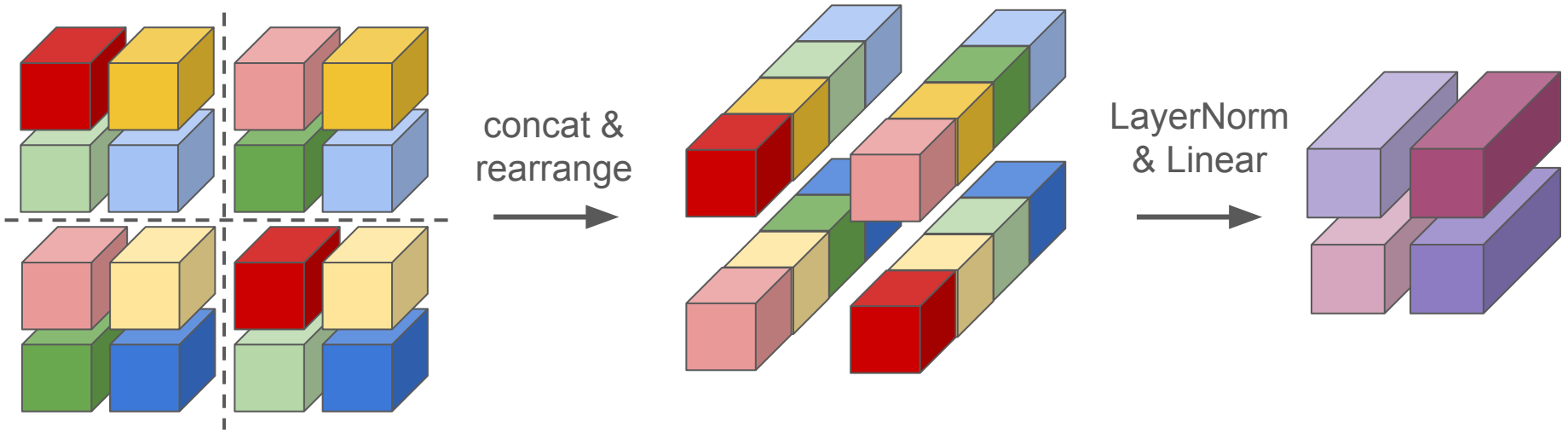


Swin: shifted windows

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



Swin: patch merging

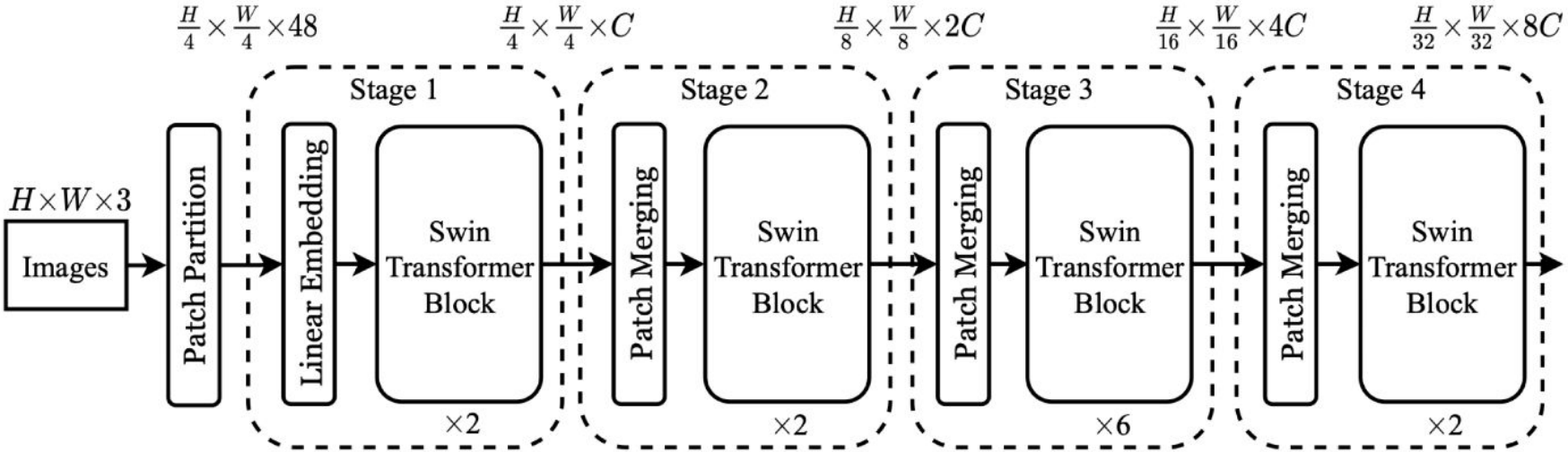


(H, W, C)

(H/2, W/2, 4C)

(H/2, W/2, 2C)

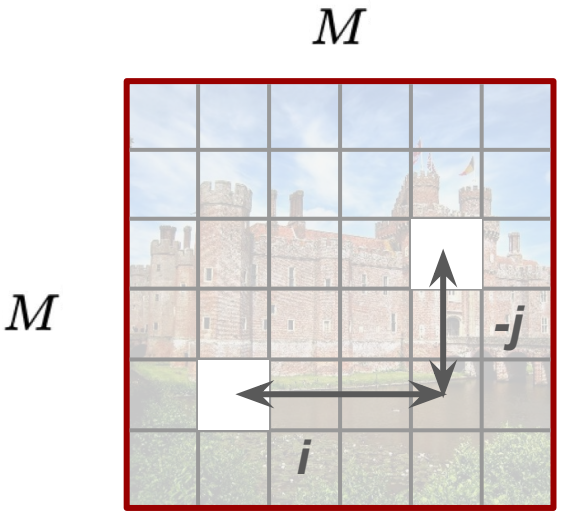
Swin: overall architecture

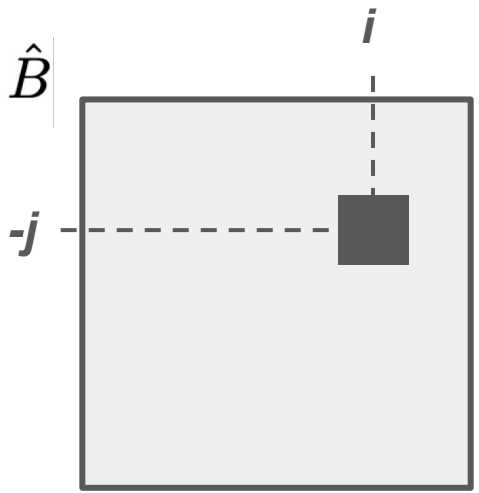


Swin: relative position bias

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d} + B)V, \qquad Q, K, V \in \mathbb{R}^{M^2 \times d}$$

$$B \in \mathbb{R}^{M^2 \times M^2} \qquad B \text{ are taken from } \hat{B}, \; \hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$$





Swin: classification results

(a) Regular ImageNet-1K trained models

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

(b) ImageNet-22K pre-trained models

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

Swin: segmentation & detection results

ADE20K semantic segmentation

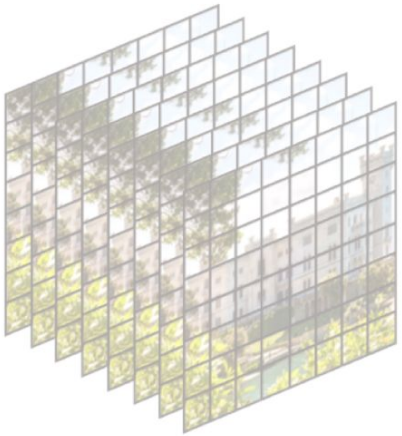
ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	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

COCO object detection

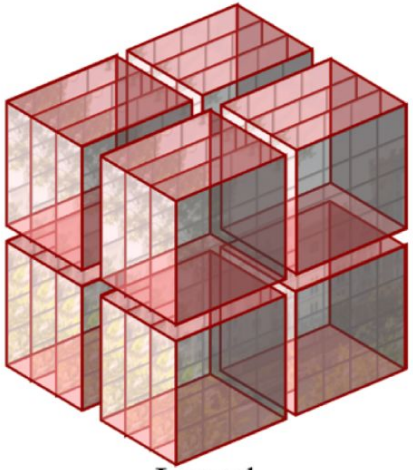
(a) Various frameworks

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

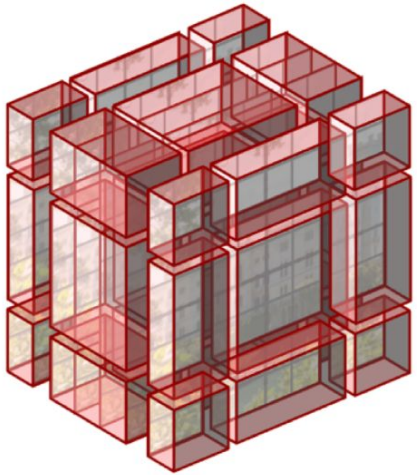
Video Swin Transformer



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



Layer l
window: 2×2×2=8



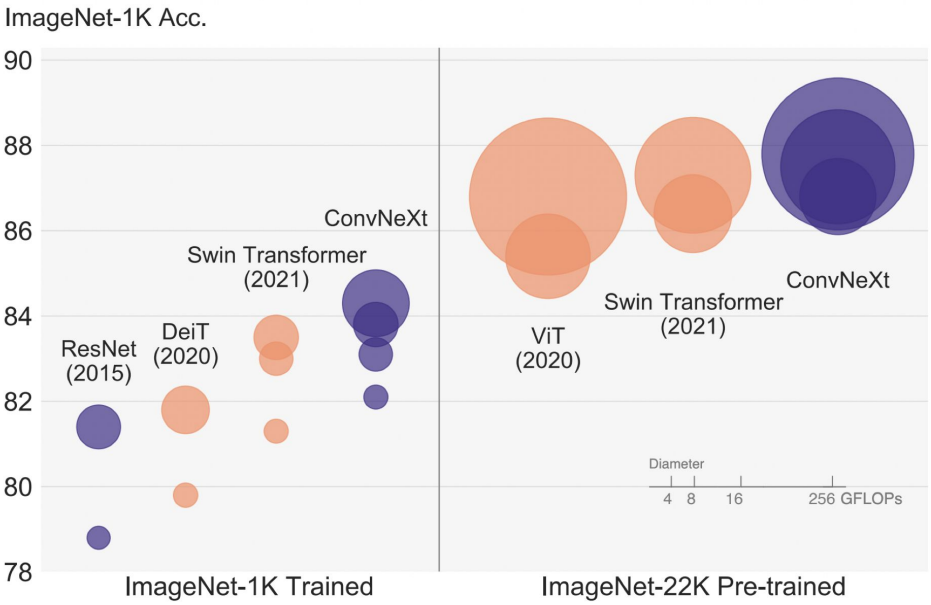
Layer l+1
window: 3×3×3=27

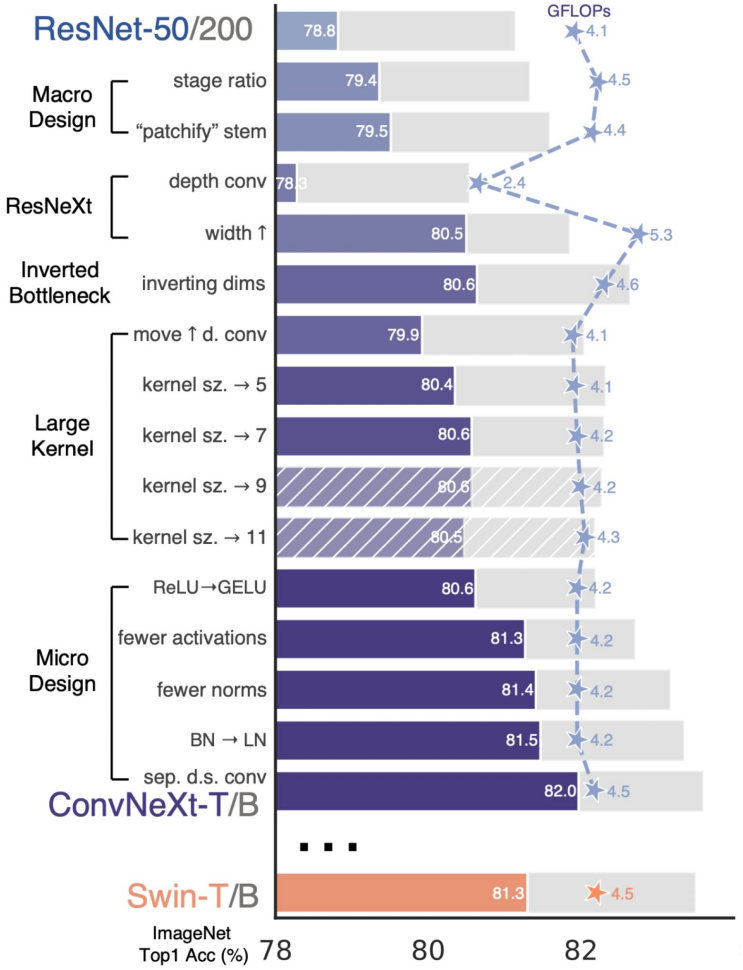
3D local window to perform self-attention

A token

Liu et al., Video Swin Transformer, 2021, https://arxiv.org/abs/2106.13230

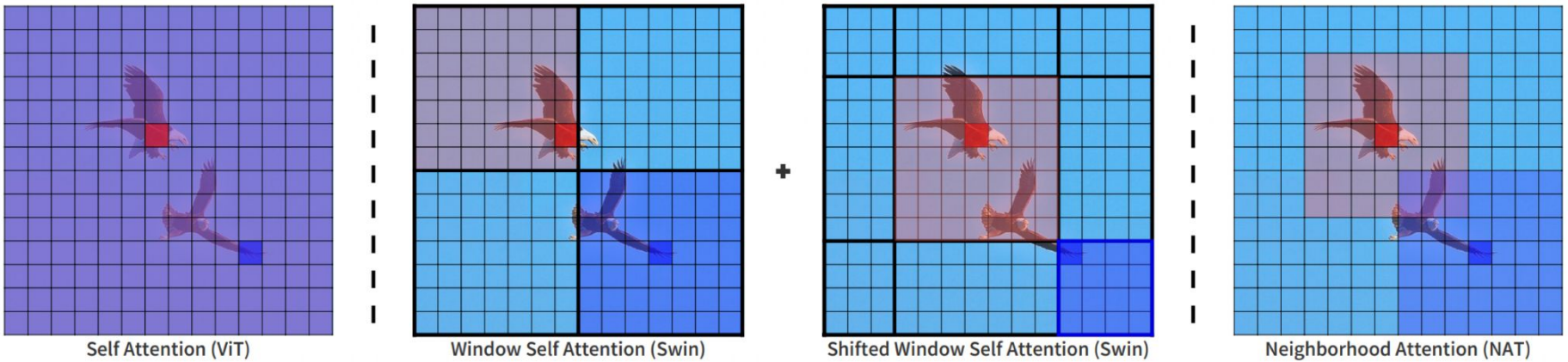
ConvNeXt





Liu et al., A ConvNet for the 2020s, 2022, https://arxiv.org/abs/2201.03545

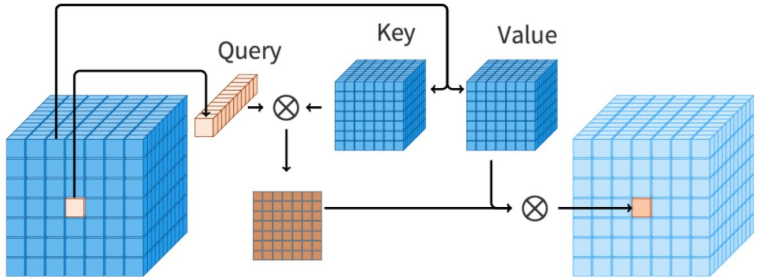
Neighborhood Attention Transformer



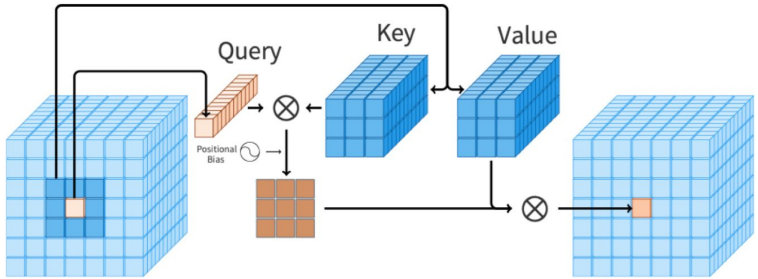
Hassani et al., Neighborhood Attention Transformer, 2022, https://arxiv.org/abs/2204.07143

Neighborhood Attention Transformer

Self Attention



Neighborhood Attention

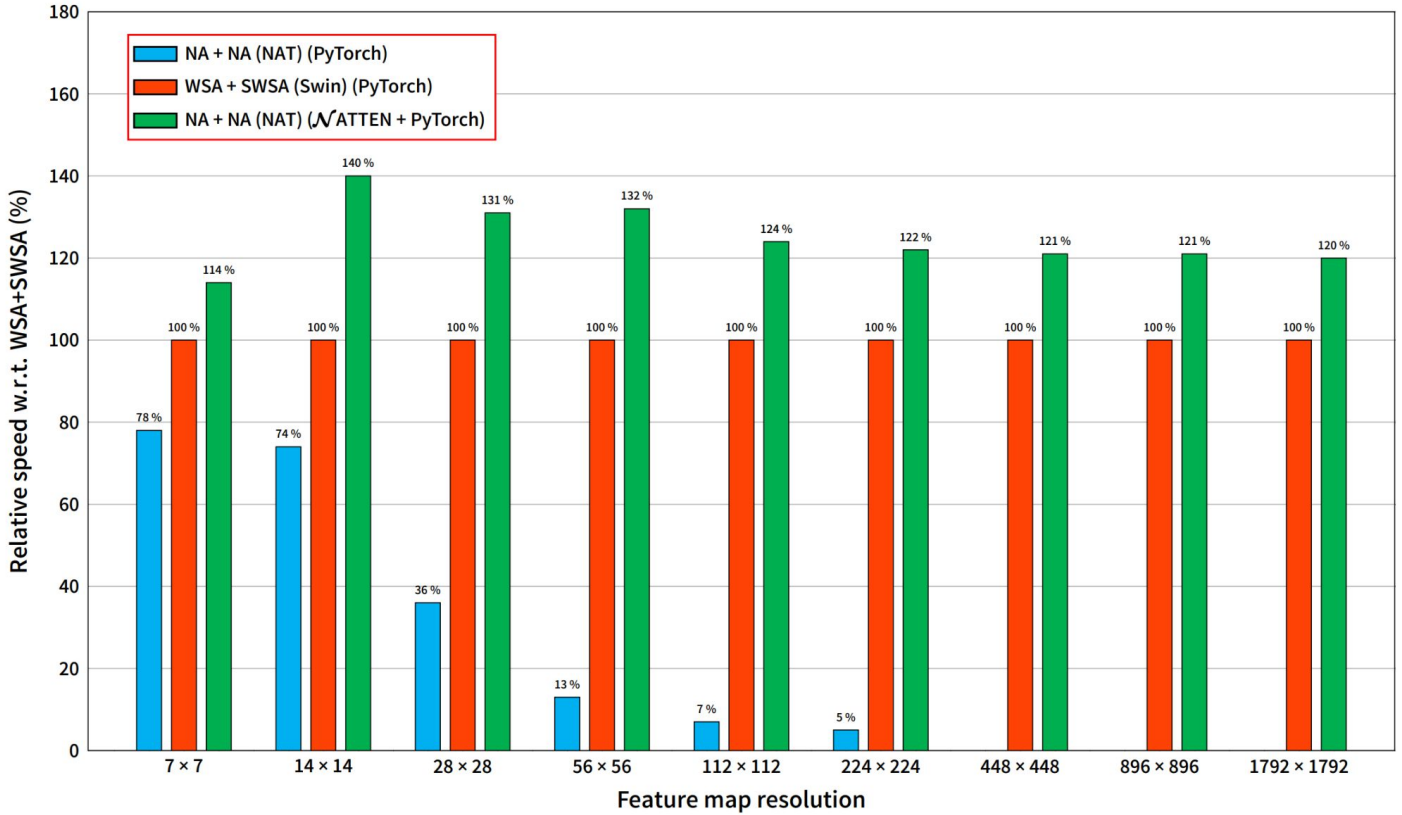


$$\mathbf{A}_i^k = \begin{bmatrix} Q_i K^T_{\rho_1(i)} + B_{(i,\rho_1(i))} \\ Q_i K^T_{\rho_2(i)} + B_{(i,\rho_2(i))} \\ \vdots \\ Q_i K^T_{\rho_k(i)} + B_{(i,\rho_k(i))} \end{bmatrix}$$

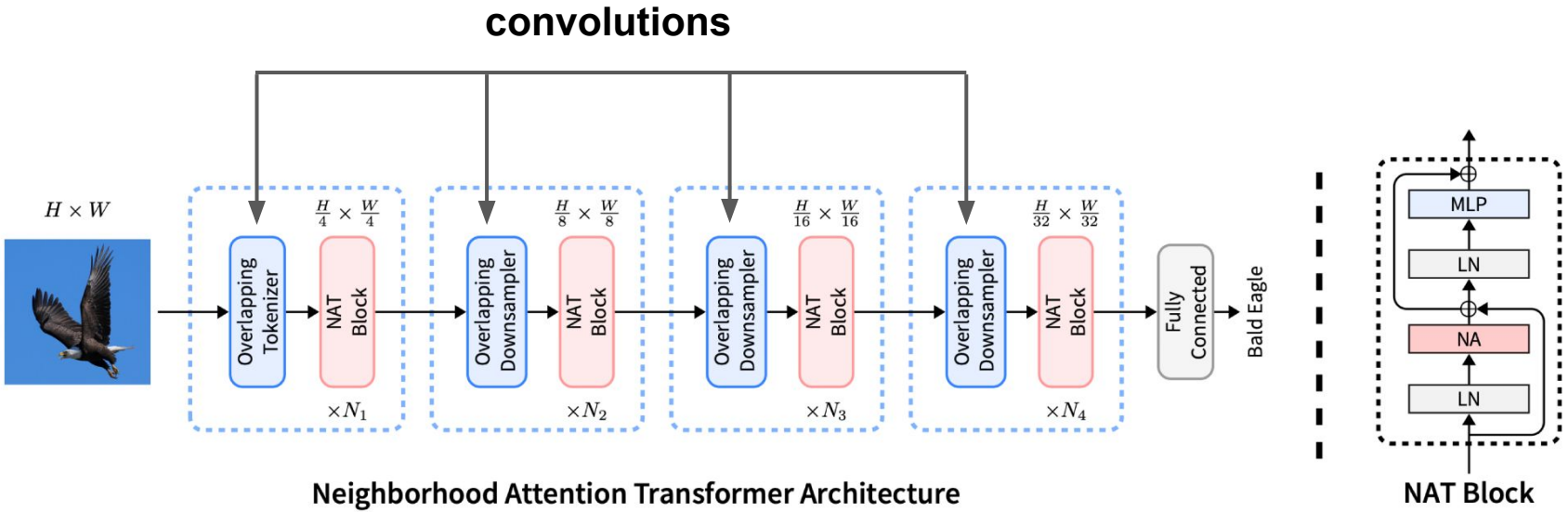
$$\mathbf{V}_i^k = \begin{bmatrix} V^T_{\rho_1(i)} & V^T_{\rho_2(i)} & \cdots & V^T_{\rho_k(i)} \end{bmatrix}^T$$

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

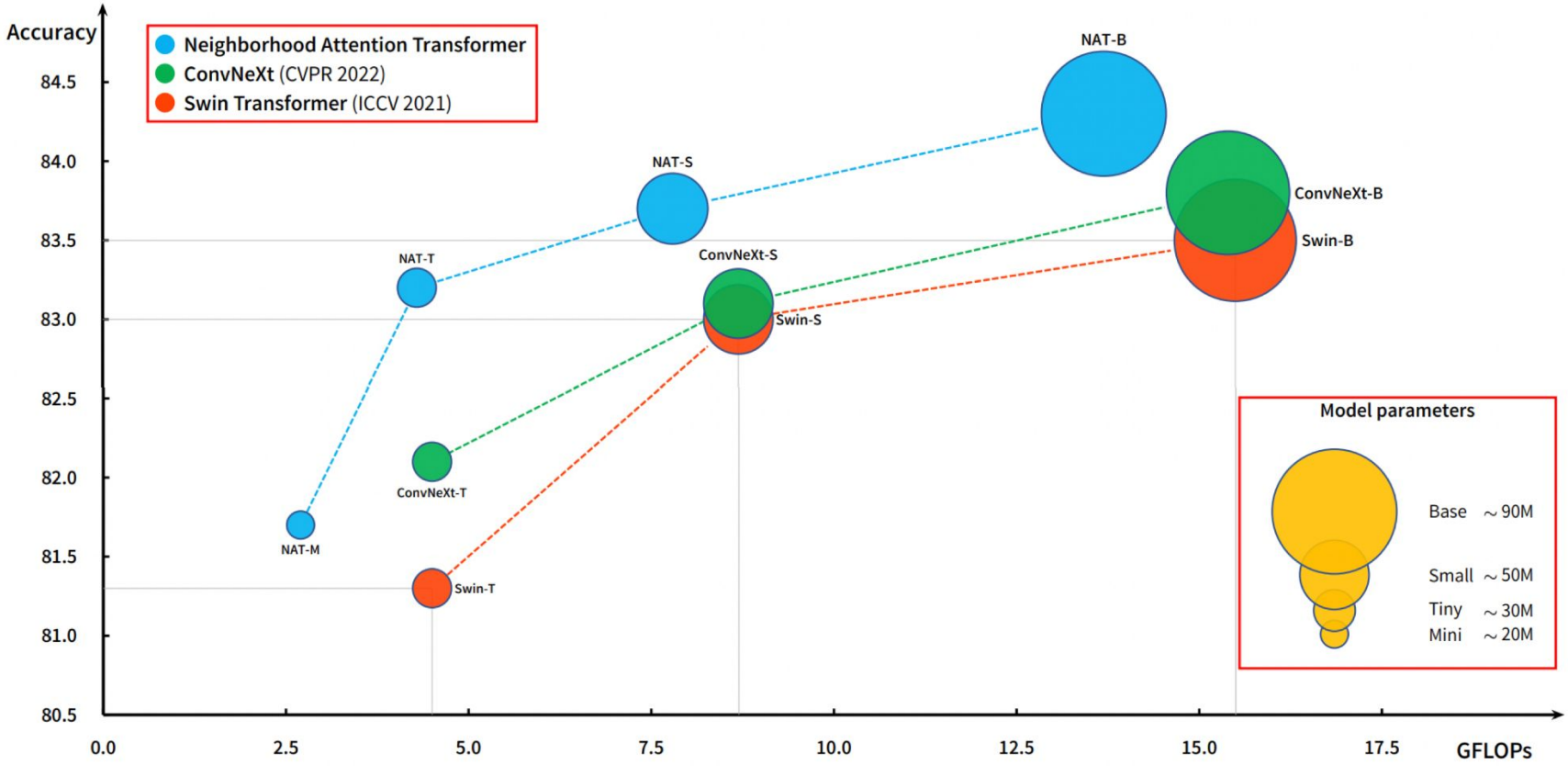
Neighborhood Attention Transformer



Neighborhood Attention Transformer



Neighborhood Attention Transformer



Conclusion

- **ConvNets** – easiest to train, best for low-data regime
- **ViT** – harder to train, requires much more data,
  but achieves superior performance
- **Swin/NAT** – local attention & hierarchical structure
- **MLP Mixer** – neither attention nor convolutions
  are required for good quality