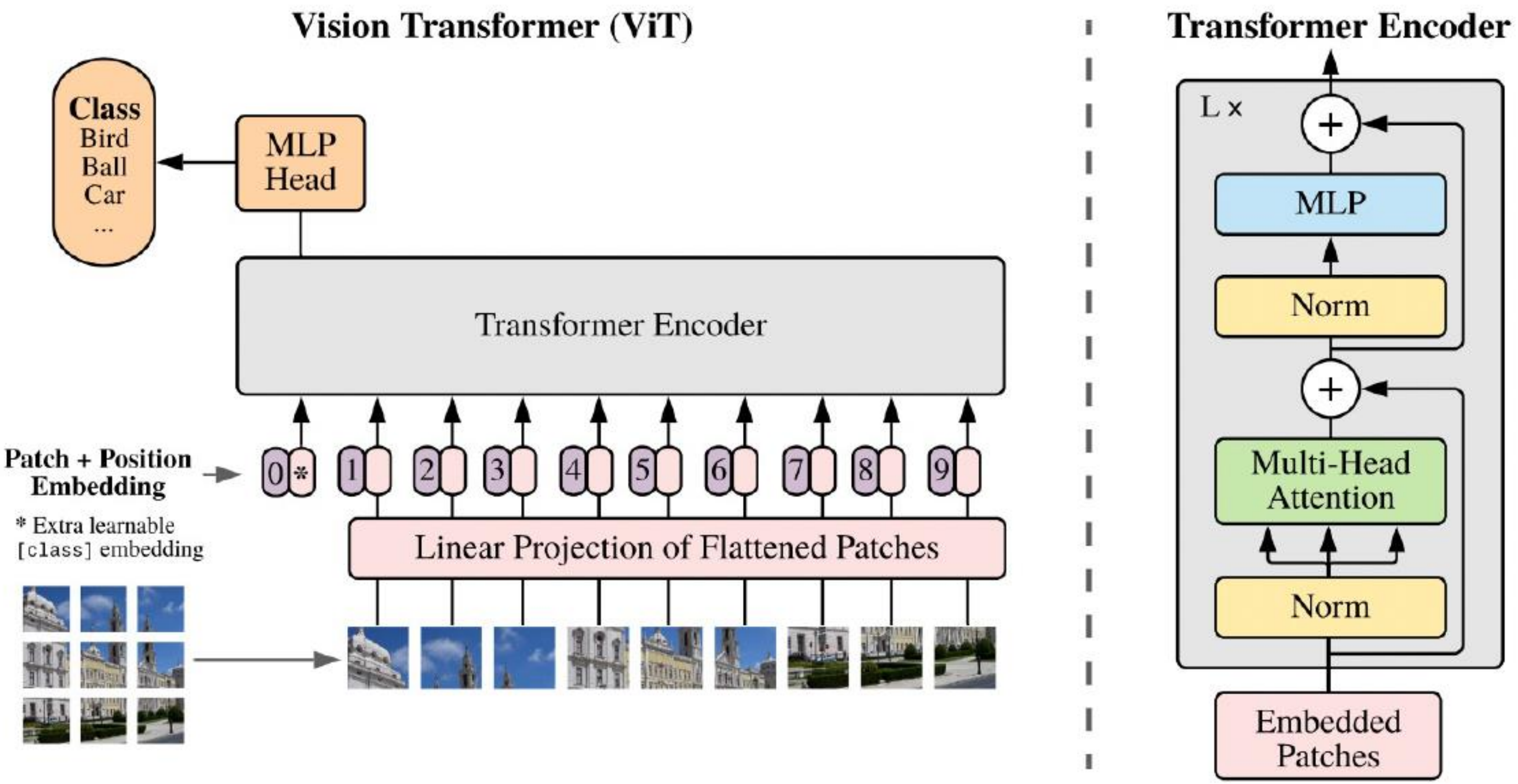
Vision transformers

Alberto Albiol

Contents

- ViT
- DeiT
- Swin, Swin2
- BeiT
- Metaformer -> Poolformer
- Twins
- Maxvit
- Avanced applications

Vit transformer (by Google)

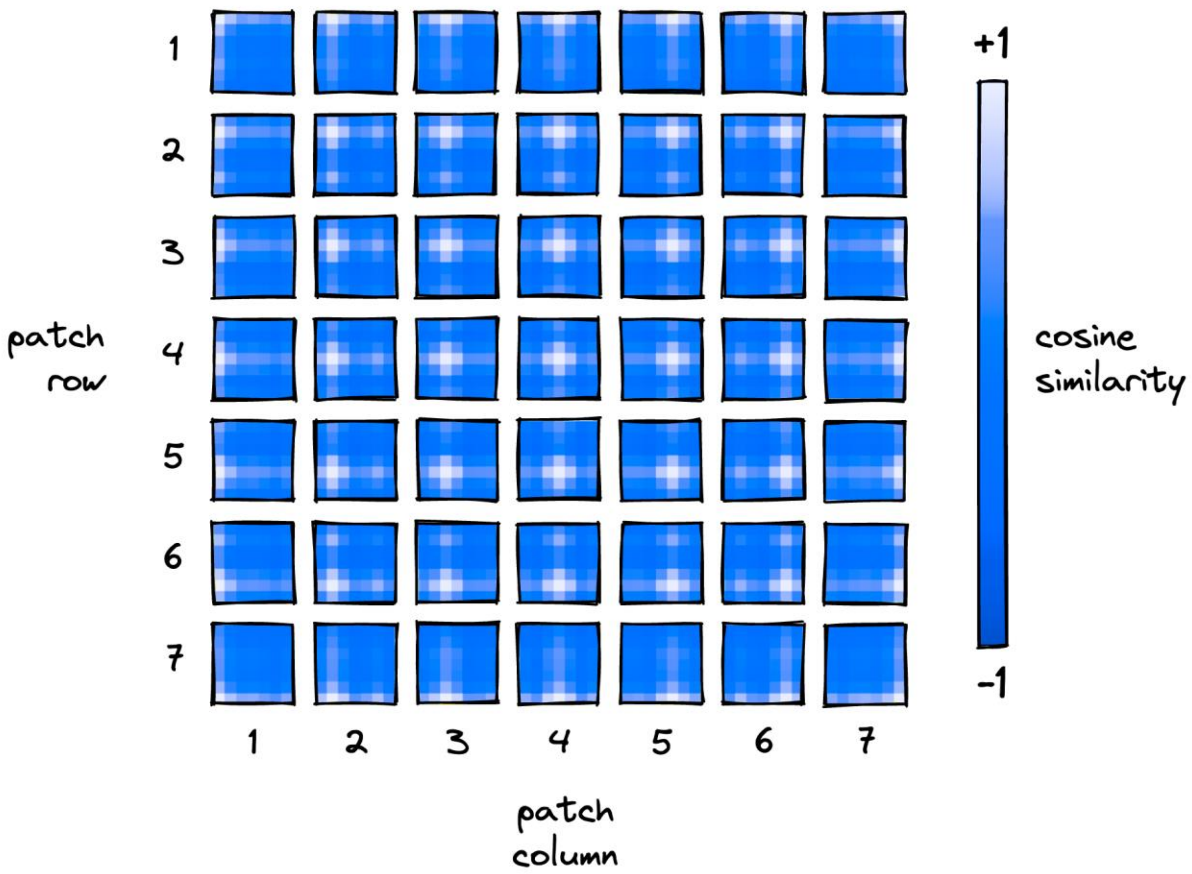


ViT transformer details

- Layer normalization is done before Multi-head self-attention
- Adds extra-token for image classification
- Extends the idea of transformers to methodology
- Replace convolutions with transformer layers (multihead attention)

ViT positional encodings

- Use learned positional encodings
- Can be finetuned
- Similarity with neighbouring positional encodings is high
- For higher resolutions, a 2D interpolation of the pre-trained position embeddings is performed.



ViT vs Convolutional networks

- For transformer based ViT architecture to learn generalized representation there is a requirement to have lots of data (100s of million images), under which threshold the model does not outperform CNN based approaches.
- Vision Transformers offer advantages in scenarios where global dependencies and contextual understanding are crucial.
- For fine-grained classification, which involves distinguishing between highly similar classes, ViTs demonstrated a stronger ability to capture subtle differences.
- ViT are more robust to adversarial perturbations

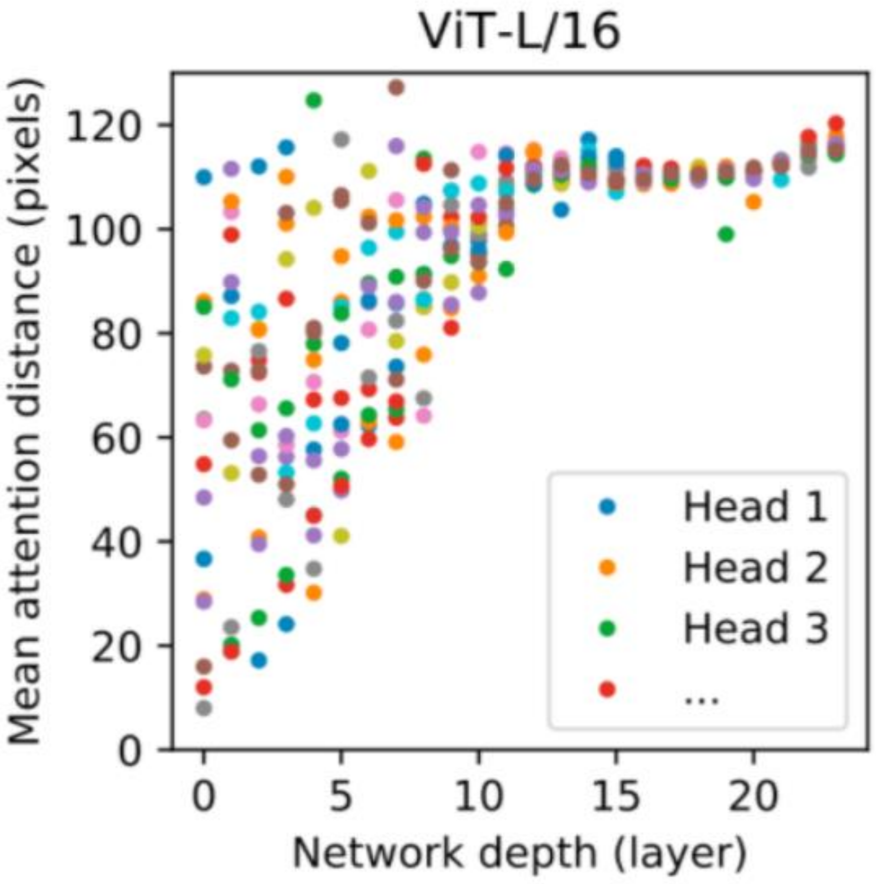
Pretrained ViT models

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

ViT performance

	ViT-H	Previous SOTA
ImageNet	88.55	88.5
ImageNet-ReaL	90.72	90.55
Cifar-10	99.50	99.37
Cifar-100	94.55	93.51
Pets	97.56	96.62
Flowers	99.68	99.63

Attention and layer depth



Visualizing attention

- Recap:
  - The unit of attention is a token (image patch)
  - Visualization is done through computing a 0–1 ligthness (i.e. attention weight) at each token
  - Attention can be visualized for at any (attention) head at any layer

Visualizing attention

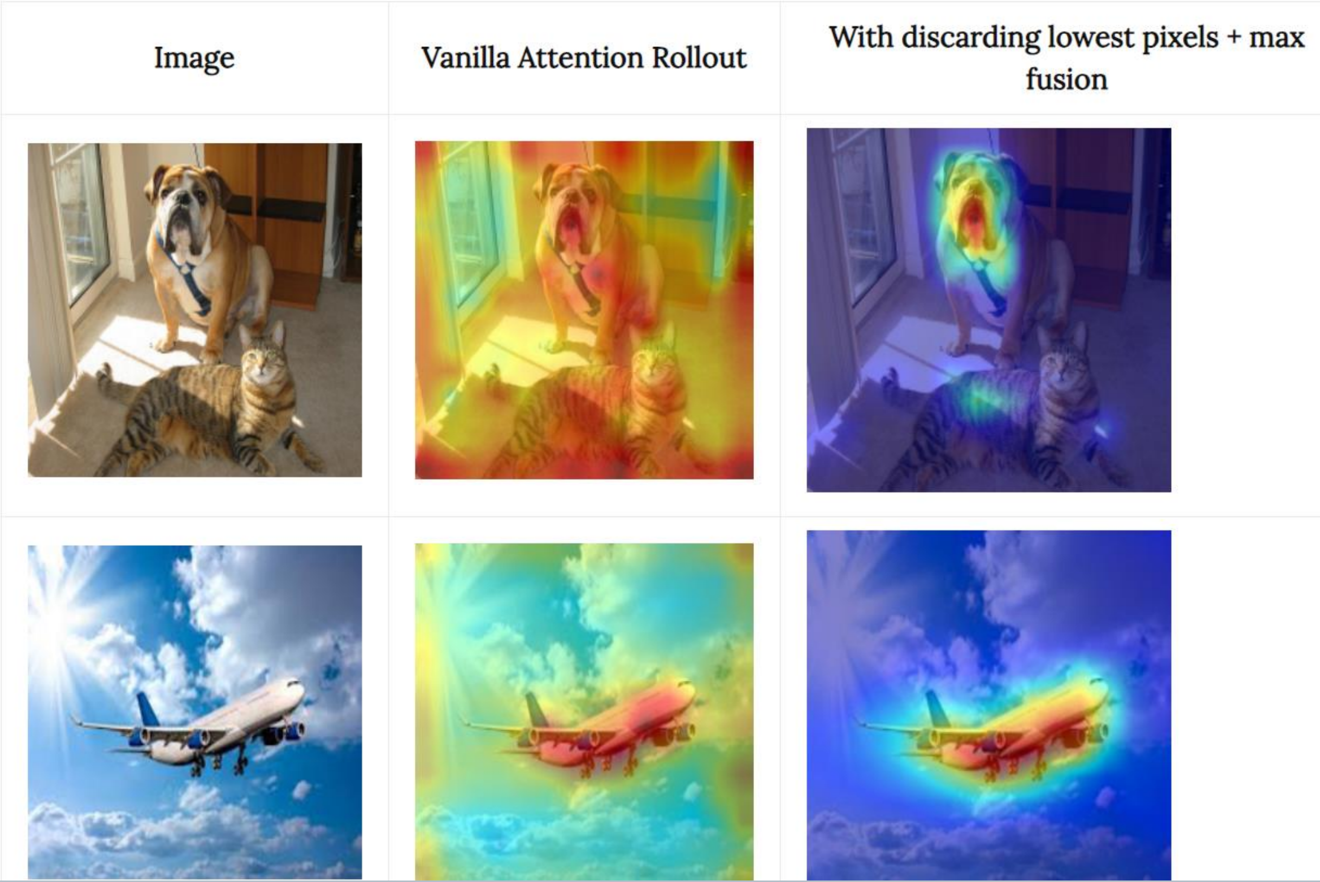
- Attention rollout:
  - Feed image in ViT ->Attention layers **[A^l_h]_l,h,** layer **l** has **h** attention heads with size (N+1)x(N+1) (N is number of image patch tokens + 1 CLS token)
  - aggregate (can be done with mean, max, min...) the attention over heads so that **[A^l_h]** reduced to **[A^l]**
  - compute the attention rollout (estimating the flow of attention in ViT network) at layer **l** through the following iterative equation:

$$A^l_{rollout} = (A^l + I)A^{l-1}_{rollout}, \text{ for } i > 1$$

$$A^1_{rollout} = A^1 + I$$

Visualizing attention (cont.)

– We take out the 1st column of **A^l_{rollout}** (estimates self-attention weights of CLS token to all tokens in the sequence)

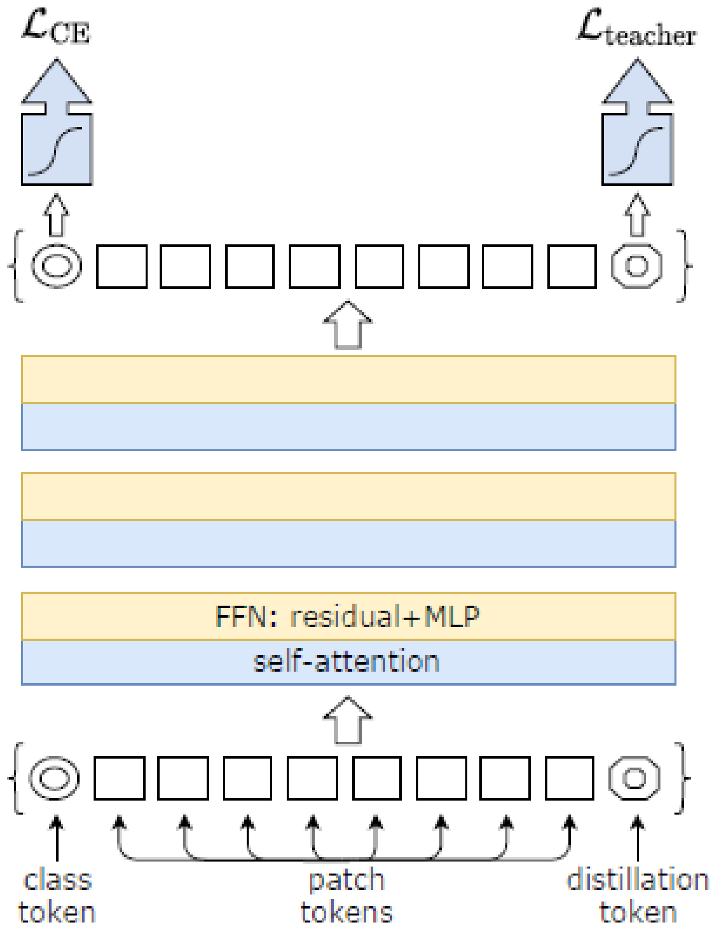


DeiT: Data efficient image Transformer (by. Meta)

- ViT does not generalize until tranined with huge datasets
- DeiT and ViT architectures are the same, but DeiT generalizes well only with imagenet data
- Teacher-student strategy is used with extra distillation token
- Idea:

> *Assume we have access to a strong image classifier as a teacher model. Can the Transformer be learnt by exploiting this teacher?*

Deit architecture



- Teacher is a CNN
- L_teacher uses a teacher
  - Soft distillation
  - Hard distillation
- L_CE uses true labels

Hard vs. soft distillation

› Soft distillation uses logits:

$$\mathcal{L}_{\text{global}} = (1-\lambda)\boxed{\mathcal{L}_{\text{CE}}(\psi(Z_{\text{s}}), y)} + \lambda\tau^2\boxed{\text{KL}(\psi(Z_{\text{s}}/\tau), \psi(Z_{\text{t}}/\tau))}.$$

Soft Distillation Objective

› Hard distillation uses labels from teacher:

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

Hard Distillation Objective

Hard vs. soft distillation

› Hard distillation produces better results than soft distillation

Comments on distillation

- When teacher's softmax distribution is much closer to original ground truth labels and has not much different effect than training with ground truth labels
- Adding a temperature term in softmax, the distribution becomes much smoother, and it allows the student to learn

- Hinton 2015 video explaining distillation:
  - https://www.youtube.com/watch?v=EK61htlw8hY

DeiT performance

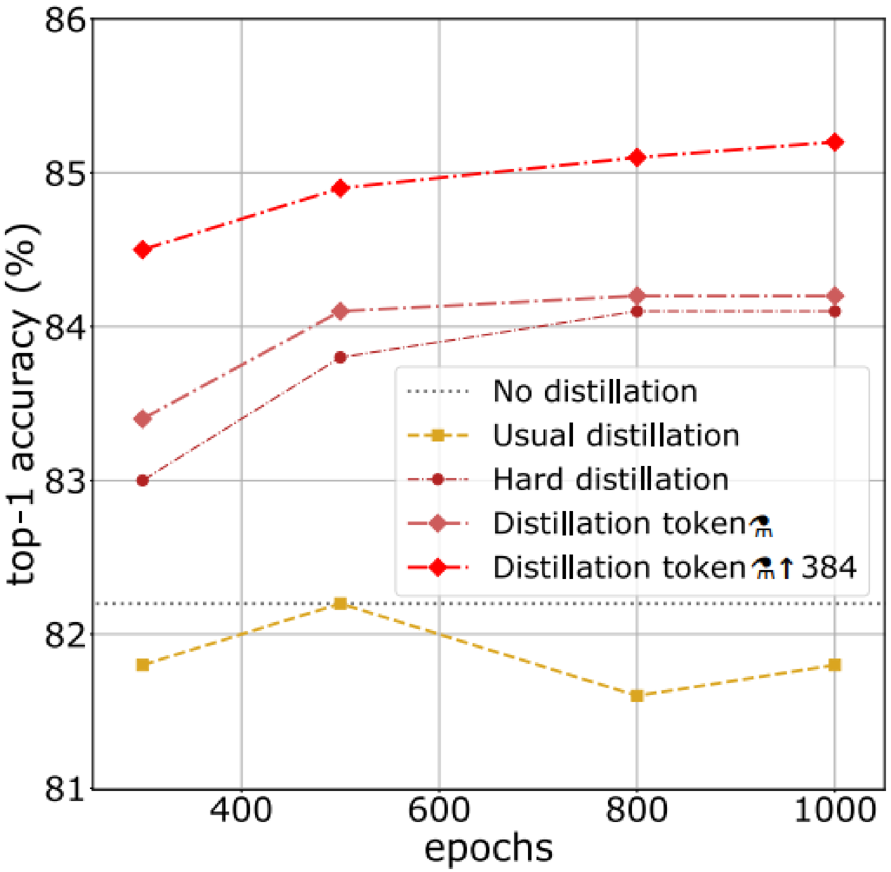


Figure 3: Distillation on ImageNet [42] with DeiT-B: performance as a function of the number of training epochs. We provide the performance without distillation (horizontal dotted line) as it saturates after 400 epochs.

Deit performance

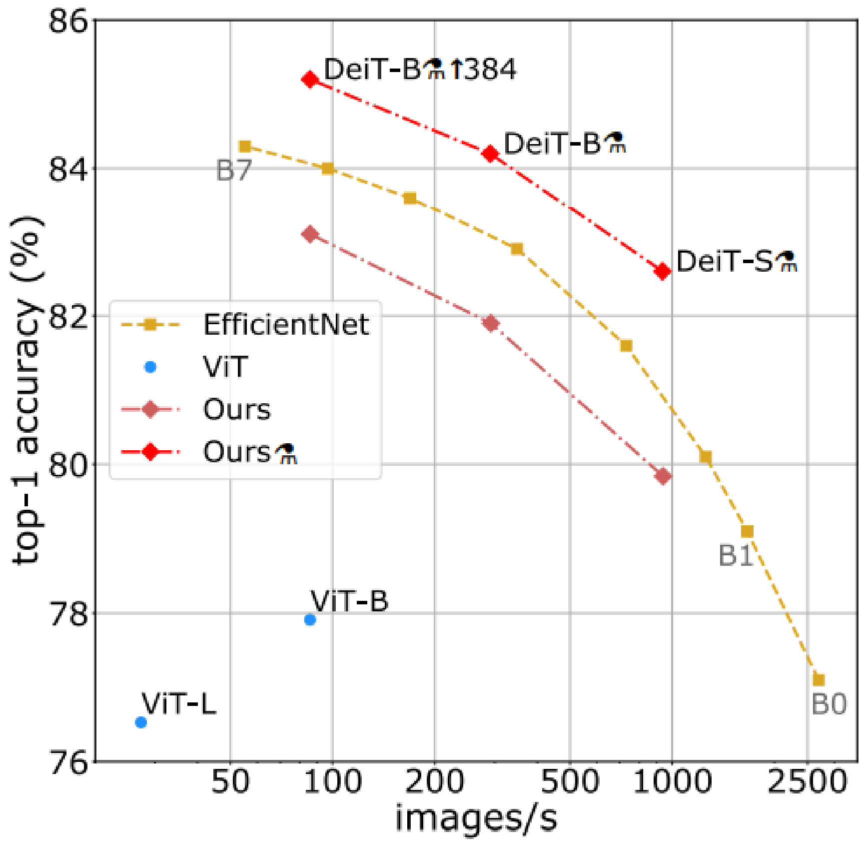
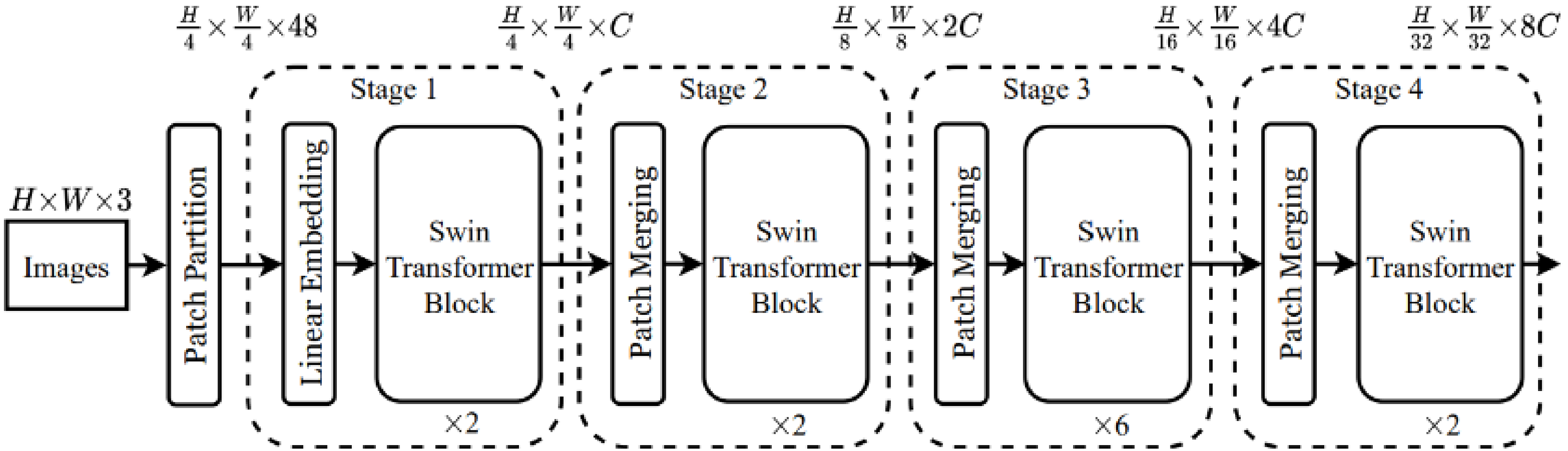


Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to VIT-B, but the training is more adapted to a data-starving regime. It is learned in a few days on one machine. The symbol ⚗ refers to models trained with our transformer-specific distillation. See Table 5 for details and more models.

Swin: Sliding window transformer) (by Microsoft)

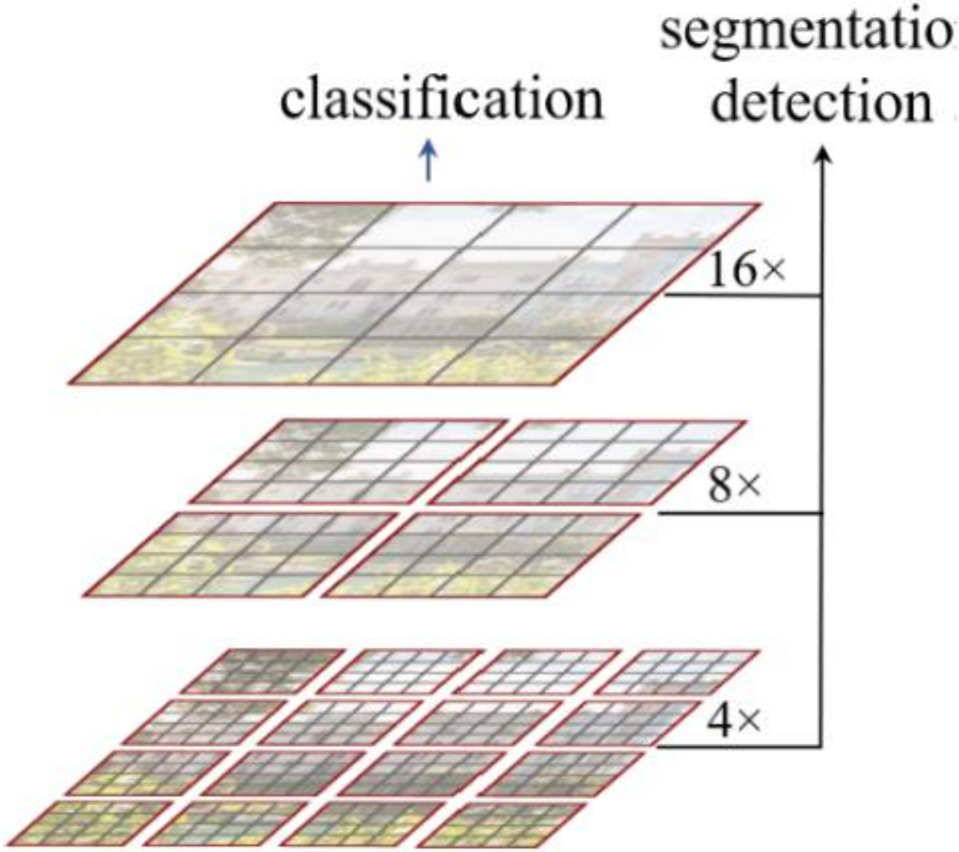


(a) Architecture (b)

Swin key ideas

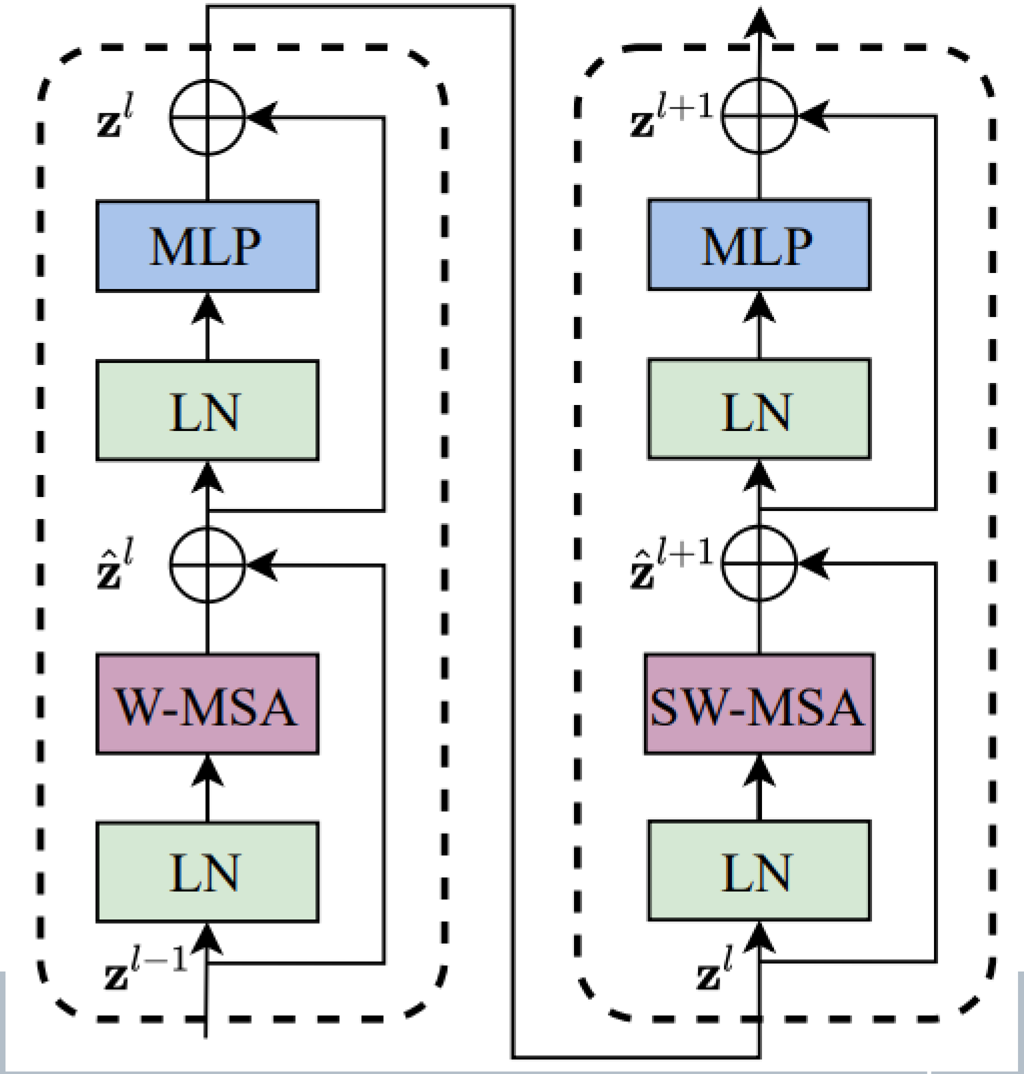
- Patch merging: Similar to image pooling
- Attention is restricted to neighbouring windows
- Replaces learned positional encodings with relative positional encodings

Patch merging

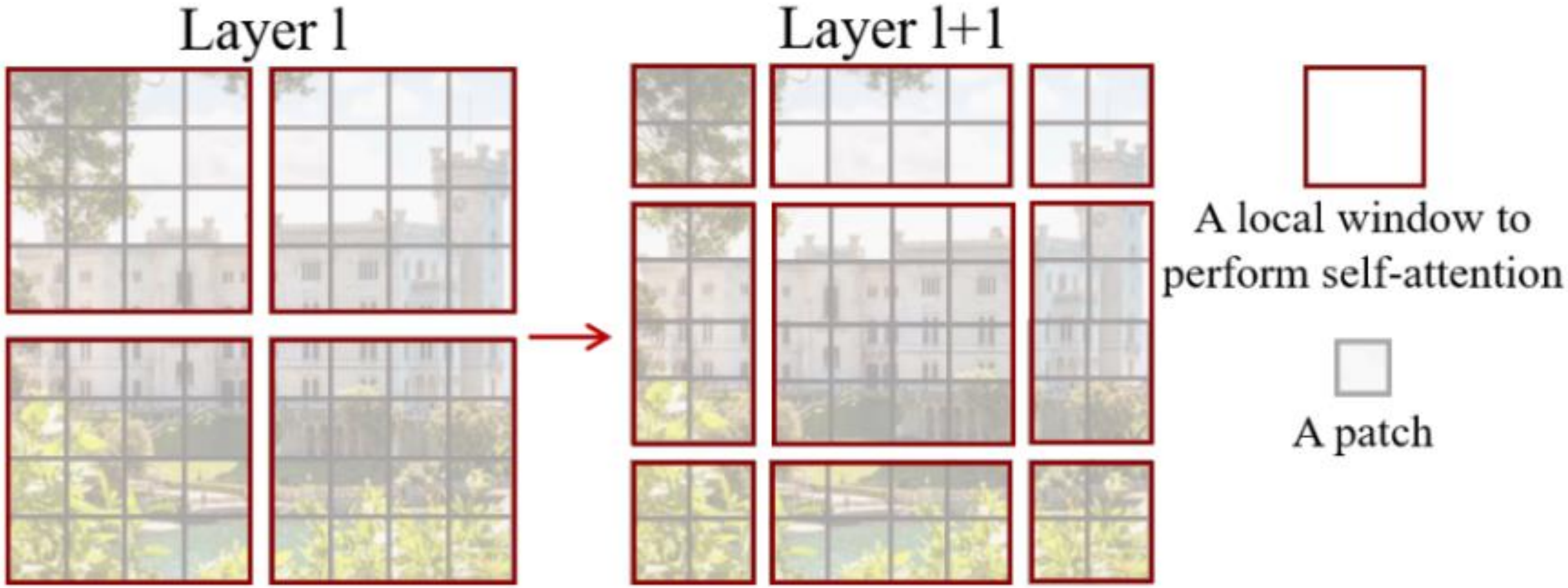


Swin transformer block

- W-MSA: windowed multi-head self-attention
- SW-MSA: Shifted windowed multi-head self-attention



Swin transformer block



- For layer l, self-attention of 4x4 tokens, and 2x2 blocks (easily done with einops)
- For layer l+1, 3x3 blocks of different sizes!

Relative positional encoding

› Positional encodings are relative between query-key vectors in each window (not absolute position)

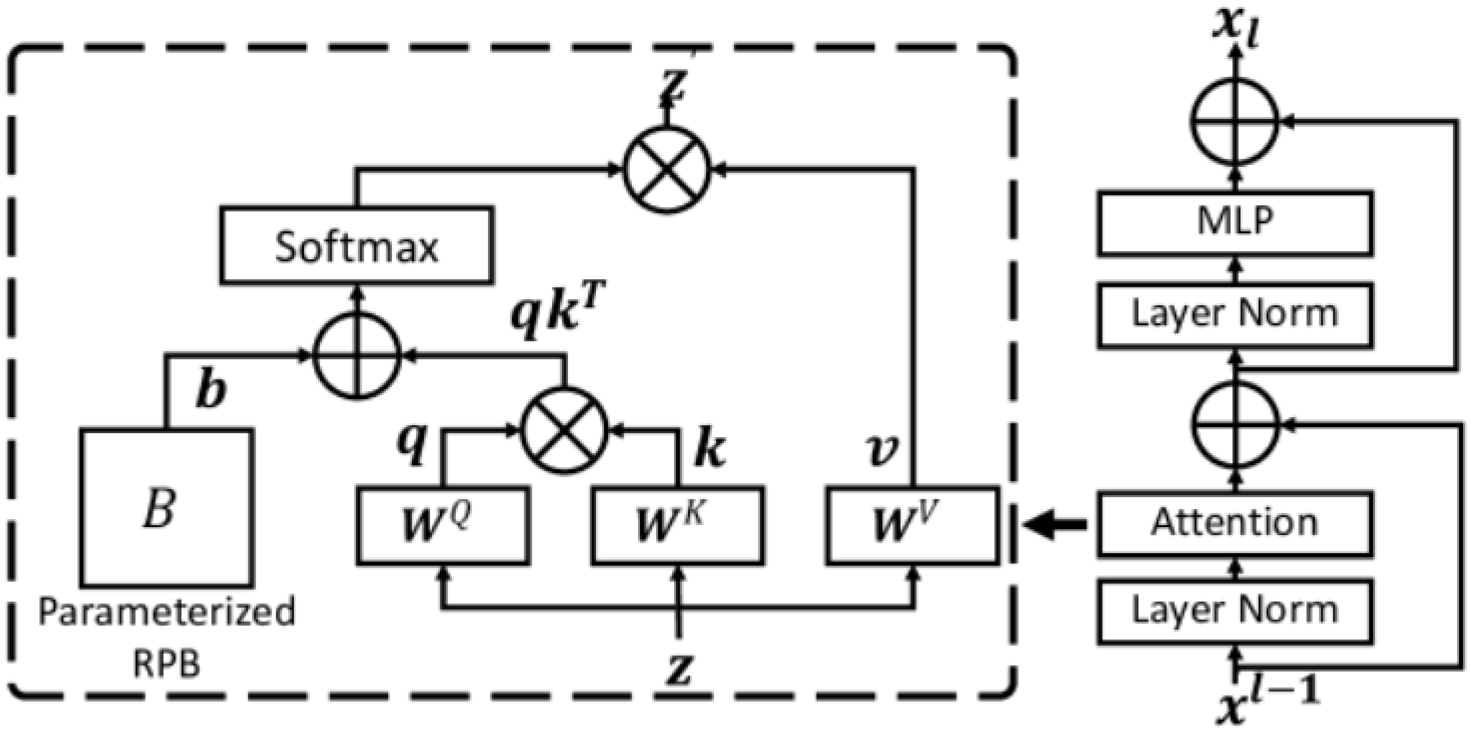
x axis

1	2	3
4	5	6
7	8	9

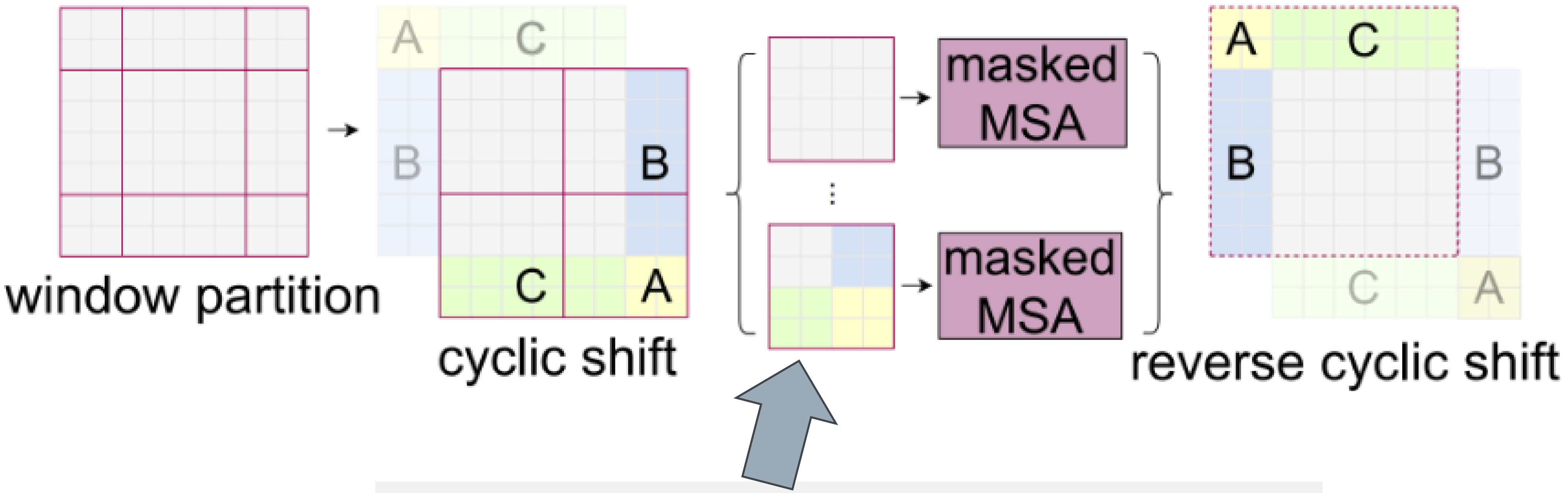
x axis matrix

	1	2	3	4	5	6	7	8	9
1	0	0	0	-1	-1	-1	-2	-2	-2
2	0	0	0	-1	-1	-1	-2	-2	-2
3	0	0	0	-1	-1	-1	-2	-2	-2
4	1	1	1	0	0	0	-1	-1	-1
5	1	1	1	0	0	0	-1	-1	-1
6	1	1	1	0	0	0	-1	-1	-1
7	2	2	2	1	1	1	0	0	0
8	2	2	2	1	1	1	0	0	0
9	2	2	2	1	1	1	0	0	0

Windowed self-attention with positional bias

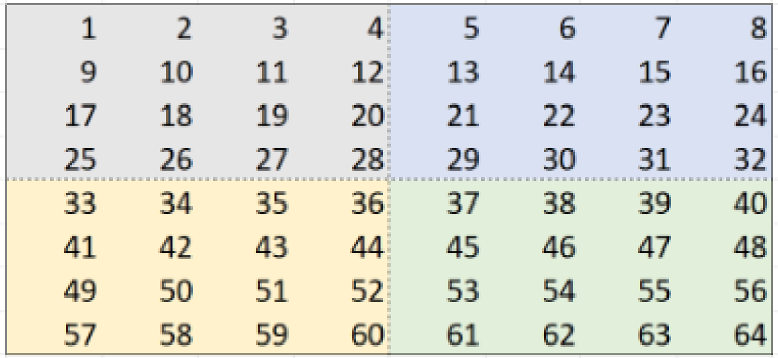


Efficient batch computation for shifted configuration

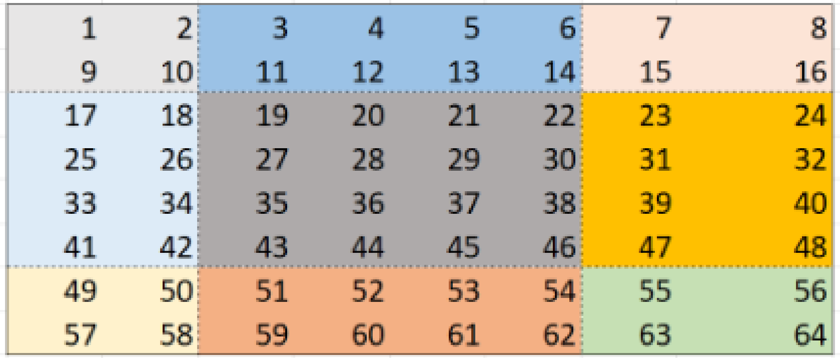


All have same size -> can be batched!
In the first one, all attend to all
In the border parches (last one) mask attention is used

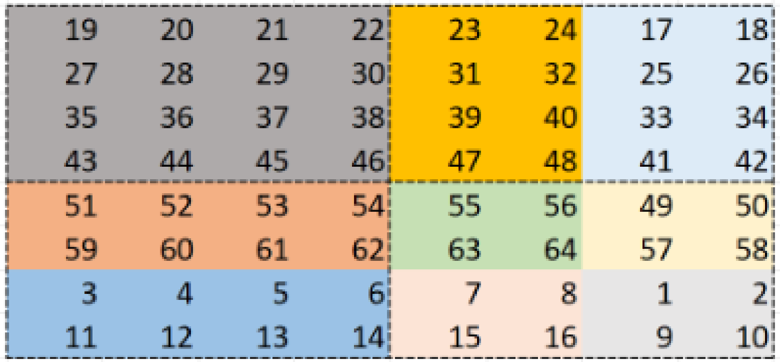
Efficient batch computation for shifted configuration



A: Attention in local windows



B: Attention in shifted windows (2,2)



C: Batched windows after cyclic shift

Efficient batch computation for shifted configuration

19	20	21	22	23	24	17	18
27	28	29	30	31	32	25	26
35	36	37	38	39	40	33	34
43	44	45	46	47	48	41	42
51	52	53	54	55	56	49	50
59	60	61	62	63	64	57	58
3	4	5	6	7	8	1	2
11	12	13	14	15	16	9	10

```
array([[0, 0, 0, 0, 1, 1, 2, 2],
       [0, 0, 0, 0, 1, 1, 2, 2],
       [0, 0, 0, 0, 1, 1, 2, 2],
       [0, 0, 0, 0, 1, 1, 2, 2],
       [3, 3, 3, 3, 4, 4, 5, 5],
       [3, 3, 3, 3, 4, 4, 5, 5],
       [6, 6, 6, 6, 7, 7, 8, 8],
       [6, 6, 6, 6, 7, 7, 8, 8]])
```

Attention mask

Efficient batch computation for shifted configuration

```
cnt = 0
m   = np.arange(1, 65).reshape(8,8)
for h in (slice(0, -4), slice(-4, -2), slice(-2, None)):
    for w in (slice(0, -4), slice(-4, -2), slice(-2, None)):
        m[h, w] = cnt
        cnt += 1
```

```
array([[0, 0, 0, 0, 1, 1, 2, 2],
       [0, 0, 0, 0, 1, 1, 2, 2],
       [0, 0, 0, 0, 1, 1, 2, 2],
       [0, 0, 0, 0, 1, 1, 2, 2],
       [3, 3, 3, 3, 4, 4, 5, 5],
       [3, 3, 3, 3, 4, 4, 5, 5],
       [6, 6, 6, 6, 7, 7, 8, 8],
       [6, 6, 6, 6, 7, 7, 8, 8]])
```

Effect of shifted window

	ImageNet		COCO		ADE20k
	Top-1	Top-5	AP^{box}	AP^{mask}	mIoU
Without shifting	80.2	95.1	47.7	41.5	43.3
With shifting	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Ablation study on the shifted window MSA approach, from Liu et al., 2021.

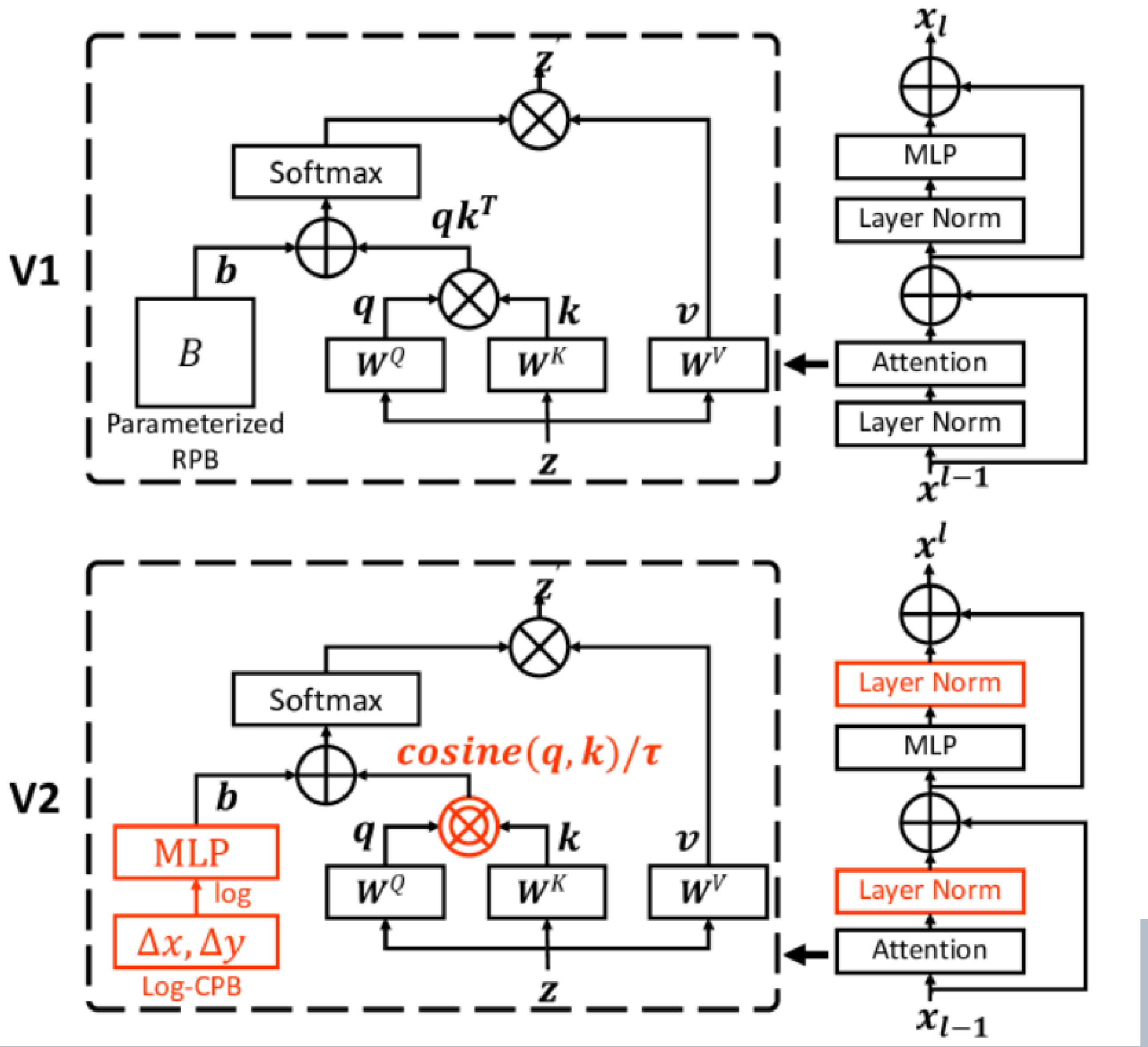
Swin performance

(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

Swin2: Scaling Up Capacity and Resolution

- Swin1 issues:
  - Stability during training for large models
  - How to upscale relative positional encodings when image resolution changes

Swin2 modifications



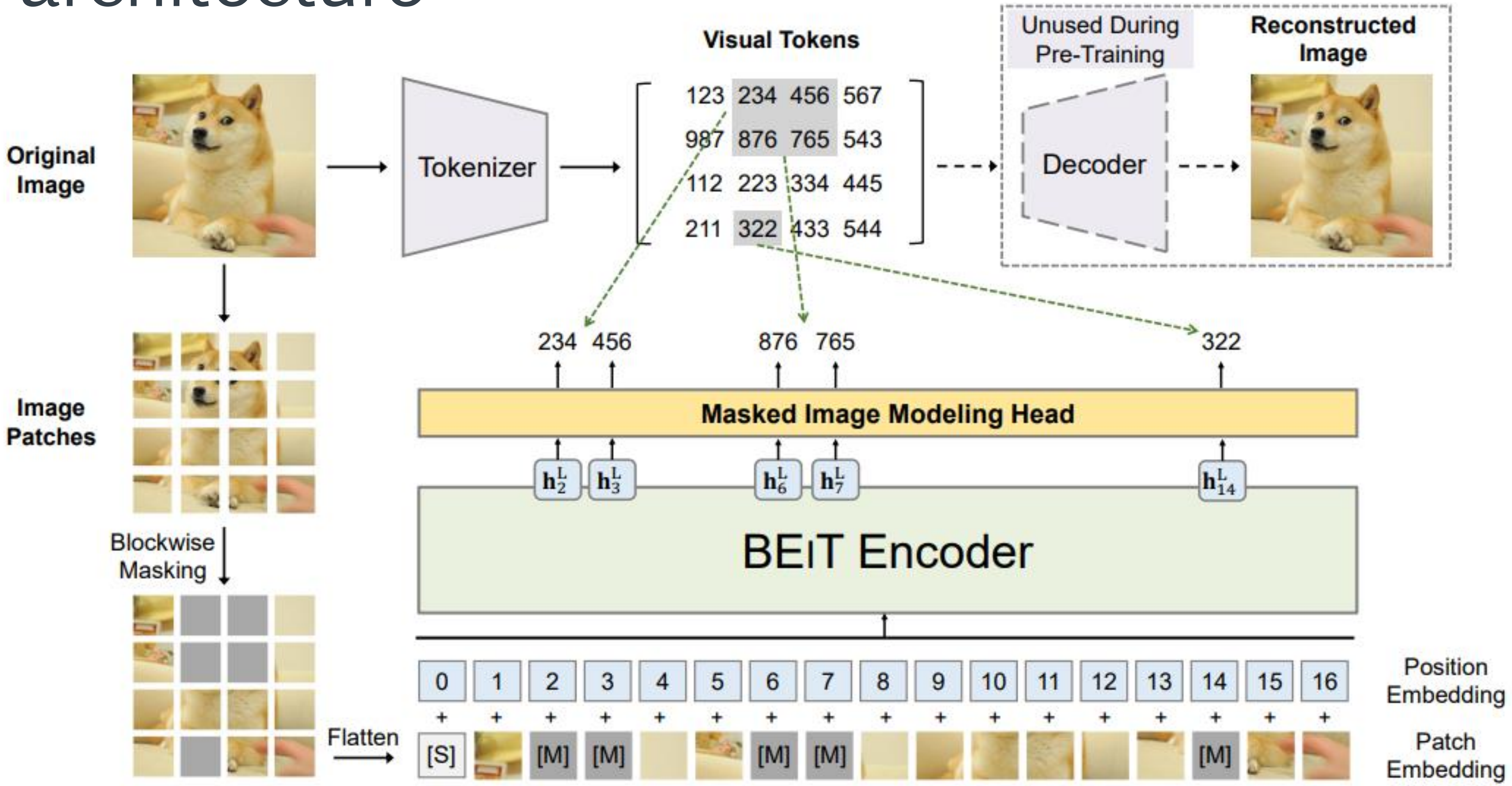
Swin2 performance

Method	param	pre-train images	pre-train length (#im)	pre-train im size	pre-train time	fine-tune im size	ImageNet-1K-V1 top-1 acc	ImaegNet-1K-V2 top-1 acc
SwinV1-B	88M	IN-22K-14M	1.3B	224^2	$< 30^\dagger$	384^2	86.4	76.58
SwinV1-L	197M	IN-22K-14M	1.3B	224^2	$< 10^\dagger$	384^2	87.3	77.46
ViT-G [80]	1.8B	JFT-3B	164B	224^2	> 30k	518^2	90.45	83.33
V-MoE [56]	14.7B*	JFT-3B	-	224^2	16.8k	518^2	90.35	-
CoAtNet-7 [17]	2.44B	JFT-3B	-	224^2	20.1k	512^2	**90.88**	-
SwinV2-B	88M	IN-22K-14M	1.3B	192^2	$< 30^\dagger$	384^2	87.1	78.08
SwinV2-L	197M	IN-22K-14M	1.3B	192^2	$< 20^\dagger$	384^2	87.7	78.31
SwinV2-G	3.0B	IN-22K-ext-70M	3.5B	192^2	$< 0.5\text{k}^\dagger$	640^2	90.17	**84.00**

BeiT: BERT Pre-Training of Image Transformers

- Key idea: Use masked image modeling task to pretrain vision Transformers. (similar to BERT)
- It is difficult to predict patch (too complex), instead they predict image tokens (after patch tokenization)
- Tokens are borrowed from: Zero-Shot Text-to-Image Generation
- Uses a ViT transformer as backbone

BeiT architecture

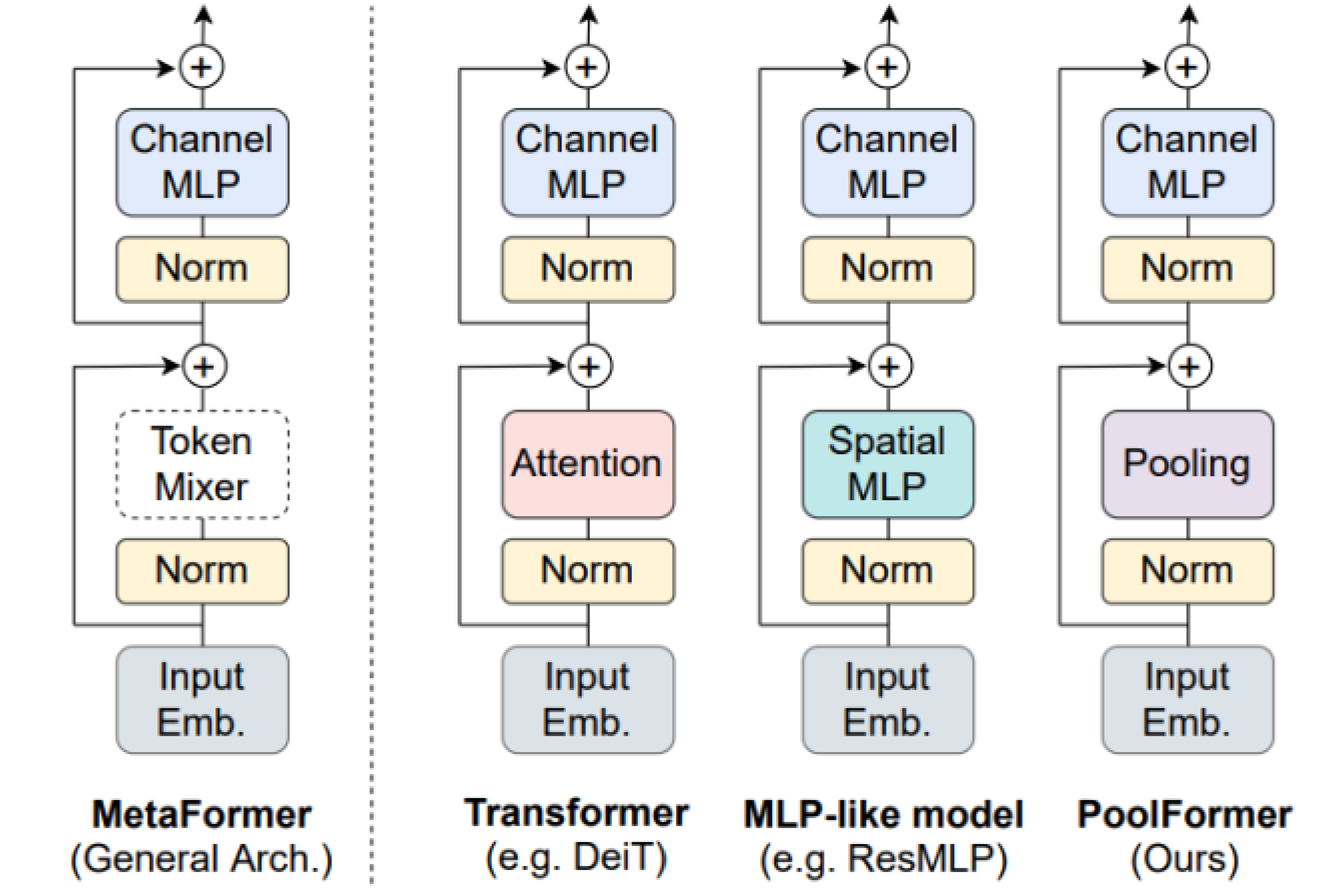


BeiT performance

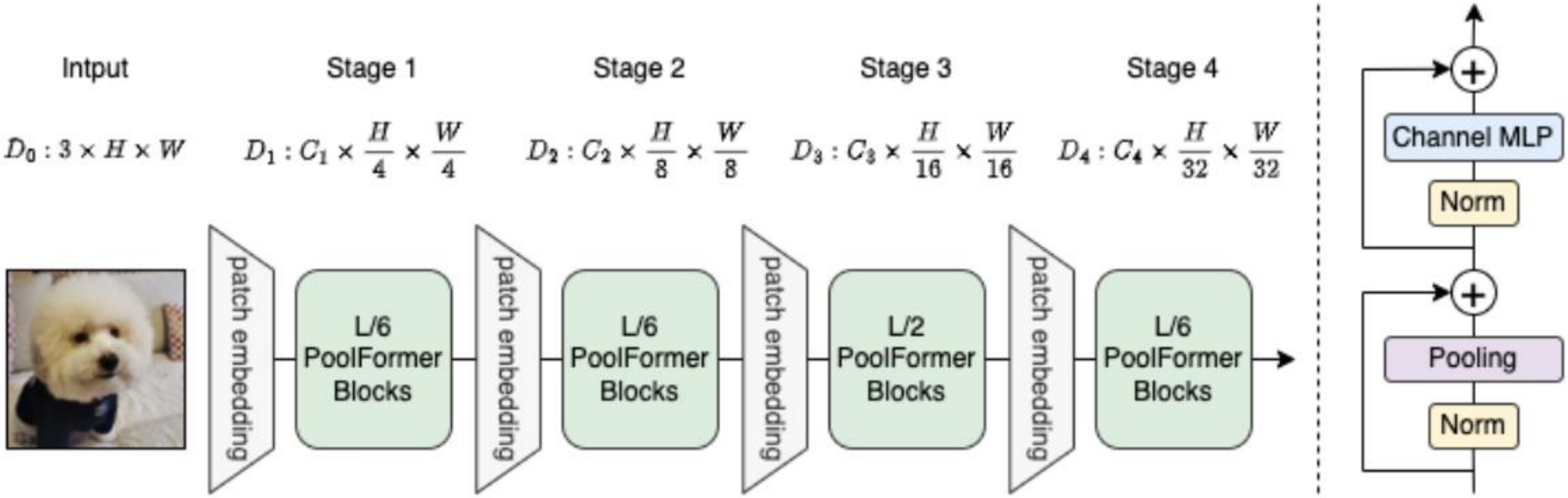
Models	Model Size	Resolution	ImageNet
Training from scratch (i.e., random initialization)			
ViT_{384}-B [DBK+20]	86M	384^2	77.9
ViT_{384}-L [DBK+20]	307M	384^2	76.5
DeiT-B [TCD+20]	86M	224^2	81.8
$DeiT_{384}$-B [TCD+20]	86M	384^2	83.1
Supervised Pre-Training on ImageNet-22K (using labeled data)			
ViT_{384}-B [DBK+20]	86M	384^2	84.0
ViT_{384}-L [DBK+20]	307M	384^2	85.2
Self-Supervised Pre-Training on ImageNet-1K (without labeled data)			
iGPT-1.36B† [CRC+20]	1.36B	224^2	66.5
ViT_{384}-B-JFT300M‡ [DBK+20]	86M	384^2	79.9
MoCo v3-B [CXH21]	86M	224^2	83.2
MoCo v3-L [CXH21]	307M	224^2	84.1
DINO-B [CTM+21]	86M	224^2	82.8
BEIT-B (ours)	86M	224^2	83.2
$BEIT_{384}$-B (ours)	86M	384^2	84.6
BEIT-L (ours)	307M	224^2	85.2
$BEIT_{384}$-L (ours)	307M	384^2	**86.3**

PoolFormer: MetaFormer Is Actually What You Need for Vision (2022)

› Rethinks the architecture of the transformer layer



Poolformer architecture



(a) Overall framework with L PoolFormer blocks

(b) PoolFormer block

Pooling for Poolformer

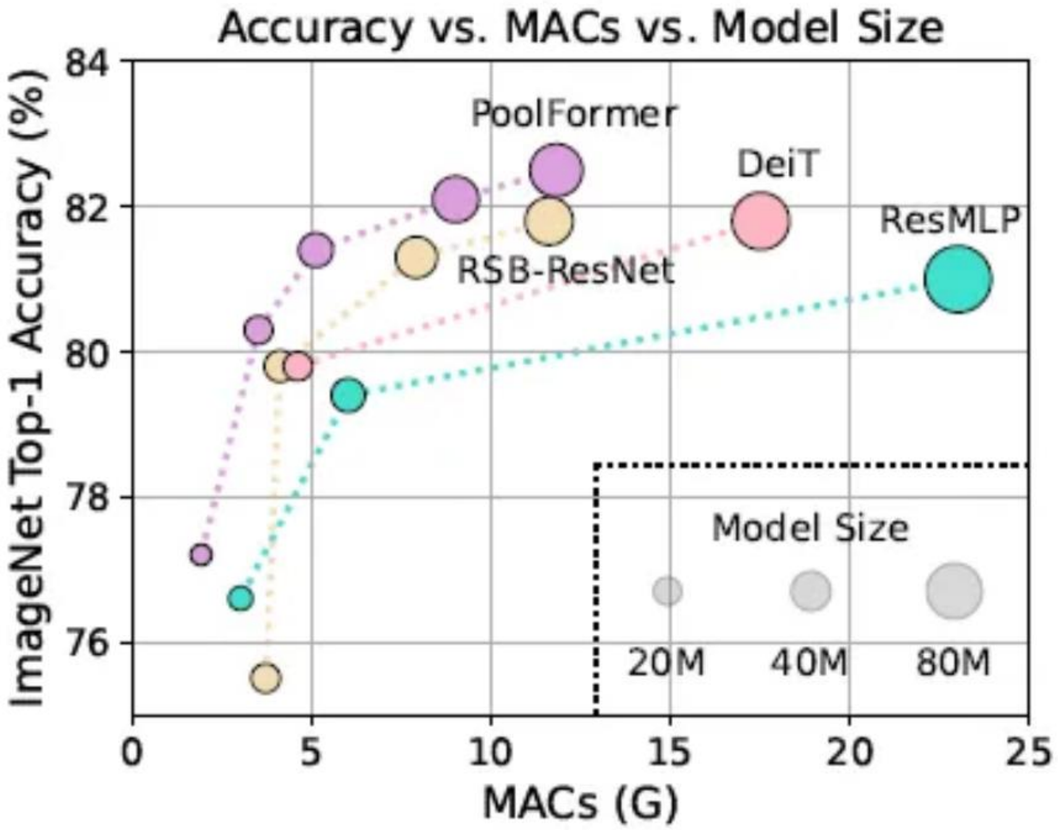
Algorithm 1 Pooling for PoolFormer, PyTorch-like Code

```
import torch.nn as nn

class Pooling(nn.Module):
   def __init__(self, pool_size=3):
      super().__init__()
      self.pool = nn.AvgPool2d(
         pool_size, stride=1,
         padding=pool_size//2,
         count_include_pad=False,
      )
   def forward(self, x):
      """
      [B, C, H, W] = x.shape
      Subtraction of the input itself is added
      since the block already has a
      residual connection.
      """
      return self.pool(x) - x
```

Pooling for PoolFormer, PyTorch-like Code.

Poolformer performance



Accuracy vs MACs

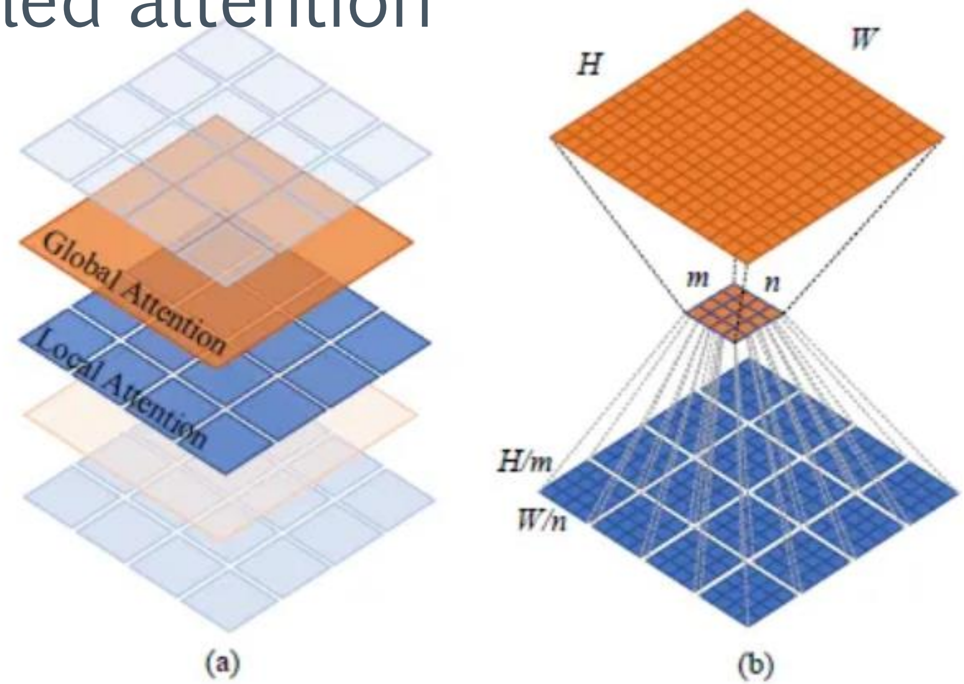
Poolformer performance

General Arch.	Token Mixer	Outcome Model	Image Size	Params (M)	MACs (G)	Top-1 (%)
Convolutional Neural Netowrks	—	RSB-ResNet-18 [57]	224	12	1.8	70.6
		RSB-ResNet-34 [57]	224	22	3.7	75.5
		RSB-ResNet-50 [57]	224	26	4.1	79.8
		RSB-ResNet-101 [57]	224	45	7.9	81.3
		RSB-ResNet-152 [57]	224	60	11.6	81.8
MetaFormer	Attention	ViT-B/16* [17]	224	86	17.6	79.7
		ViT-L/16* [17]	224	307	63.6	76.1
		DeiT-S [51]	224	22	4.6	79.8
		DeiT-B [51]	224	86	17.5	81.8
		PVT-Tiny [55]	224	13	1.9	75.1
		PVT-Small [55]	224	25	3.8	79.8
		PVT-Medium [55]	224	44	6.7	81.2
		PVT-Large [55]	224	61	9.8	81.7
	Spatial MLP	MLP-Mixer-B/16 [49]	224	59	12.7	76.4
		ResMLP-S12 [50]	224	15	3.0	76.6
		ResMLP-S24 [50]	224	30	6.0	79.4
		ResMLP-B24 [50]	224	116	23.0	81.0
		Swin-Mixer-T/D24 [35]	256	20	4.0	79.4
		Swin-Mixer-T/D6 [35]	256	23	4.0	79.7
		Swin-Mixer-B/D24 [35]	224	61	10.4	81.3
		gMLP-S [34]	224	20	4.5	79.6
		gMLP-B [34]	224	73	15.8	81.6
	Pooling	PoolFormer-S12	224	12	1.9	77.2
		PoolFormer-S24	224	21	3.5	80.3
		PoolFormer-S36	224	31	5.1	81.4
		PoolFormer-M36	224	56	9.0	82.1
		PoolFormer-M48	224	73	11.8	82.5

Performance of different types of models on ImageNet-1K classification.

Twins: Revisiting the Design of Spatial Attention in Vision Transformers

- Similar to swin tries to reduce the complexity of global attention
- Proposes global subsampled attention



(a) Twins-SVT interleaves locally-grouped attention (LSA) and global sub-sampled attention (GSA). (b) Schematic view of the locally-grouped attention (LSA) and global sub-sampled attention (GSA).

Twins performance

› Similar to swin

Twins-PCPVT-L(ours)	60.9	9.8	367	83.1 (+5.2)
Swin-B [4]	88	15.4	275	83.3
Twins-SVT-L (ours)	99.2	15.1	288	83.7 (+5.8)

MaxViT: Multi-Axis Vision Transformer

› Hibryd architecture

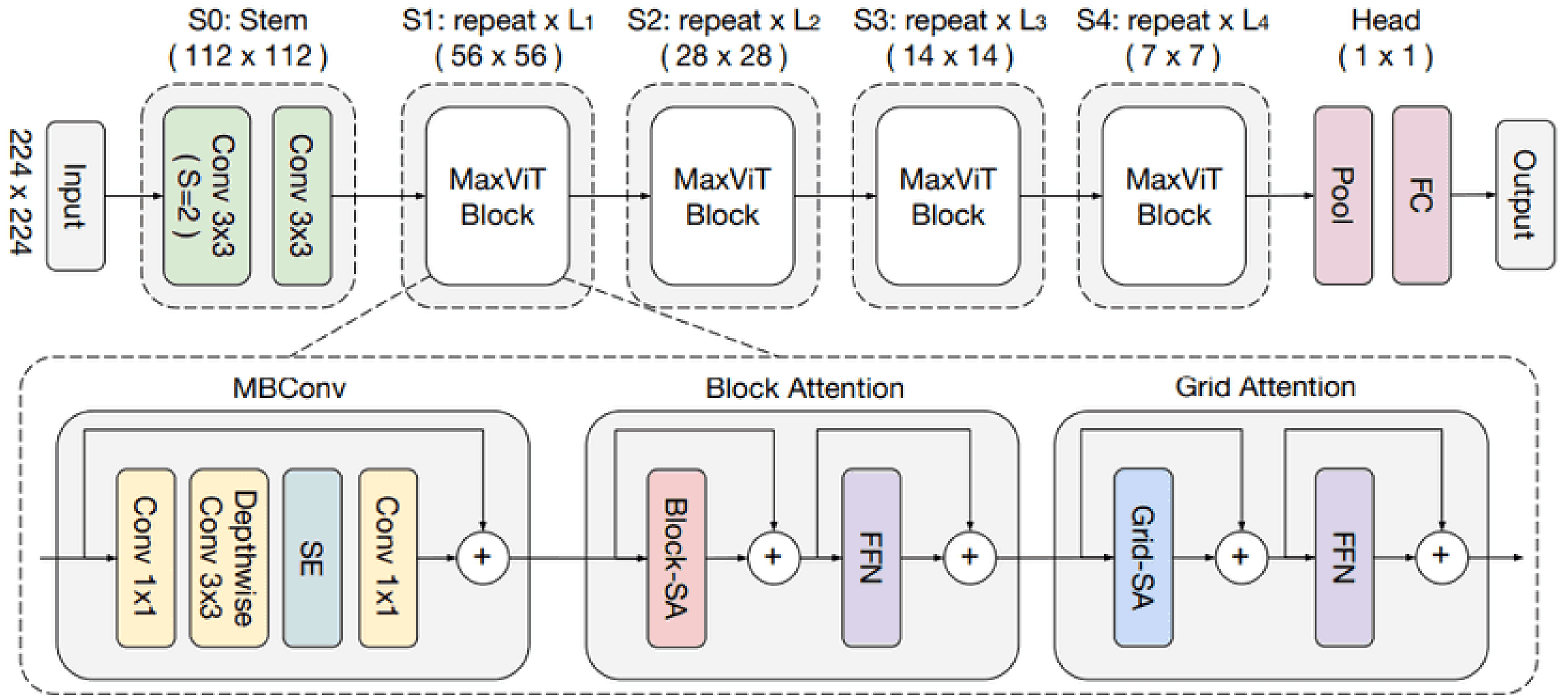


Fig. 2: **MaxViT architecture.** We follow a typical hierarchical design of ConvNet practices (e.g., ResNet) but instead build a new type of basic building block that unifies MBConv, block, and grid attention layers. Normalization and activation layers are omitted for simplicity.

MaxViT: Multiaxis attention

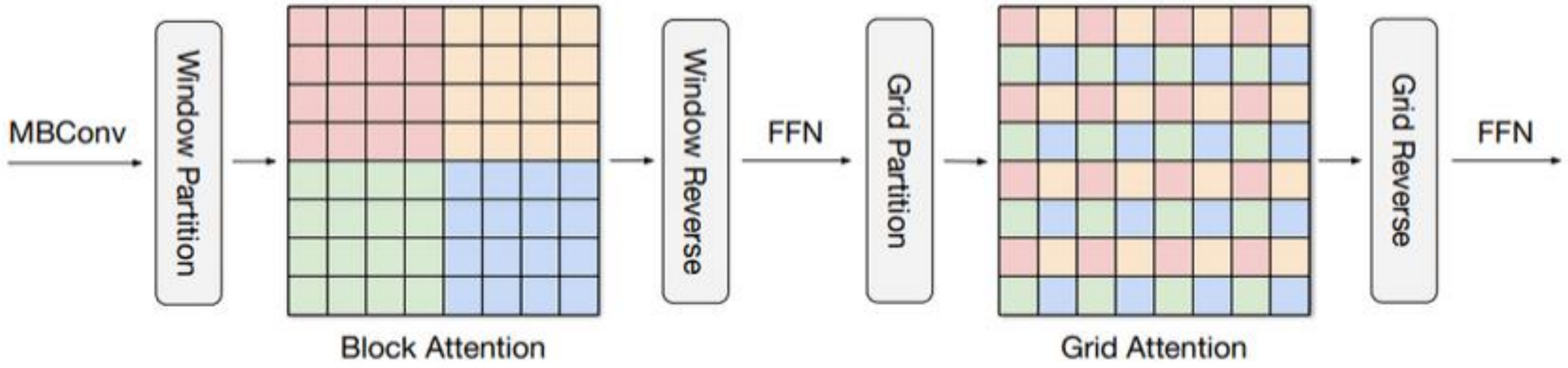
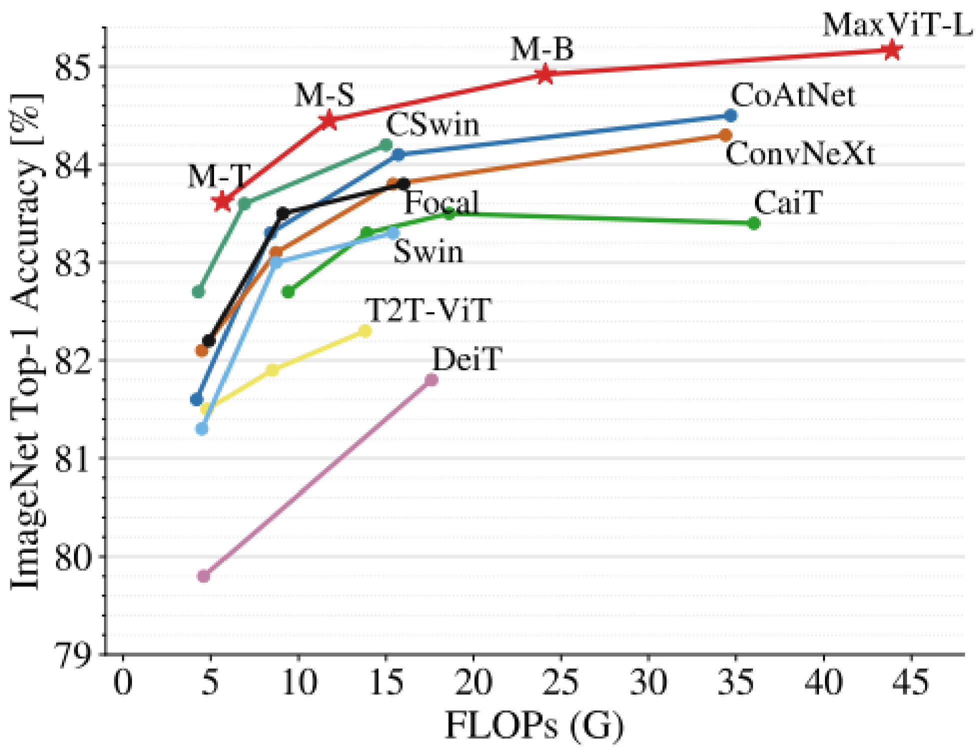


Fig. 3: **Multi-axis self-attention (Max-SA)** (best viewed in color). An illustration of the multi-axis approach for computing self-attention (window/grid size is 4×4). The block-attention module performs self-attention within windows, while the grid-attention module attends globally to pixels in a sparse, uniform grid overlaid on the entire 2D space, with both having linear complexity against input size, as we use fixed attention footage. The same colors are spatially mixed by the self-attention operation.

MaxViT Performance

› Training from scratch on Imagenet-1K



(a) Accuracy *vs.* FLOPs performance scaling curve under ImageNet-1K training setting at input resolution 224×224.

Other applications of vision transformers

- Object detection
- Image segmentation
- Image to text

DeTR

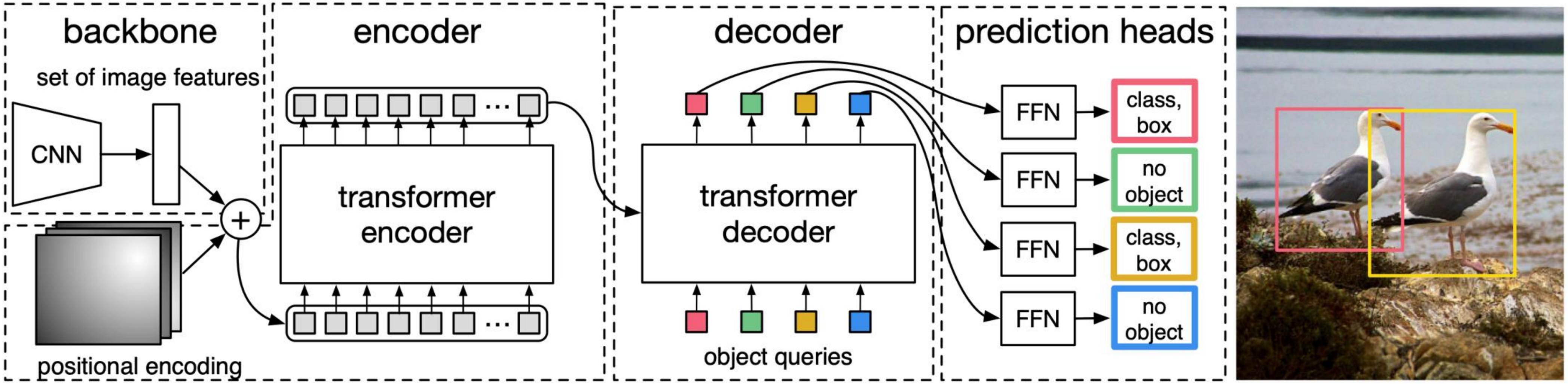
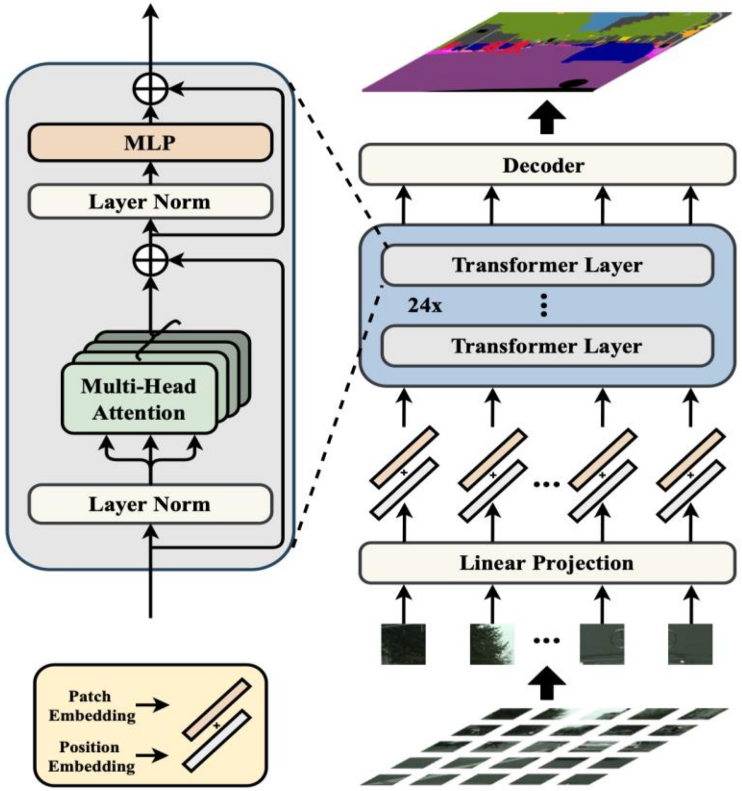
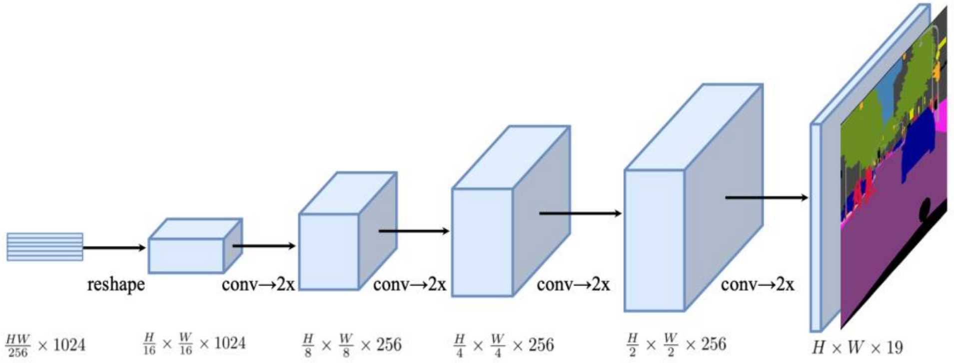

backbone
set of image features
CNN
positional encoding
encoder
transformer encoder
decoder
transformer decoder
object queries
prediction heads
FFN
class, box
no object
FFN
no object
FFN
class, box
FFN
no object

Image segmentation SeTR





Instance segmentation: Maskformer