DeepIntoDeep

Application of Transformers (ViT & Swin Transformer)

발표자: 전성후

Application of Transformers
(ViT & Swin Transformer)

전성후

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

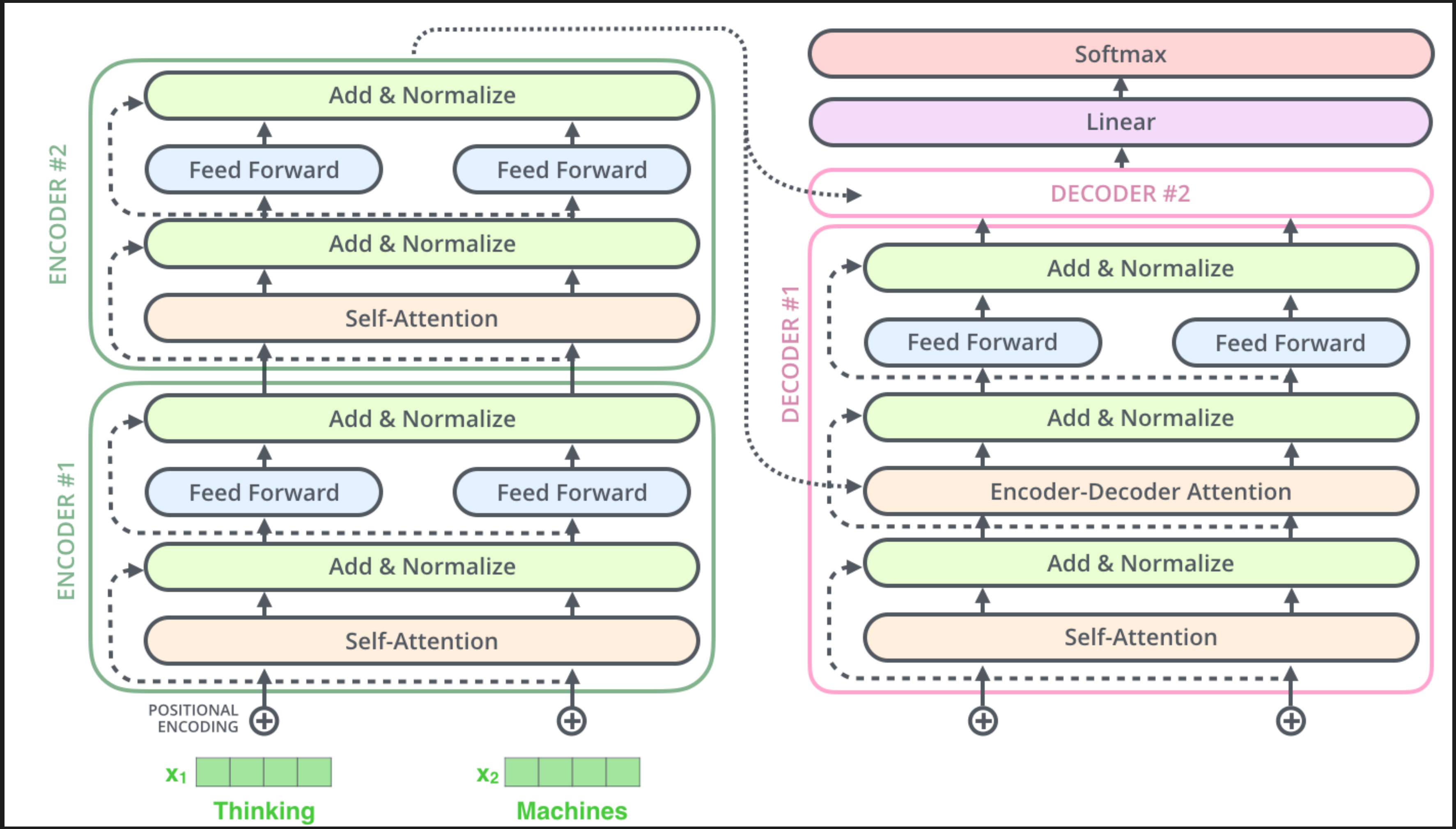
목차

- (Recap) Transformer
- Vision In Transformer (ViT)
  - How to use attention/Transformer in vision?
  - Architecture
  - Inductive Bias
- Swin Transformer
  - Relative Positional Encoding
- Is Attention All You Need?
  - MLP-Mixer
  - MetaFormer

과제2

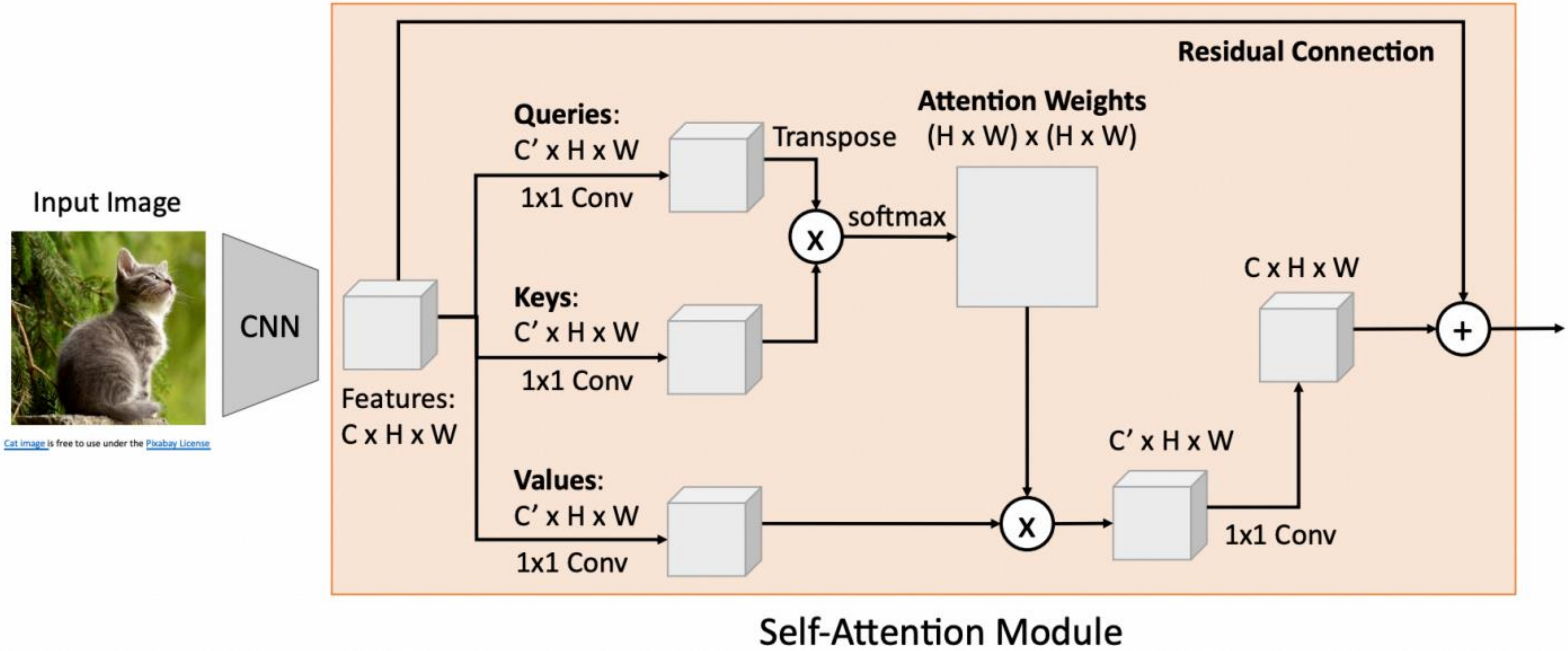
- LSTM 구현 + Transformer Encoder 구현하기
- Sentiment Classification

(Recap) Transformer



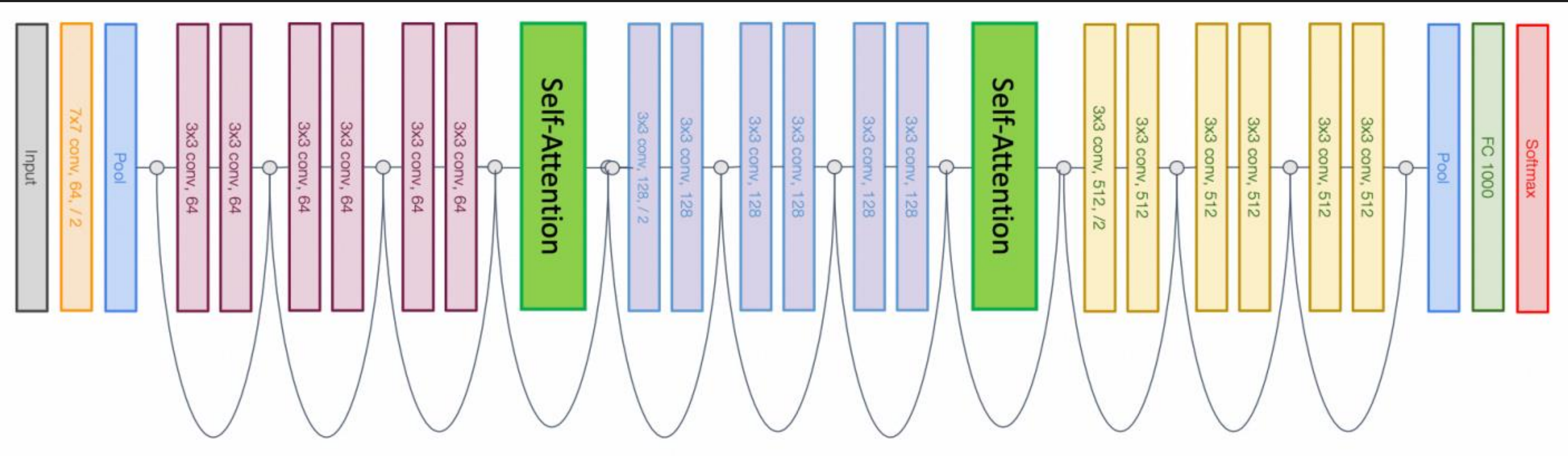
How to use Attention / Transformers for Vision?

Idea #1. Add Attention to CNNs



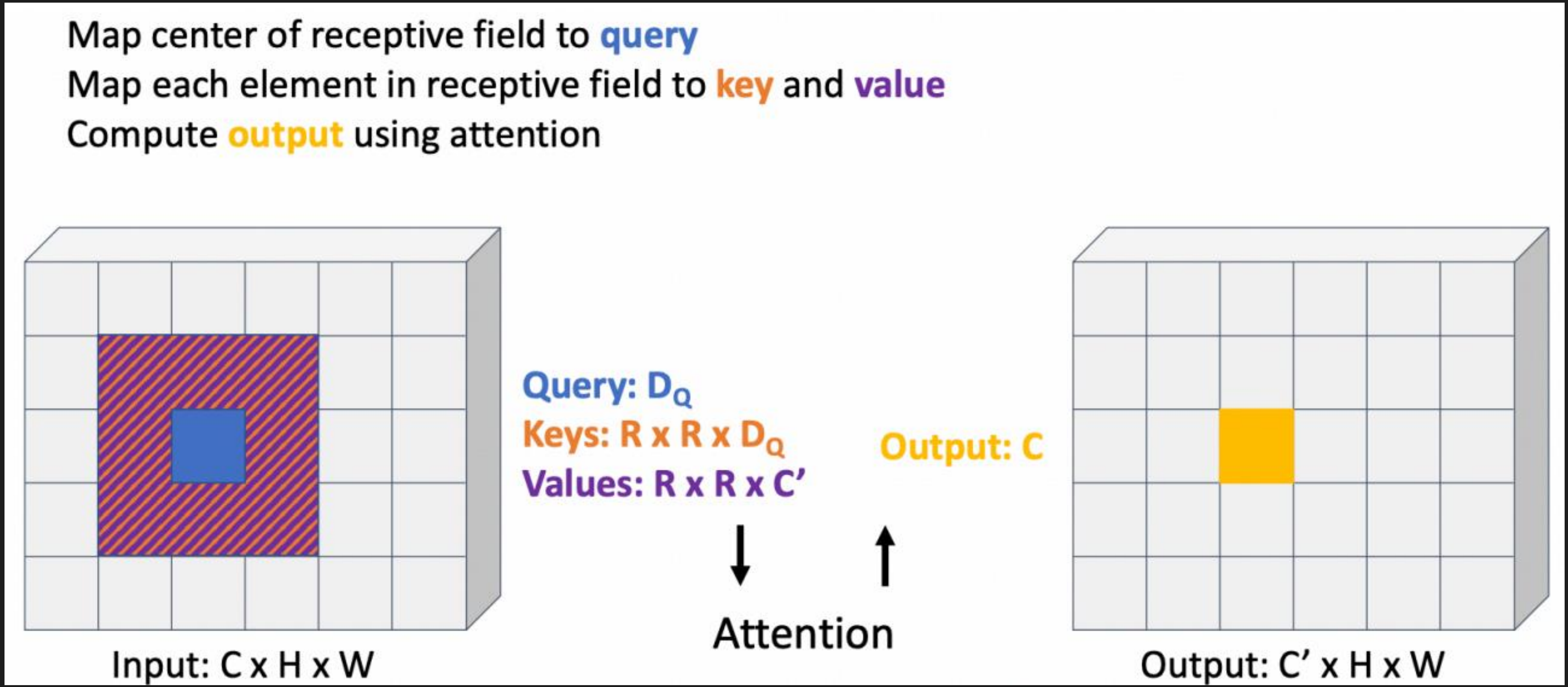
Idea #1. Add Attention to CNNs

- Quite used architecture nowadays
- But model is still a CNN; Can we replace convolution entirely?

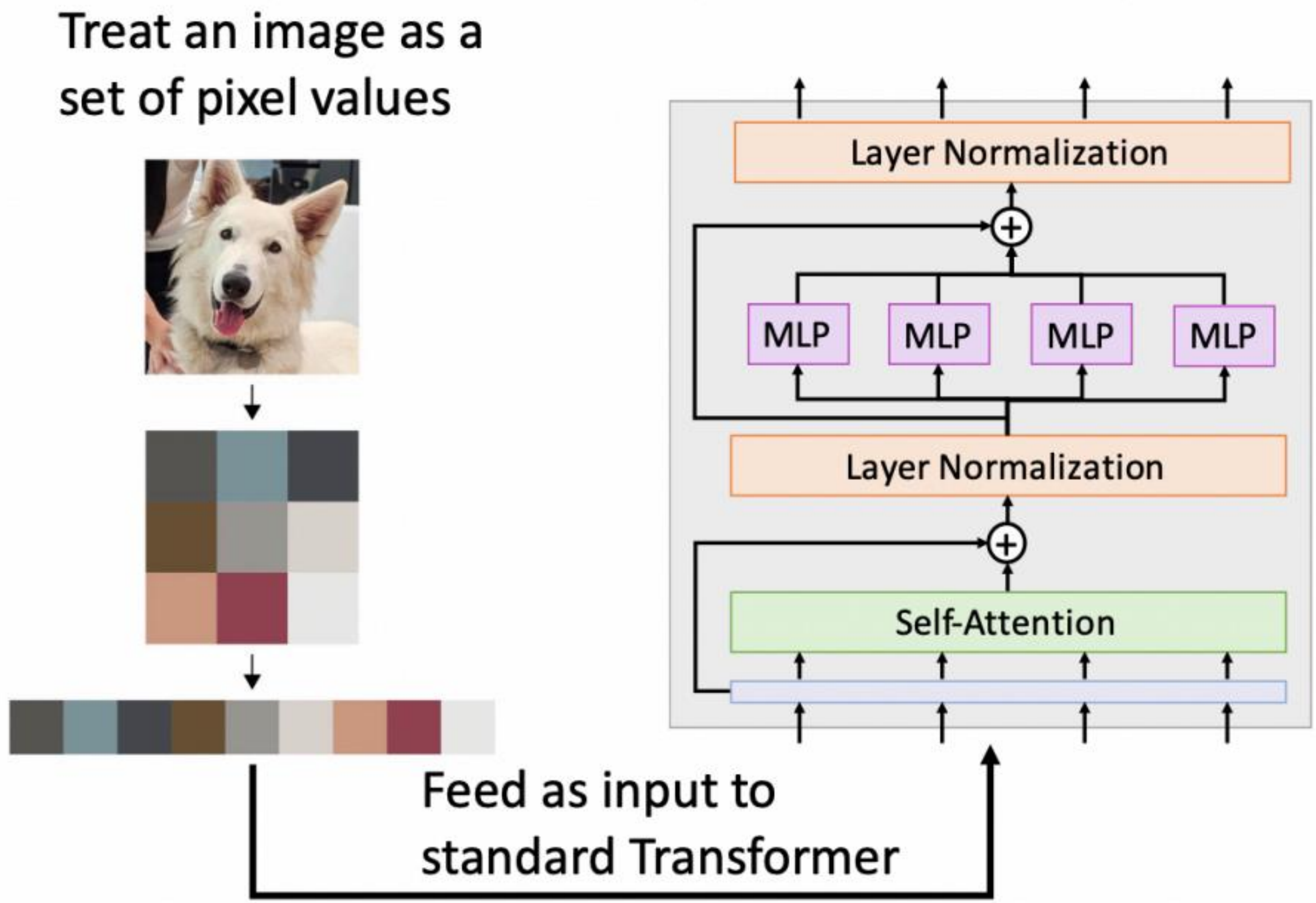


Idea #2. Replace Conv with "Local Attention"

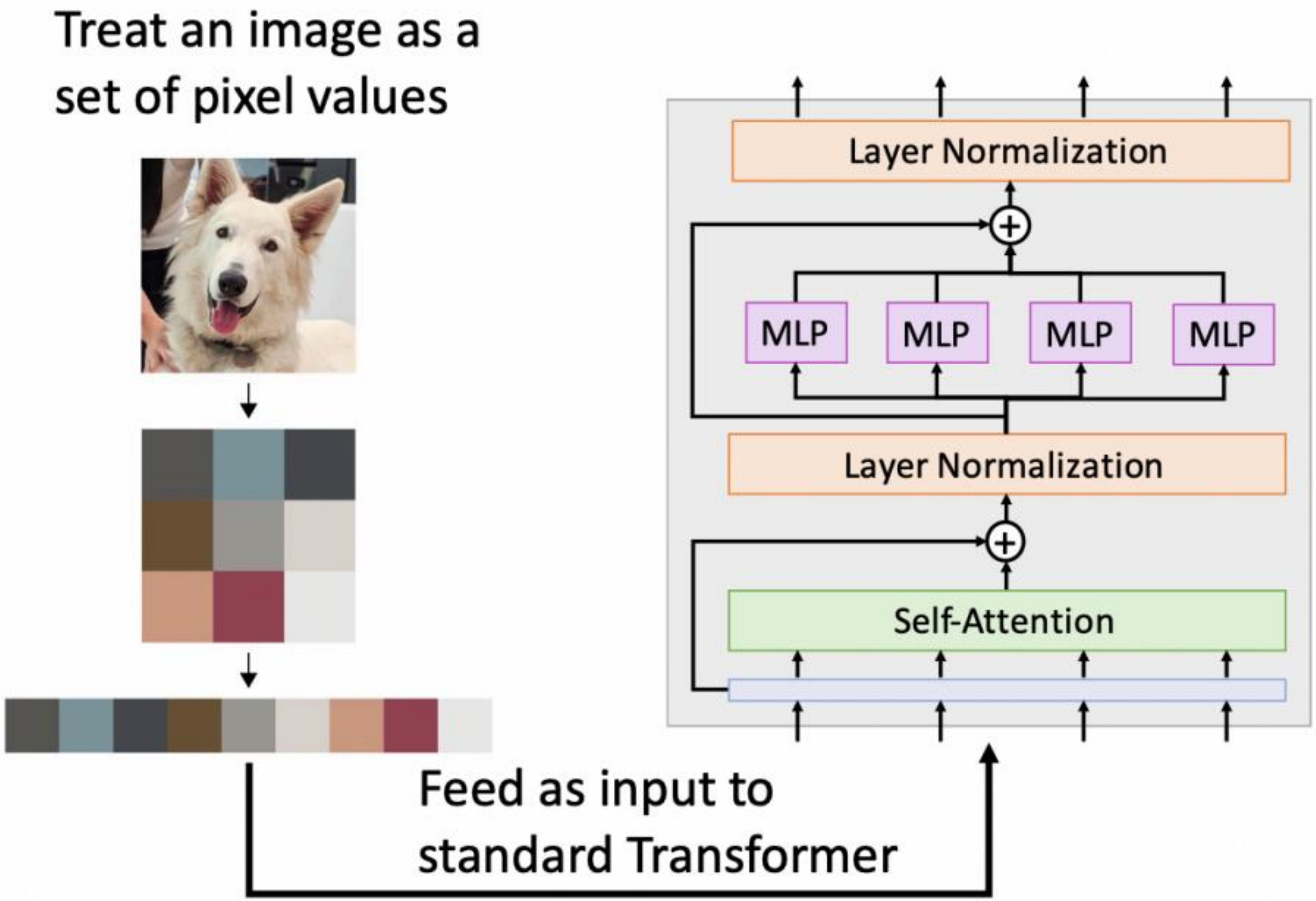
- Lots of tricky details; hard to implement and marginally better than CNNs



Idea #3. Standard Transformers on Pixels



Idea #3. Standard Transformers on Pixels



Problem: Memory use!

R x R image needs R^4 elements per attention matrix

R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example...

Idea #4. Standard Transformers on Patches

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

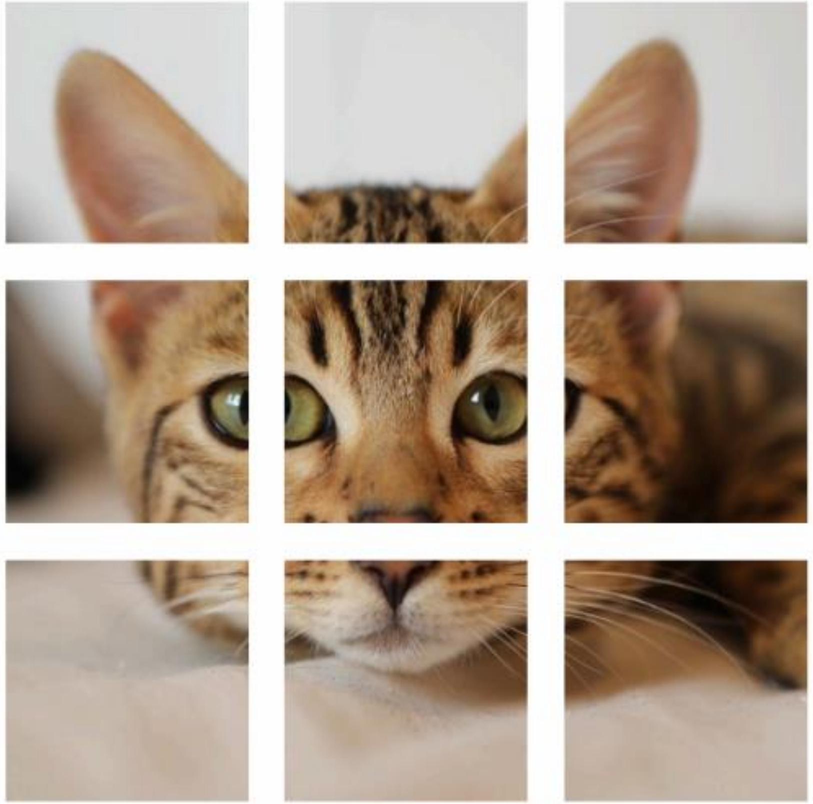
Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†

*equal technical contribution, †equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

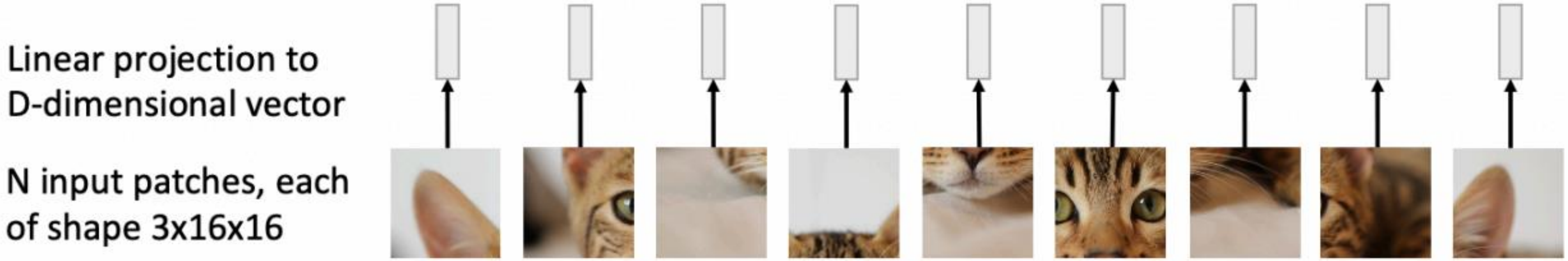
Idea #4. Standard Transformers on Patches



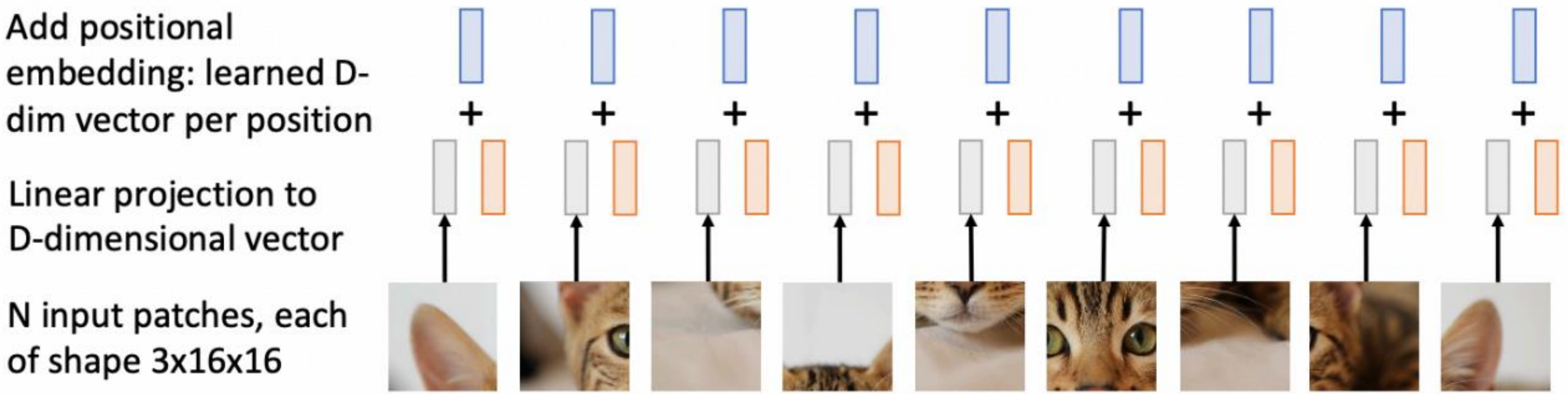
Idea #4. Standard Transformers on Patches

N input patches, each of shape 3x16x16

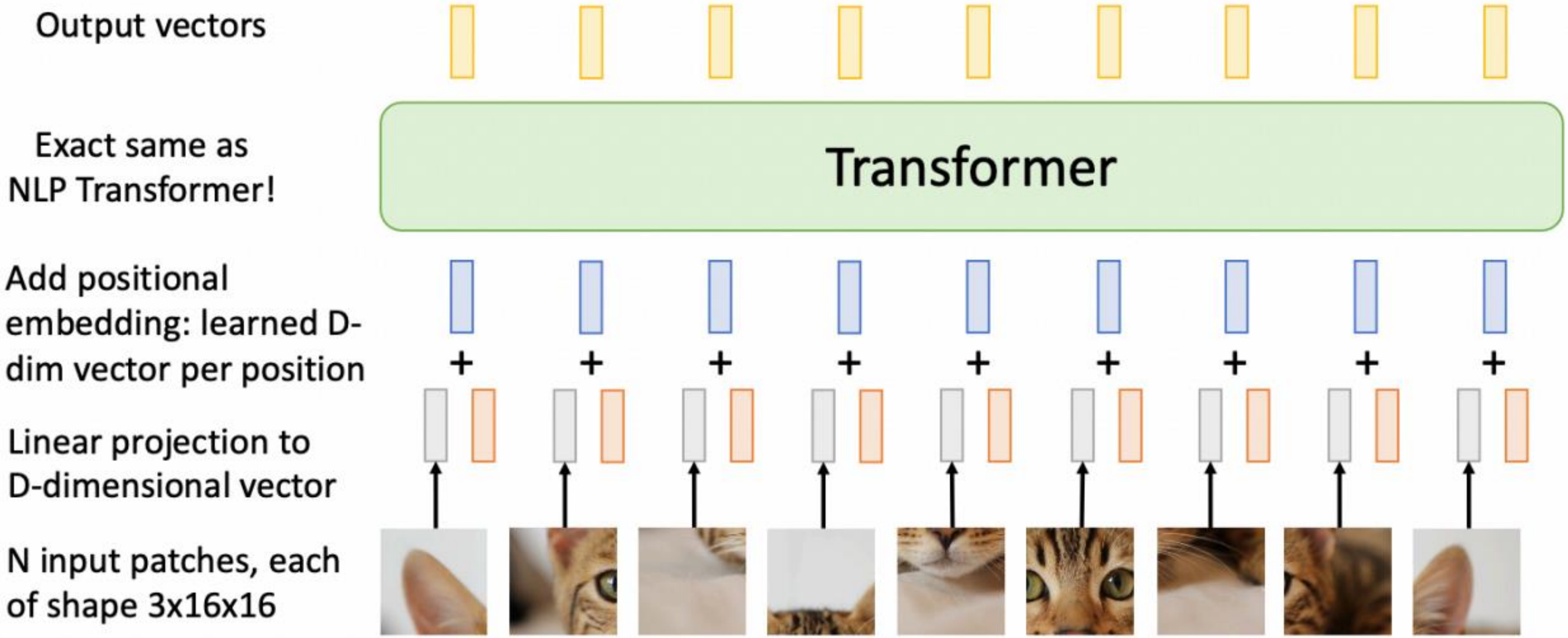
Idea #4. Standard Transformers on Patches



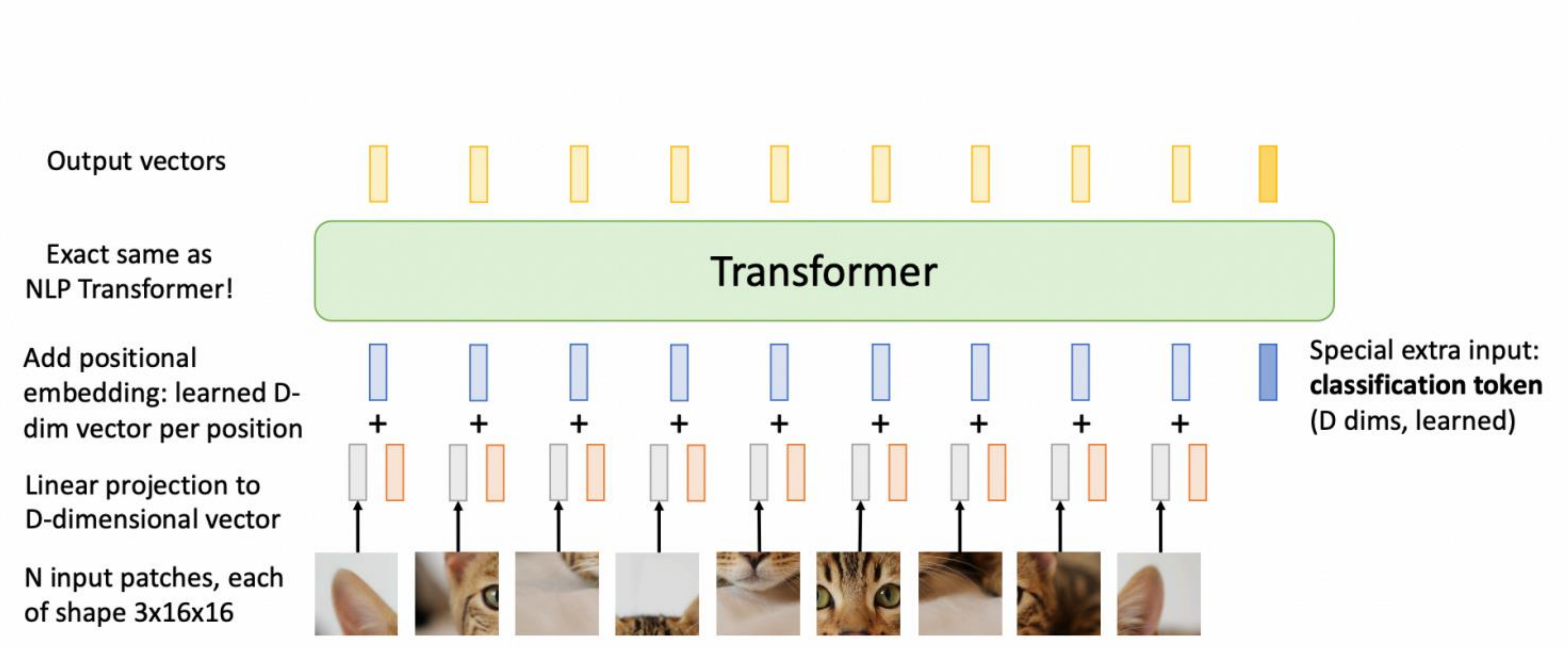
Idea #4. Standard Transformers on Patches



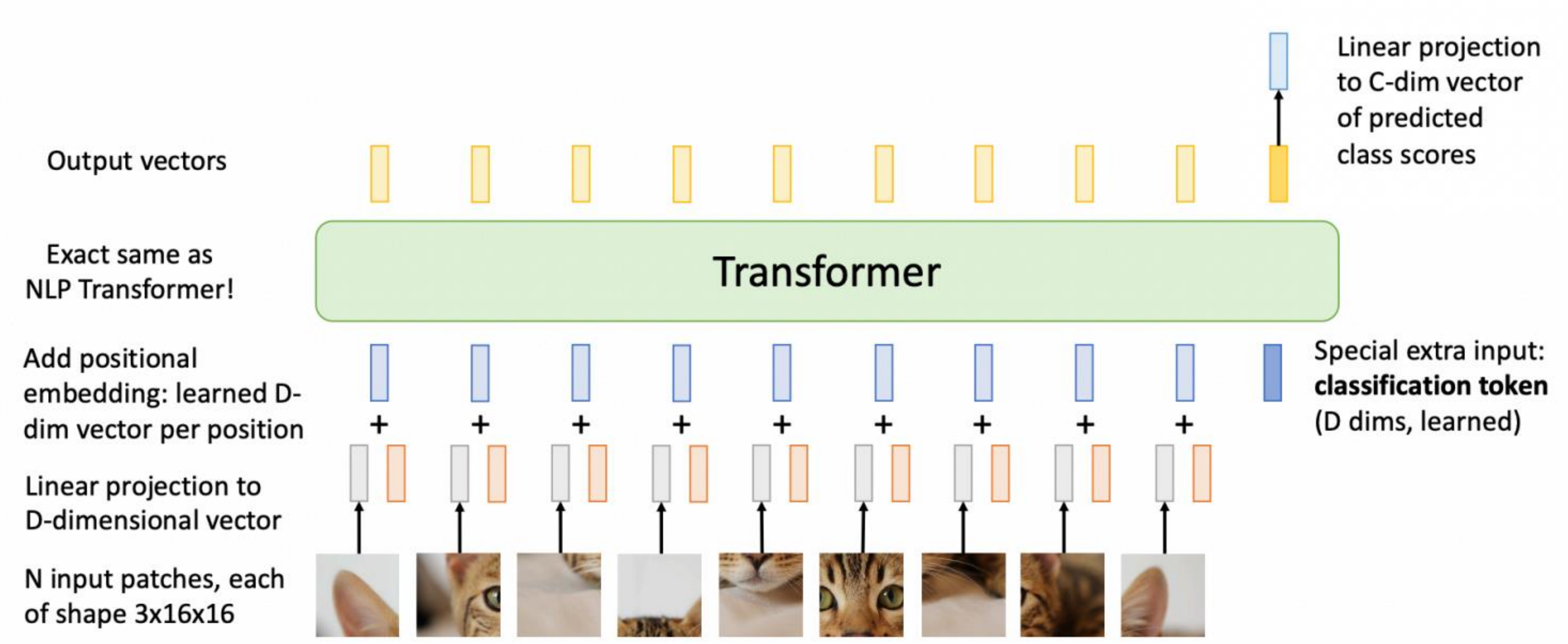
Idea #4. Standard Transformers on Patches



Idea #4. Standard Transformers on Patches

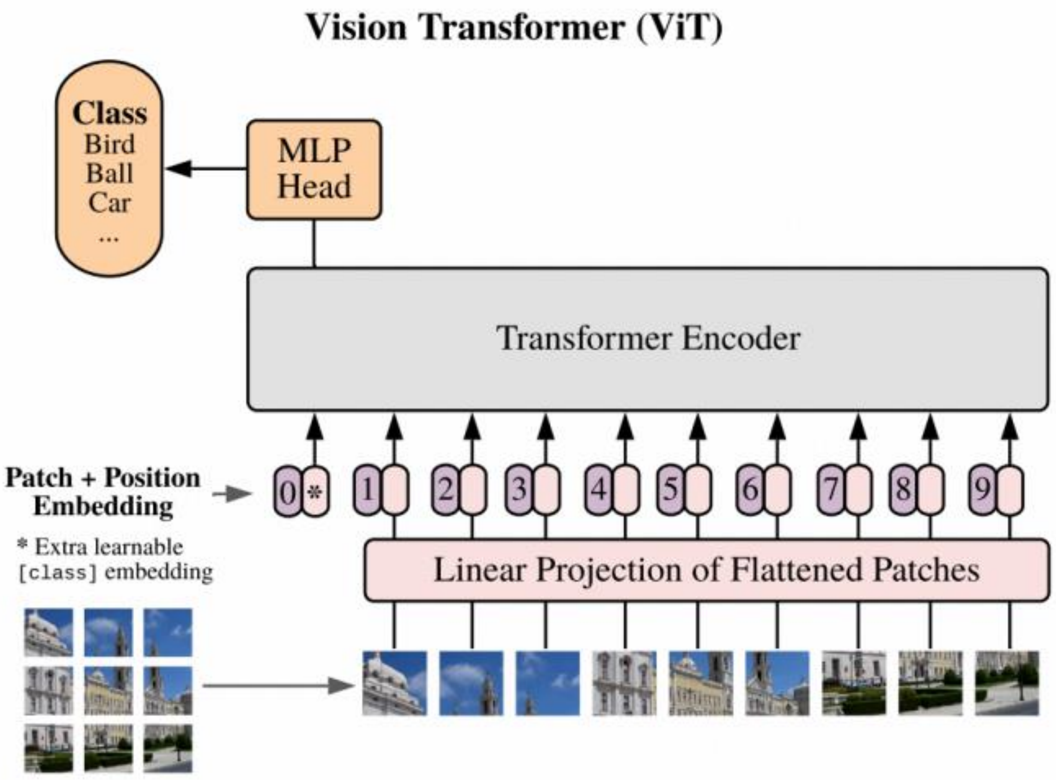


Idea #4. Standard Transformers on Patches



Idea #4. Standard Transformers on Patches

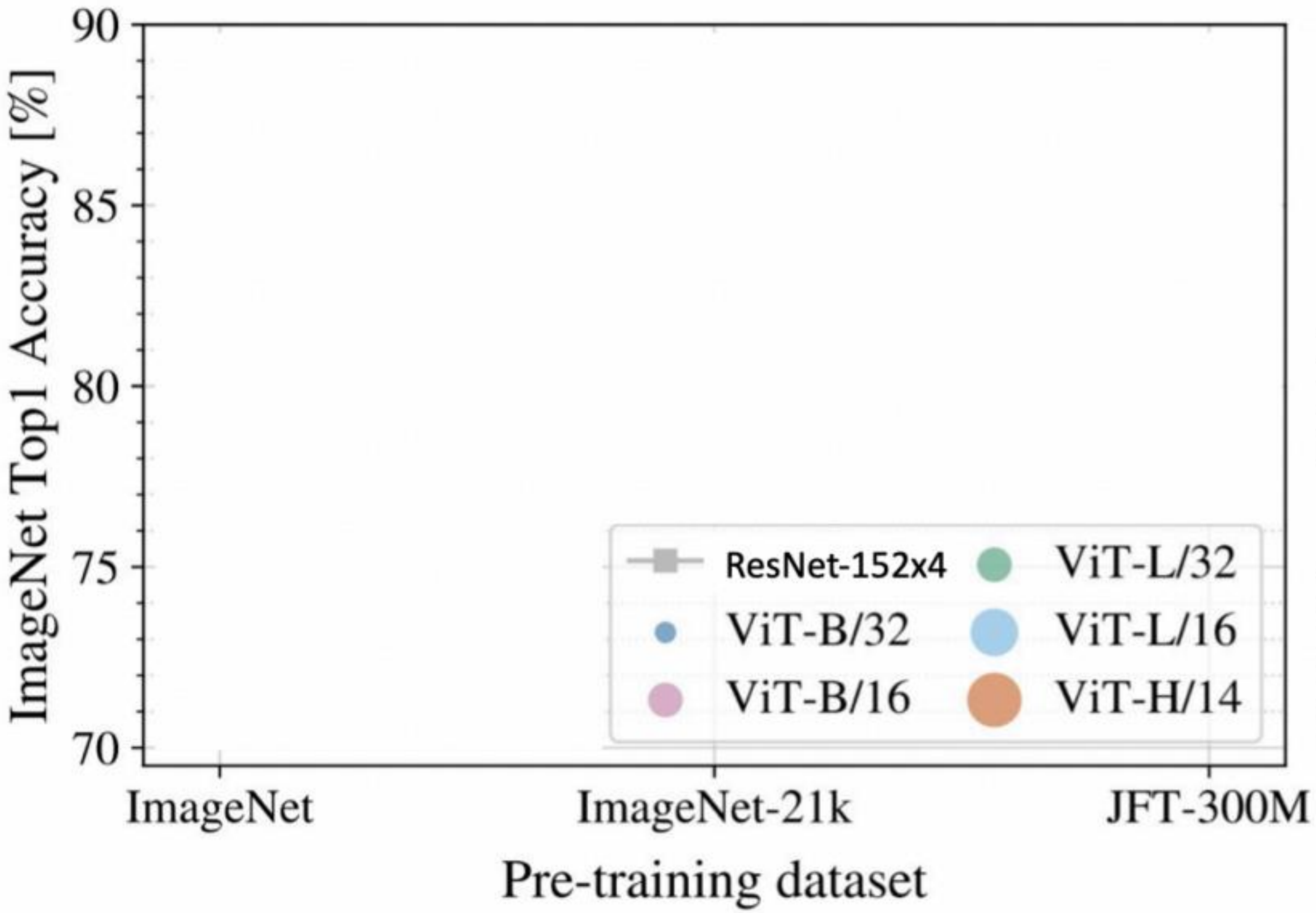
- [CLS] token



Idea #4. Standard Transformers on Patches

- Less computation resources
  - In practice: take 224x224 input image, divide into 14x14 grid of 16x16 pixel patches (or 16x16 grid of 14x14 patches)
  - With 48 layers, 16 heads per layer, all attention matrices take 112 MB (or 192MB)

ViT vs. ResNets



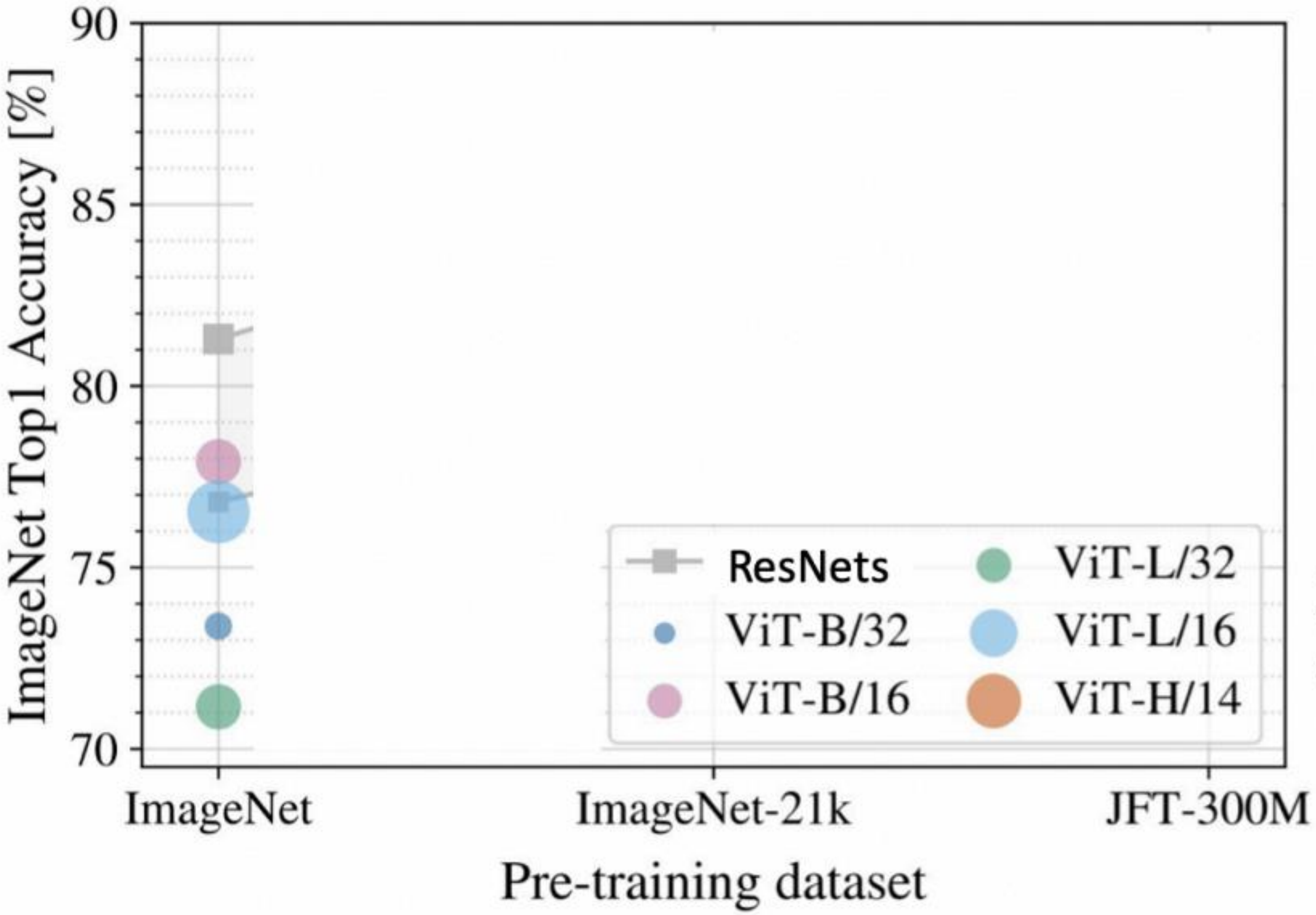
B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

ViT vs. ResNets

Recall: ImageNet dataset has 1k categories, 1.2M images

When trained on ImageNet, ViT models perform worse than ResNets

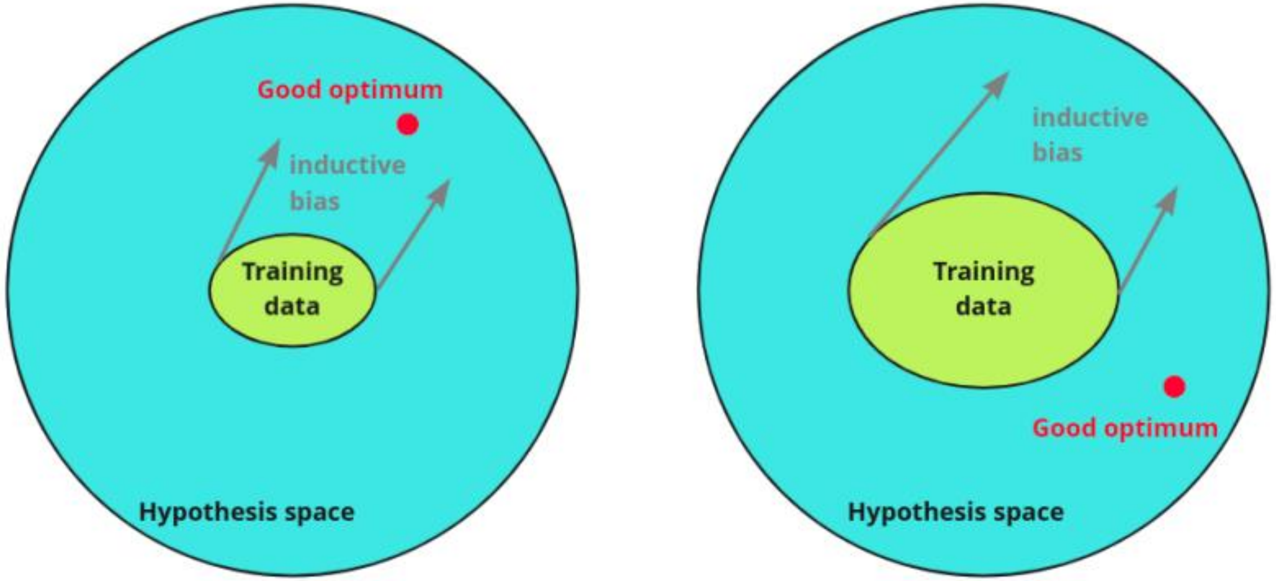


B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

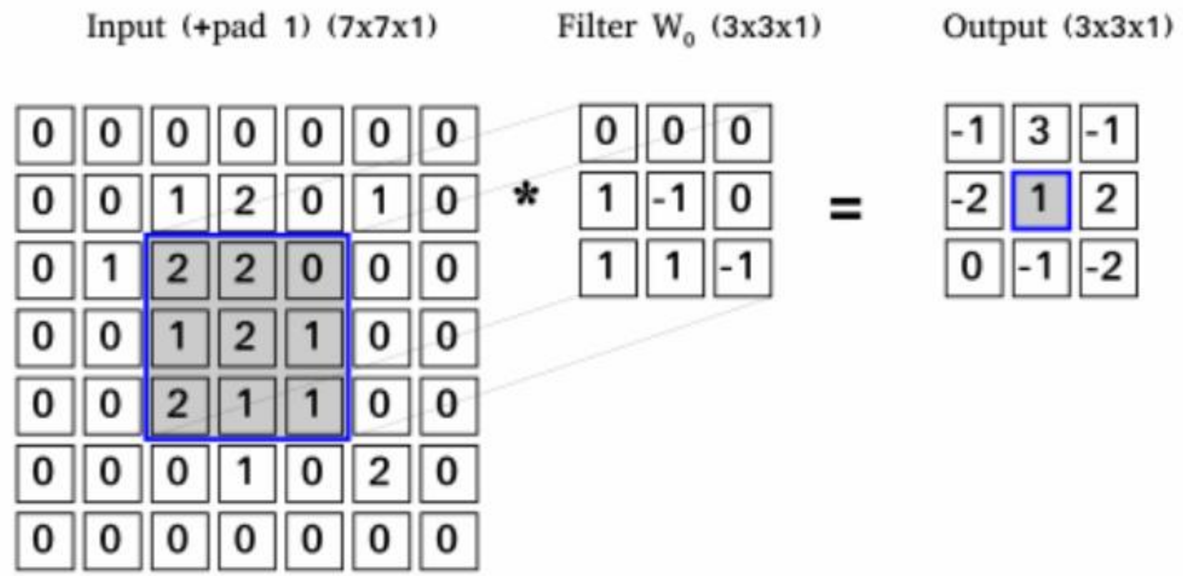
Inductive Bias

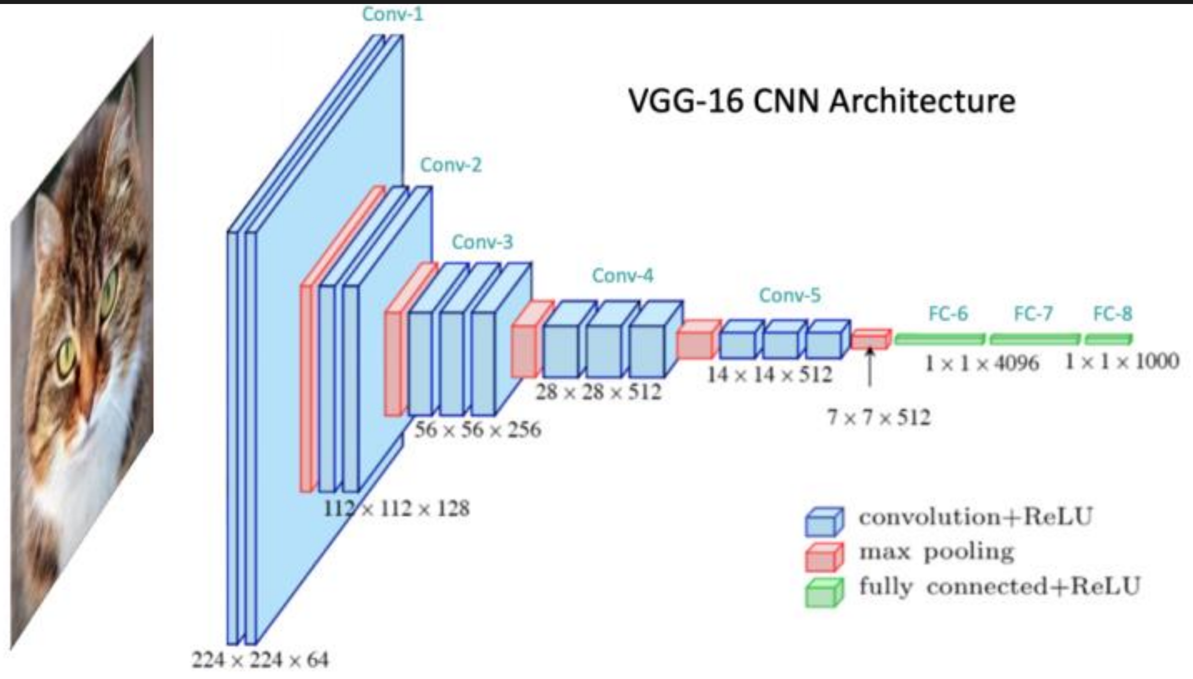
- Learning is an ill-posed problem; data is not sufficient to find a unique solution.
- Human-designed assumptions about hypothesis space are needed; Inductive Bias



Inductive Bias of CNNs

- Why CNNs have great performance on vision tasks?
- Local kernel processing / Hierarchical processing
- Translation invariance (Shared kernels and sliding windows)



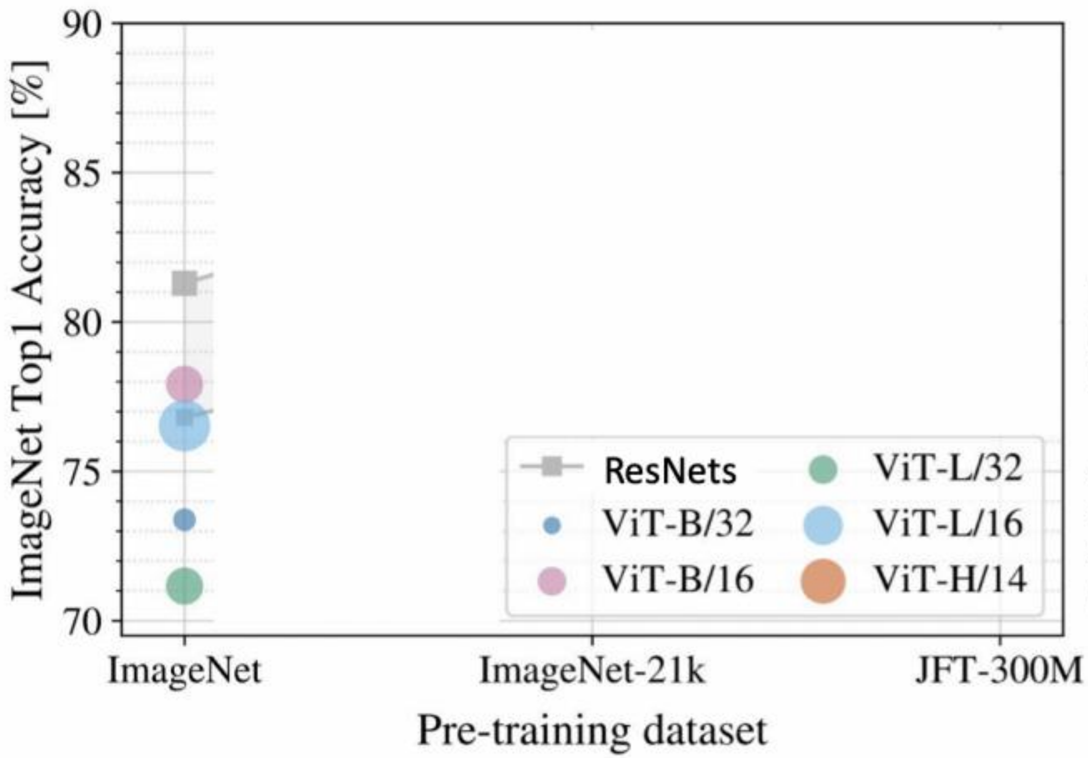


Inductive Bias of CNNs

- Strong inductive bias can make models robust to overfitting. (Regularization?)

Recall: ImageNet dataset has 1k categories, 1.2M images

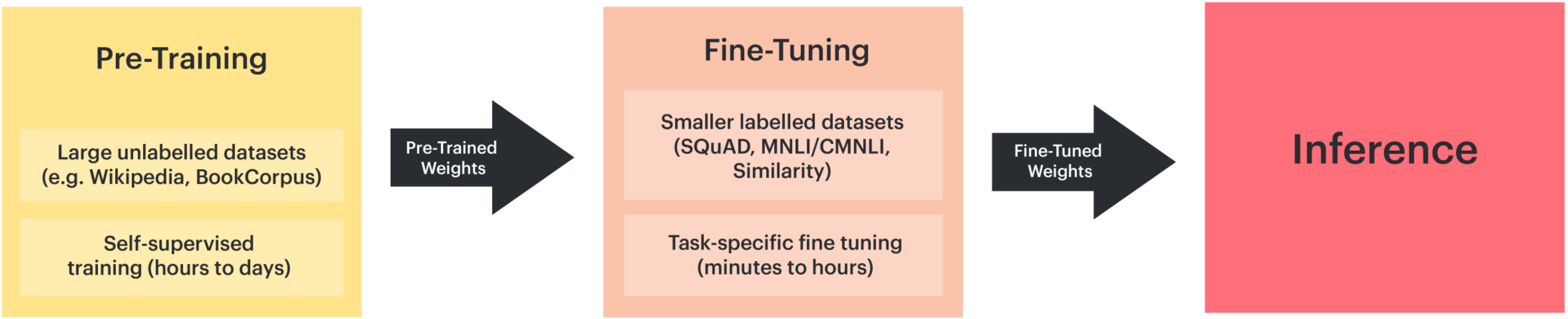
When trained on ImageNet, ViT models perform worse than ResNets



B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

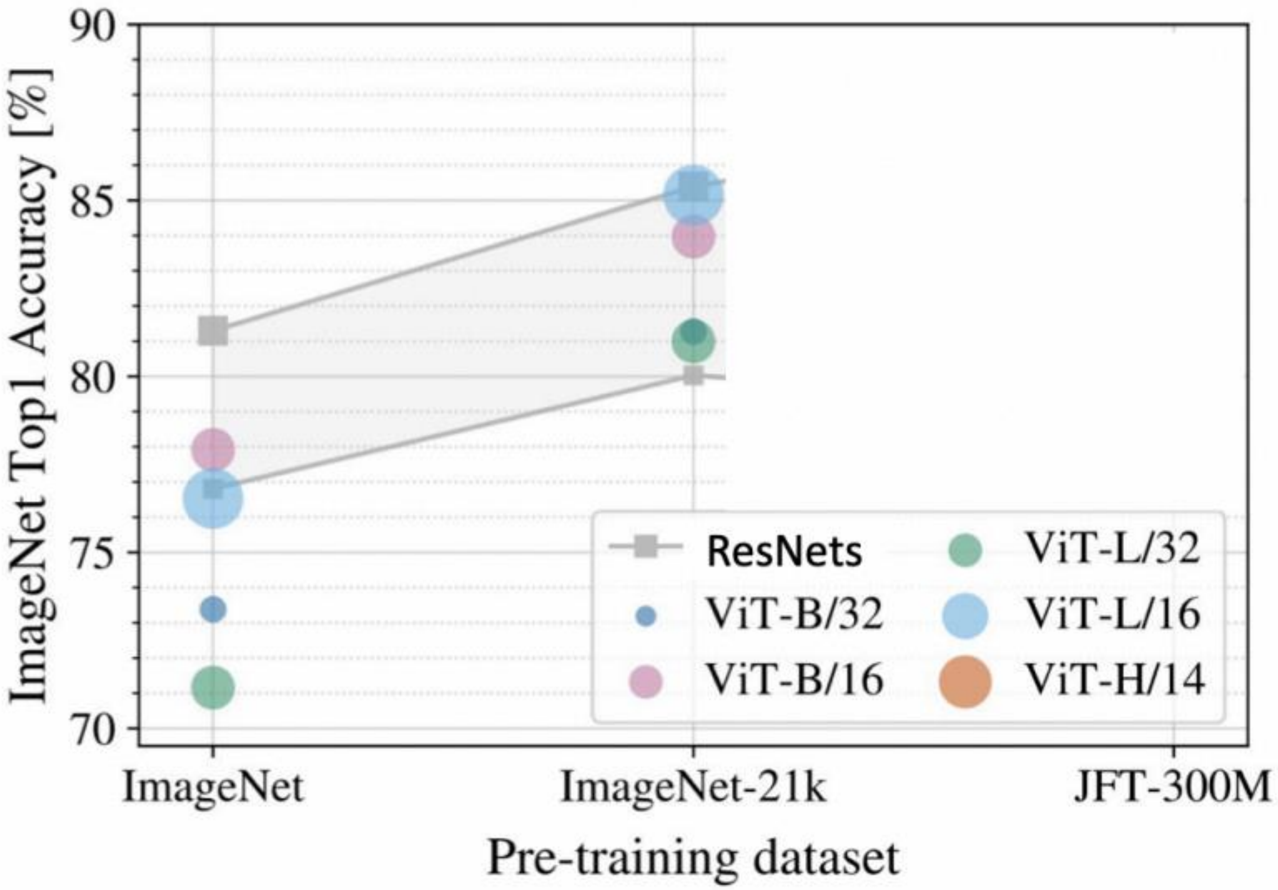
Pretraining and Finetuning



ViT vs. ResNets

ImageNet-21k has 14M images with 21k categories

If you pretrain on ImageNet-21k and fine-tune on ImageNet, ViT does better: big ViTs match big ResNets



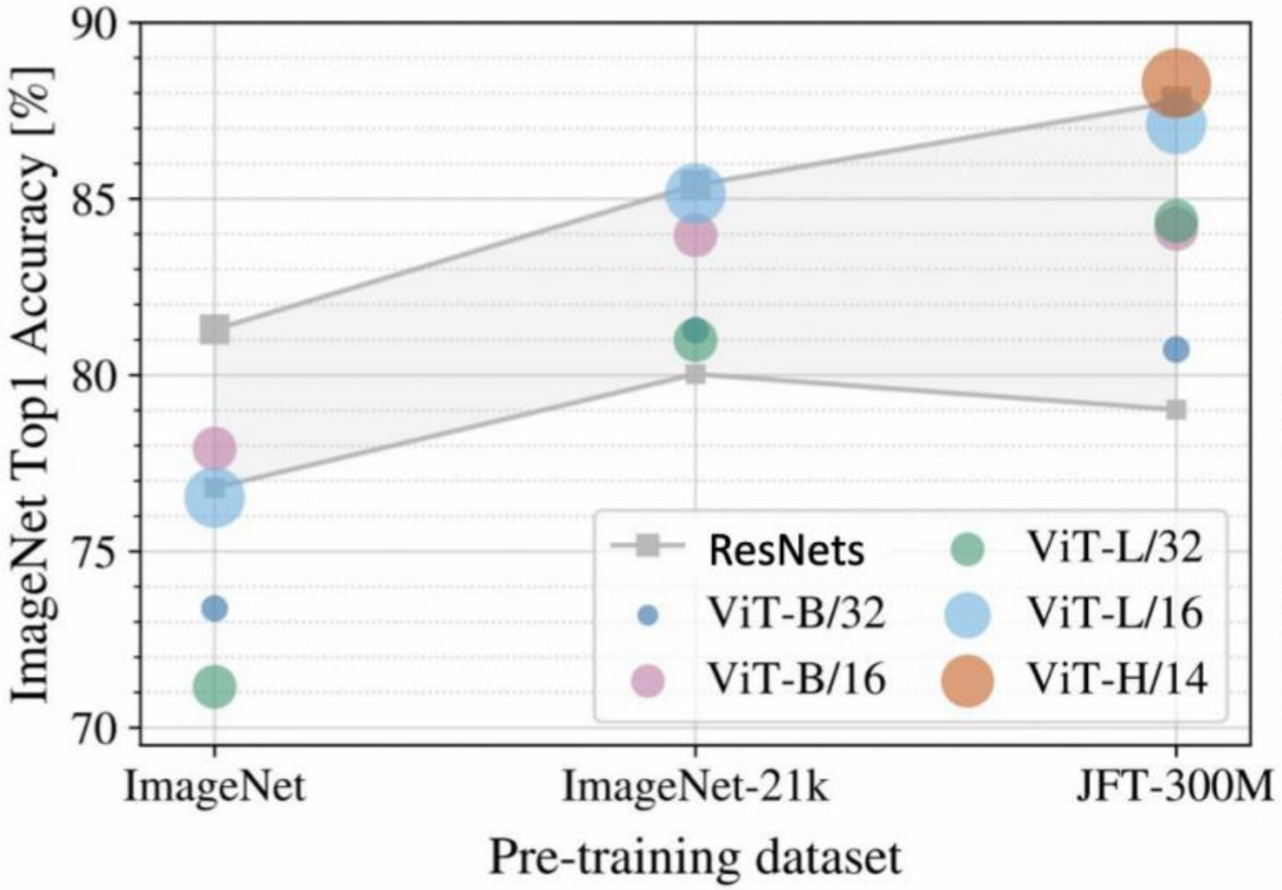
B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

ViT vs. ResNets

JFT-300M is an internal Google dataset with 300M labeled images

If you pretrain on JFT and finetune on ImageNet, large ViTs outperform large ResNets



B = Base
L = Large
H = Huge

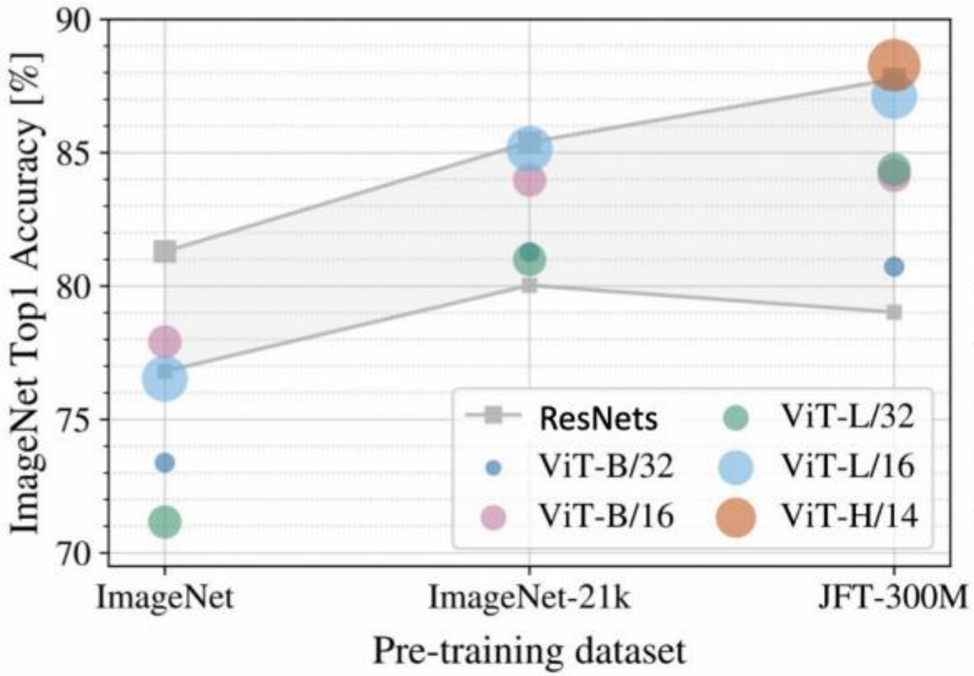
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

ViT vs. ResNets

- ViT models have less ‘inductive bias’ than ResNets
- Need more pretraining data to learn good features
- ViTs make more efficient use of GPU / TPU hardware (Why?)

JFT-300M is an internal Google dataset with 300M labeled images

If you pretrain on JFT and finetune on ImageNet, large ViTs outperform large ResNets



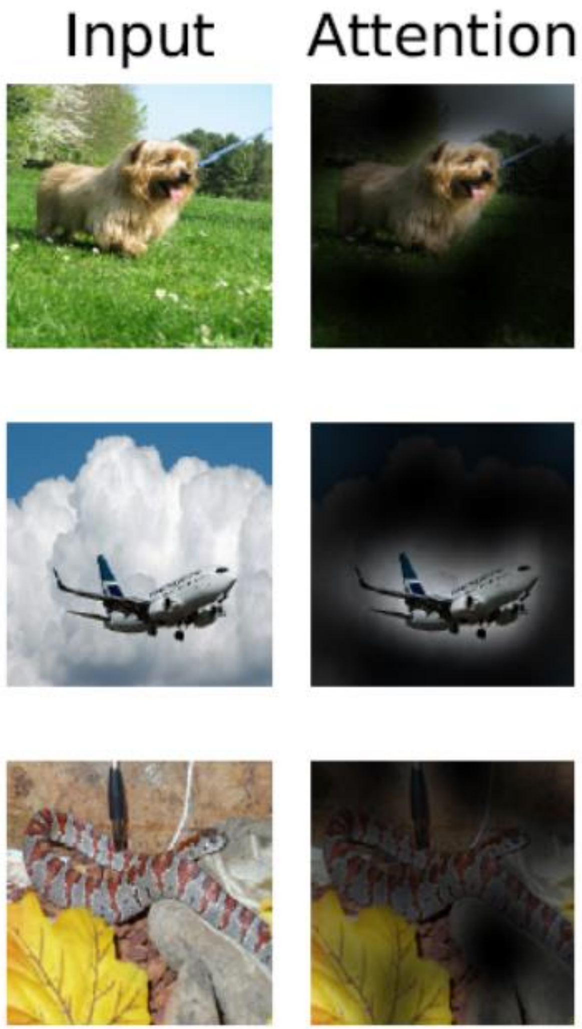
B = Base
L = Large
H = Huge

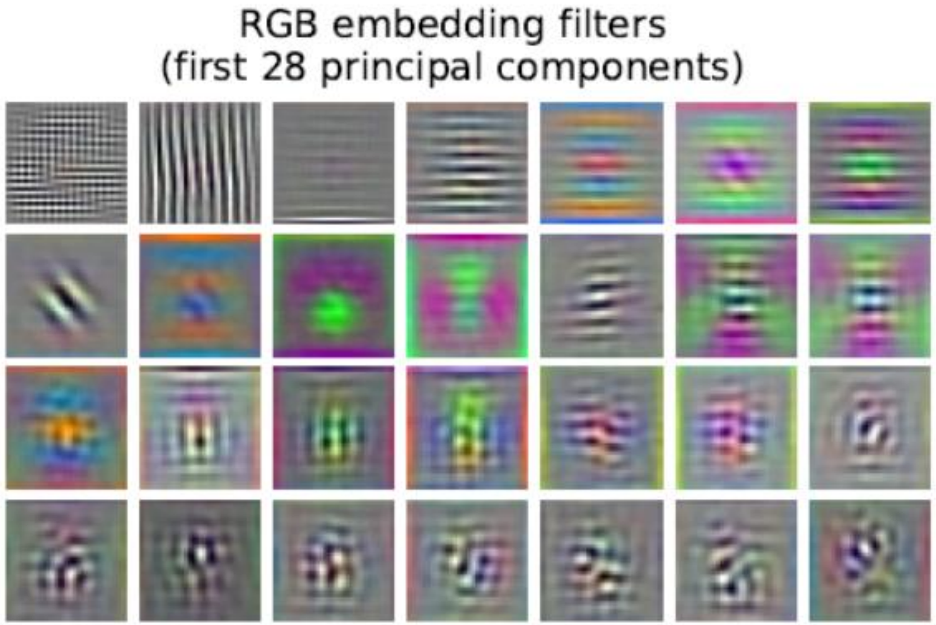
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

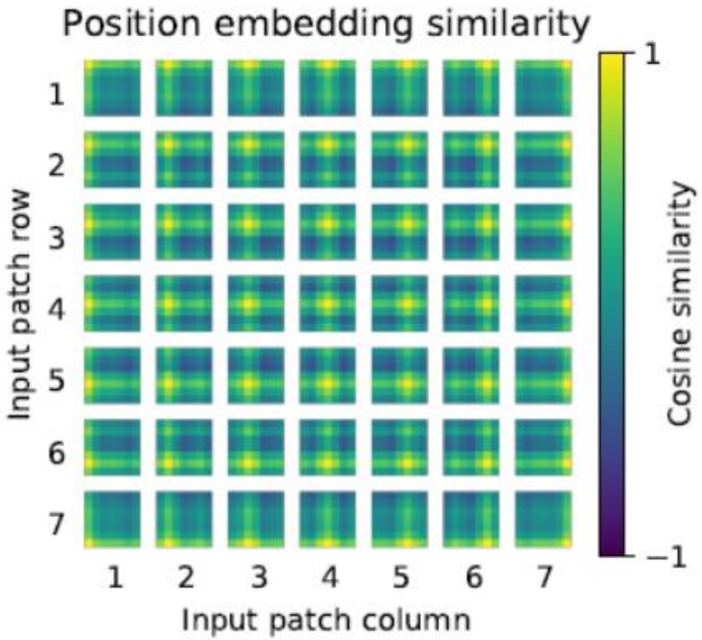
ViT: 2.5k TPU-v3 core days of training

ResNet: 9.9k TPU-v3 core days of training

What ViT accomplished

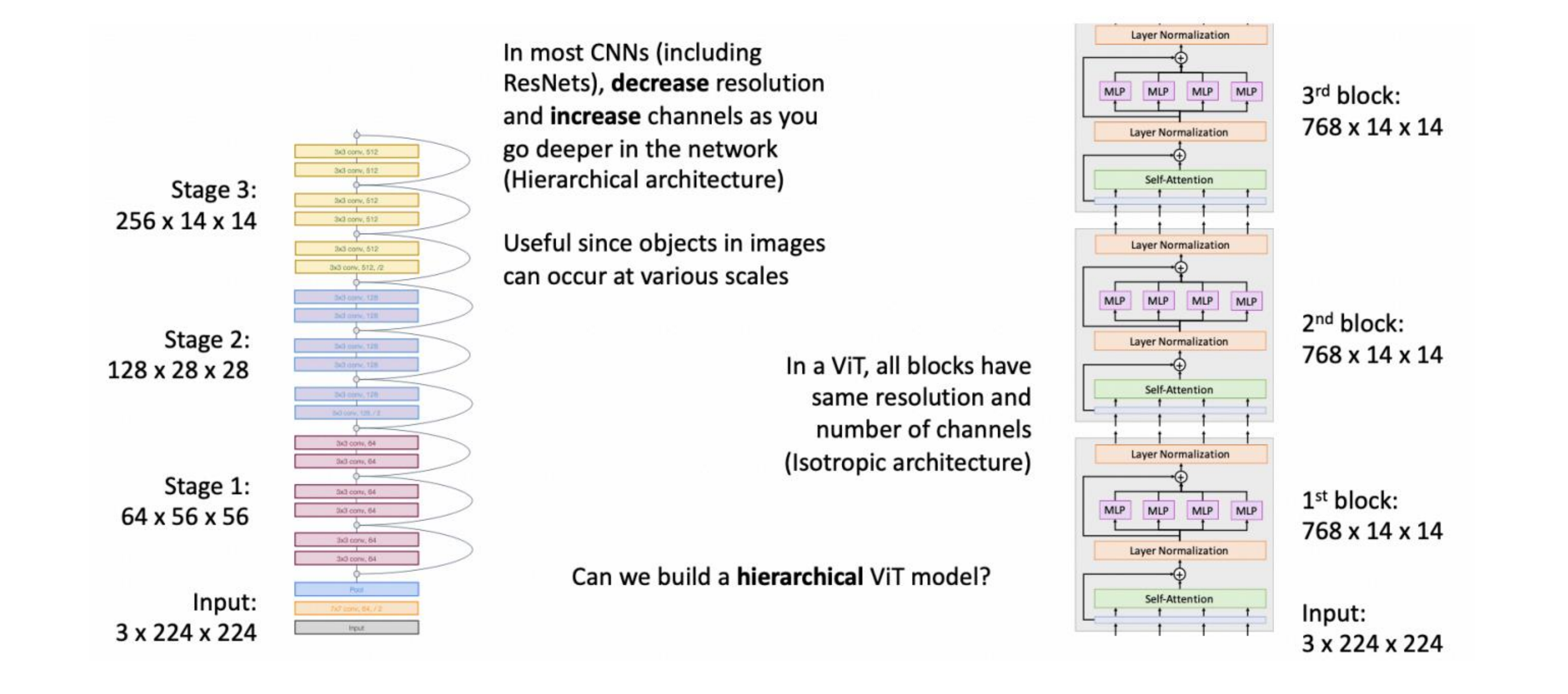






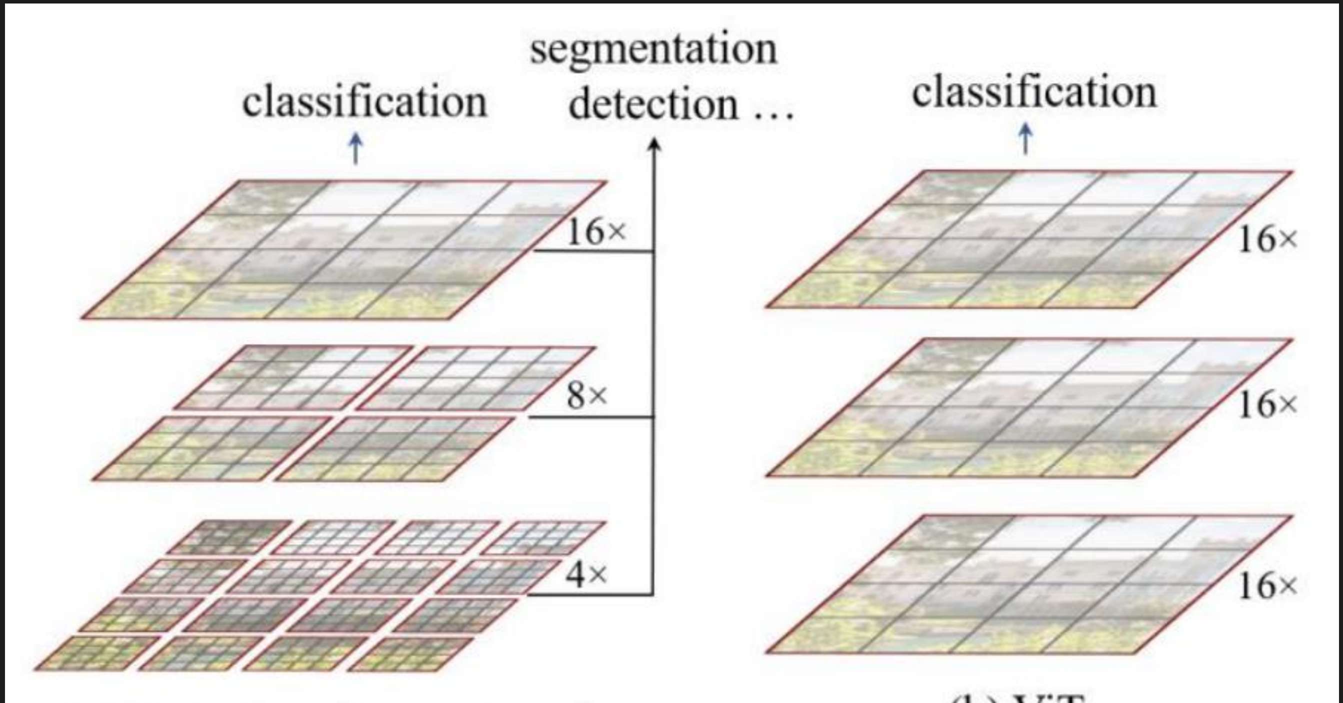
Inductive Bias : Hierarchicality

- ViT can only produce feature maps of single low resolution
  - Visual elements has various scales, unlike the word tokens
- We cannot attention on pixels (or very small patches) for high-res feature map, due to quadratic computational complexity of self-attention



Swin Transformer

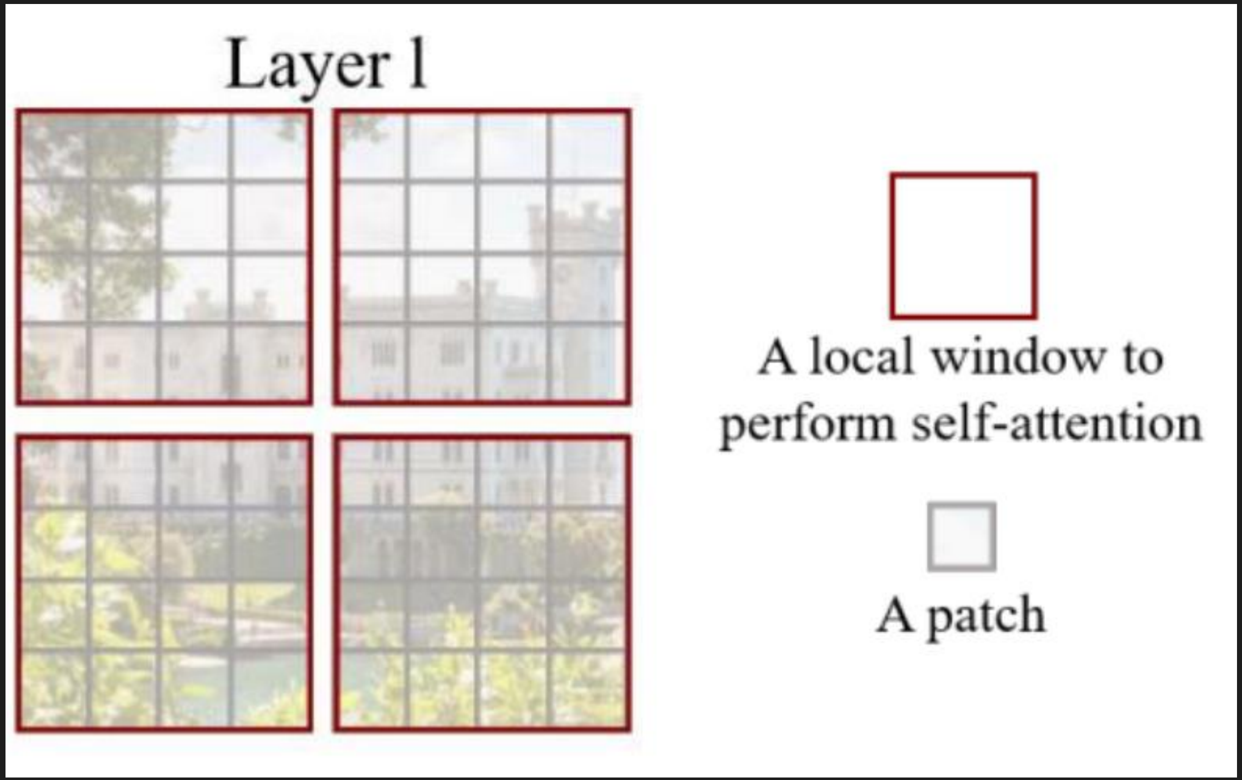
- Expand the applicabaility of Transformer to general-purpos backbone for CV
  - Introduce 'window' concept
  - Starts from small-sized patches, gradually merges neigboring patches



(a) Swin Transformer (ours)   (b) ViT

Swin Transformer

- Local Window attention
  - Compute self-attention locally within non-overlapping windows



Computational Complexity of Self-attention

- Global (Multihead) Self-Attention

$$hw \times C \quad \begin{array}{l} \xrightarrow{\mathbf{W}_Q \in \mathbb{R}^{C\times C}} \mathbf{Q} \\ \xrightarrow{\mathbf{W}_K \in \mathbb{R}^{C\times C}} \mathbf{K} \\ \xrightarrow{\mathbf{W}_V \in \mathbb{R}^{C\times C}} \mathbf{V} \end{array} \rightarrow \mathbf{Z} = \left(\mathbf{QK}^T\right)\mathbf{V} \xrightarrow{W_{\mathbf{z}} \in \mathbb{R}^{C\times C}} out$$

$(hw \times C) \times (C \times C)$

$(hw \times C) \times (C \times hw)$

$(hw \times hw) \times (hw \times C)$

$(hw \times C) \times (C \times C)$

$$\begin{aligned} \Omega\left(\mathbf{MSA}\right) &= 3hw \times C^2 + 2(hw)^2C + hw \times C^2 \\ &= 4hwC^2 + 2(hw)^2C \end{aligned}$$

Computational Complexity of Self-attention

- Windowed (Multihead) Self-Attention

[for $\lfloor \frac{h}{M} \rfloor \lfloor \frac{w}{M} \rfloor$ windows]

$M^2 \times C$

$\xrightarrow{\mathbf{W}_Q \in \mathbb{R}^{C \times C}} \mathbf{Q}$

$\xrightarrow{\mathbf{W}_K \in \mathbb{R}^{C \times C}} \mathbf{K}$

$\xrightarrow{\mathbf{W}_V \in \mathbb{R}^{C \times C}} \mathbf{V}$

$(M^2 \times C) \times (C \times C)$

$$\mathbf{Z} = \left(\mathbf{Q}\mathbf{K}^T\right)\mathbf{V}$$

$(M^2 \times C) \times (C \times M^2)$

$(M^2 \times M^2) \times (M^2 \times C)$

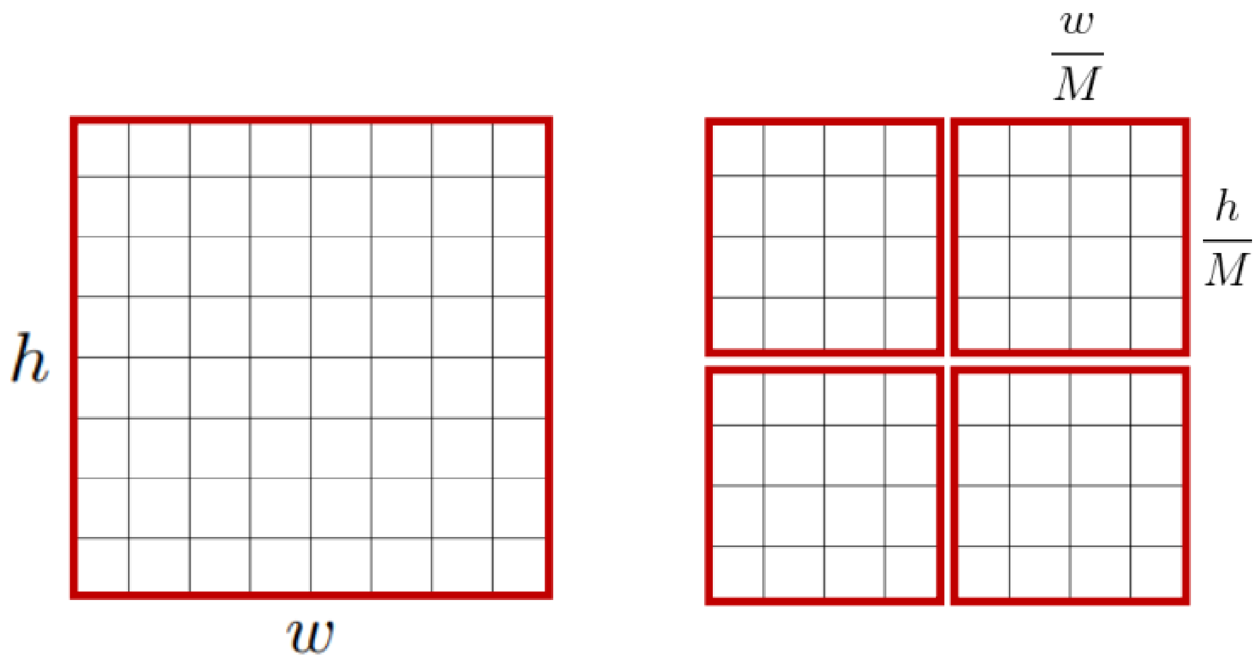
$\xrightarrow{W_{\mathbf{z}} \in \mathbb{R}^{C \times C}} out$

$(M^2 \times C) \times (C \times C)$

Computational Complexity of Self-attention

- W-MSA has linear complexity to image size (# of patches, hw)

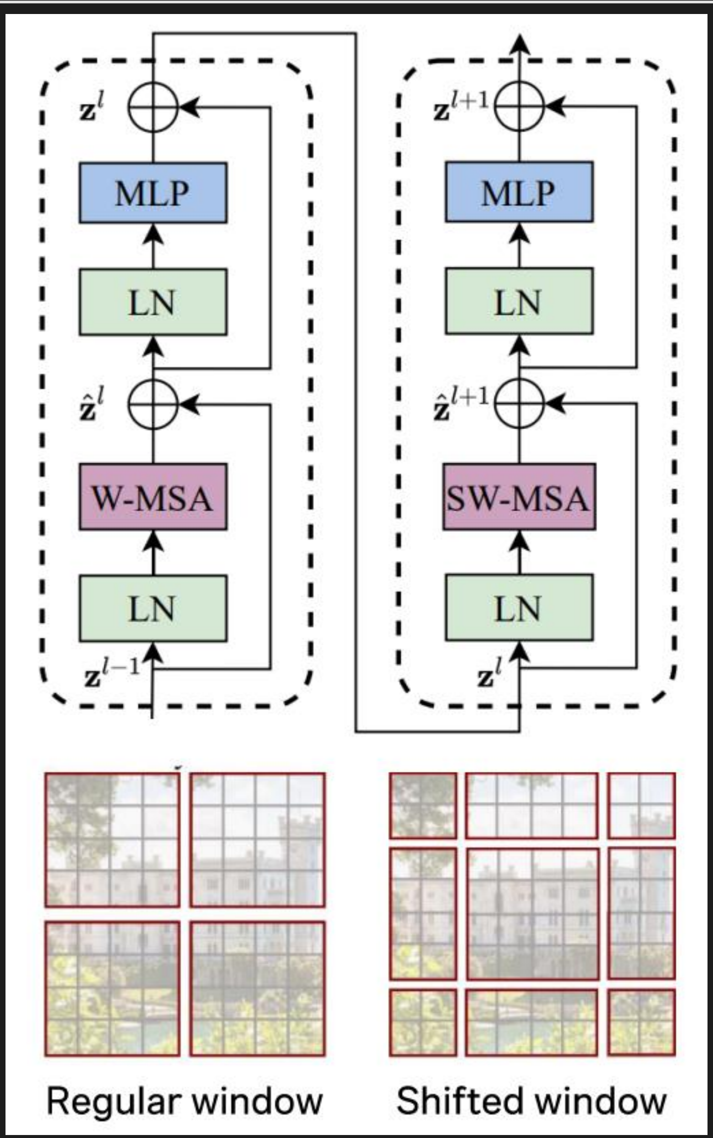
$$\Omega(\text{MSA}) = 4hwC^2 + 2\boxed{(hw)^2}C,$$
$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2\boxed{hw}C,$$



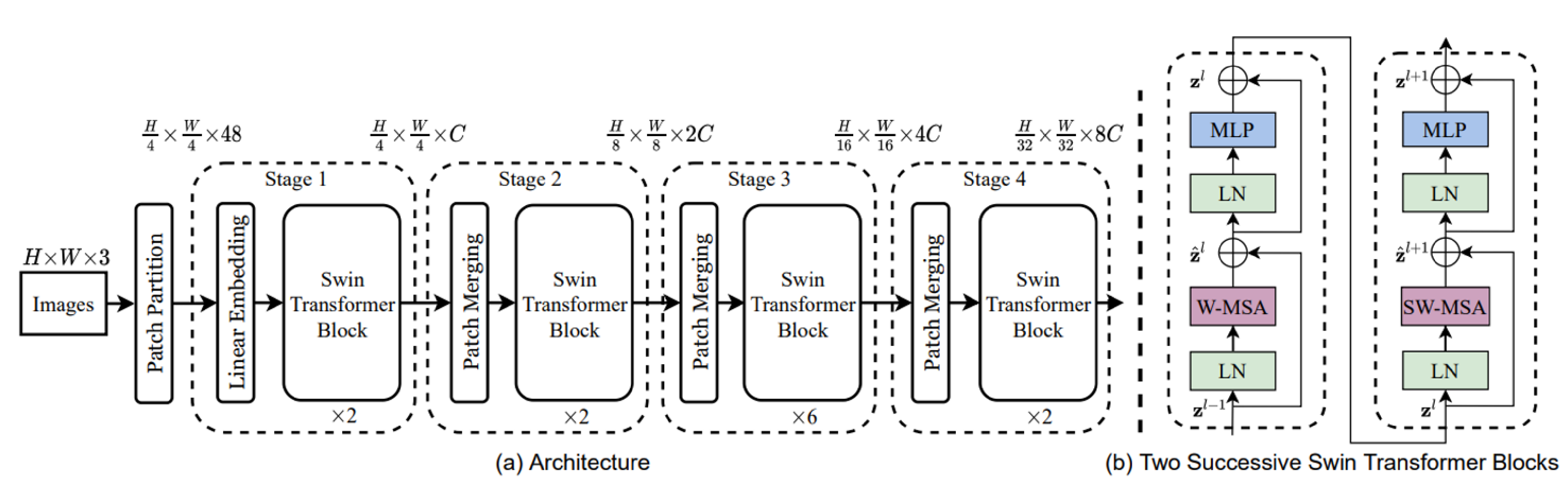
Shifted window

- Problem of W-MSA?
  - Lack of connectivity between windows
  - Introduces Shifted Window
  - Two Successive Swin Transformer Blocks have W-MSA and SW-MSA

  - W-MSA uses regular window
  - SW-MSA uses shifted window

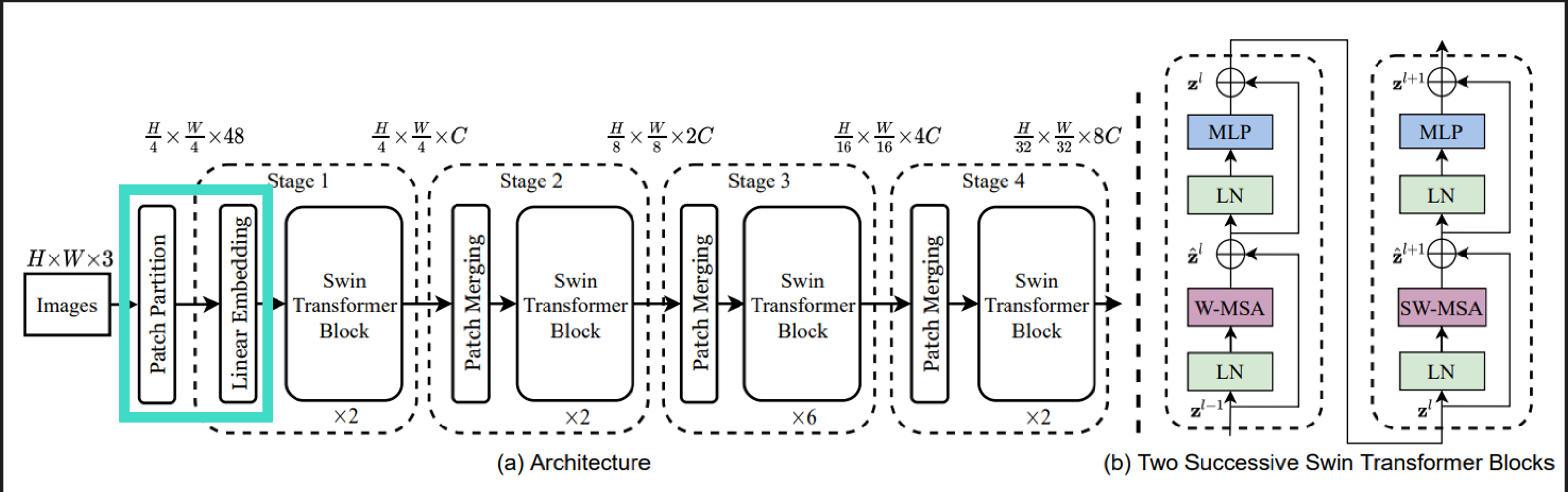


Swin Transformer



- Patch Partition & linear Embedding (w/ Relative Positional Bias)
- Swin Transformer Blocks
- Patch Merging

Swin Transformer

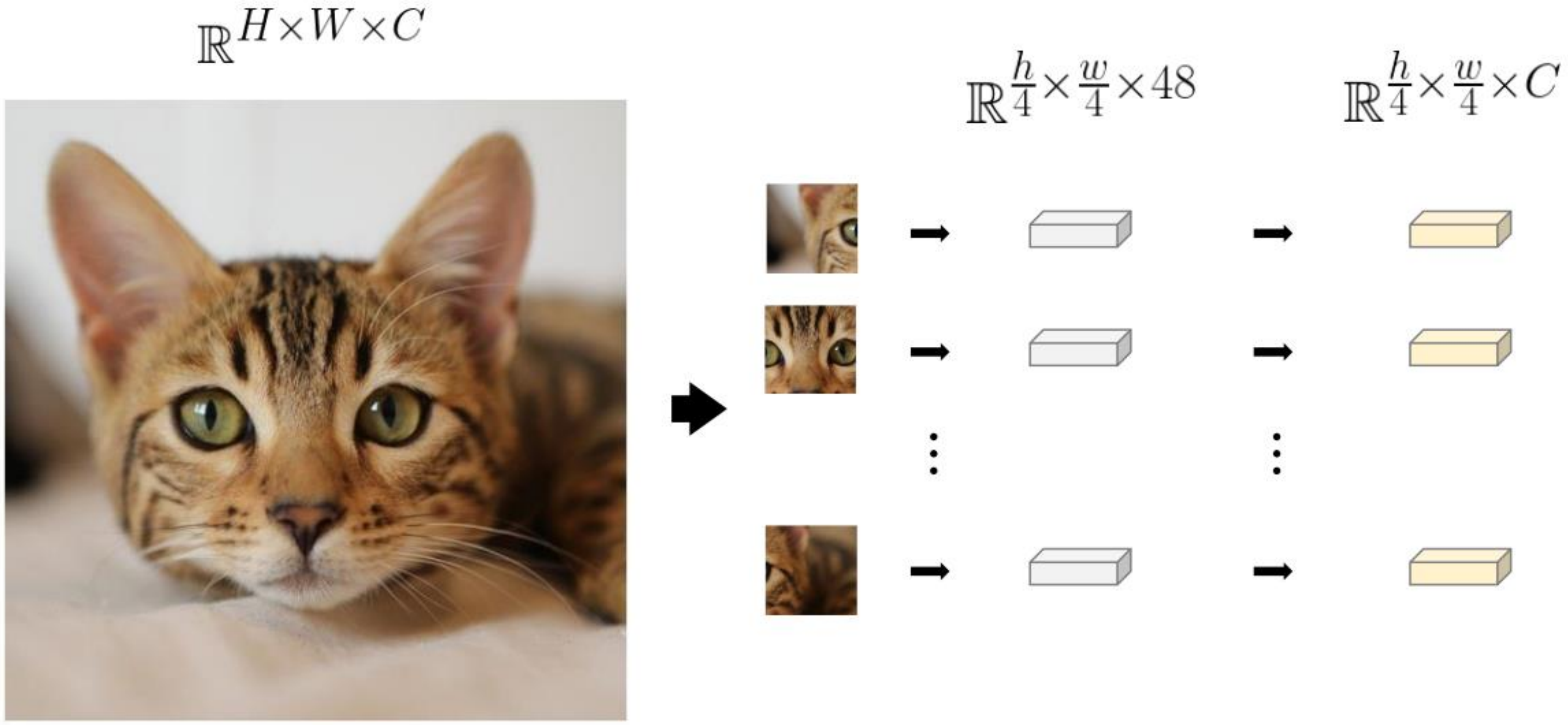


(a) Architecture

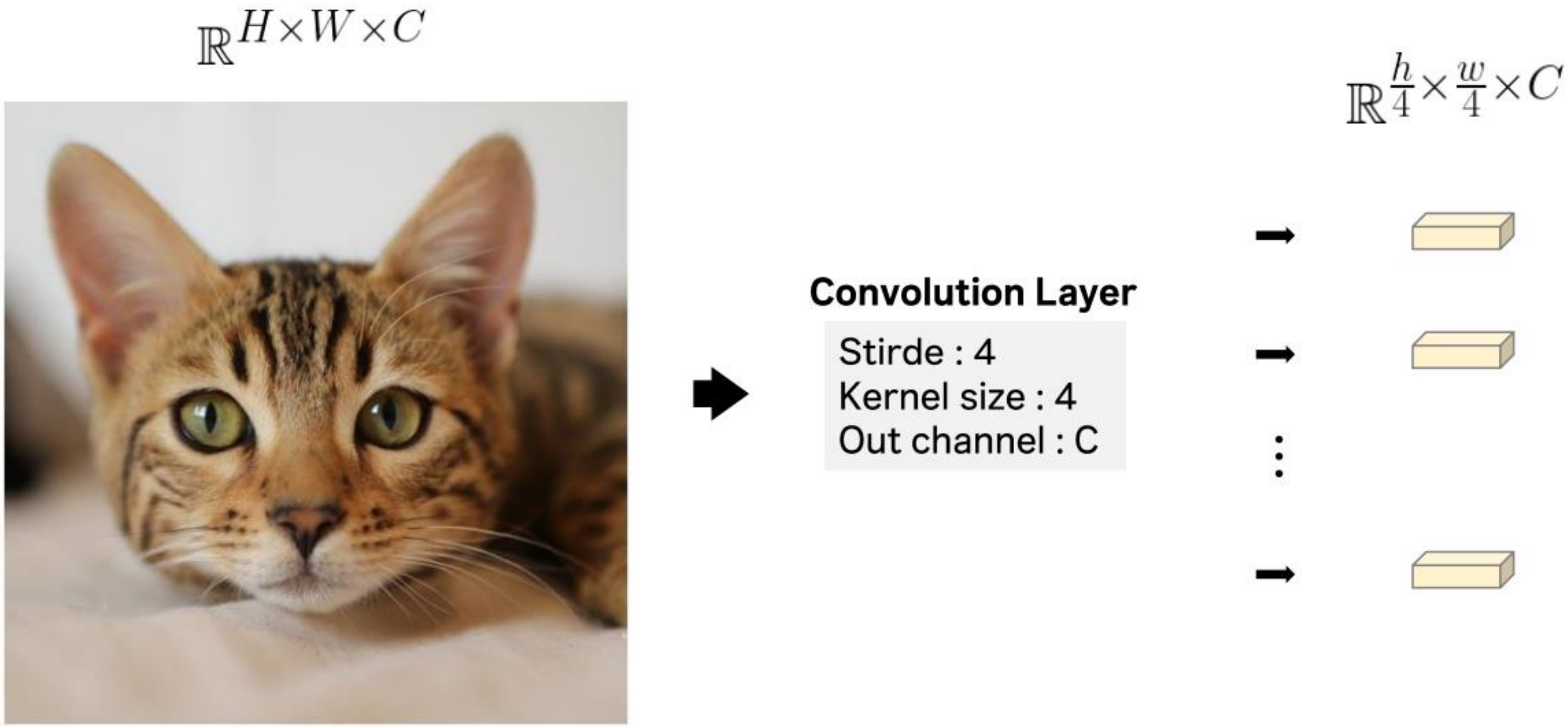
(b) Two Successive Swin Transformer Blocks

- Patch Partition & linear Embedding (w/ Relative Positional Bias)
- Swin Transformer Blocks
- Patch Merging

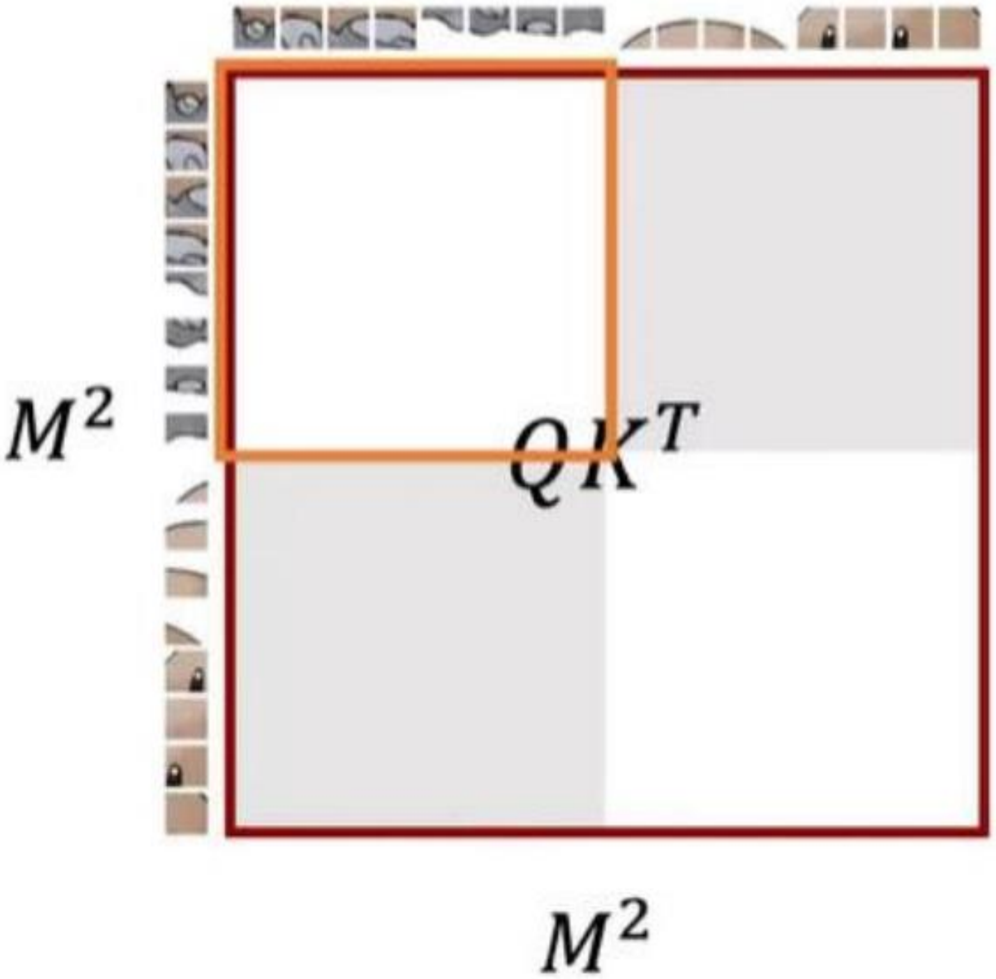
Patch Partition & linear Embedding



Patch Partition & linear Embedding



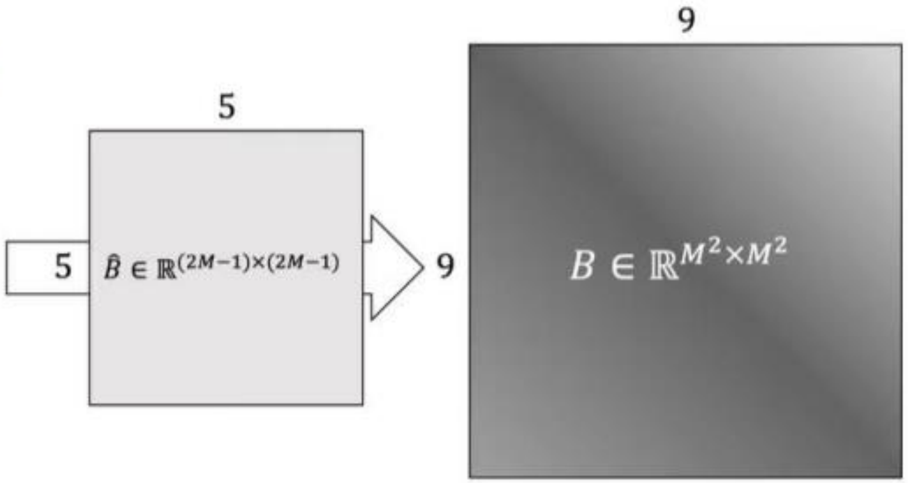
Relative Positional Bias



Window size (M) = 3

Relative Position index

12	11	10	7	6	5	2	1	0
13	12	11	8	7	6	3	2	1
14	13	12	9	8	7	4	3	2
17	16	15	12	11	10	7	6	5
18	17	16	13	12	11	8	7	6
19	18	17	14	13	12	9	8	7
22	21	20	17	16	15	12	11	10
23	22	21	18	17	16	13	12	11
24	23	22	19	18	17	14	13	12

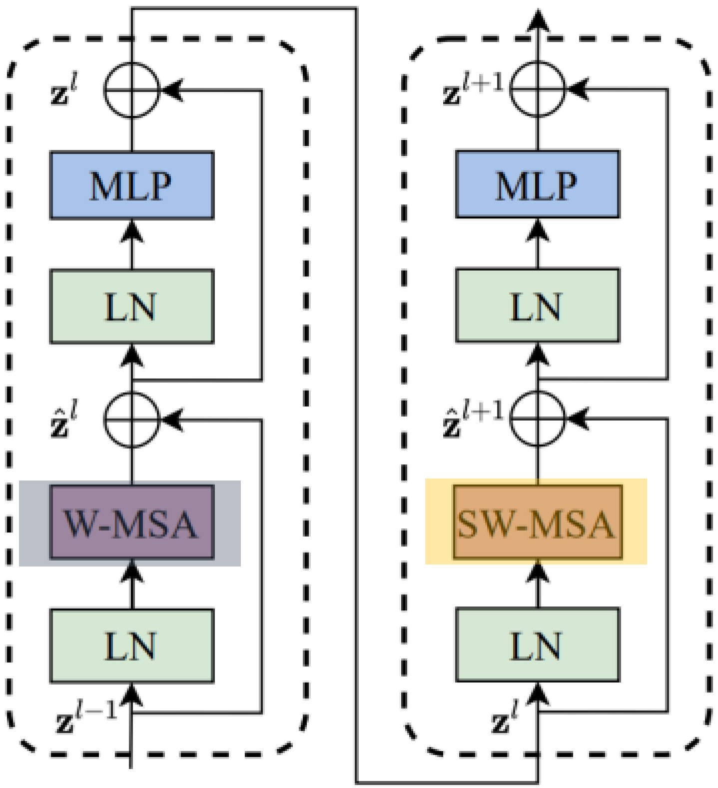


Swin Transformer Block

: W-MSA

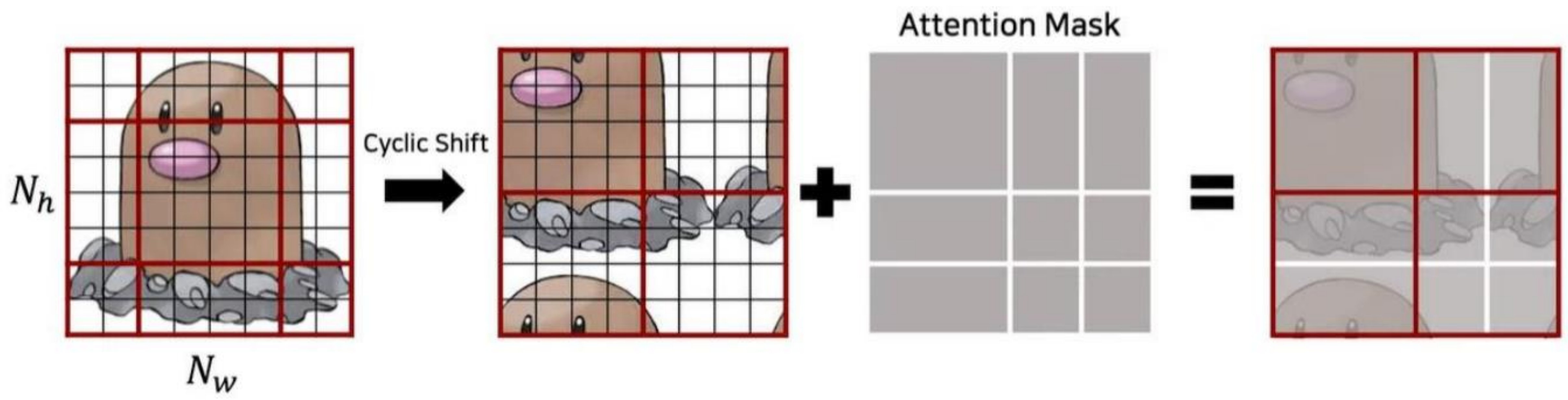
: SW-MSA

: Relative Position Bias

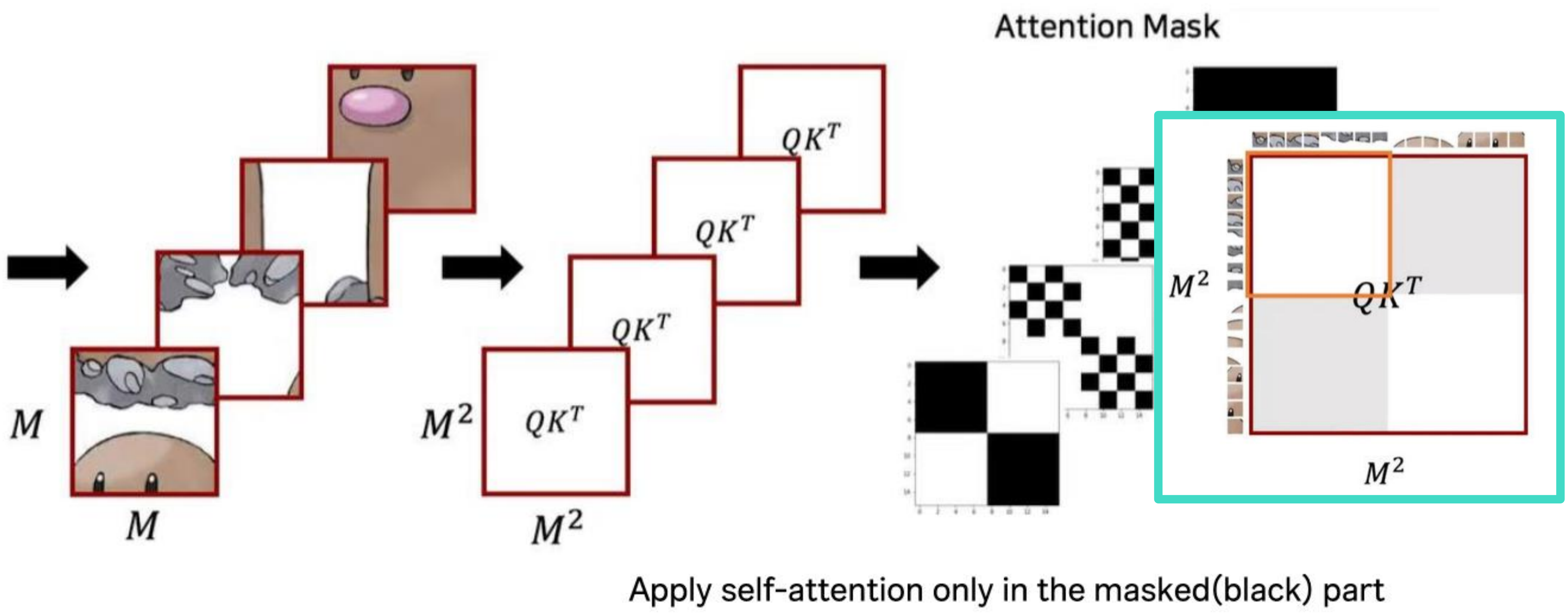


$$\mathrm{Attention}(Q, K, V) = \mathrm{SoftMax}(QK^T/\sqrt{d} + B)V,$$

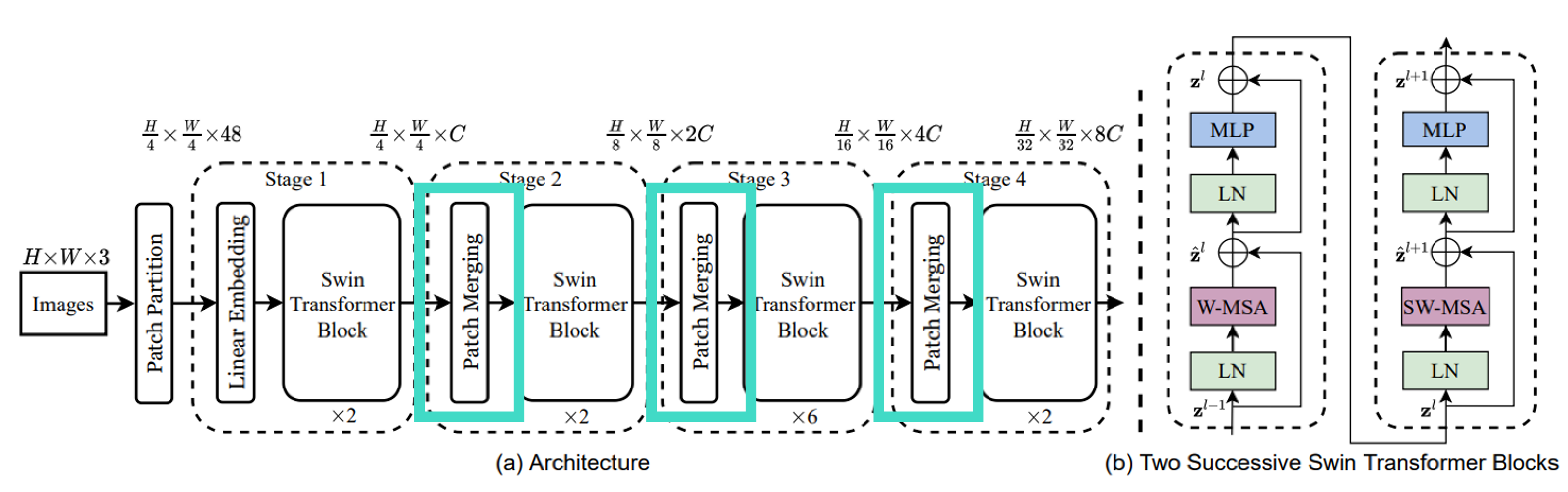
SW-MSA



SW-MSA


Attention Mask
M
M
M^2
QK^T
QK^T
QK^T
QK^T
M^2
M^2
QK^T
M^2
Apply self-attention only in the masked(black) part

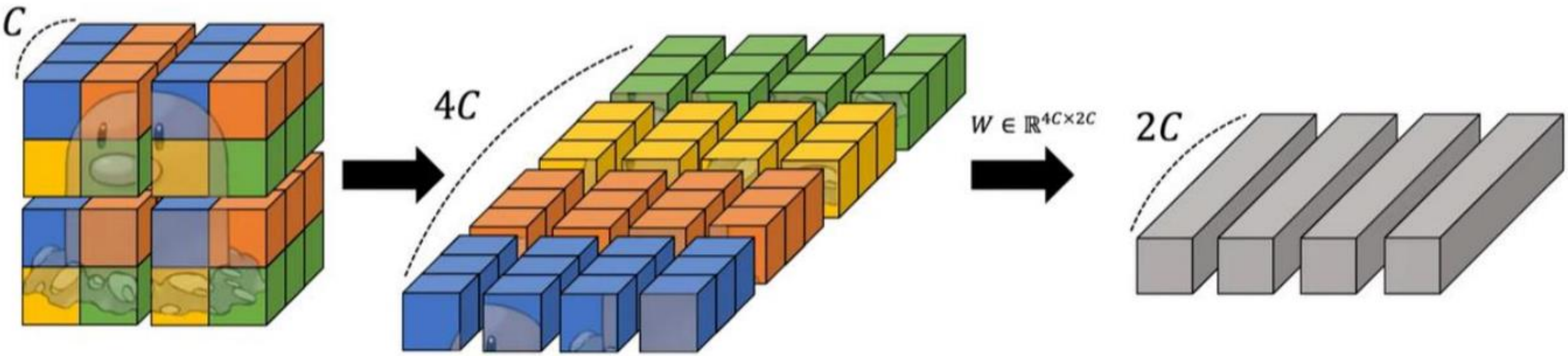
Swin Transformer



(a) Architecture
(b) Two Successive Swin Transformer Blocks

- Patch Partition & linear Embedding (w/ Relative Positional Bias)
- Swin Transformer Blocks
- Patch Merging

Patch Merging



- Concatenates patches in each window and projects from 4C to 2C
- Mix the information of the patches on the channel axis
- Prevent from too large channel size

Swin Transformer

3번 문항

(A) Table 1

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
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
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(B) Table 2

(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
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

질문 1 두 Table에 명시된 결과를 자유롭게 분석하세요.
답변에 따라 추가 질문이 있을 수 있습니다.

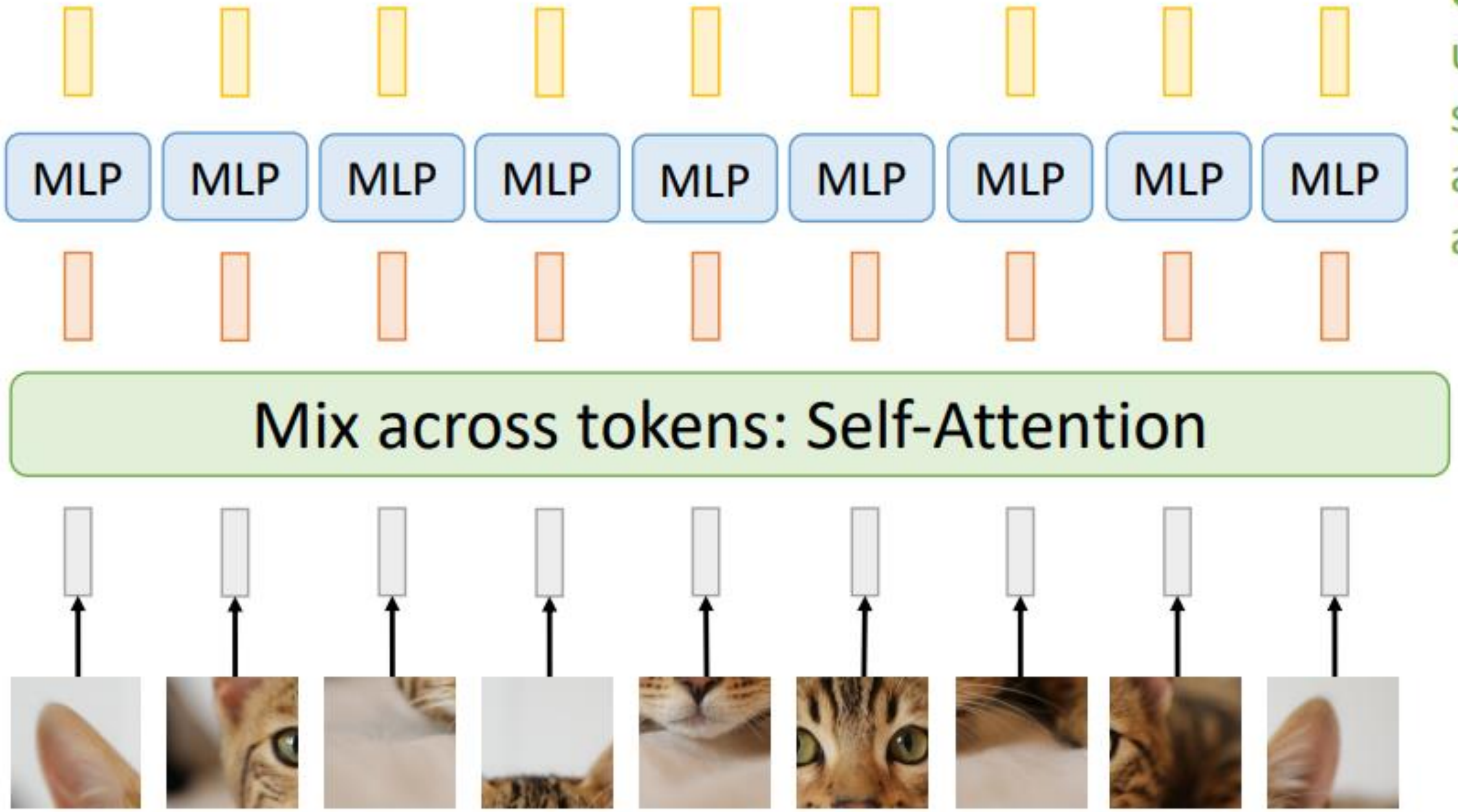
Vision Transformer: Another Look

Apply (D->D)
MLP to each of
the N tokens

Mix tokens with
self-attention

Input vectors N x D

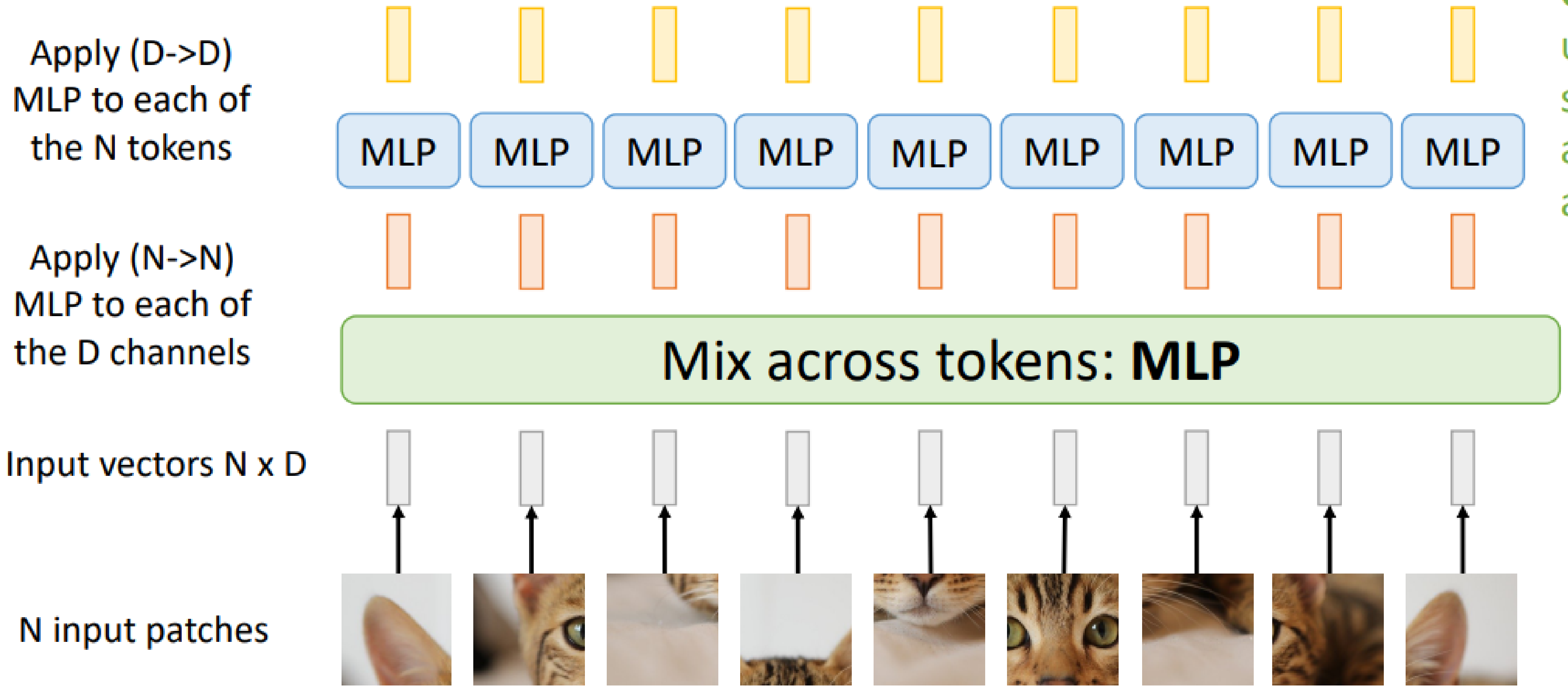
N input patches



Question: Can we use something simpler than self-attention to mix across tokens?

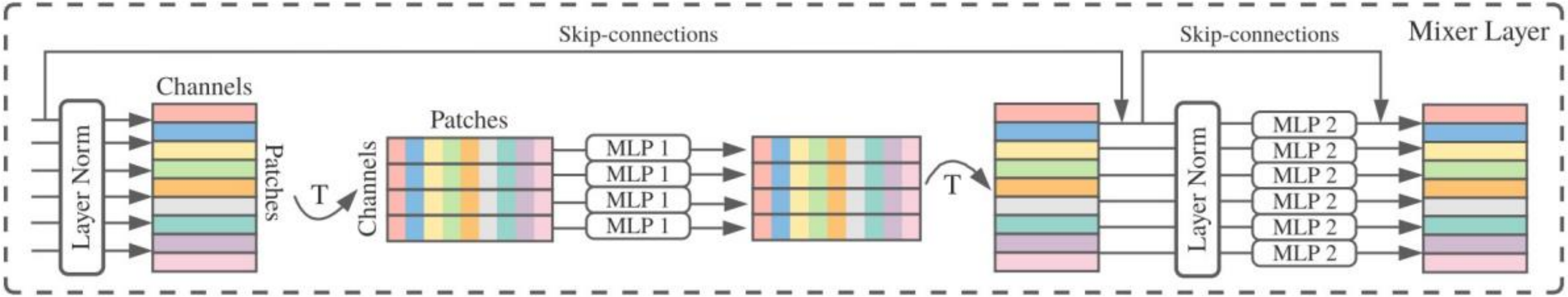
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformer: Another Look



Question: Can we use something simpler than self-attention to mix across tokens?

MLP-Mixer

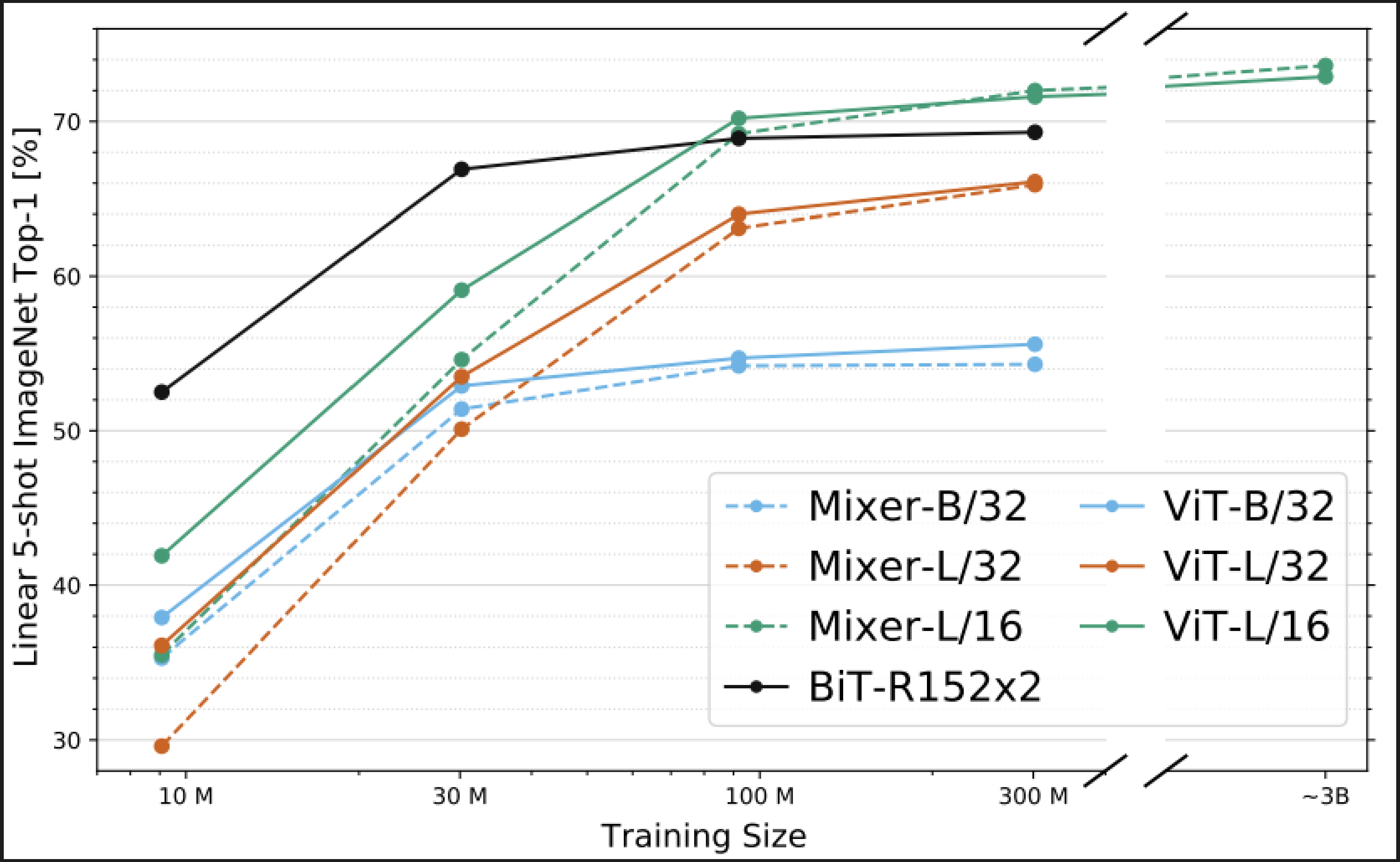


Input: N x C
N patches with
C channels each

MLP 1: C -> C,
apply to each of
the **N patches**

MLP 2: N -> N,
apply to each of
the **C channels**

MLP-Mixer



Linear 5-shot ImageNet Top-1 [%]
70
60
50
40
30
10 M
30 M
100 M
300 M
~3B
Training Size
Mixer-B/32
Mixer-L/32
Mixer-L/16
BiT-R152x2
ViT-B/32
ViT-L/32
ViT-L/16

Is “*Attention*” All You Need (in CV)?

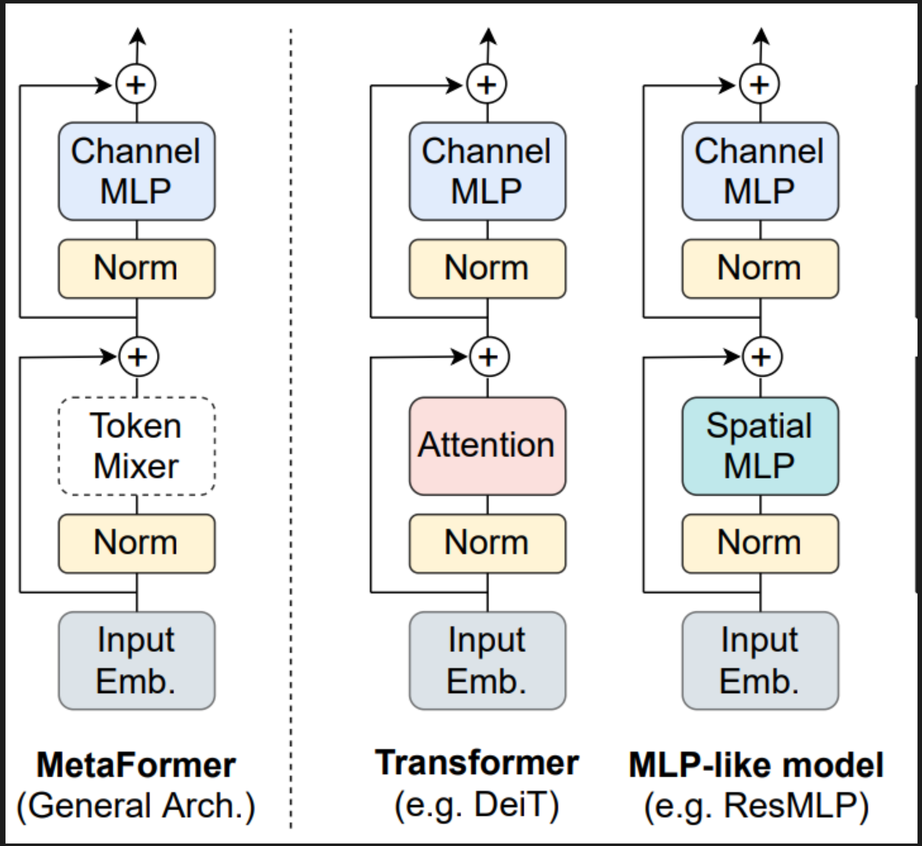
Is "*Attention*" All You Need?

"*Transformer Architecture*" is All You Need!

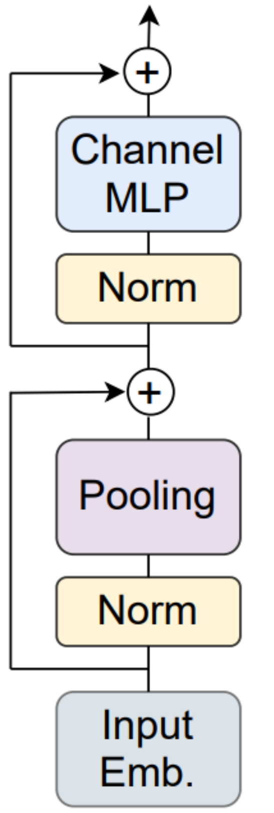
Transformer vs. MLP-Mixer

- The architecture of Transformer and MLP-Mixer are fundamentally same.
  - Token Mixer : mix information between (spatial) tokens
  - Channel Mixer : mix information between channels

- Traditionally, the success of Transformers has been attributed to attention-based token mixers
  - But, why MLP Mixer works very well?
  - Is Transformer Architecture all we need?



(Note) PoolFormer



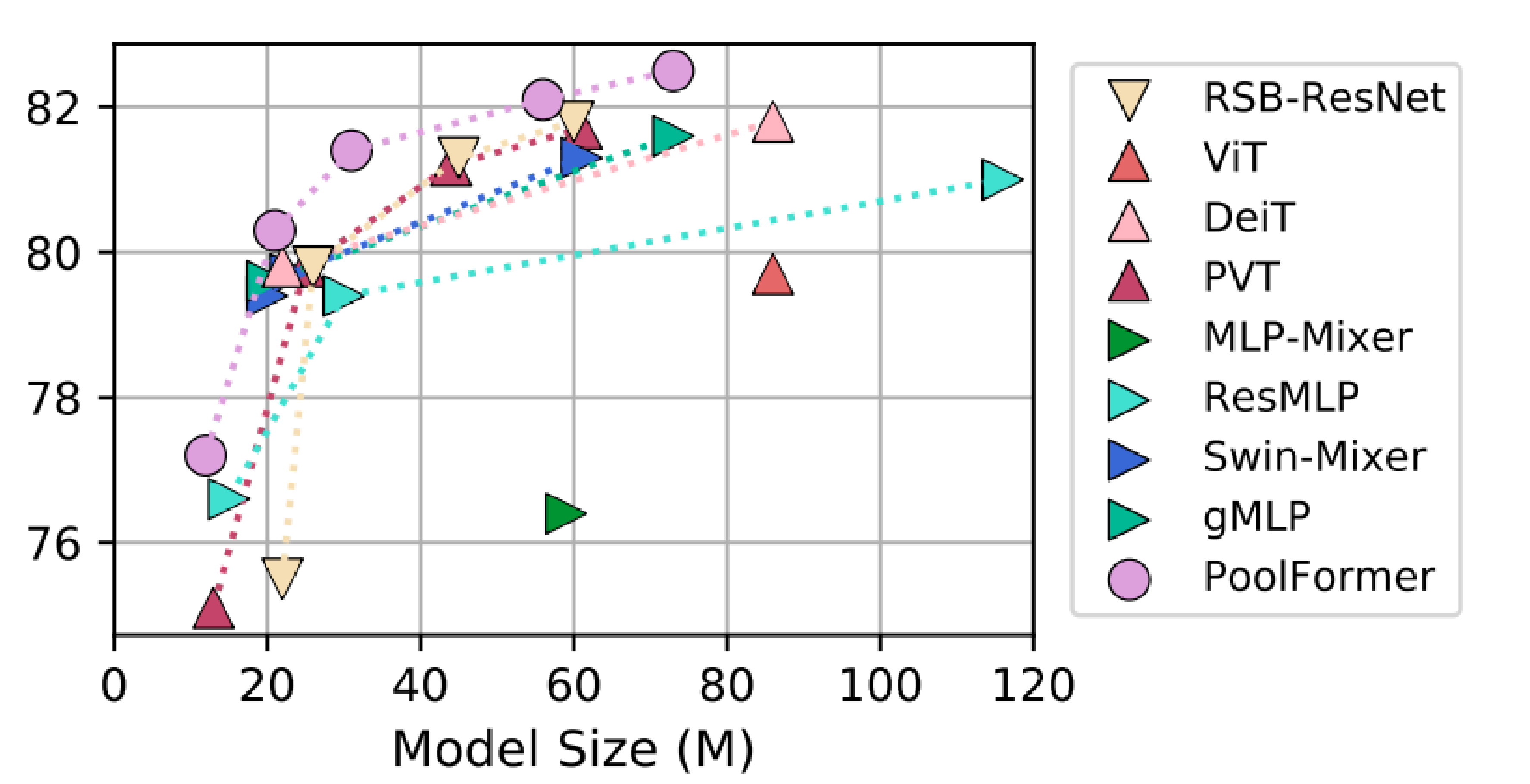
PoolFormer
(Ours)

- To test the hypothesis, PoolFormer is introduced.
  - Pooling Token mixing layer : highly simplistic and non-parametric

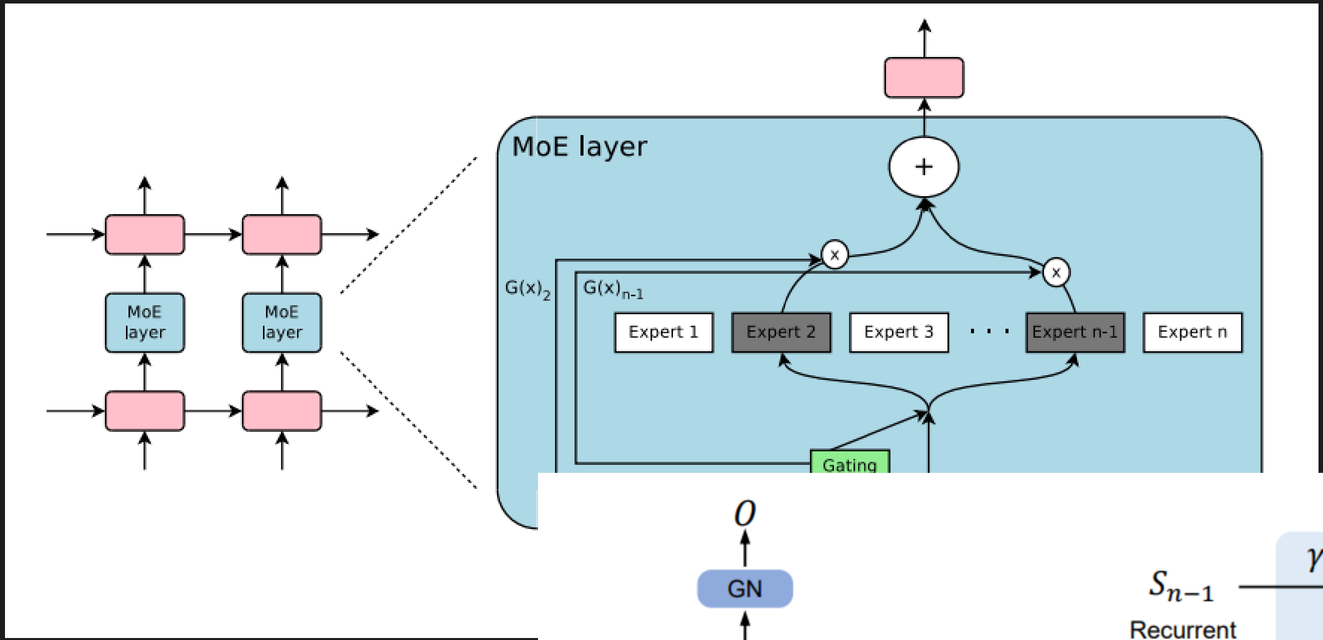
Algorithm 1 Pooling for PoolFormer, PyTorch-like Code

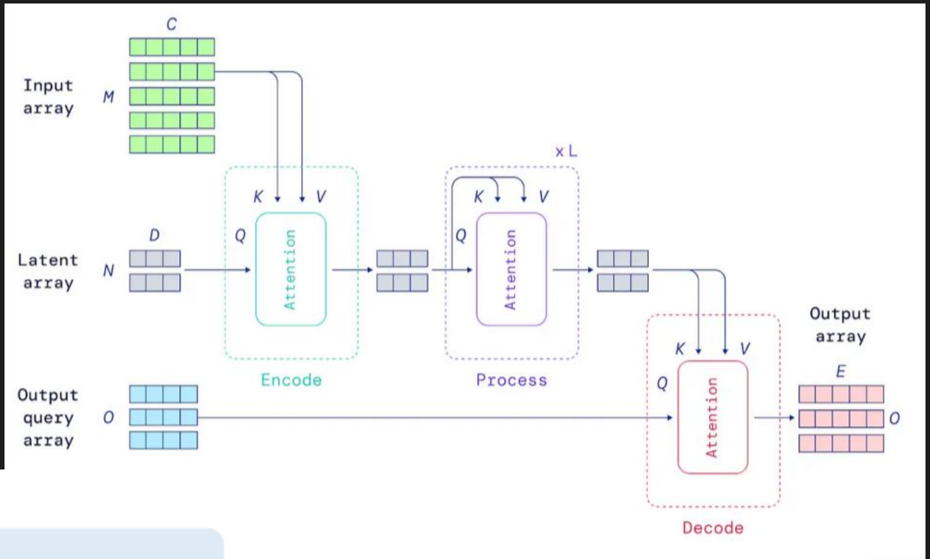
```
import torch.nn as nn

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
```

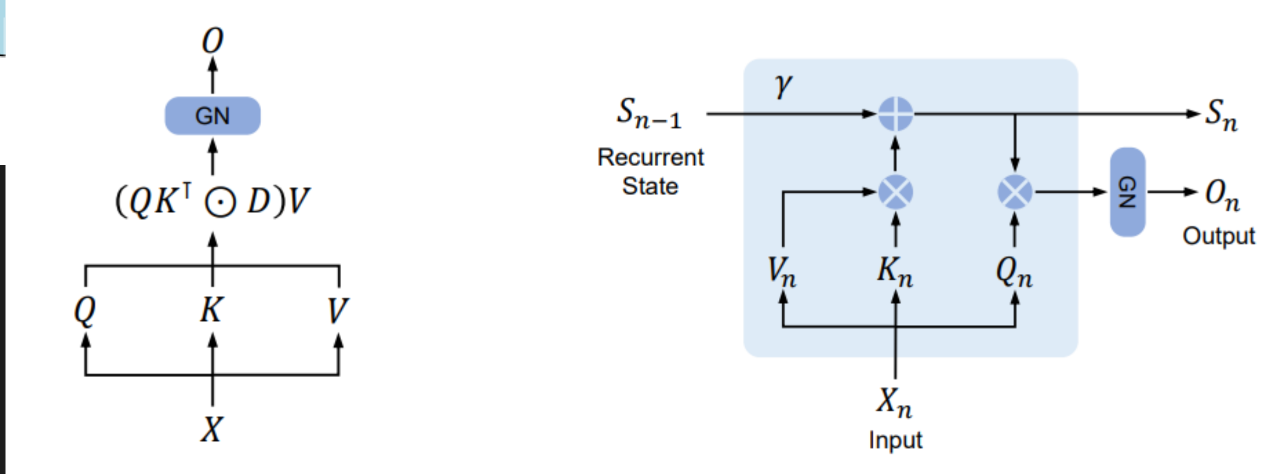
(Note) PoolFormer



MoE, Perciver IO, RetNet, ⋯







(a) Parallel representation.

(b) Recurrent representation.

디스커션 과제

- 개인 과제 : CNN vs. Transformer
  - 현재 CV 분야에서 활용되는 다양한 모델들의 대부분이 backbone architecture로 CNN 기반 혹은 Transformer 기반 모델들을 활용하고 있습니다. 그렇다면 항상 Transformer을 사용하는 것이 좋을까요? 그렇지 않다면 어떤 경우에 CNN이 더 좋은 성능을 낼까요?
- 팀 과제 : Inductive Bias
  - 두 표의 결과가 왜 이렇게 나오게 되었는지, inductive bias와 pretraining dataset size의 관점에서 분석해 봅시다.

(A) Table 1

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
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
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(B) Table 2

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
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

감사합니다.