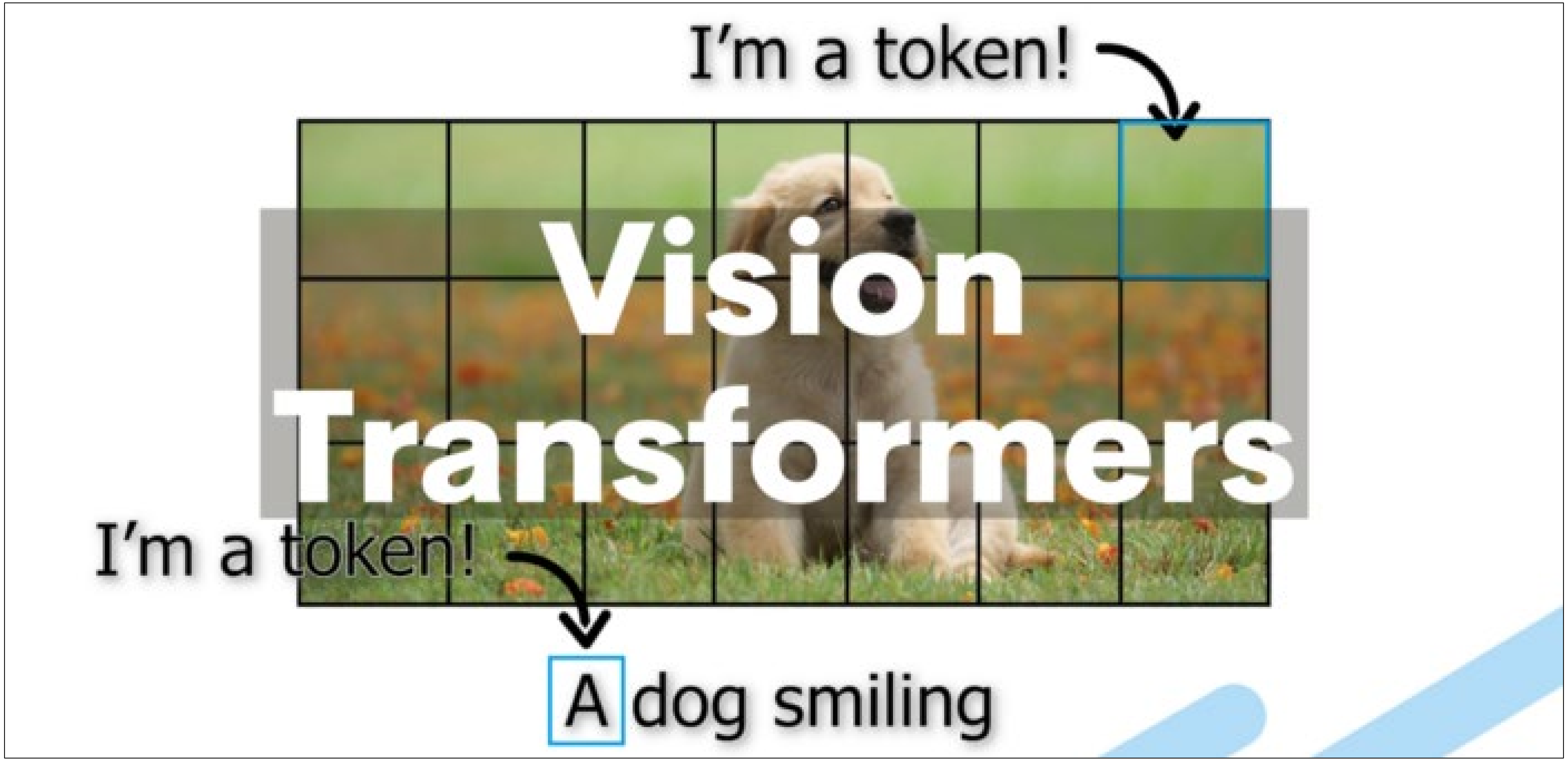
CMSC 475/675 Neural Networks



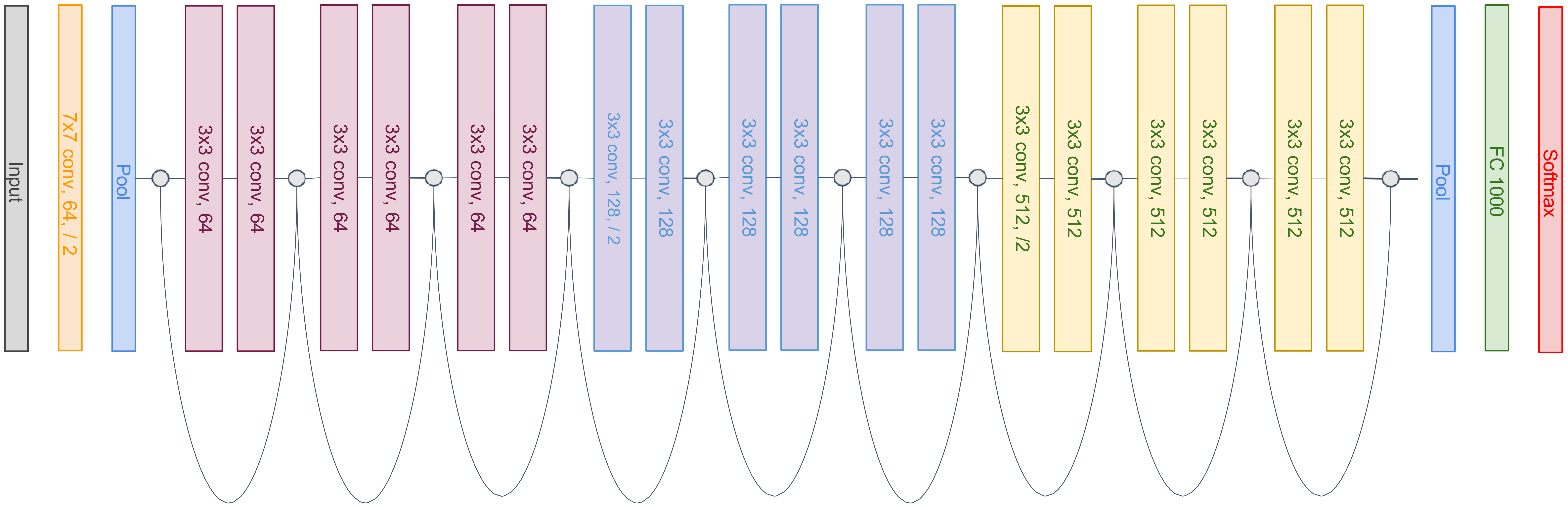
UMBC

Most slides from Justin Johnson

How to use Attention / Transformers for Vision?

Idea #1: Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)

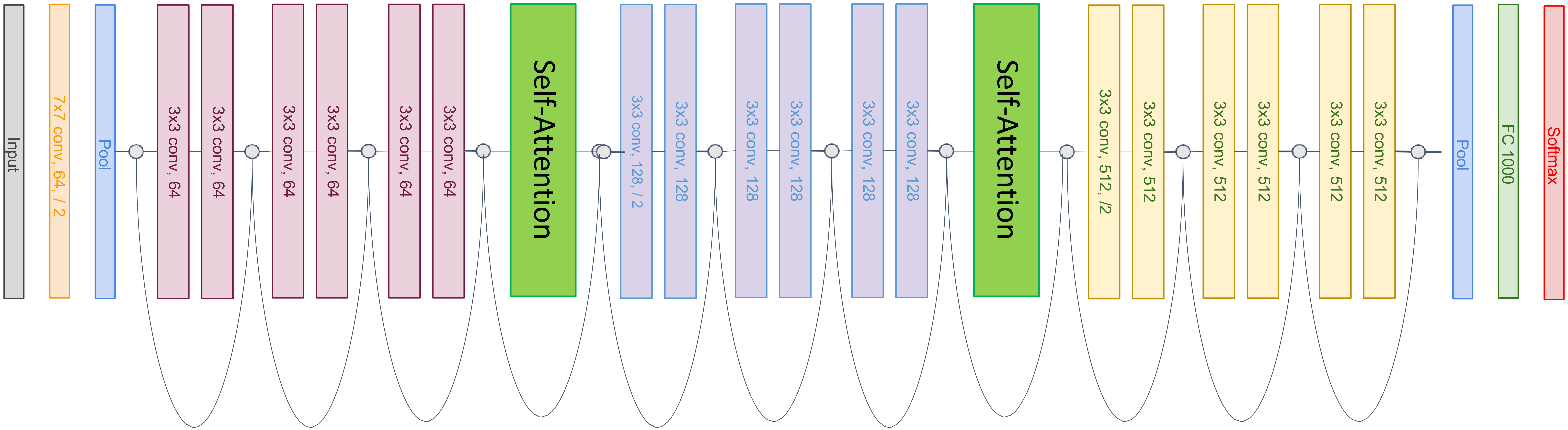


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)

Add Self-Attention blocks between existing ResNet blocks



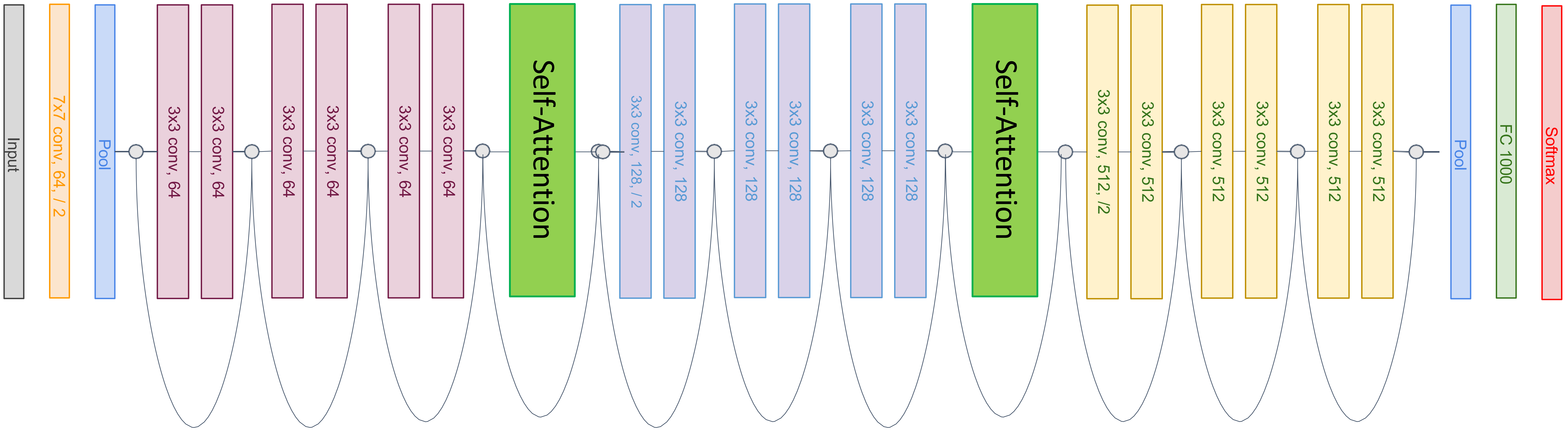
Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Model is still a CNN!
Can we replace convolution entirely?

Start from standard CNN architecture (e.g. ResNet)

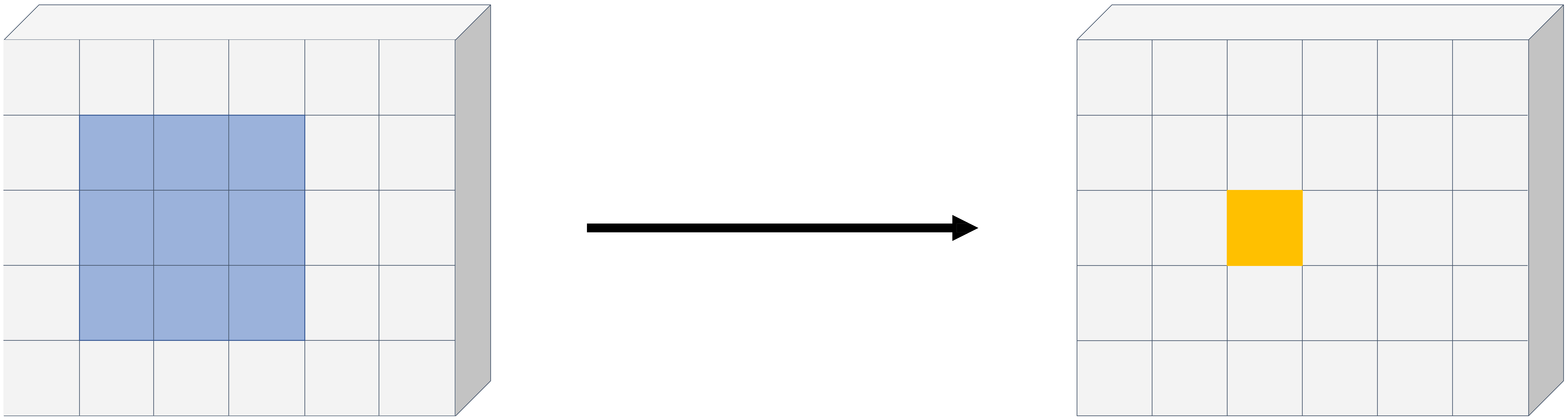
Add Self-Attention blocks between existing ResNet blocks



Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #2: Replace Convolution with "Local Attention"

Convolution: Output at each position is inner product of conv kernel with receptive field in input



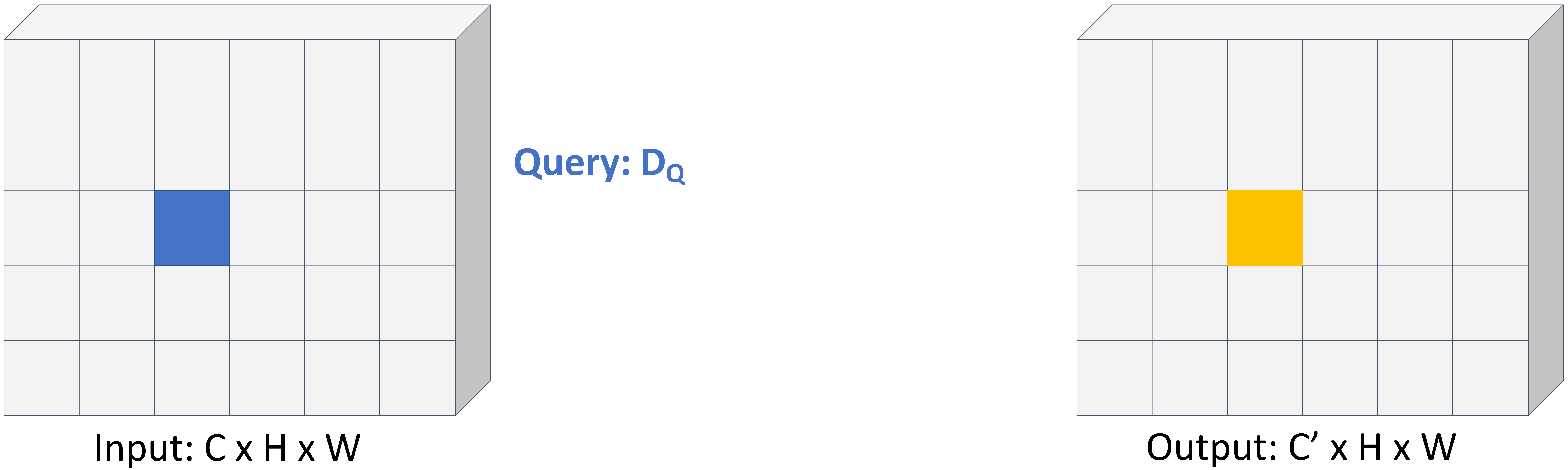
Input: C x H x W

Output: C' x H x W

Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to **query**

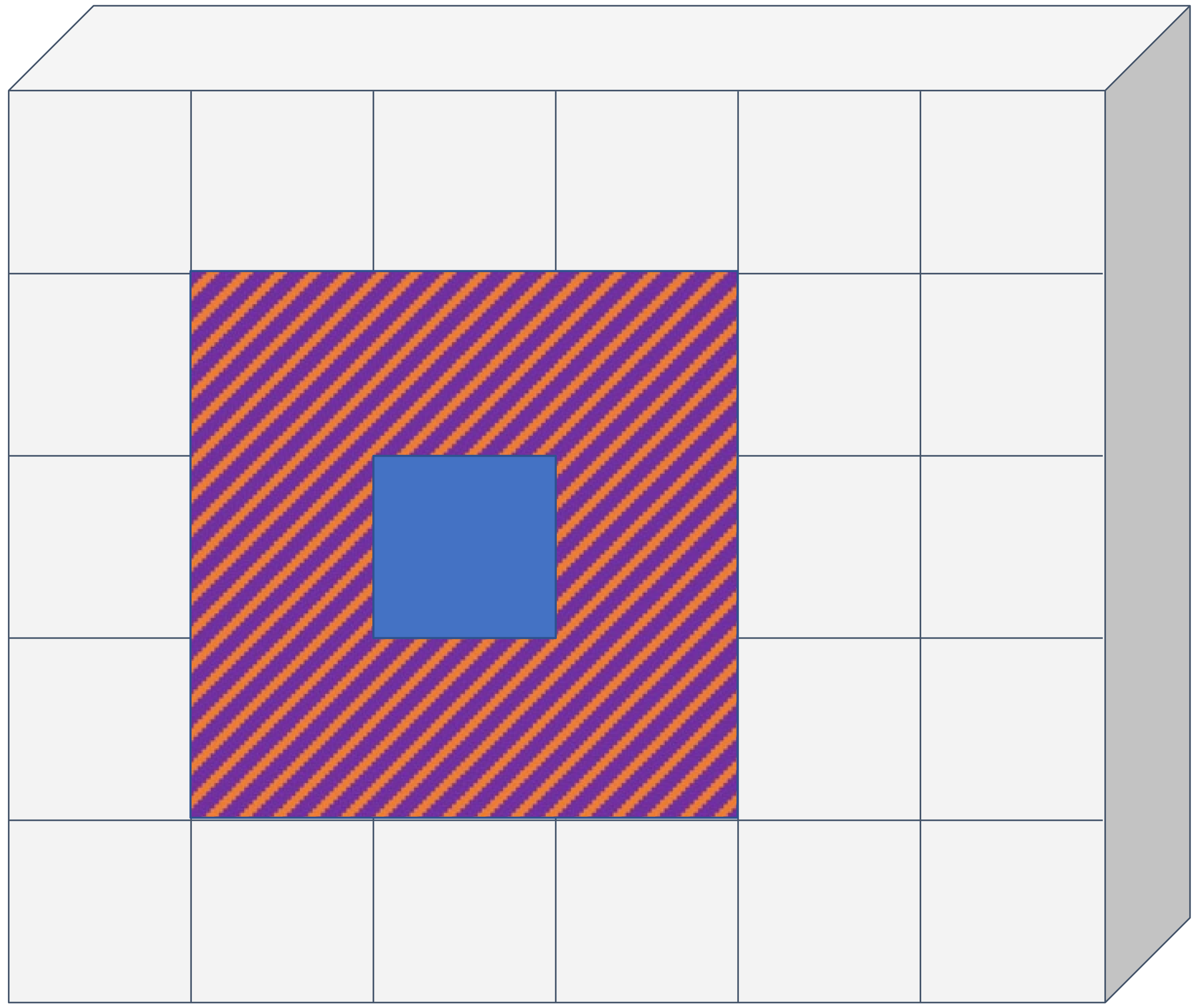


Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to **query**

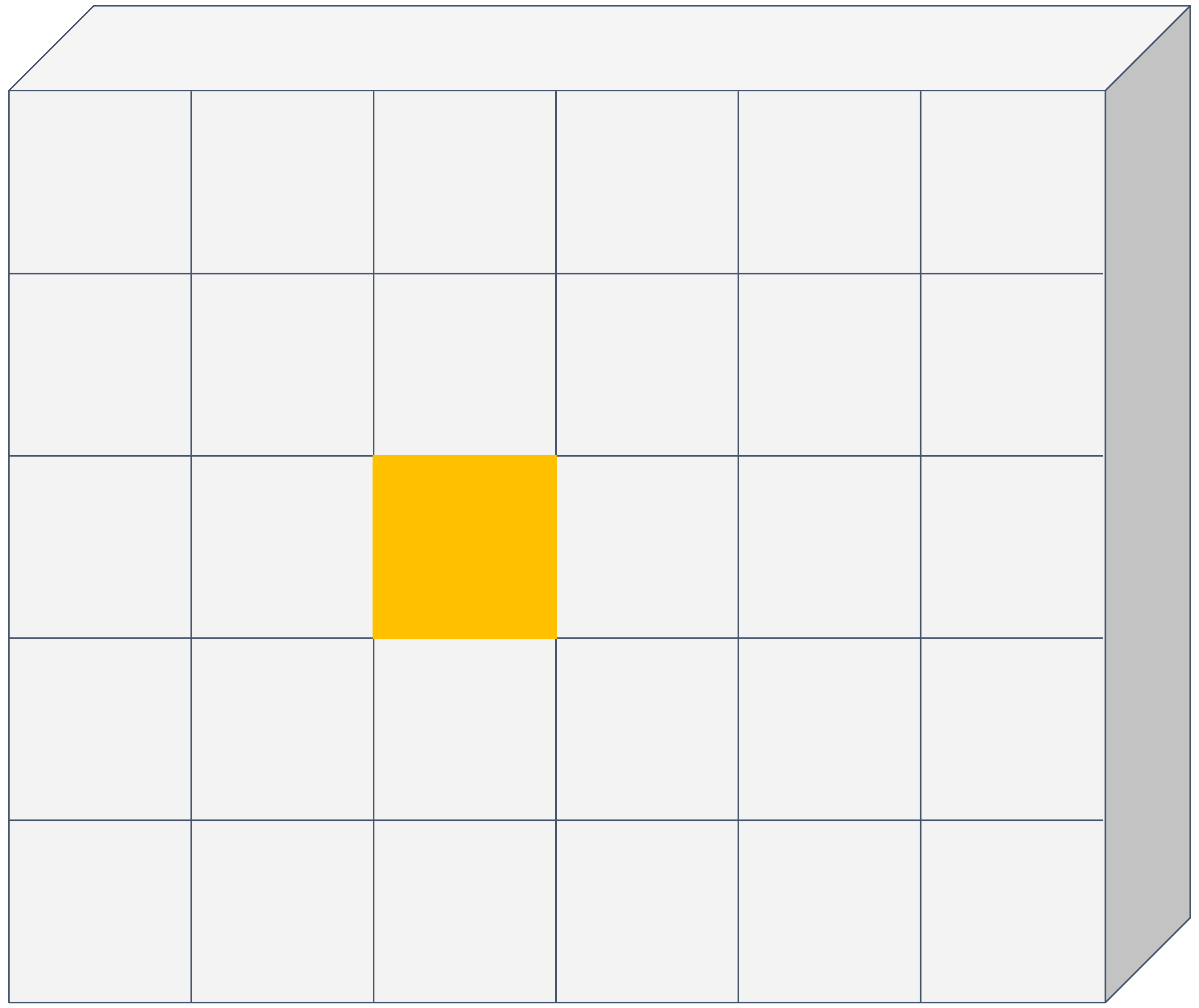
Map each element in receptive field to **key** and **value**



Query: D_Q

Keys: $R \times R \times D_Q$

Values: $R \times R \times C'$



Input: C x H x W

Output: C' x H x W

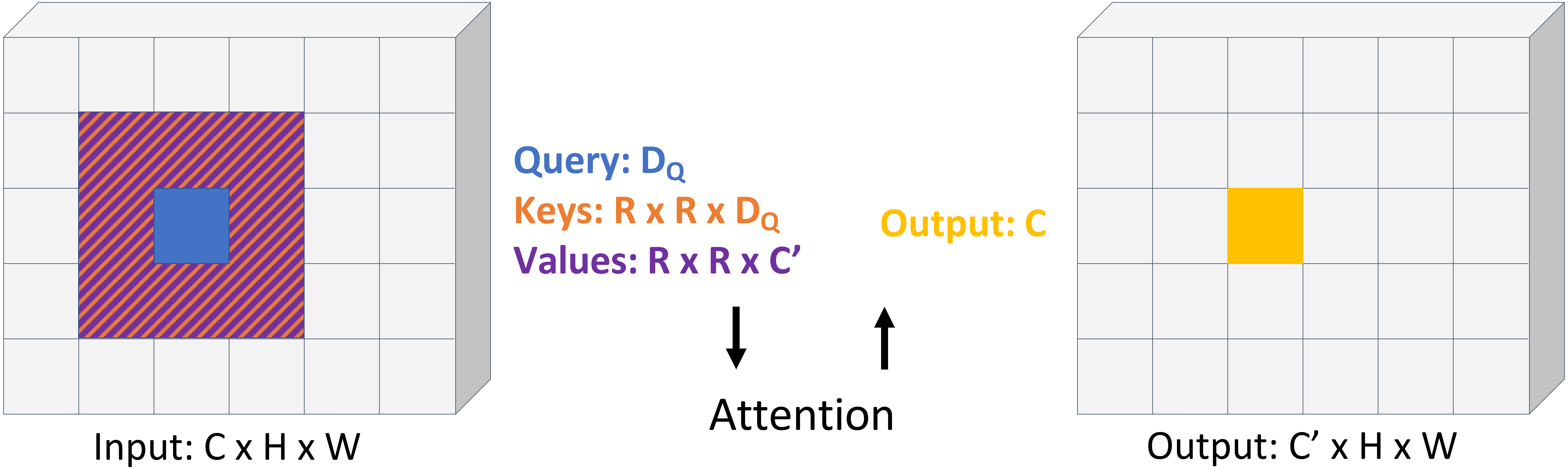
Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to **query**

Map each element in receptive field to **key** and **value**

Compute **output** using attention



Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to **query**
Map each element in receptive field to **key** and **value**
Compute **output** using attention
Replace all conv in ResNet with local attention

LR = "Local Relation"

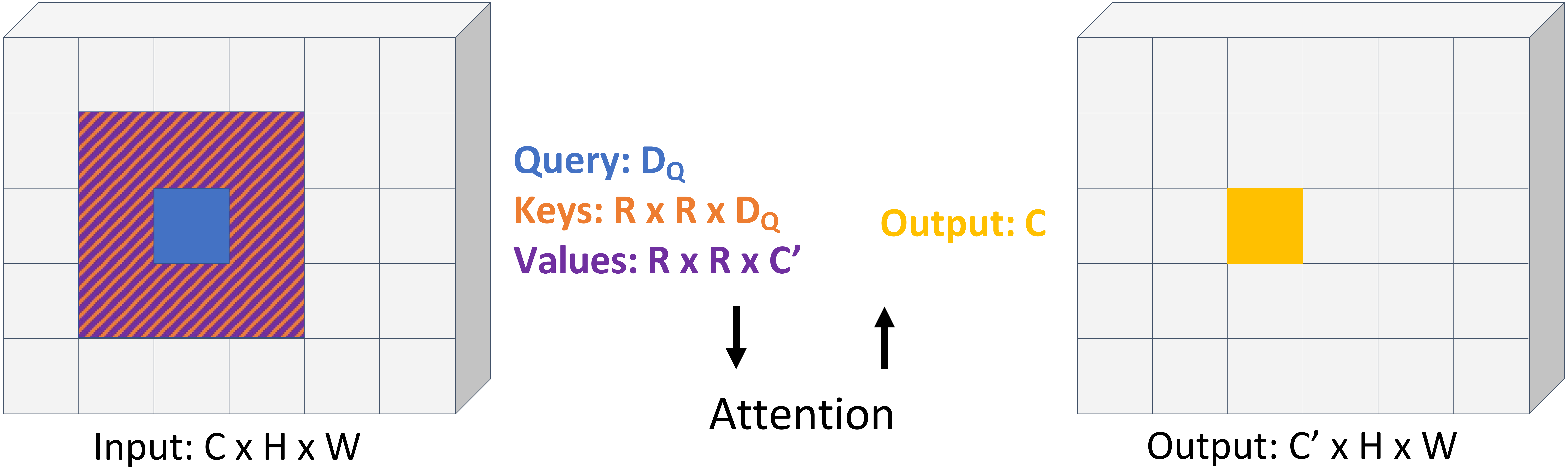
stage	output	ResNet-50		LR-Net-50 (7×7, m=8)	
res1	112×112	7×7 conv, 64, stride 2		**1×1, 64** **7×7 LR, 64, stride 2**	
res2	56×56	3×3 max pool, stride 2		3×3 max pool, stride 2	
		[1×1, 64 3×3 conv, 64 1×1, 256]	×3	[1×1, 100 **7×7 LR, 100** 1×1, 256]	×3
res3	28×28	[1×1, 128 3×3 conv, 128 1×1, 512]	×4	[1×1, 200 **7×7 LR, 200** 1×1, 512]	×4
res4	14×14	[1×1, 256 3×3 conv, 256 1×1, 1024]	×6	[1×1, 400 **7×7 LR, 400** 1×1, 1024]	×6
res5	7×7	[1×1, 512 3×3 conv, 512 1×1, 2048]	×3	[1×1, 800 **7×7 LR, 800** 1×1, 2048]	×3
	1×1	global average pool 1000-d fc, softmax		global average pool 1000-d fc, softmax	
# params		**25.5**$\times 10^6$		**23.3**$\times 10^6$	
FLOPs		**4.3**$\times 10^9$		**4.3**$\times 10^9$	

Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019;
Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to **query**
Map each element in receptive field to **key** and **value**
Compute **output** using attention
Replace all conv in ResNet with local attention

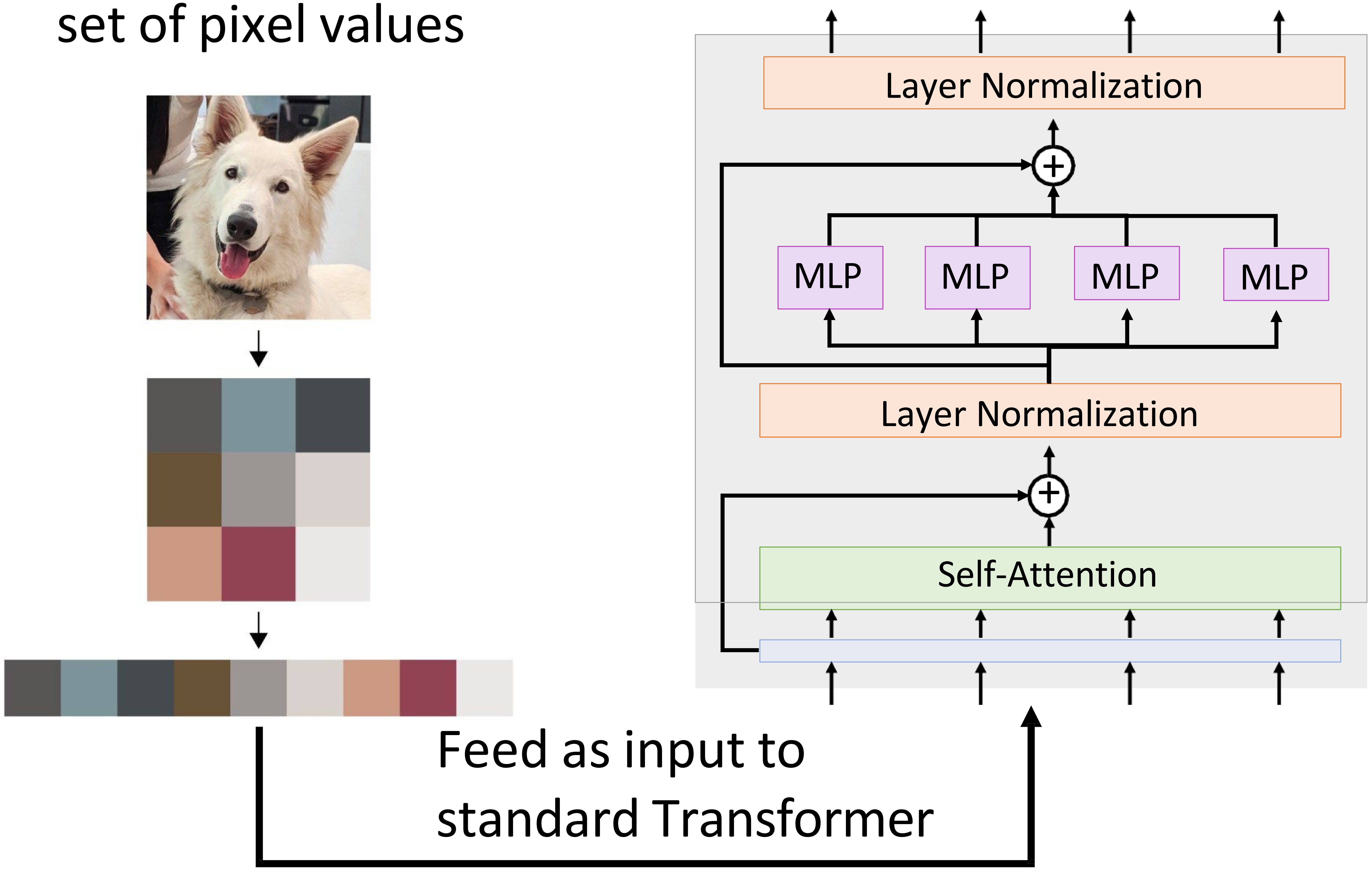
Lots of tricky details, hard to implement, only marginally better than ResNets



Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019; Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #3: Standard Transformer on Pixels

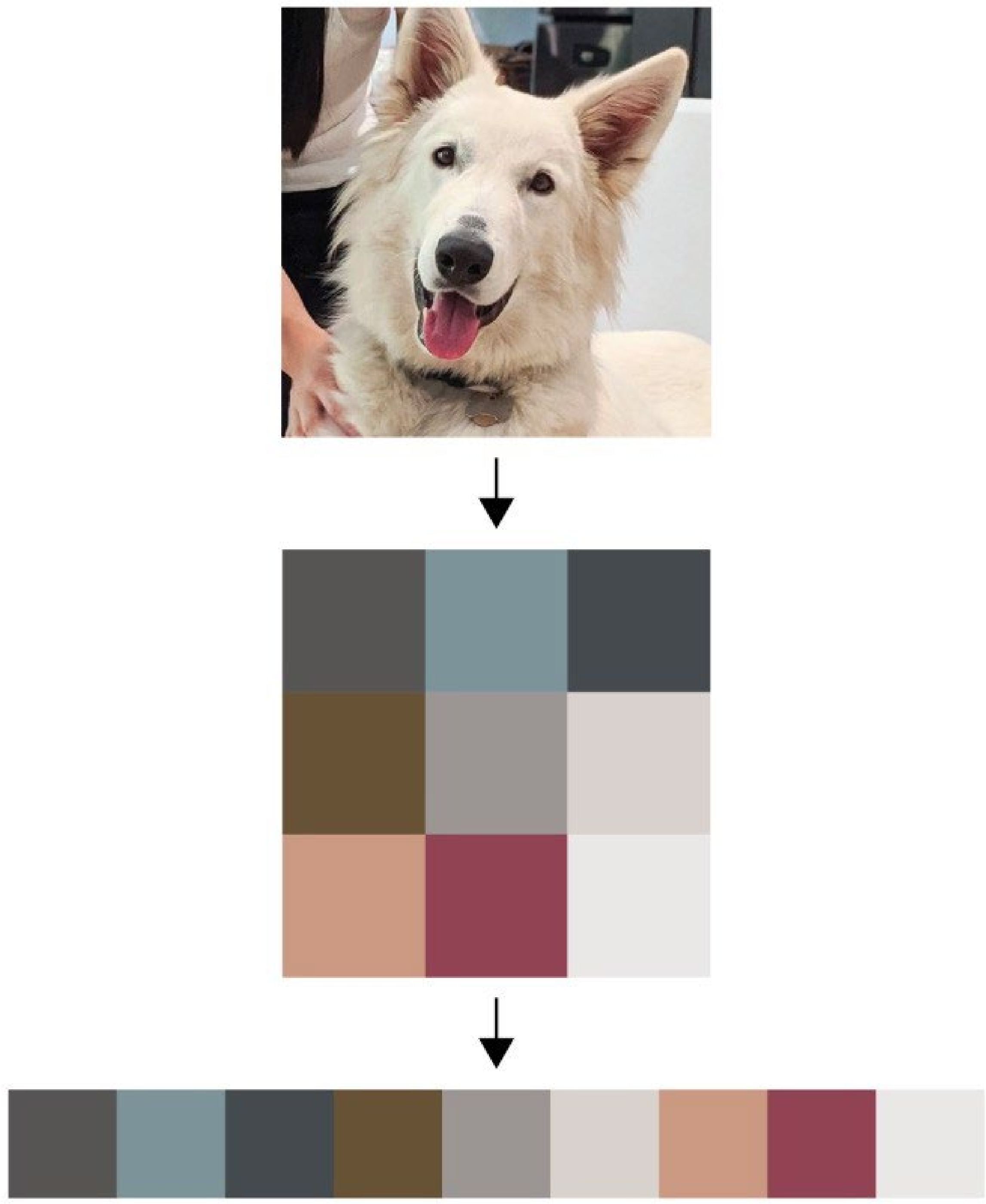
Treat an image as a set of pixel values

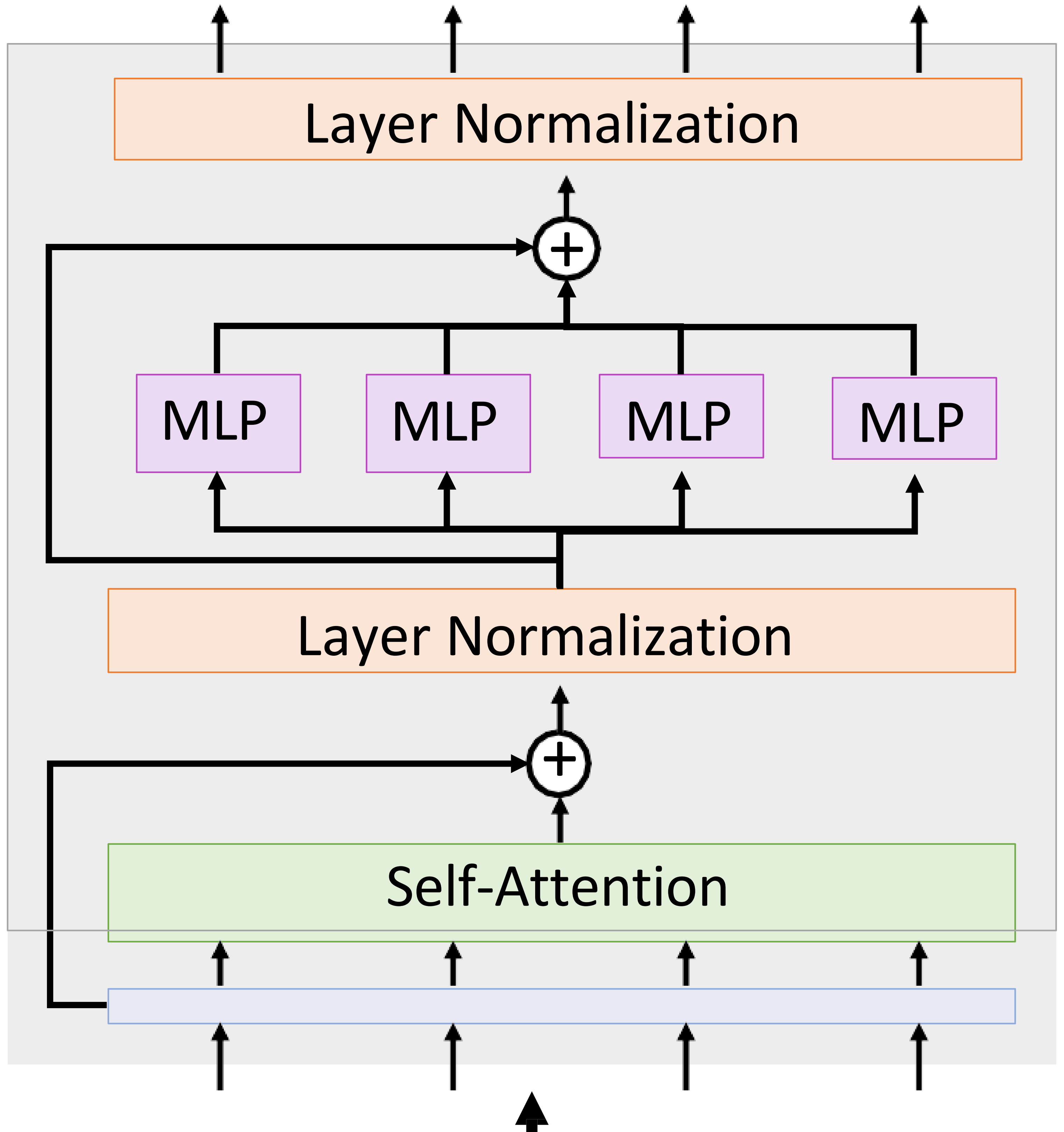


Chen et al, "Generative Pretraining from Pixels", ICML 2020

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values





Feed as input to standard Transformer

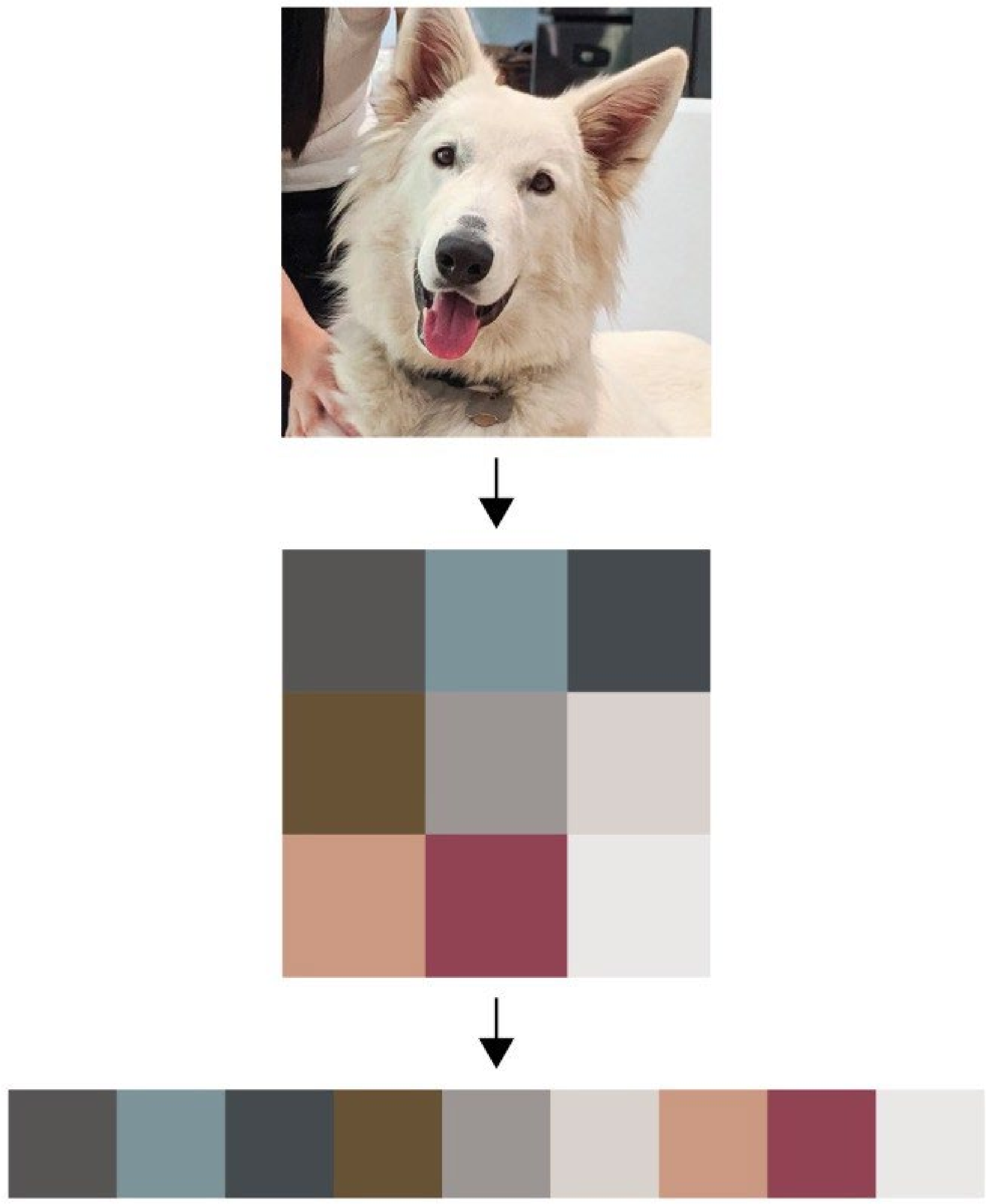
Problem: Memory use!

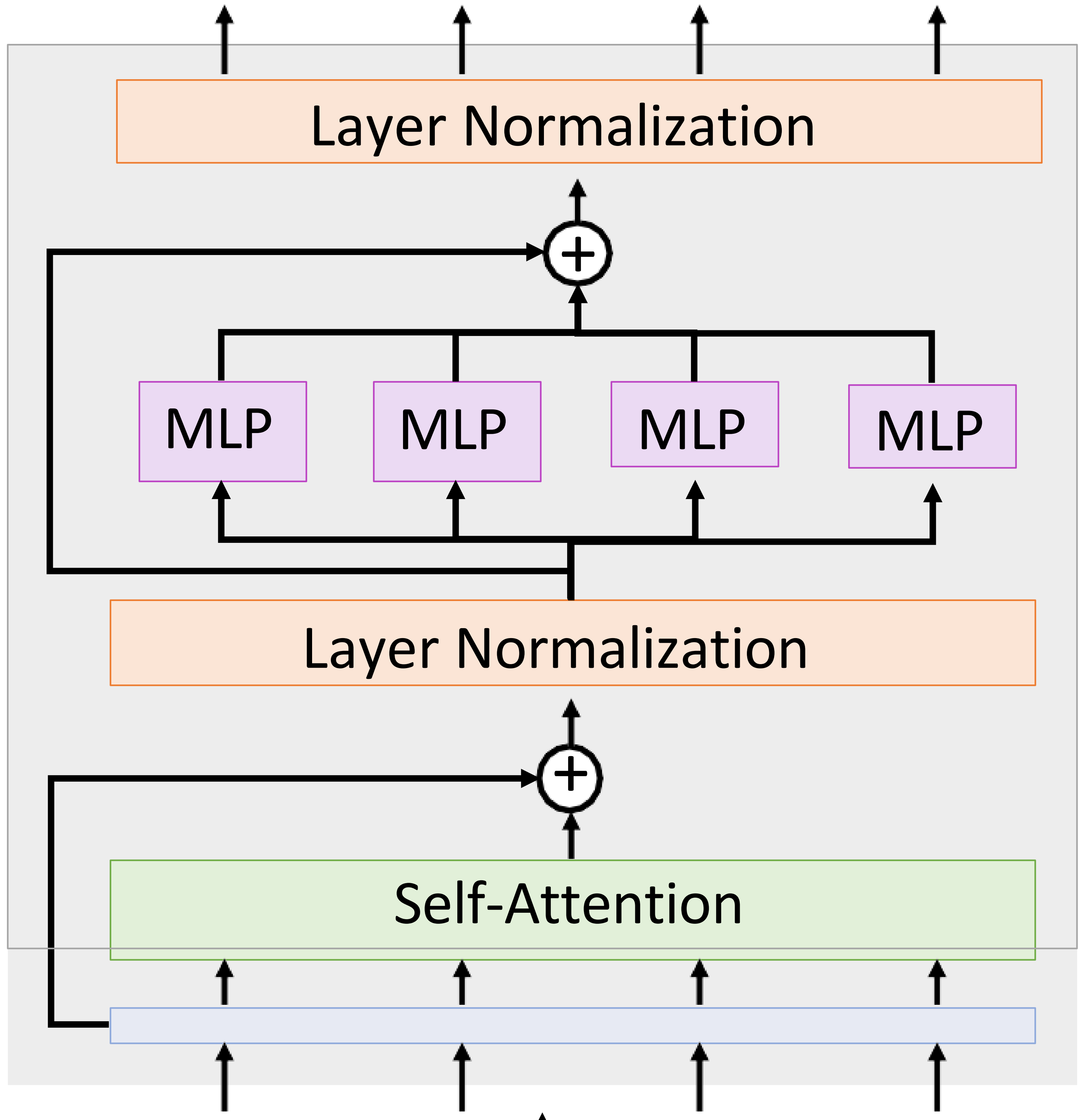
R x R image needs R^4 elements per attention matrix

Chen et al, "Generative Pretraining from Pixels", ICML 2020

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values





Feed as input to standard Transformer

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…

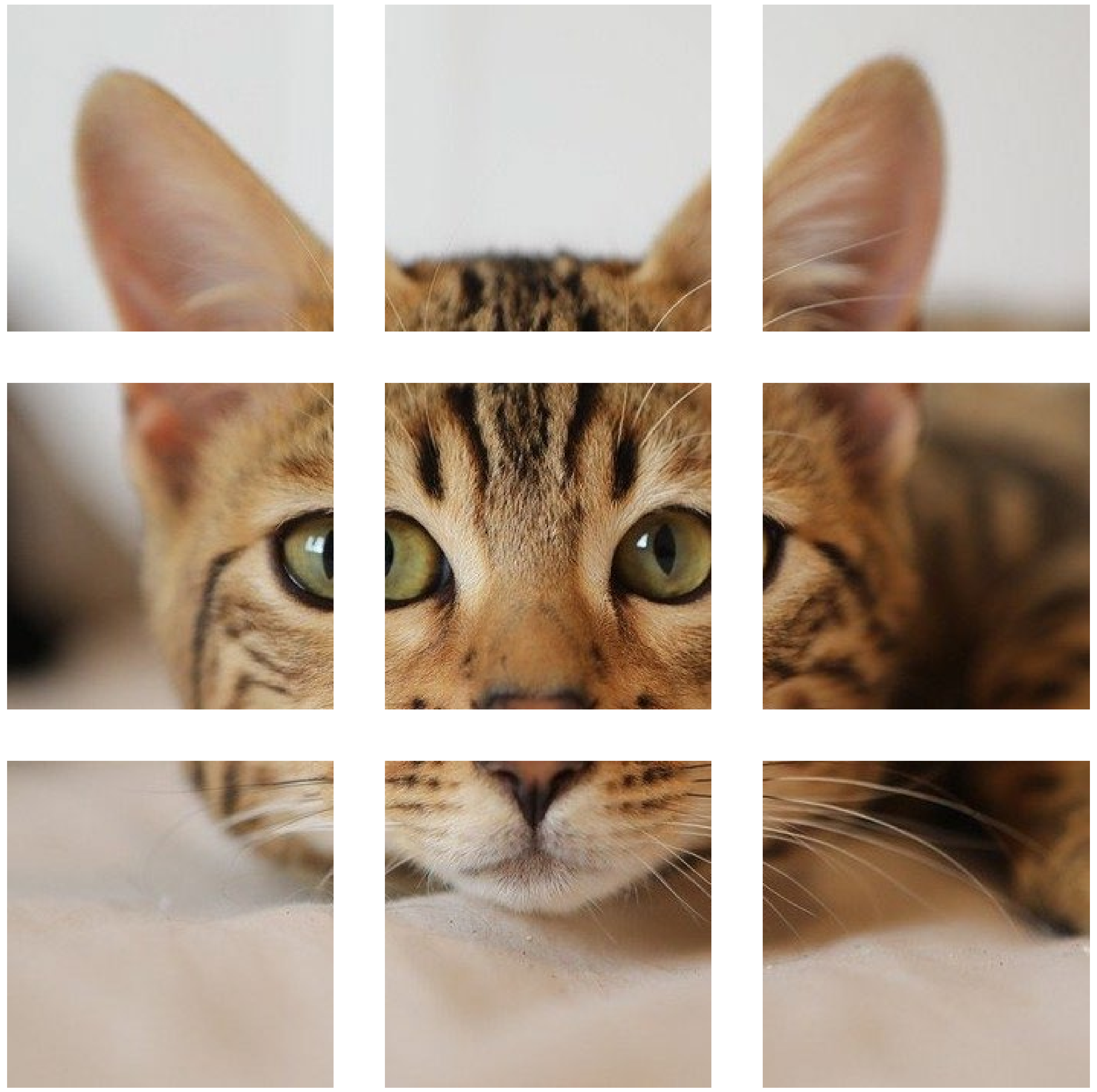
Chen et al, "Generative Pretraining from Pixels", ICML 2020

Idea #4: Standard Transformer on Patches

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

Cat image is free for commercial use under a Pixabay license

Idea #4: Standard Transformer on Patches



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

Cat image is free for commercial use under a Pixabay license

Idea #4: Standard Transformer on Patches

N input patches, each of shape 3x16x16

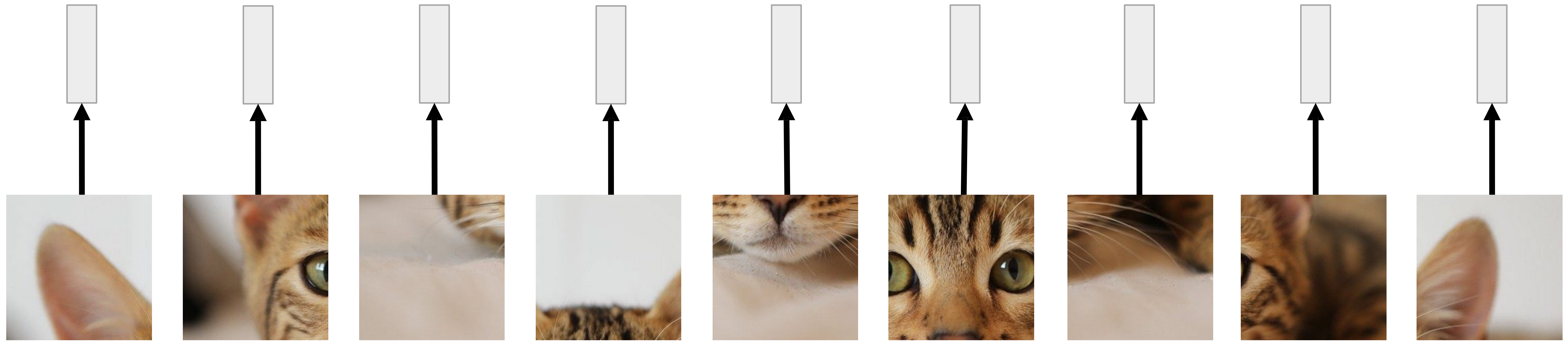
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea #4: Standard Transformer on Patches

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16



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

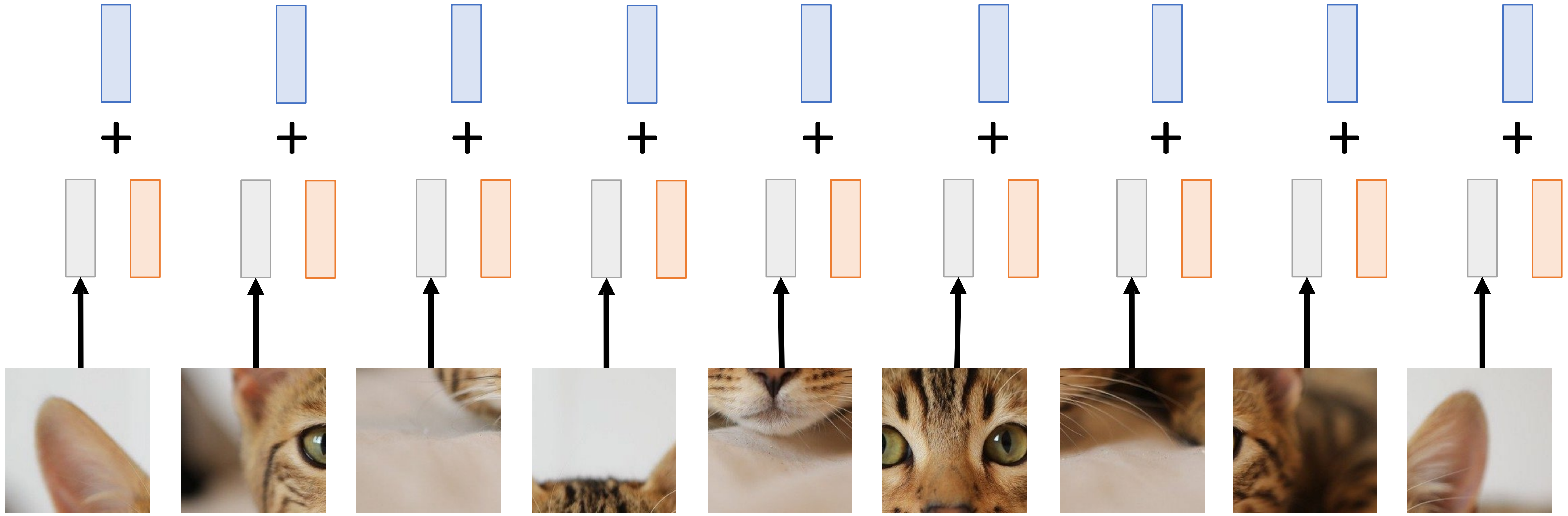
Cat image is free for commercial use under a Pixabay license

Idea #4: Standard Transformer on Patches

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

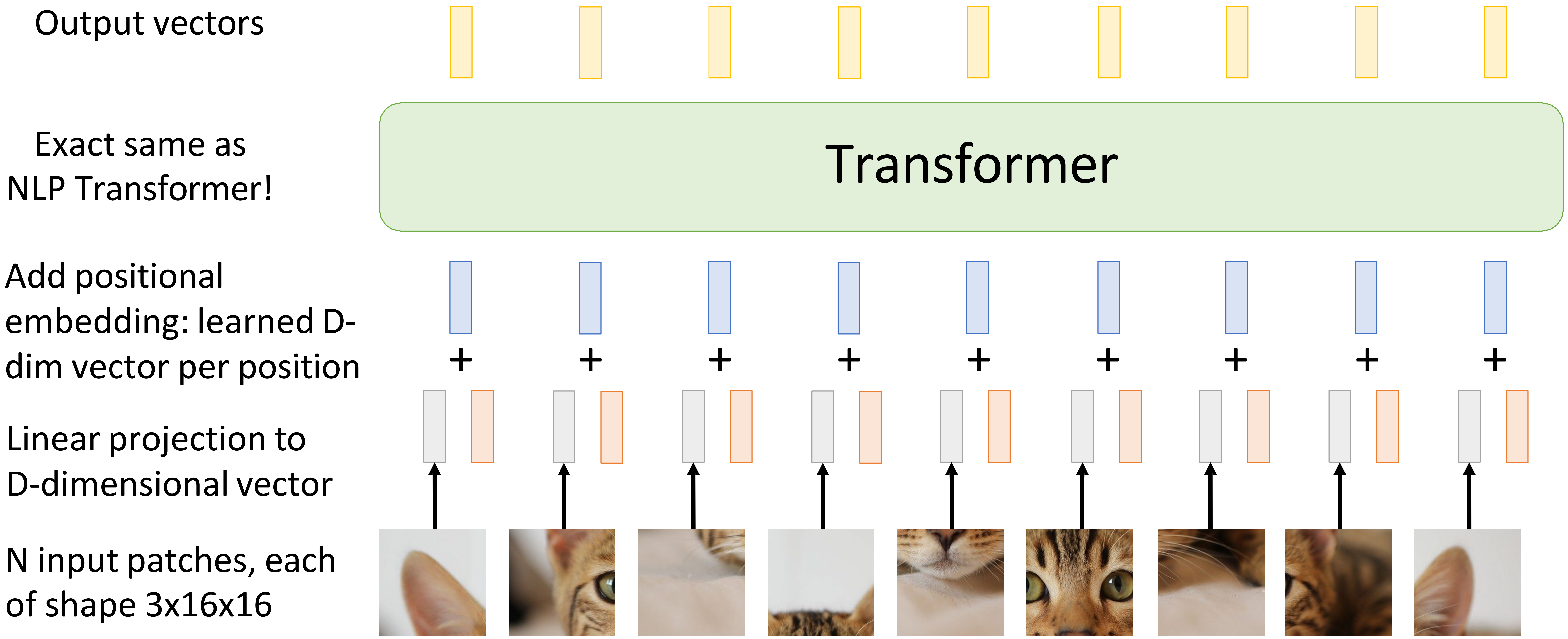
N input patches, each of shape 3x16x16



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

Cat image is free for commercial use under a Pixabay license

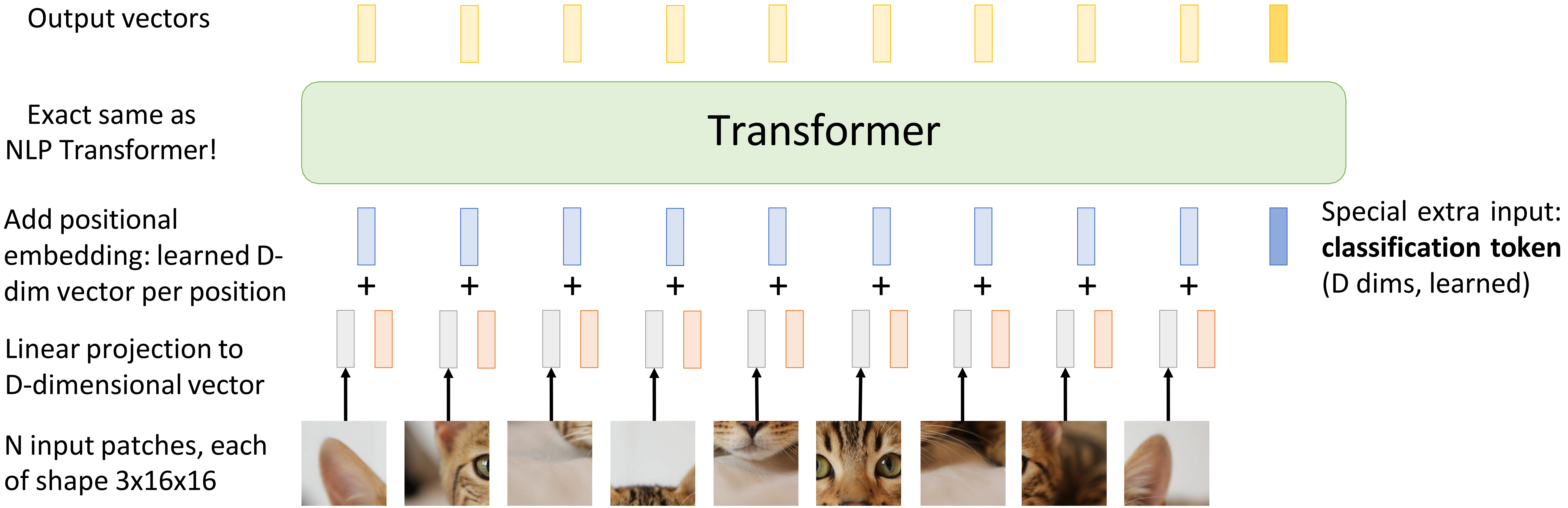
Idea #4: Standard Transformer on Patches



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

[Cat image](#) is free for commercial use under a [Pixabay license](#)

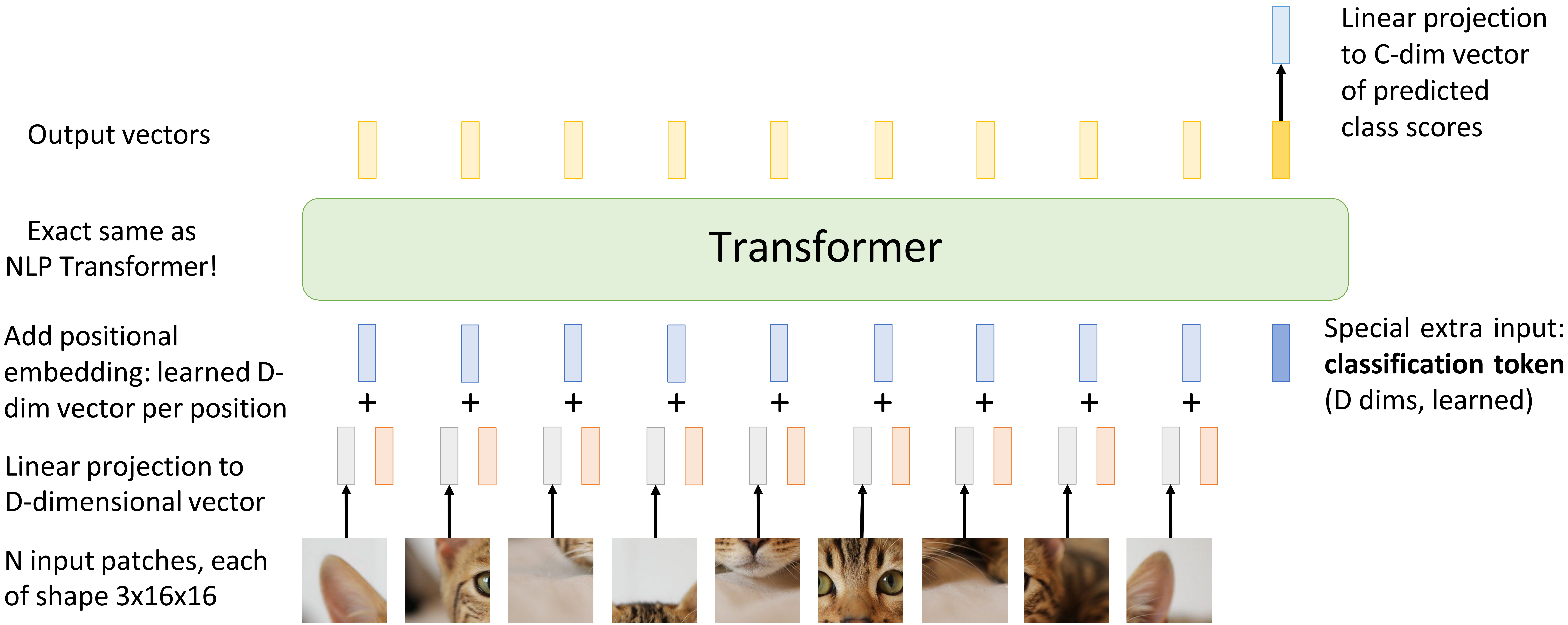
Idea #4: Standard Transformer on Patches



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

Cat image is free for commercial use under a Pixabay license

Idea #4: Standard Transformer on Patches

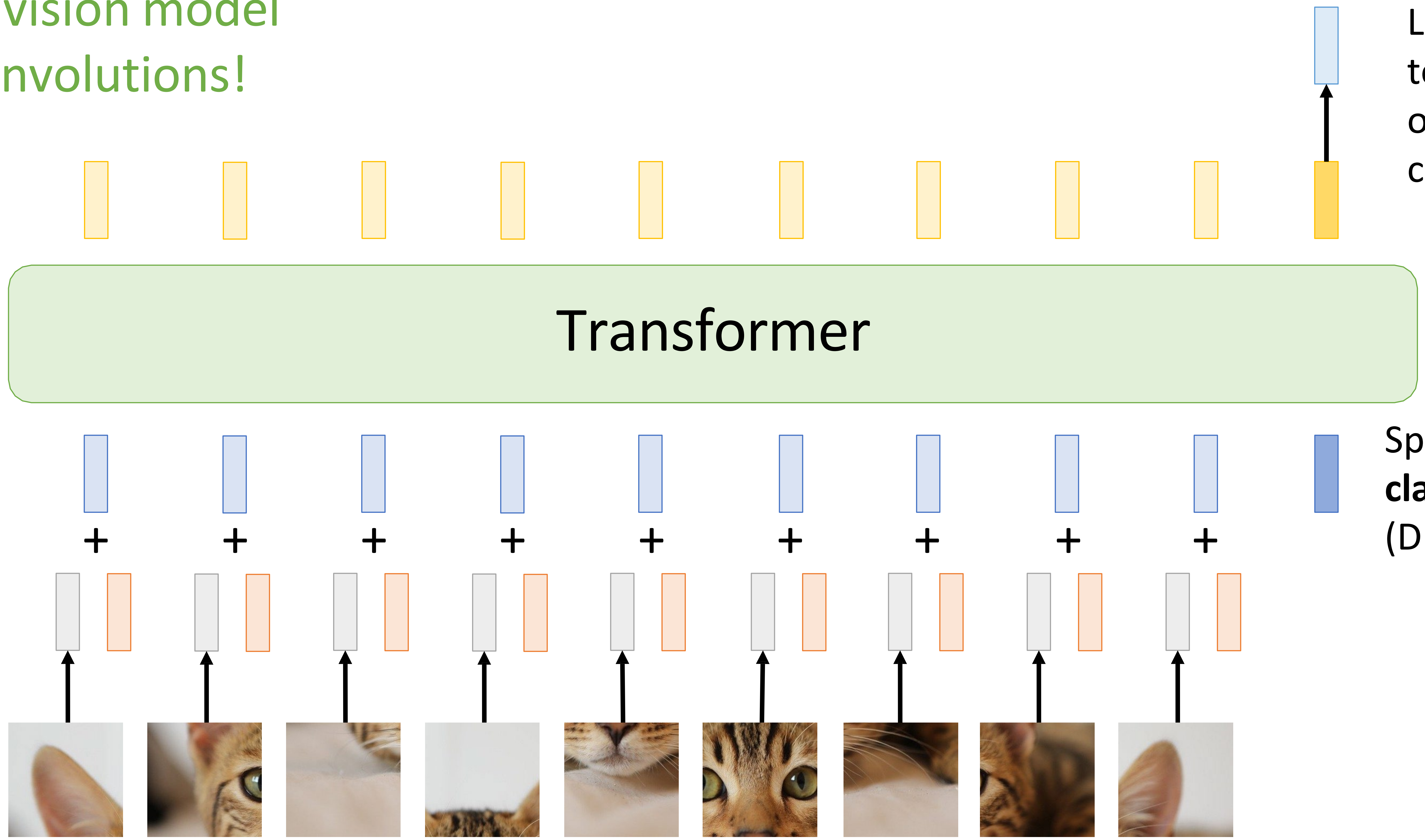


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

[Cat image](#) is free for commercial use under a [Pixabay license](#)

Vision Transformer (ViT)

Computer vision model
with no convolutions!



Output vectors

Exact same as
NLP Transformer!

Add positional
embedding: learned D-
dim vector per position

Linear projection to
D-dimensional vector

N input patches, each
of shape 3x16x16

Linear projection
to C-dim vector
of predicted
class scores

Special extra input:
classification token
(D dims, learned)

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

Cat image is free for commercial
use under a Pixabay license

Vision Transformer (ViT)

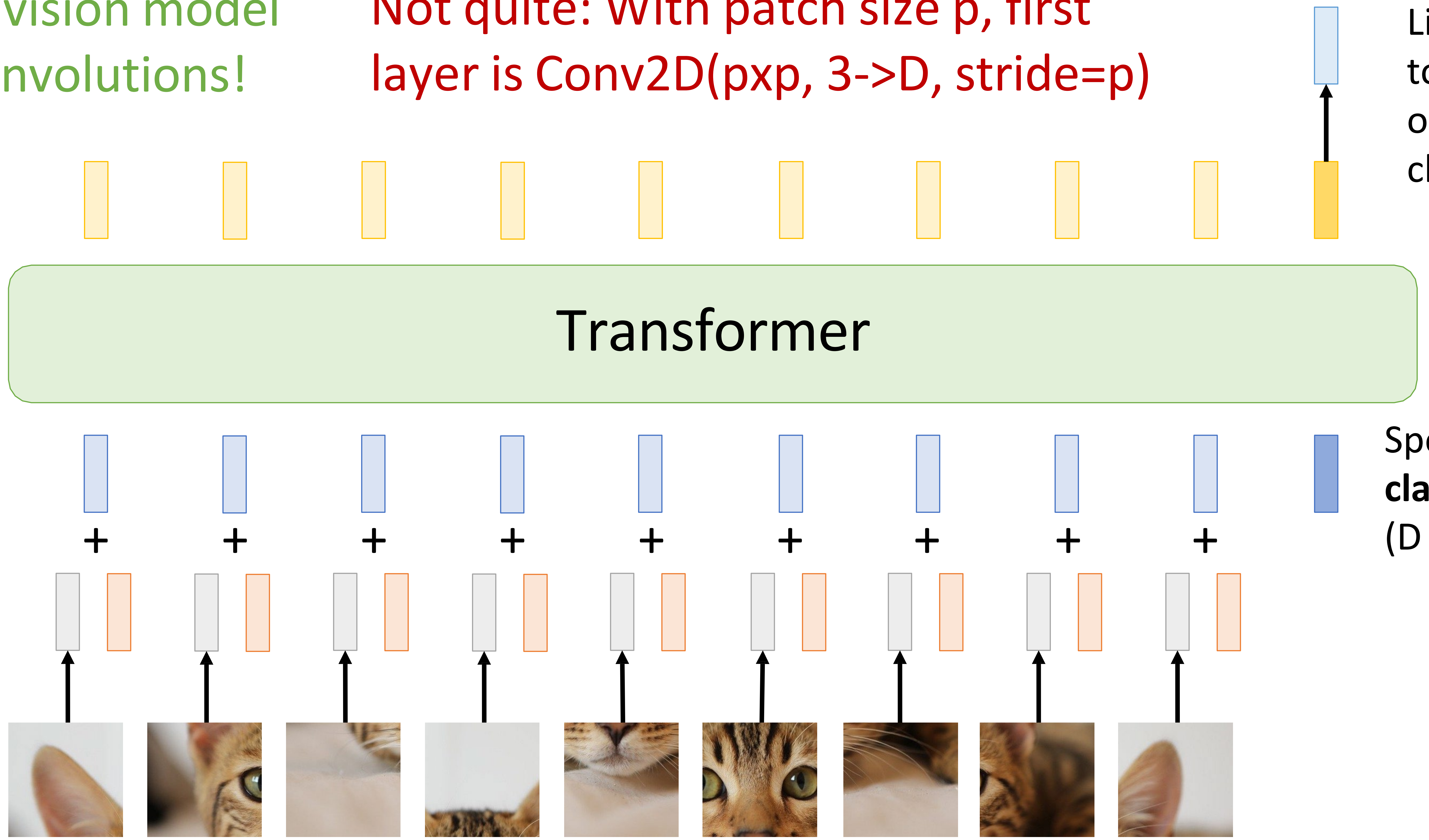
Computer vision model with no convolutions!

Not quite: With patch size p, first layer is Conv2D(pxp, 3->D, stride=p)

Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!



Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

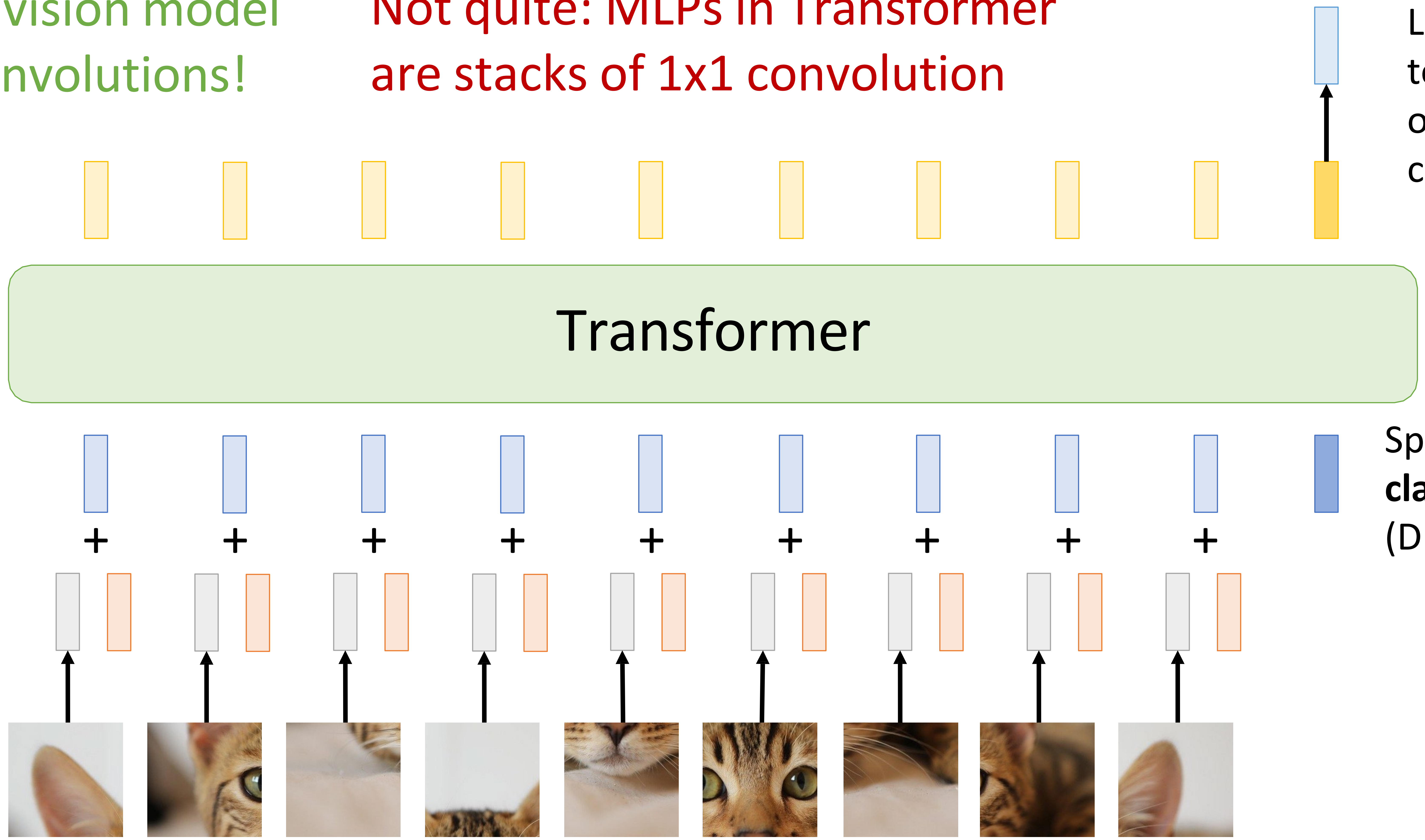
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

Computer vision model with no convolutions!

Not quite: MLPs in Transformer are stacks of 1x1 convolution



Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

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

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

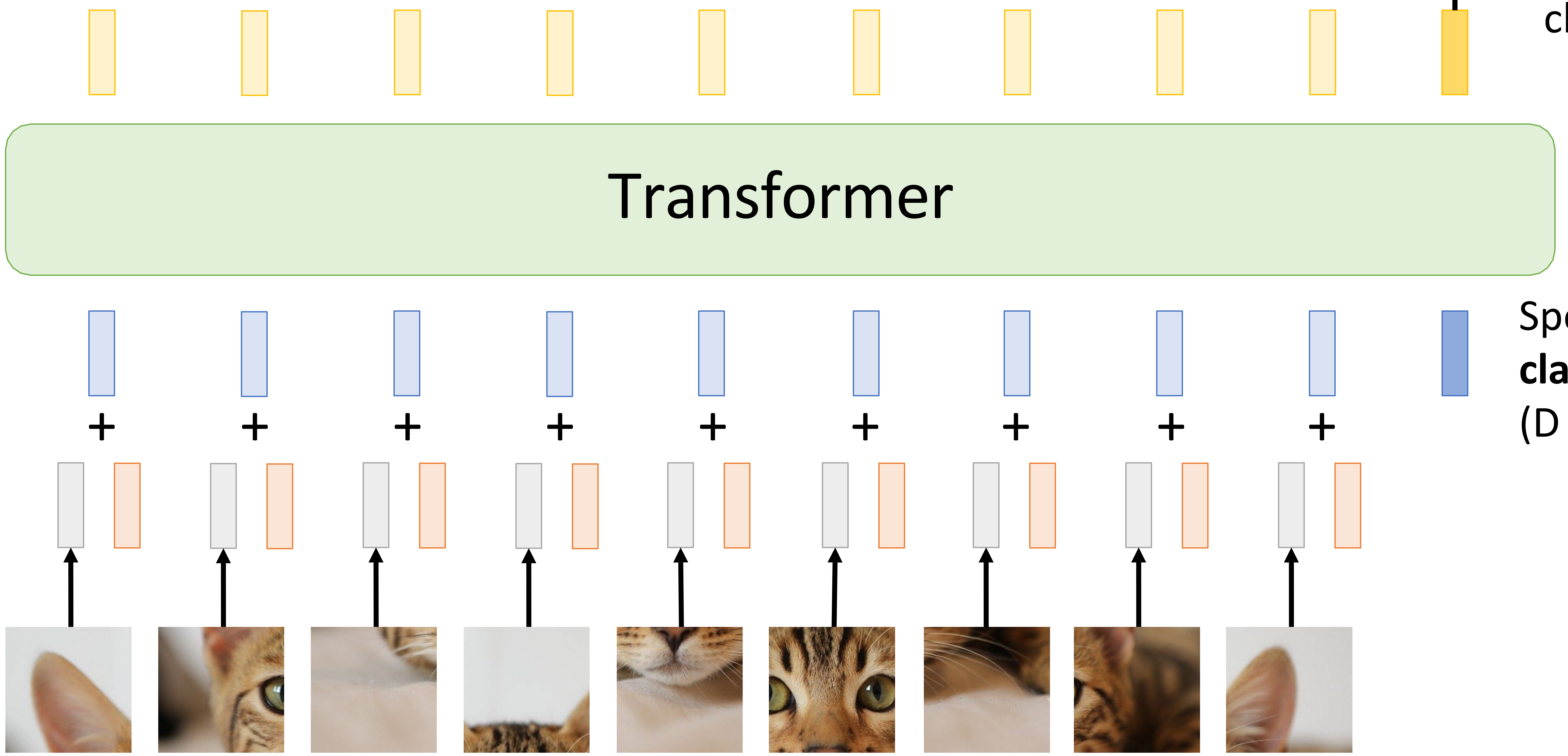
In practice: take 224x224 input image, divide into 14x14 grid of 16x16 pixel patches (or 16x16 grid of 14x14 patches)

Each attention matrix has $14^4 = 38,416$ entries, takes 150 KB
(or 65,536 entries, takes 256 KB)

Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!



Special extra input: **classification token** (D dims, learned)

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

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

[Cat image](#) is free for commercial use under a [Pixabay license](#)

Vision Transformer (ViT)

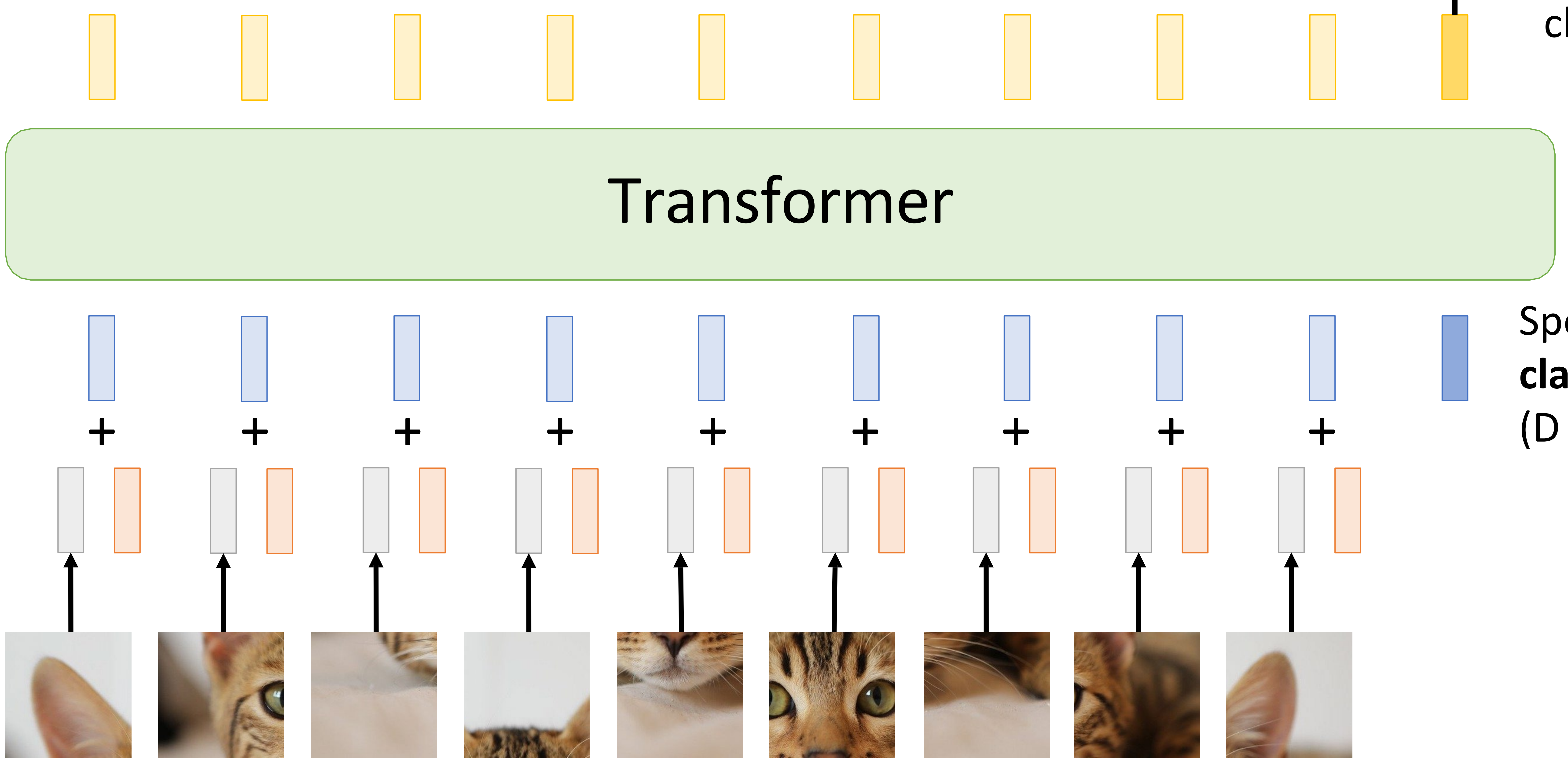
In practice: take 224x224 input image, divide into 14x14 grid of 16x16 pixel patches (or 16x16 grid of 14x14 patches)

With 48 layers, 16 heads per layer, all attention matrices take 112 MB (or 192MB)

Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!



Special extra input: **classification token** (D dims, learned)

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

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

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

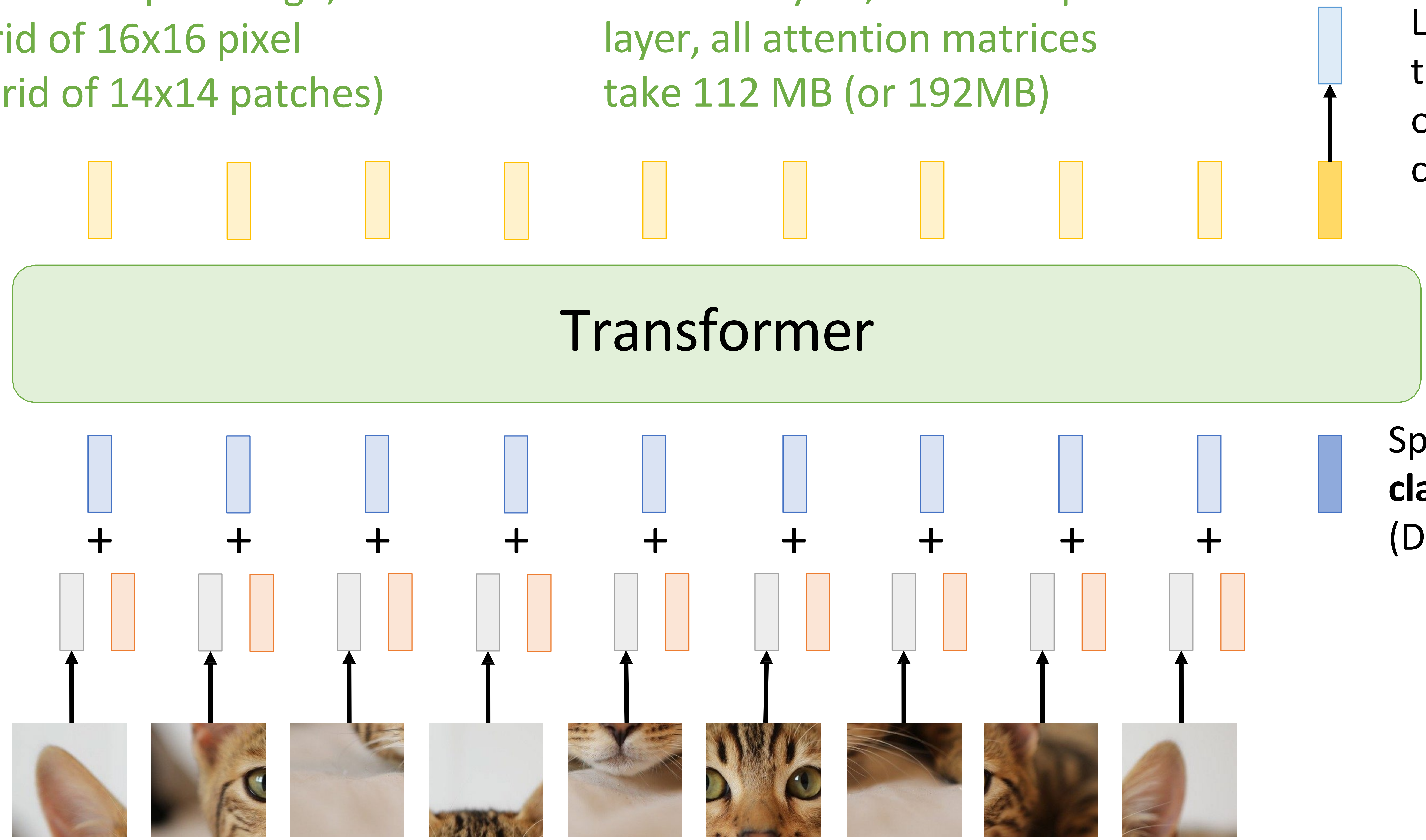
In practice: take 224x224 input image, divide into 14x14 grid of 16x16 pixel patches (or 16x16 grid of 14x14 patches)

With 48 layers, 16 heads per layer, all attention matrices take 112 MB (or 192MB)

Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!



Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

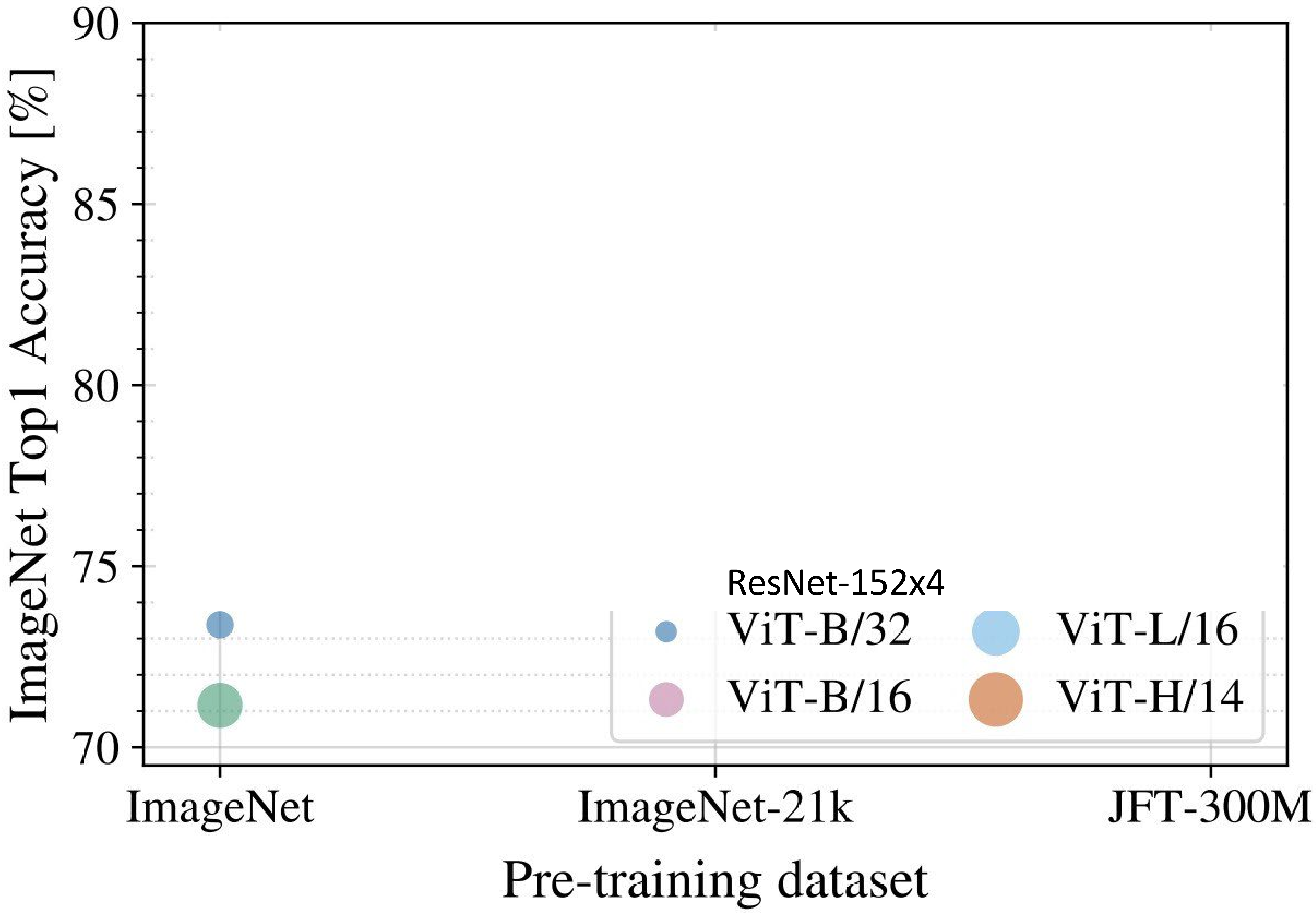
Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

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

[Cat image](#) is free for commercial use under a [Pixabay license](#)

Vision Transformer (ViT) vs ResNets



B = Base
L = Large
H = Huge

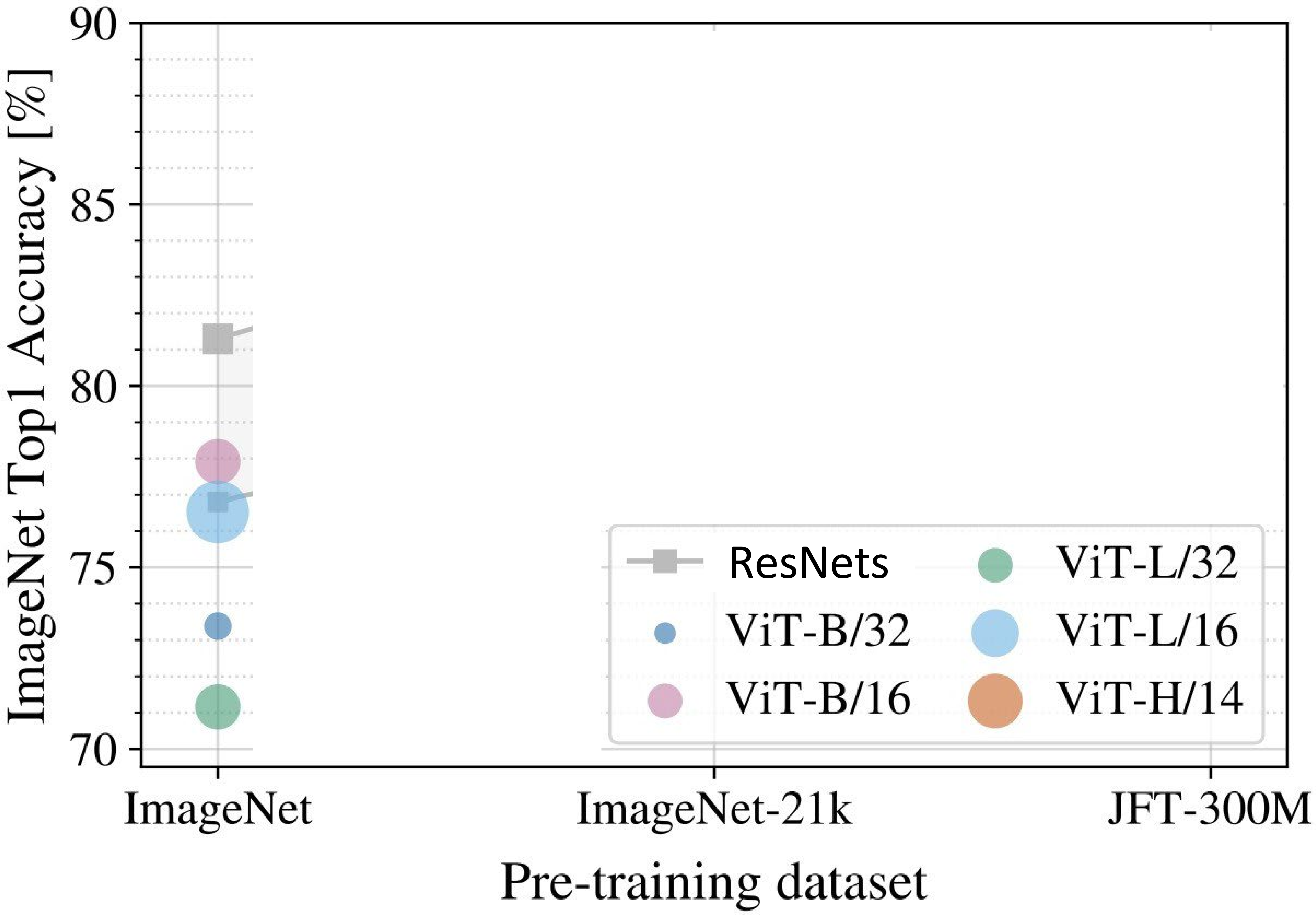
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

Vision Transformer (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

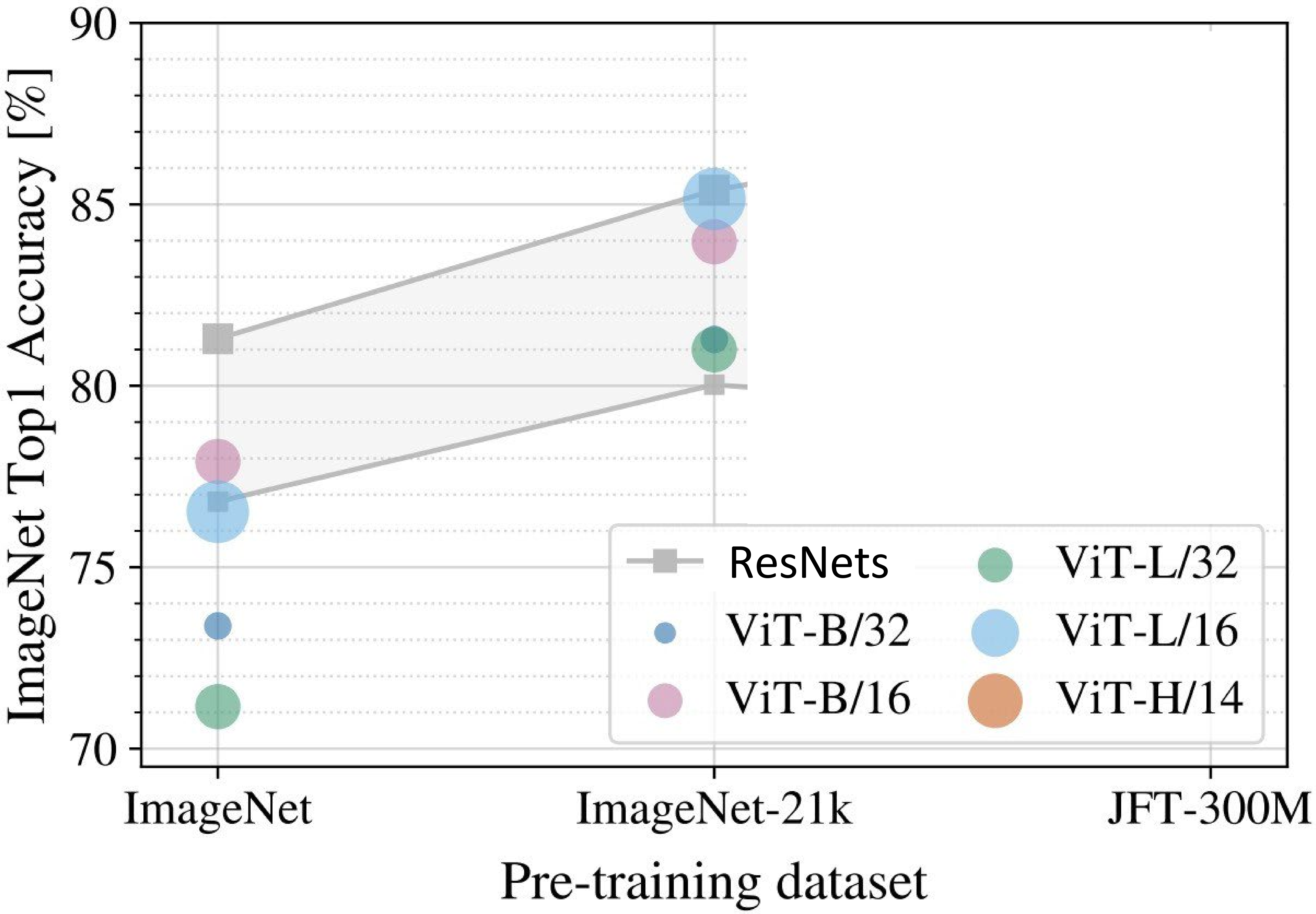
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

Vision Transformer (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

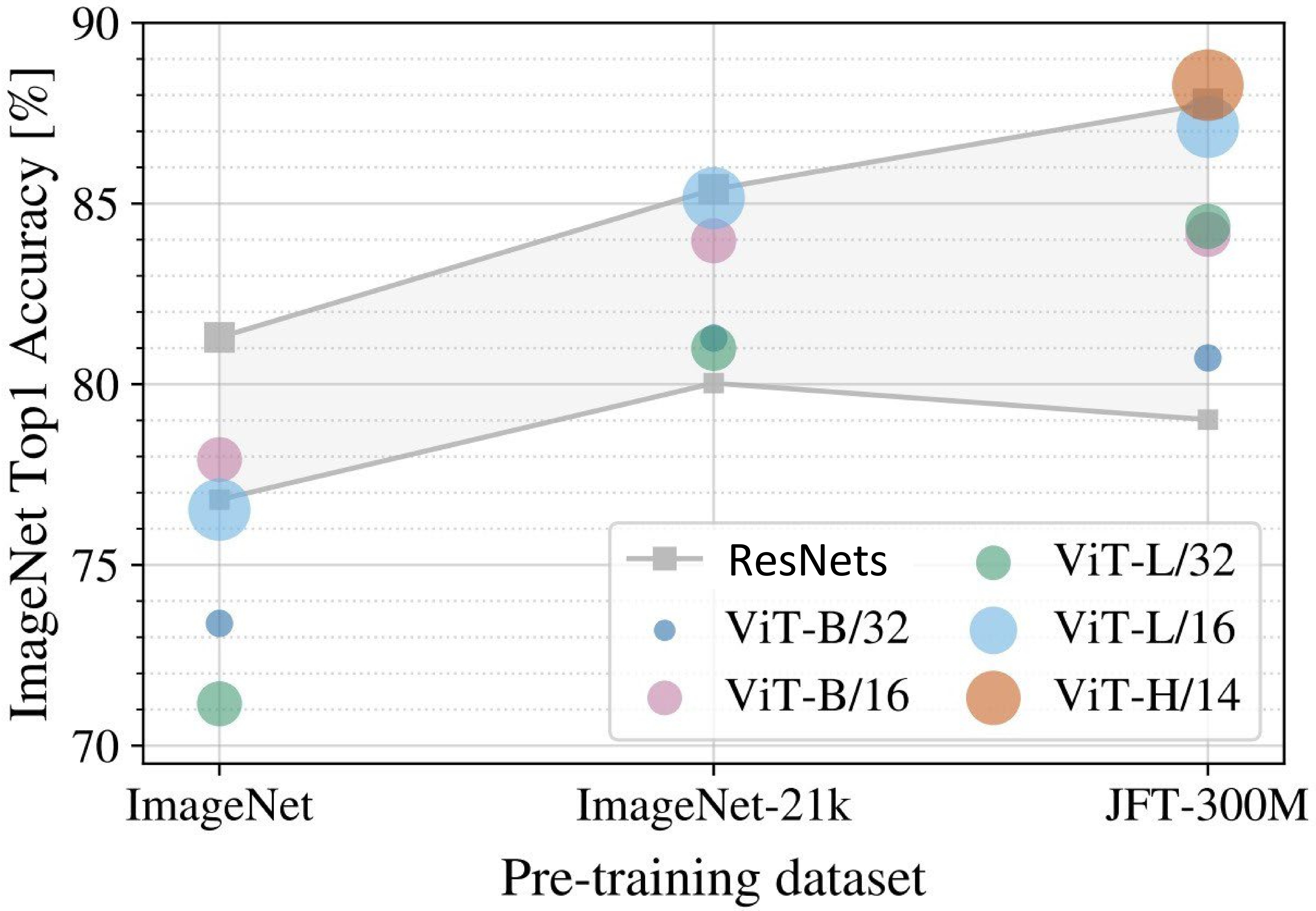
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

Vision Transformer (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

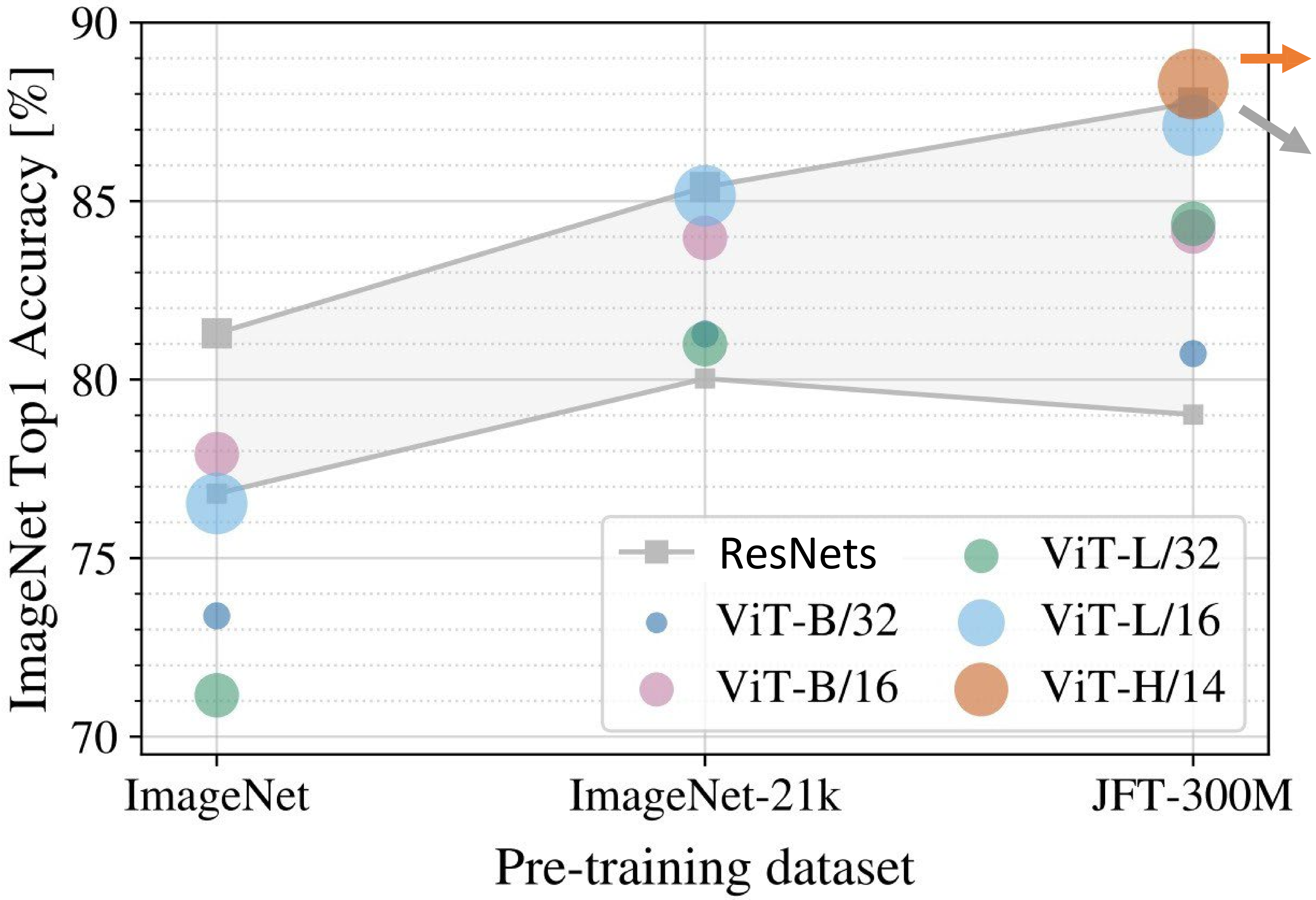
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

Vision Transformer (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



ViT: 2.5k TPU-v3 core days of training

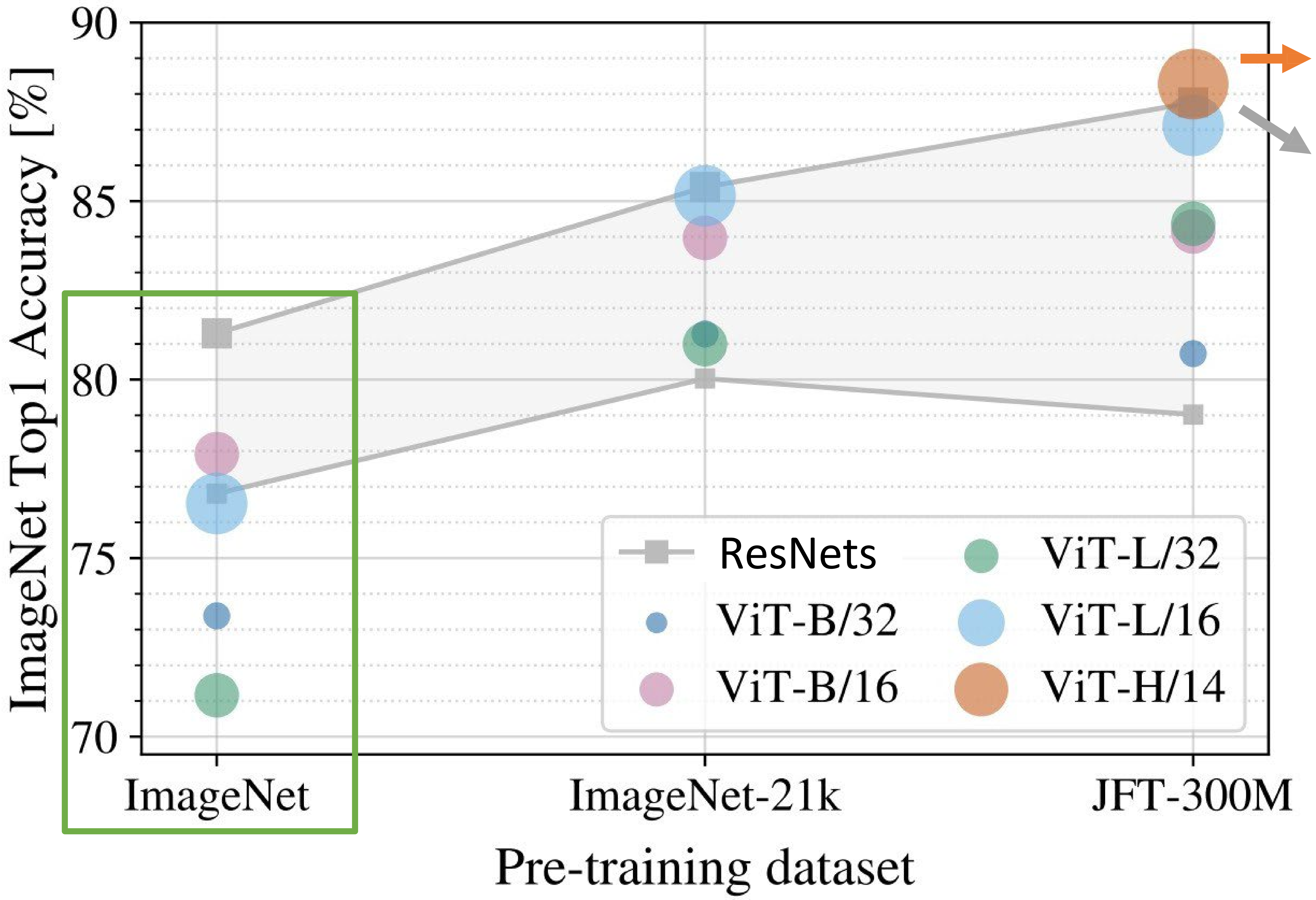
ResNet: 9.9k TPU-v3 core days of training

ViTs make more efficient use of GPU / TPU hardware (matrix multiply is more hardware-friendly than conv)

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

Vision Transformer (ViT) vs ResNets

How can we improve the performance of ViT models on ImageNet?



ViT: 2.5k TPU-v3 core days of training

ResNet: 9.9k TPU-v3 core days of training

ViTs make more efficient use of GPU / TPU hardware (matrix multiply is more hardware-friendly than conv)

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

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

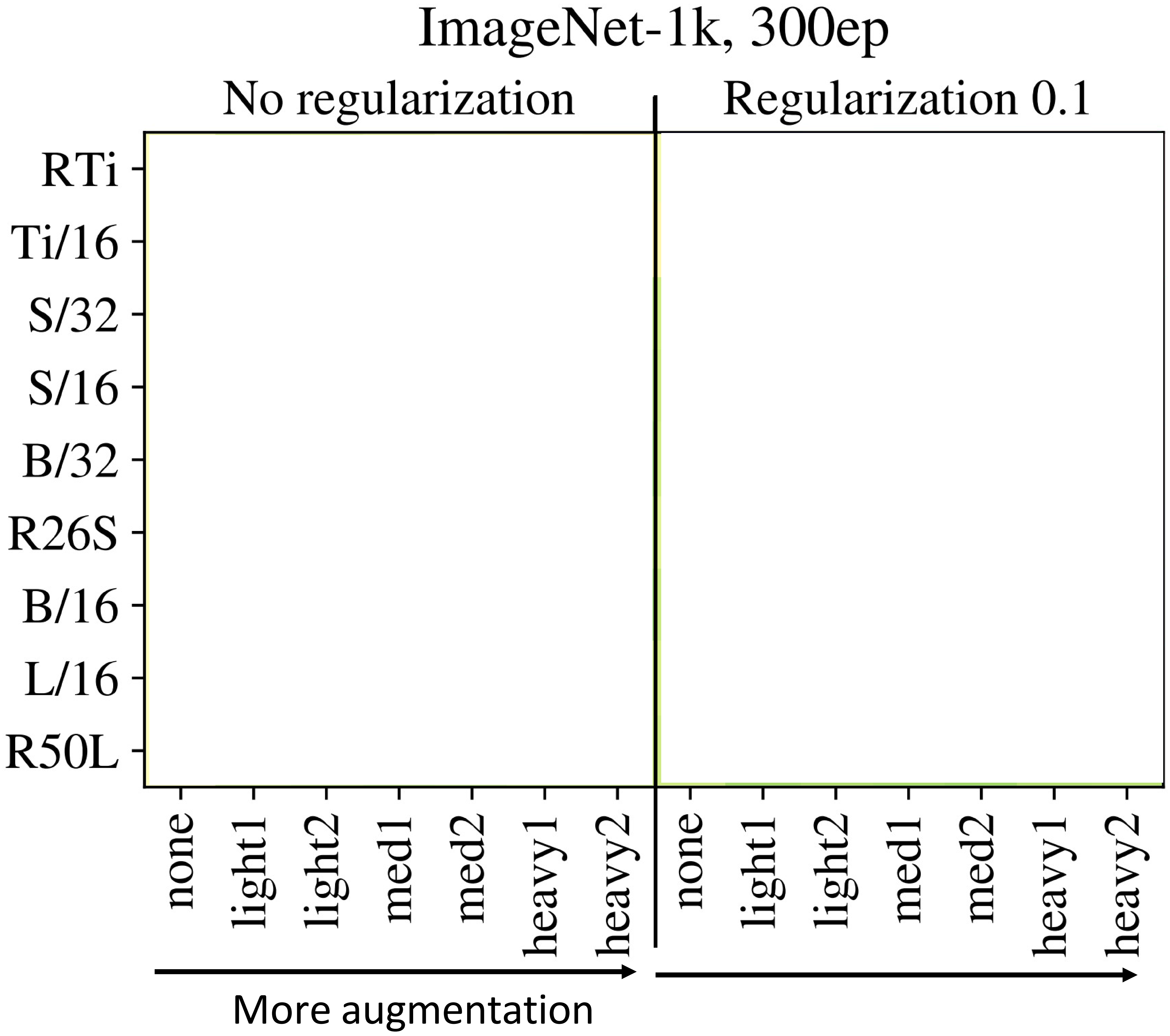
Data Augmentation for ViT models:
- MixUp
- RandAugment

Steiner et al, “How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers”, arXiv 2021

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

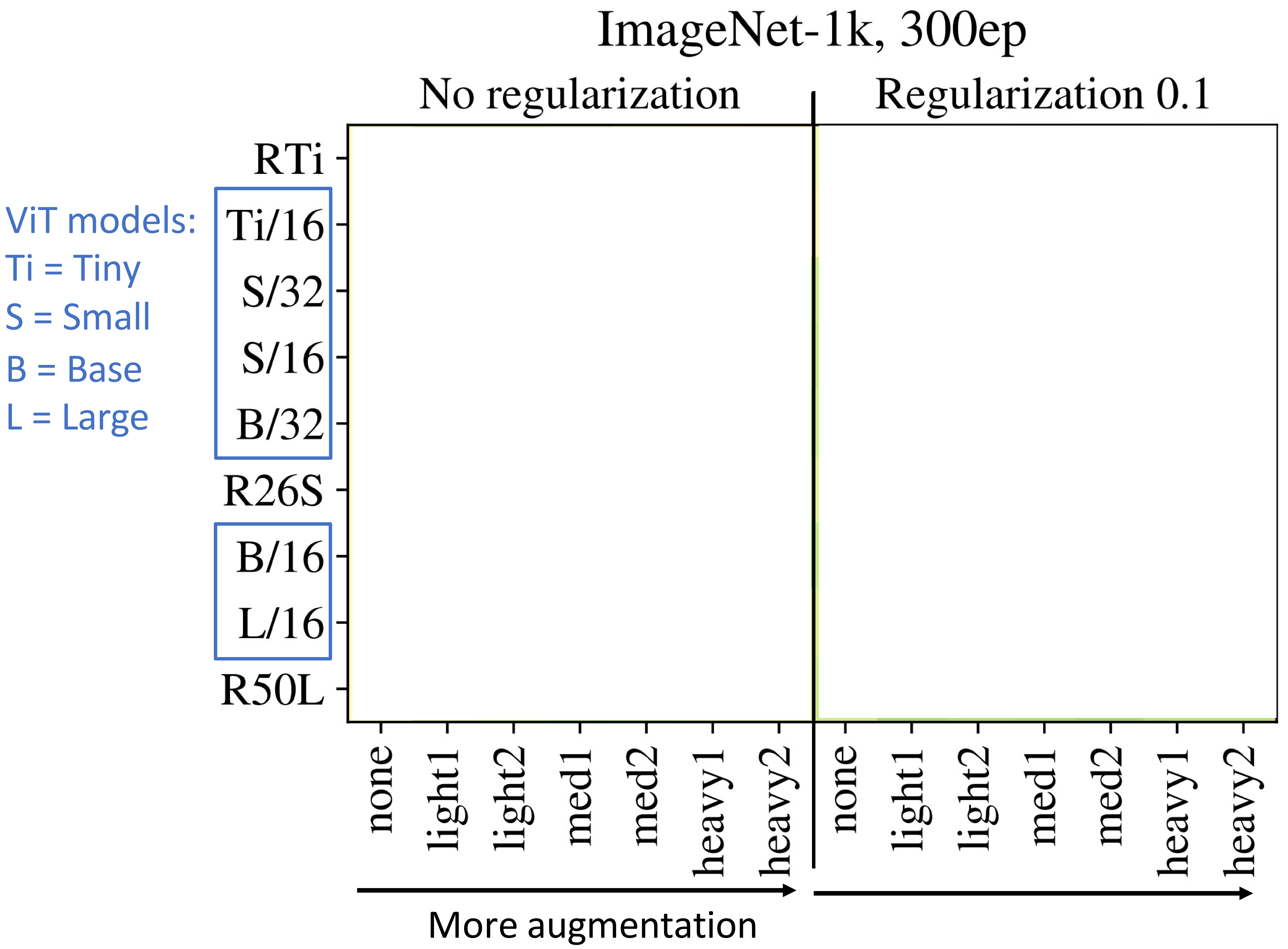


Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment



Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

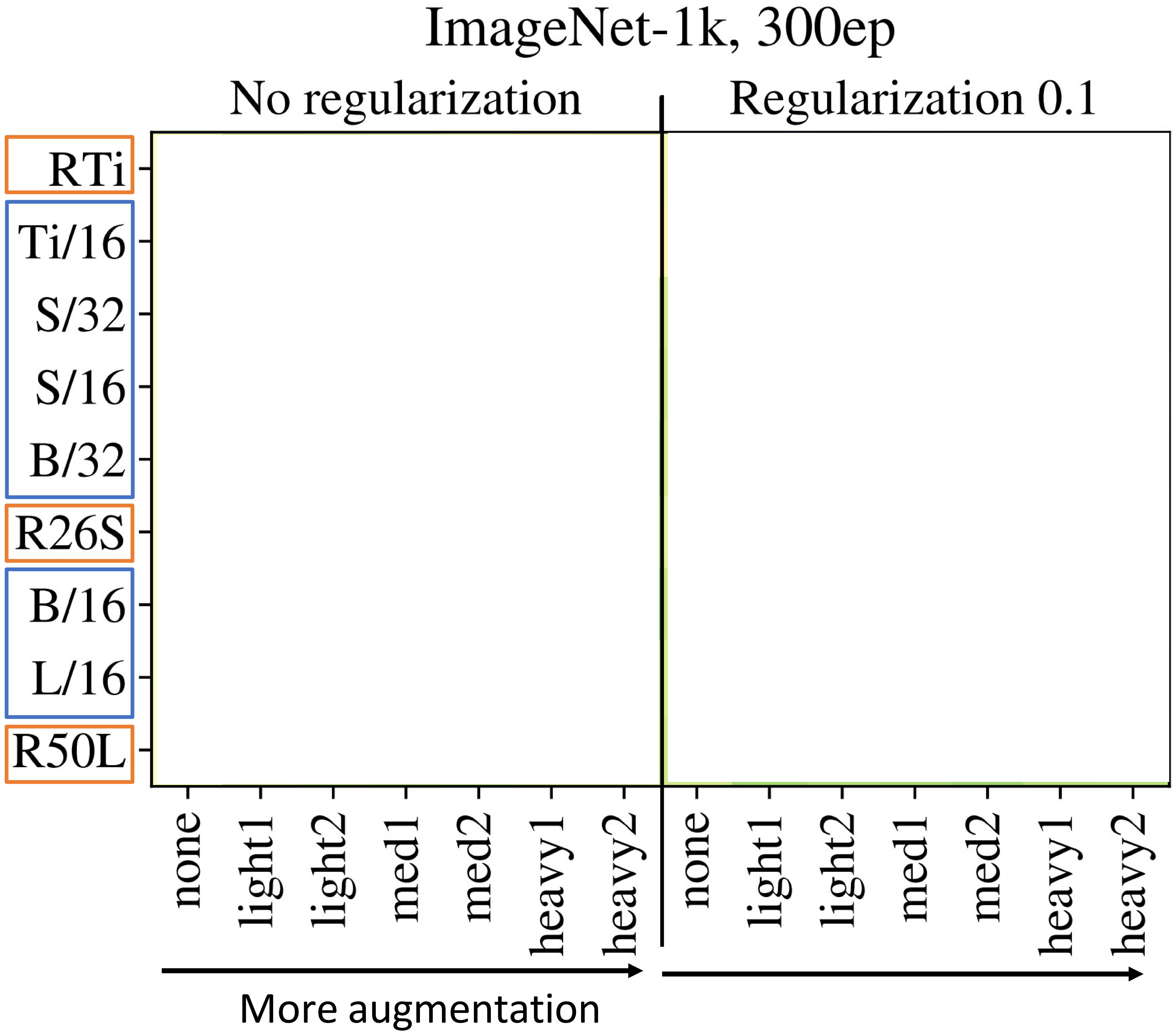
Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

ViT models:
Ti = Tiny
S = Small
B = Base
L = Large



Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

Improving ViT: Augmentation and Regularization

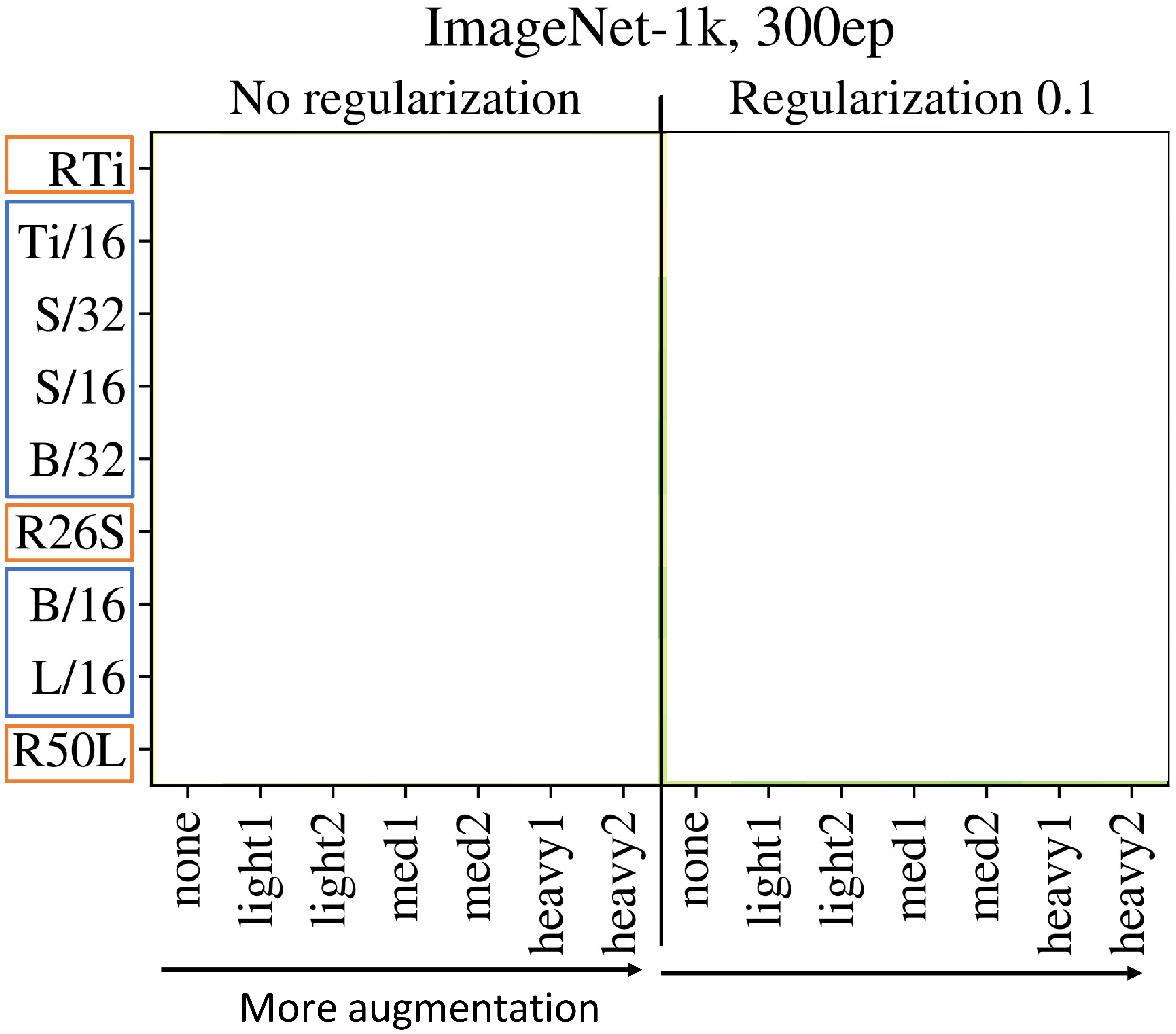
Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

ViT models:
Ti = Tiny
S = Small
B = Base
L = Large

Original Paper:
77.9
76.53



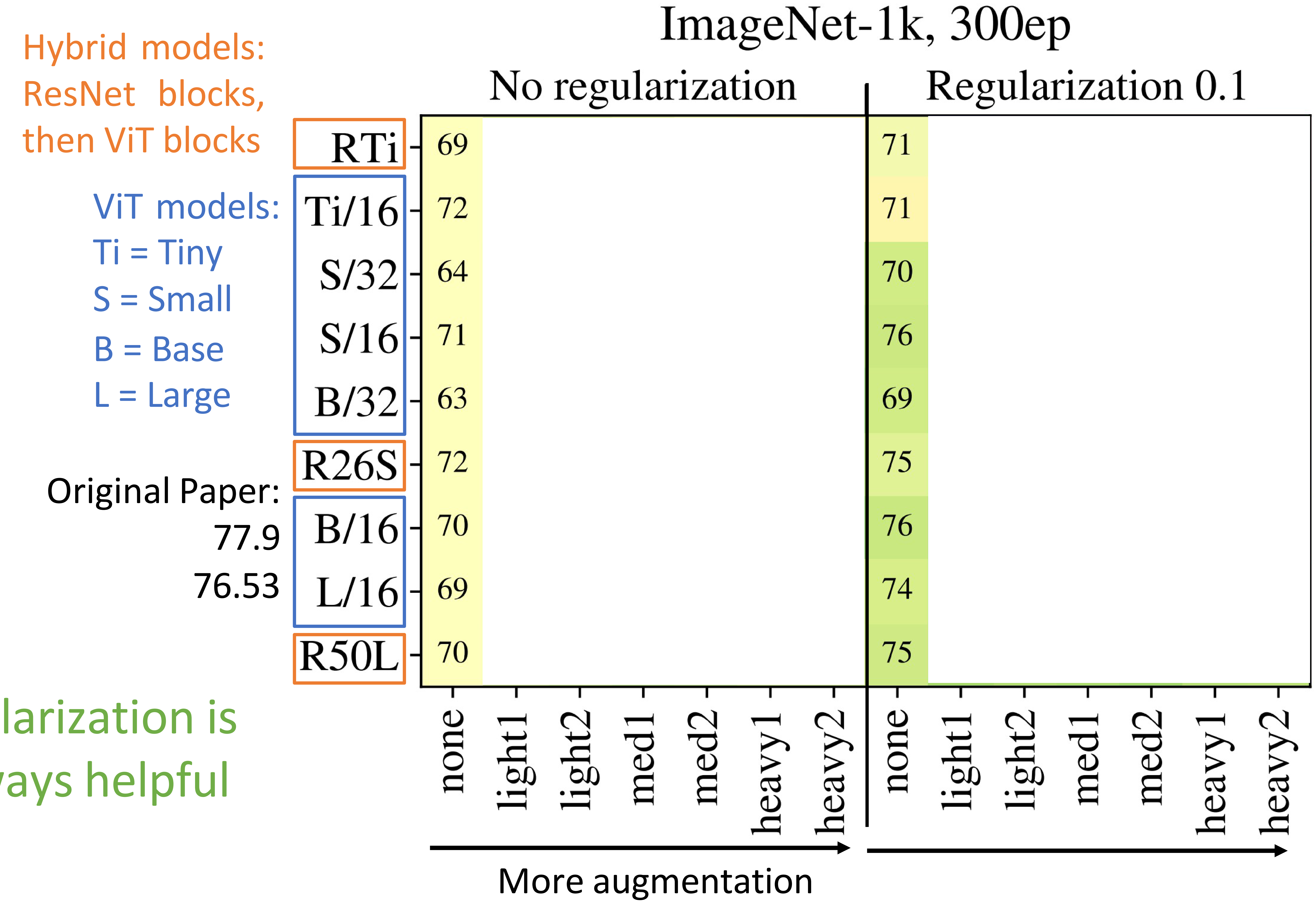
Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

Adding regularization is (almost) always helpful



Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

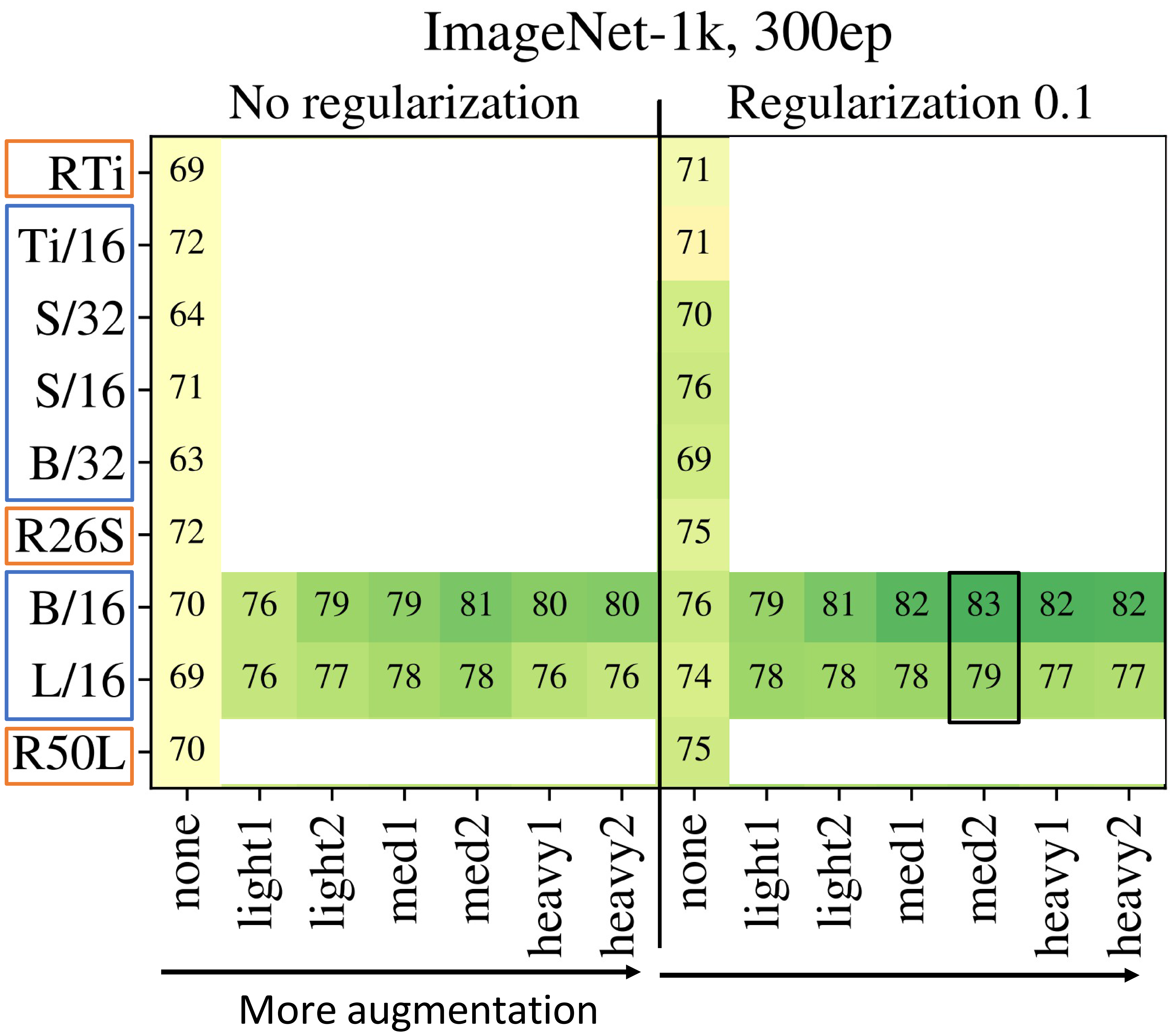
Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

ViT models:
Ti = Tiny
S = Small
B = Base
L = Large

Original Paper:
77.9
76.53

Regularization + Augmentation gives big improvements over original results



Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

Improving ViT: Augmentation and Regularization

Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

ViT models:
Ti = Tiny
S = Small
B = Base
L = Large

Original Paper:
77.9
76.53

Lots of other patterns in full results

ImageNet-1k, 300ep

	No regularization							Regularization 0.1						
	none	light1	light2	med1	med2	heavy1	heavy2	none	light1	light2	med1	med2	heavy1	heavy2
RTi	69	73	73	72	70	69	68	71	70	67	65	63	62	61
Ti/16	72	76	75	75	74	72	71	71	72	68	65	63	63	62
S/32	64	71	76	76	76	74	74	70	72	72	71	71	69	68
S/16	71	77	79	81	82	80	80	76	79	80	79	79	77	77
B/32	63	70	73	75	76	75	76	69	74	77	77	78	77	77
R26S	72	76	78	79	80	80	80	75	78	81	82	82	81	81
B/16	70	76	79	79	81	80	80	76	79	81	82	83	82	82
L/16	69	76	77	78	78	76	76	74	78	78	78	79	77	77
R50L	70	75	76	77	77	76	76	75	78	78	78	79	77	77

More augmentation

Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

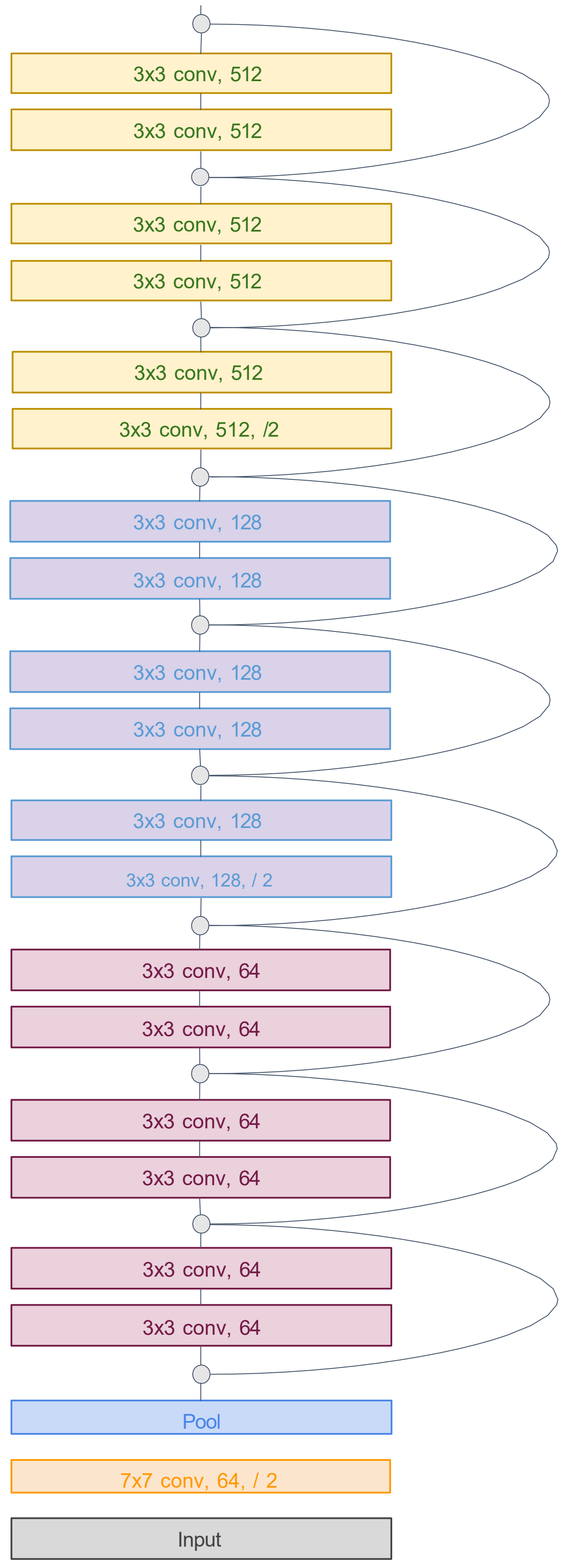
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224



In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

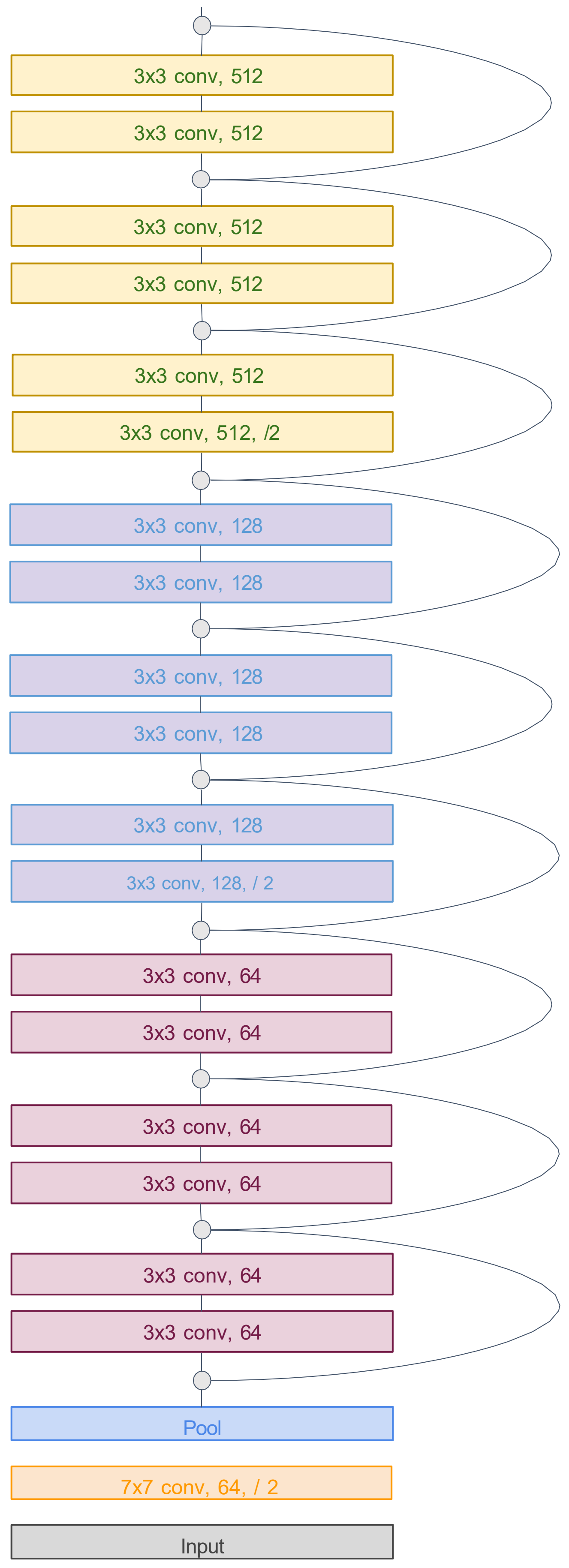
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

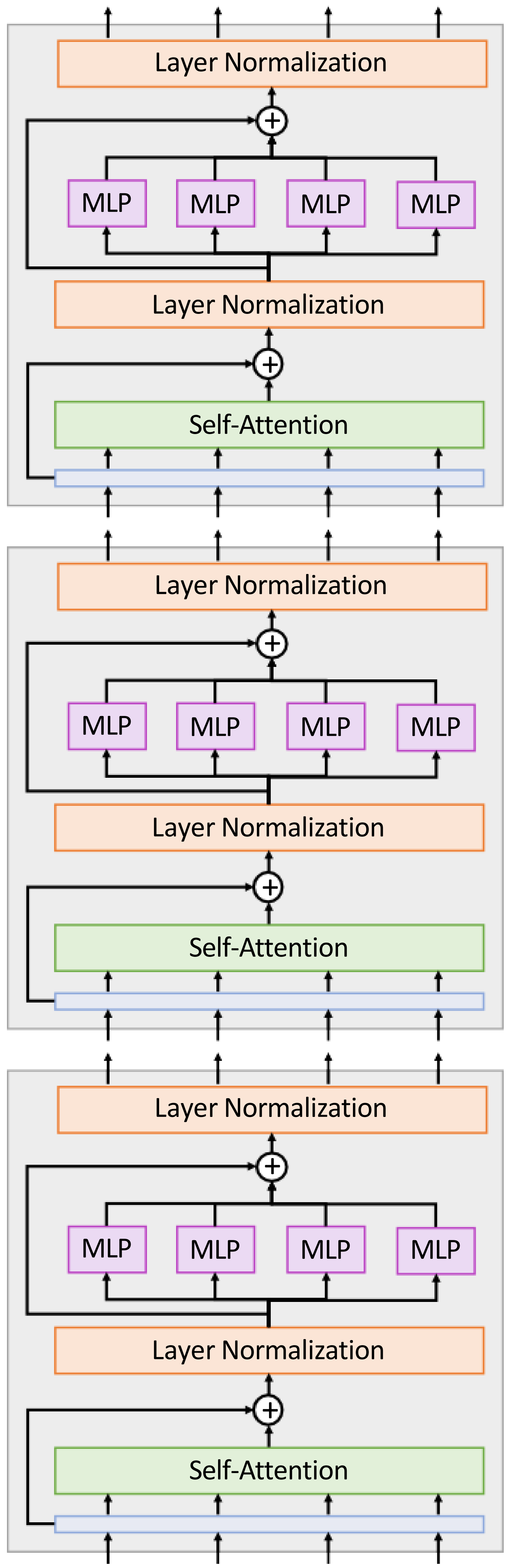
Input:
3 x 224 x 224



In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

In a ViT, all blocks have same resolution and number of channels (Isotropic architecture)



3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
3 x 224 x 224

Additional Slides (Advanced Optional Reading)

ViT vs CNN

In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

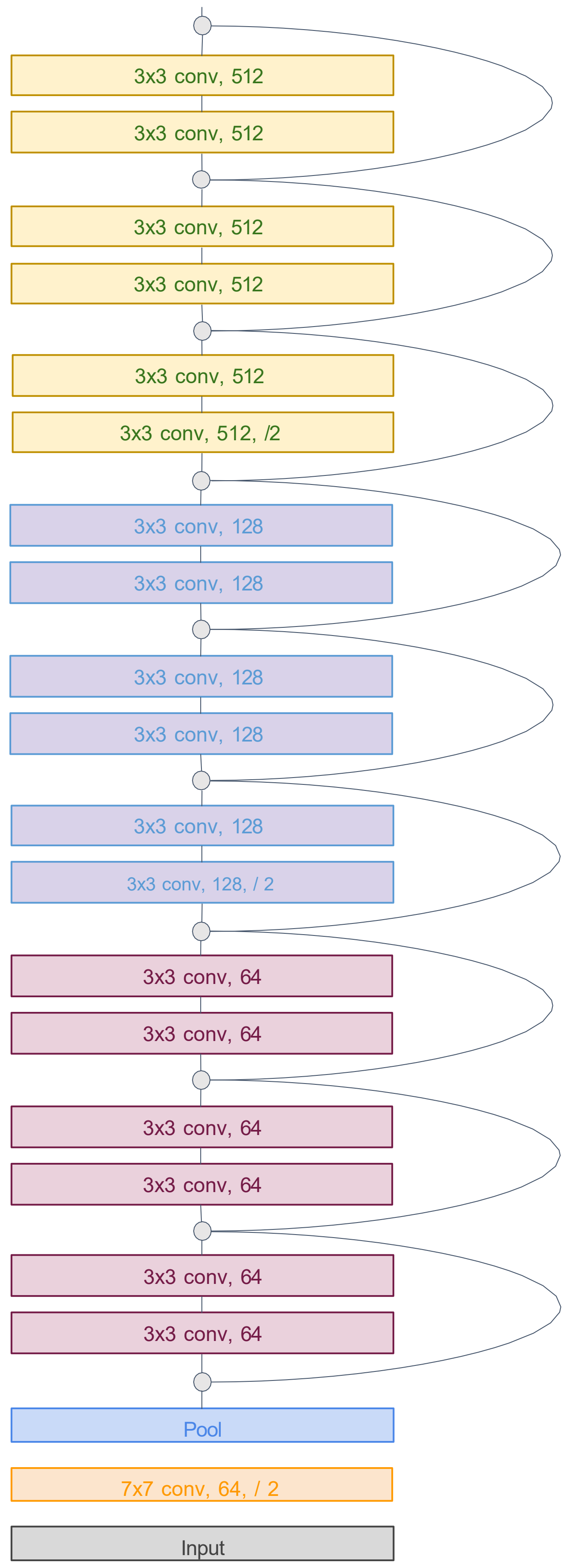
Useful since objects in images can occur at various scales

Stage 3:
256 x 14 x 14

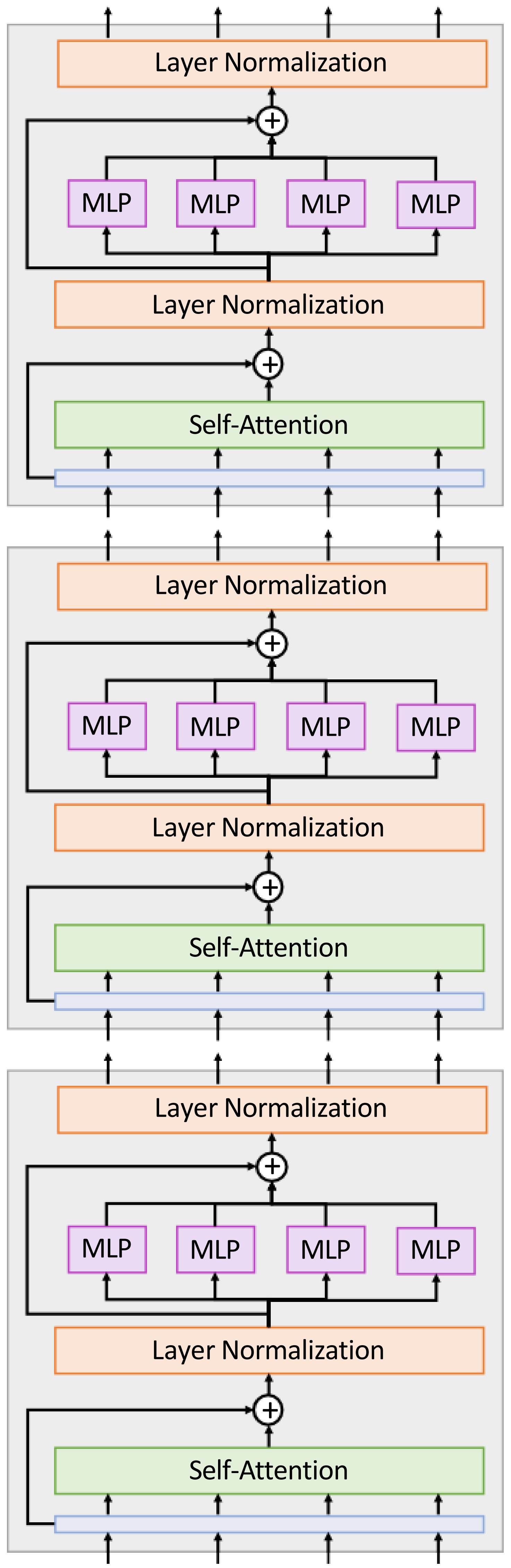
Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224



In a ViT, all blocks have same resolution and number of channels (Isotropic architecture)



3rd block:
768 x 14 x 14

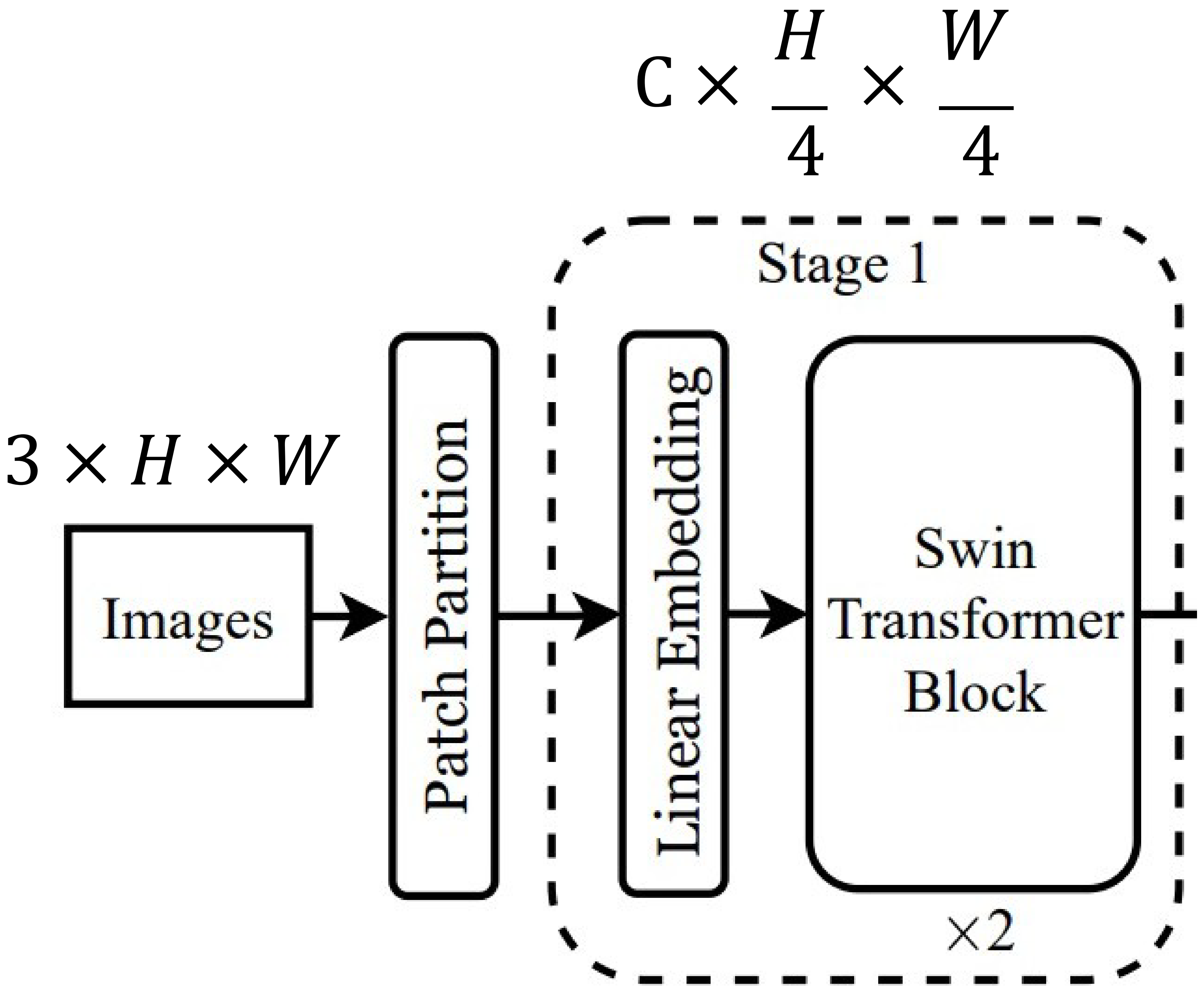
2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
3 x 224 x 224

Can we build a **hierarchical** ViT model?

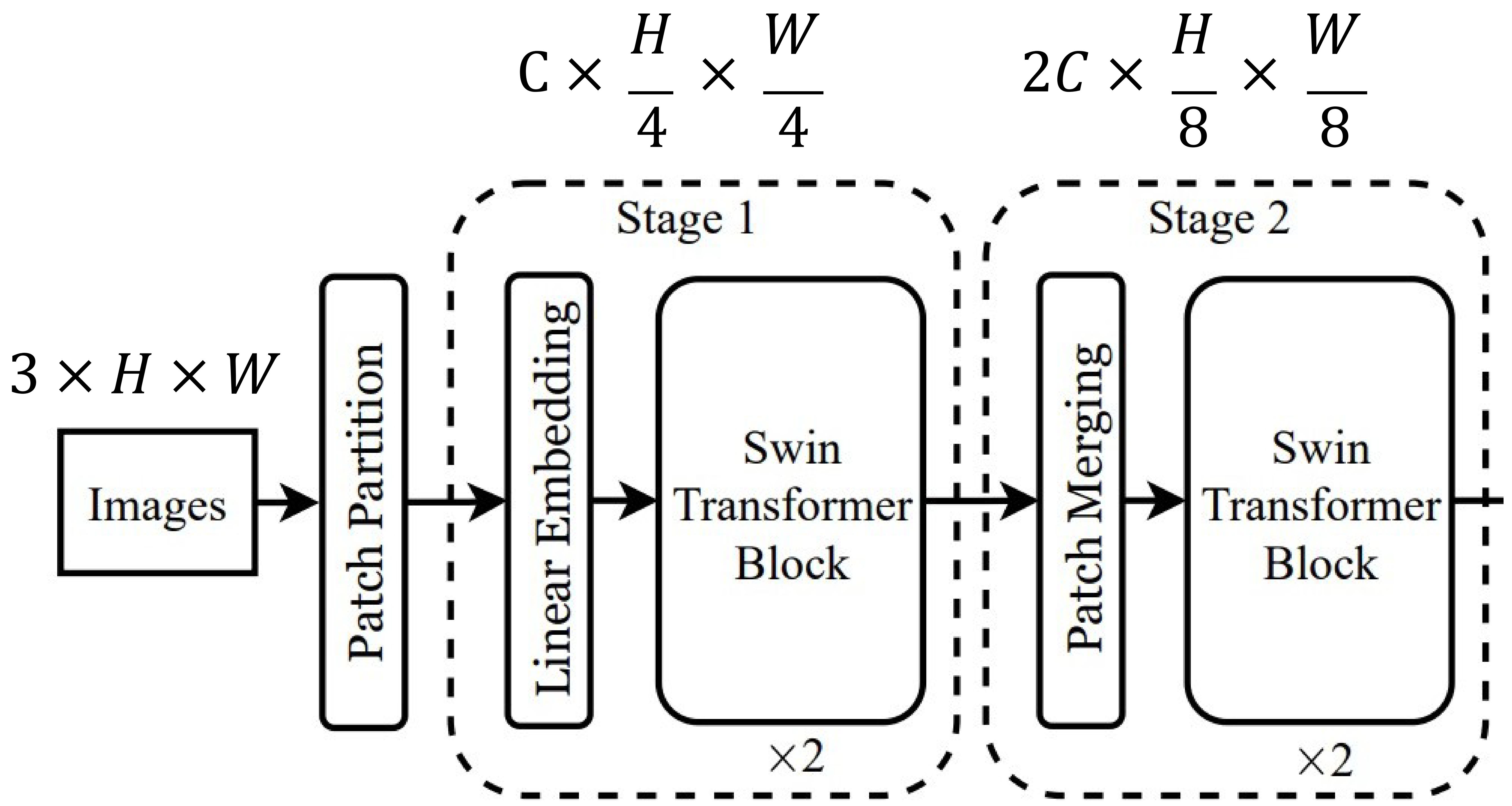
Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer

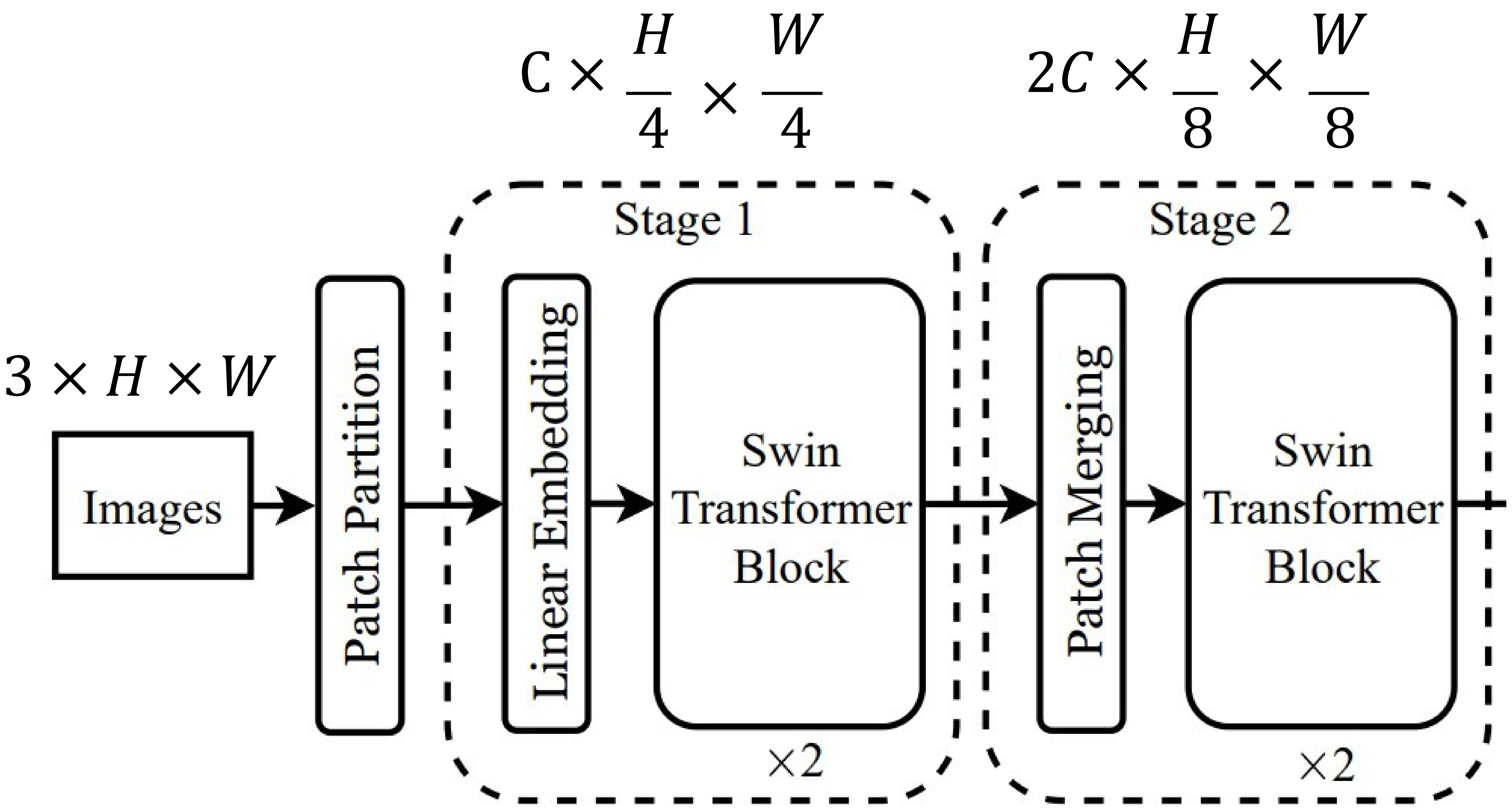


Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

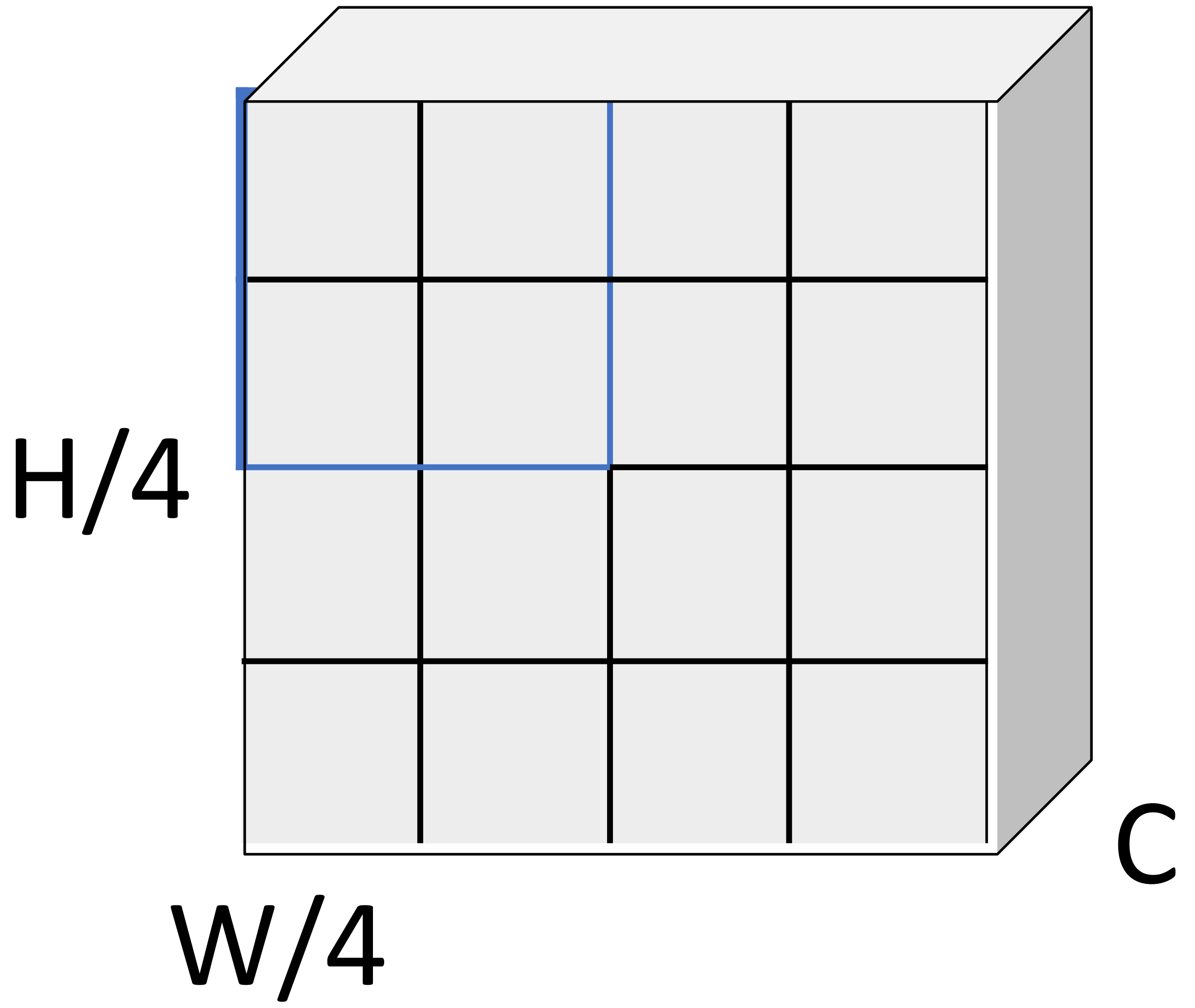
Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer



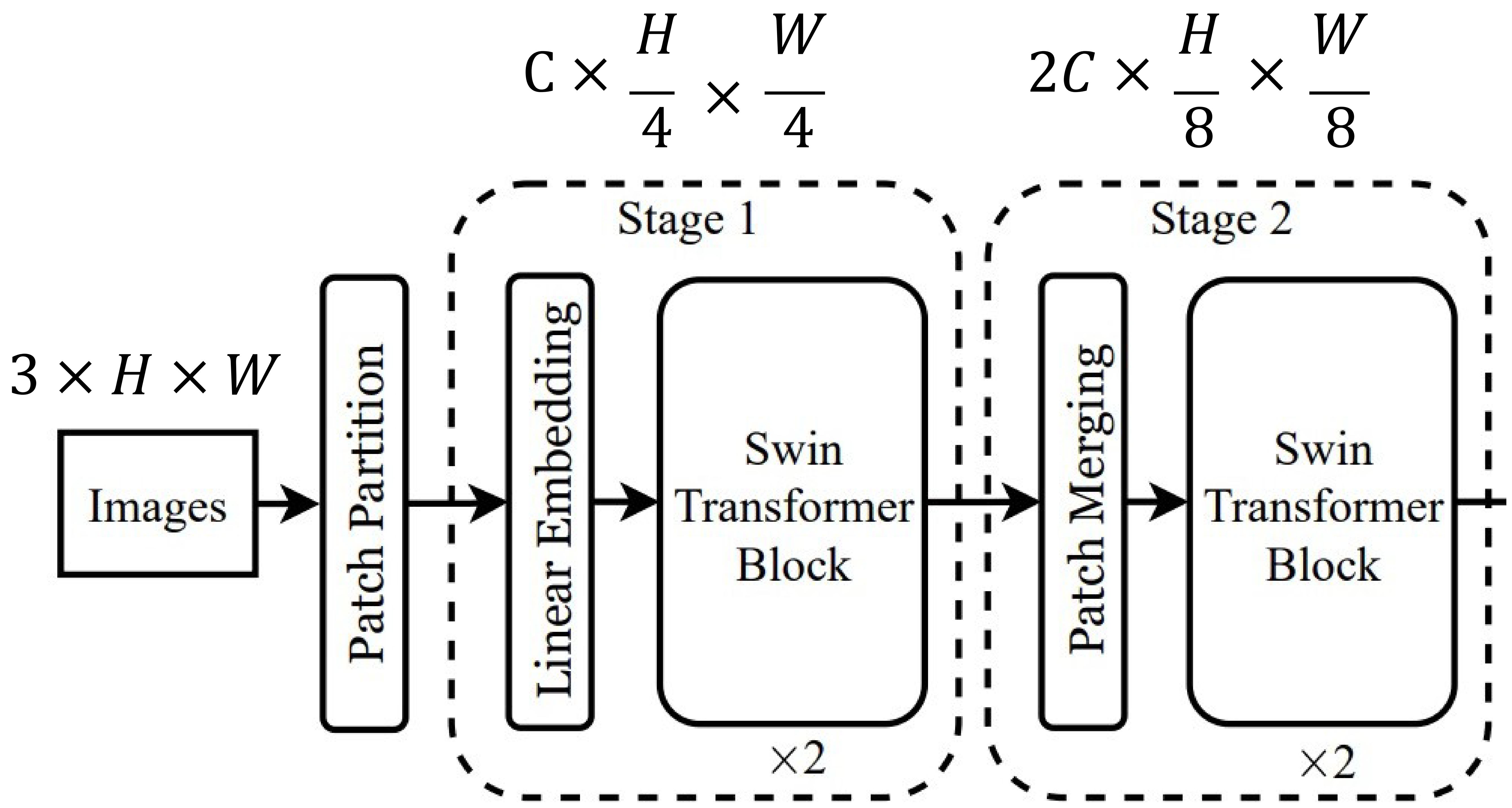
Divide image into 4x4 patches and project to C dimensions

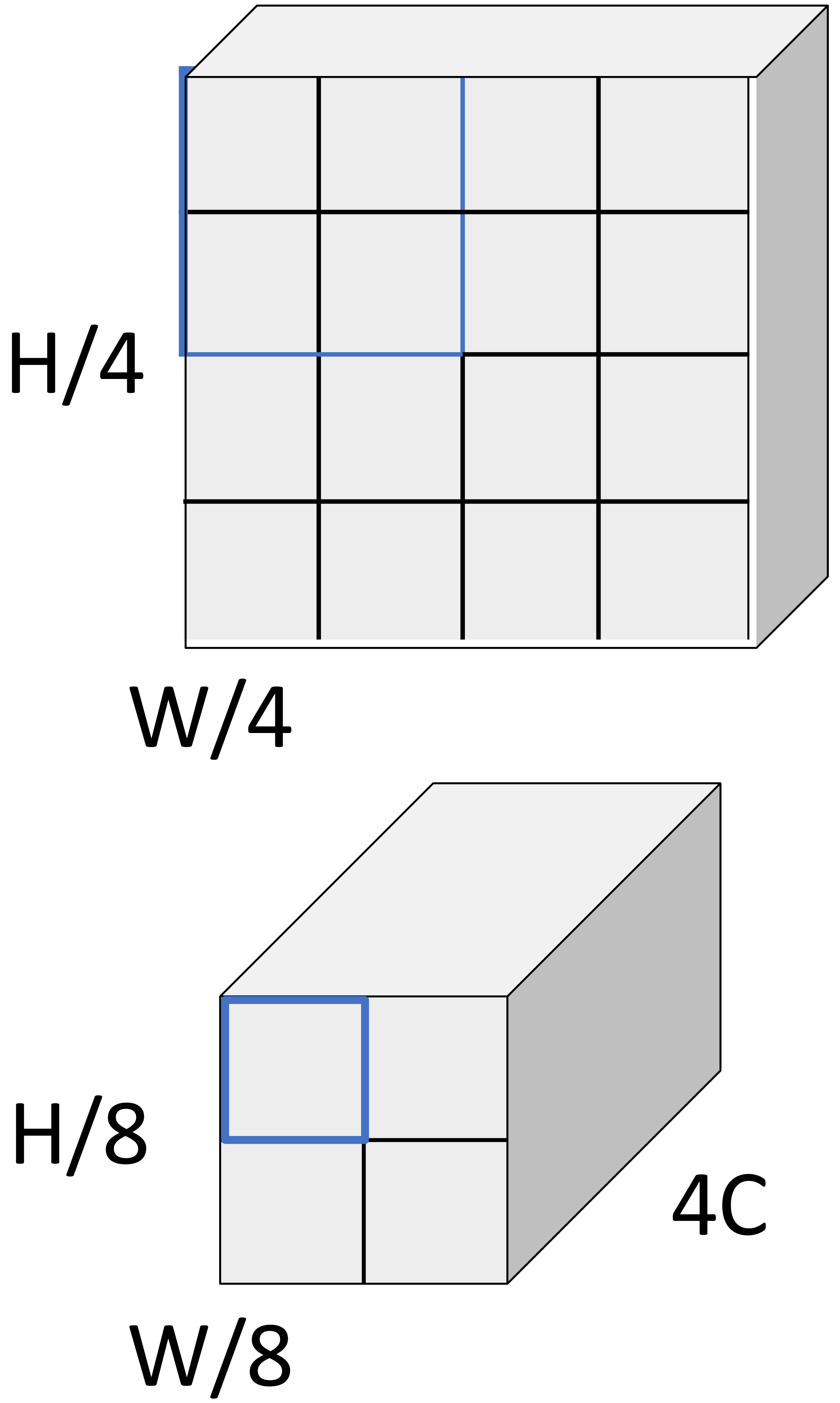
Merge 2x2 neighborhoods; now patches are (effectively) 8x8



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer





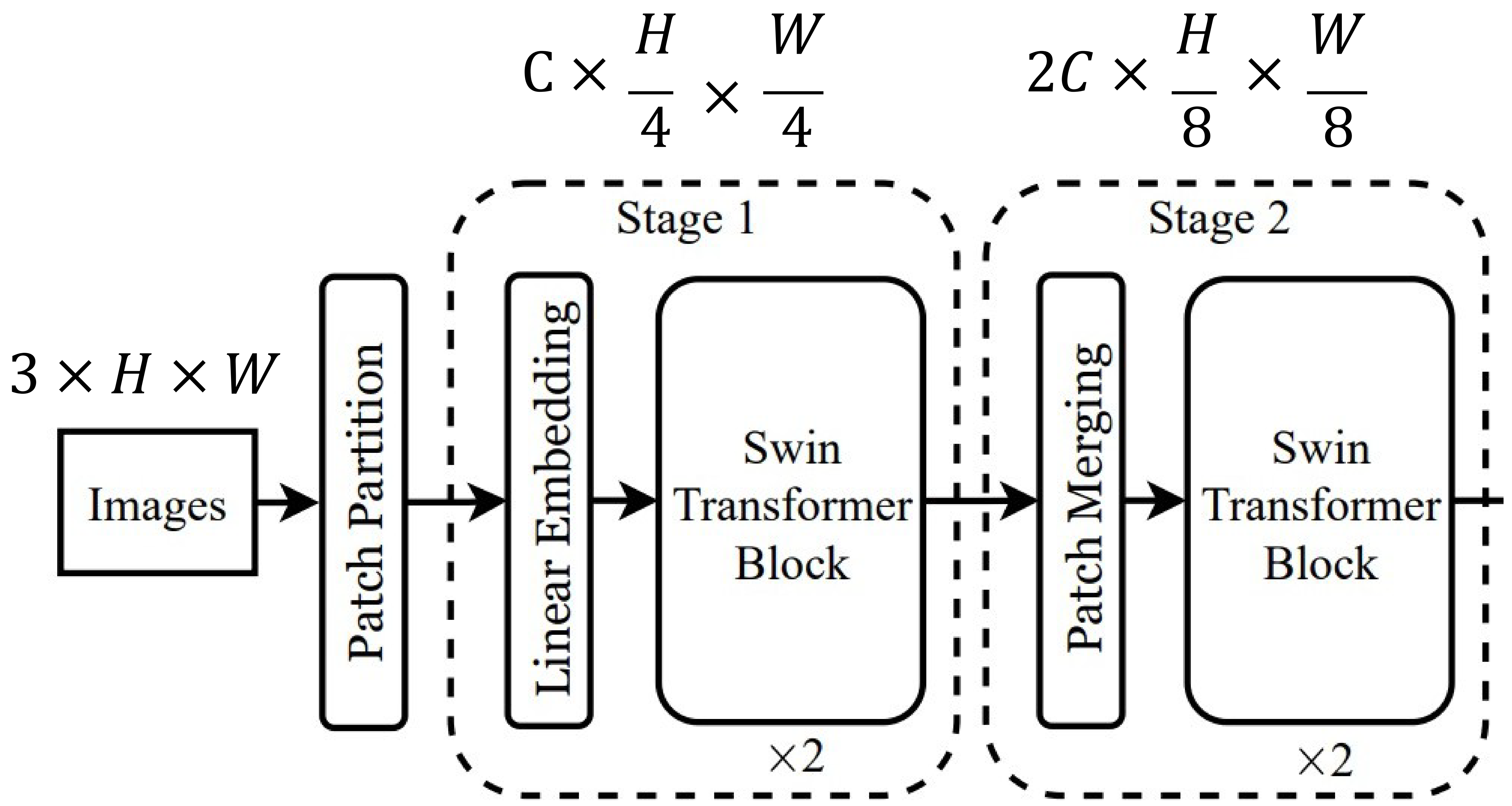
Concatenate groups of 2x2 features

Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

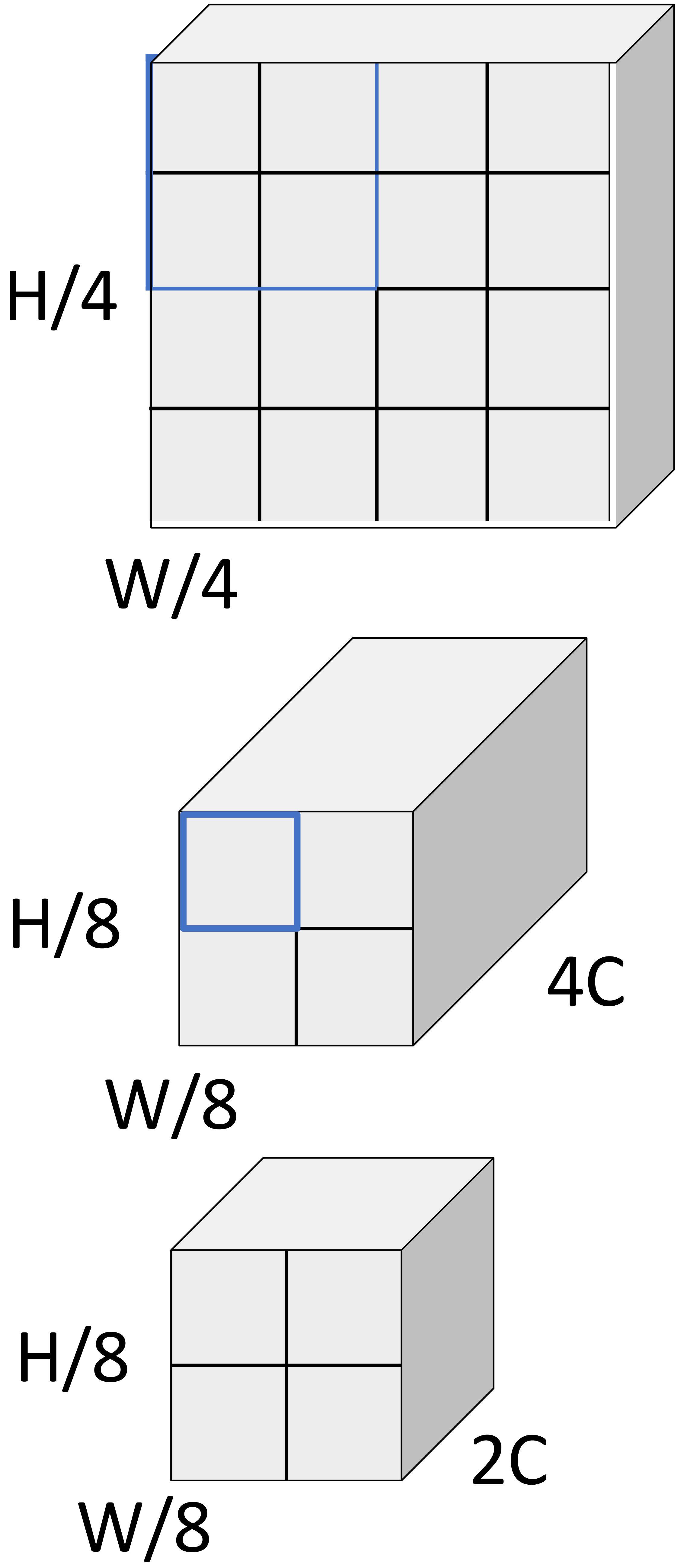
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

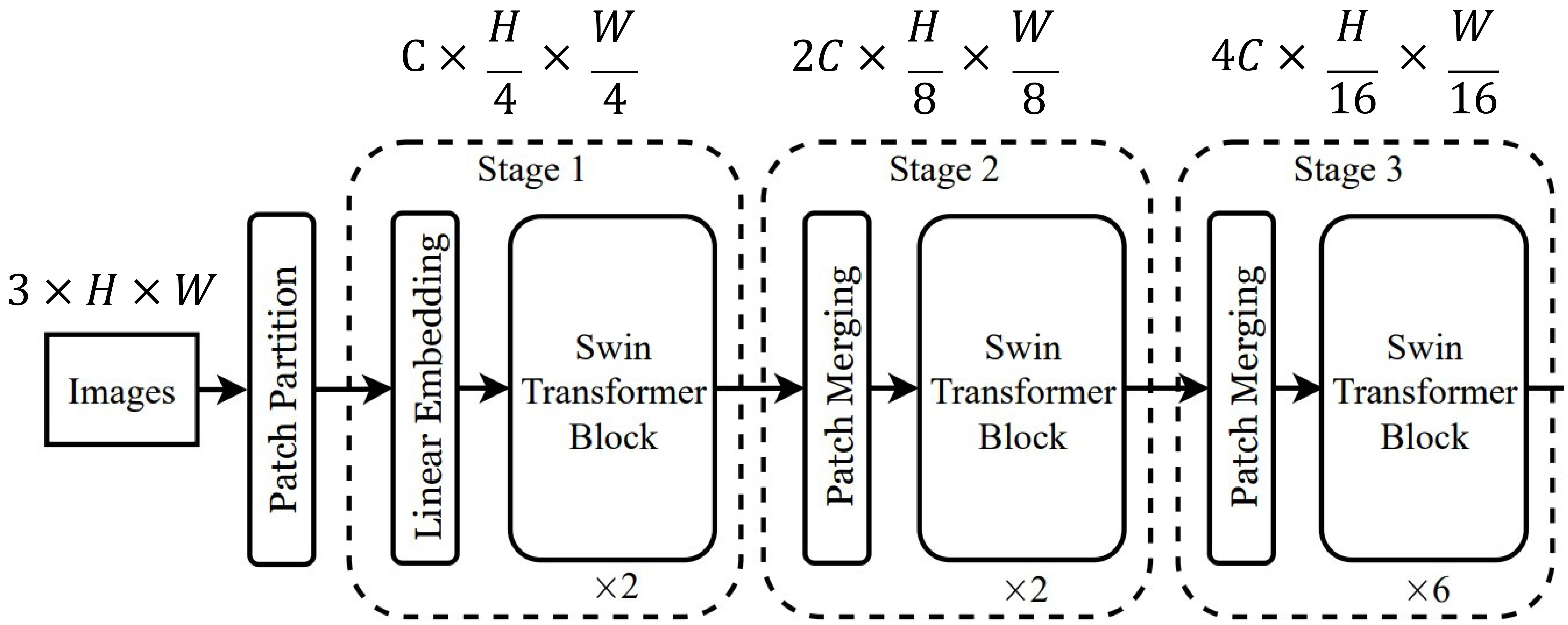


Concatenate groups of 2x2 features

Linear projection from 4C to 2C channels (1x1 conv)

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



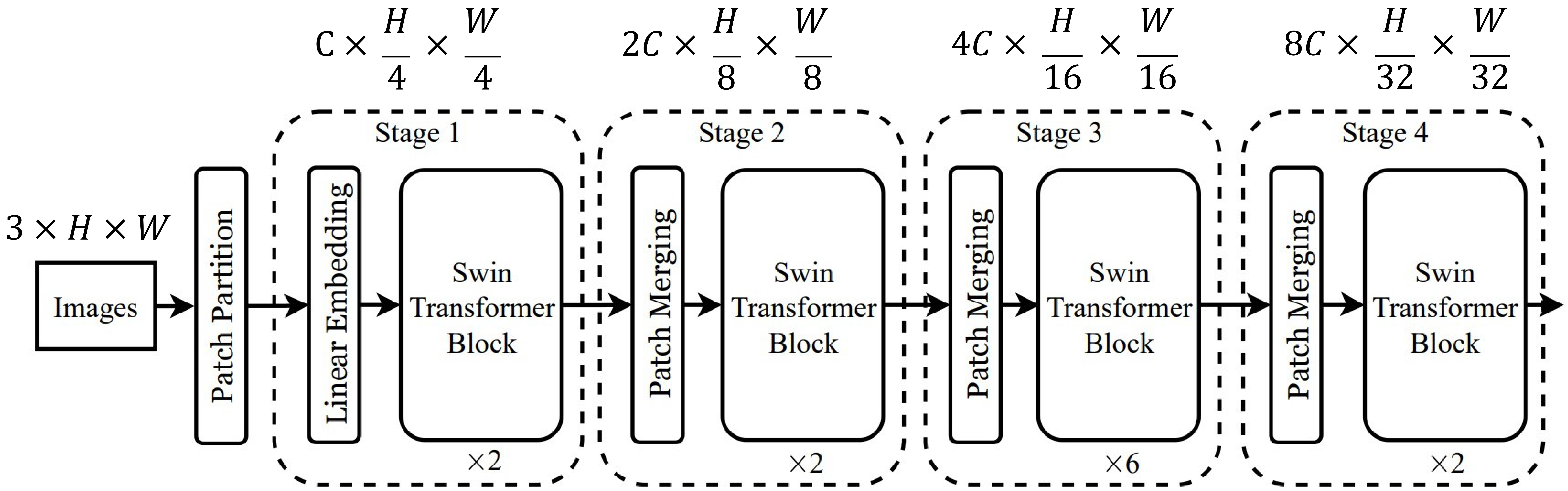
Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

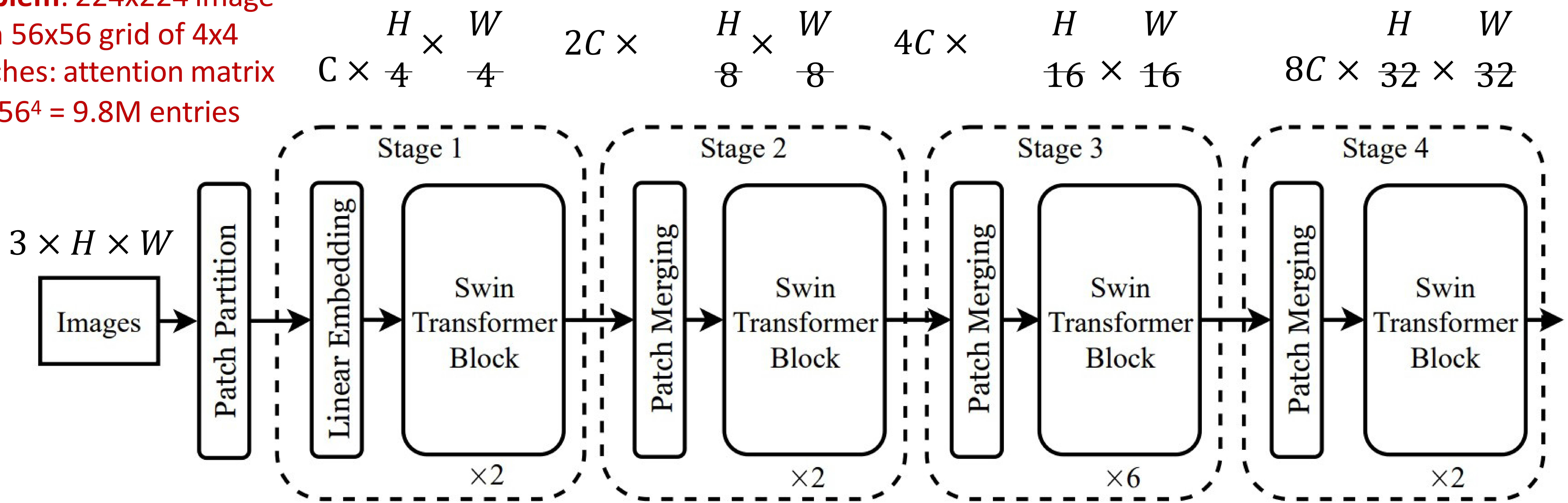
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer

Problem: 224x224 image with 56x56 grid of 4x4 patches: attention matrix has 56^4 = 9.8M entries



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

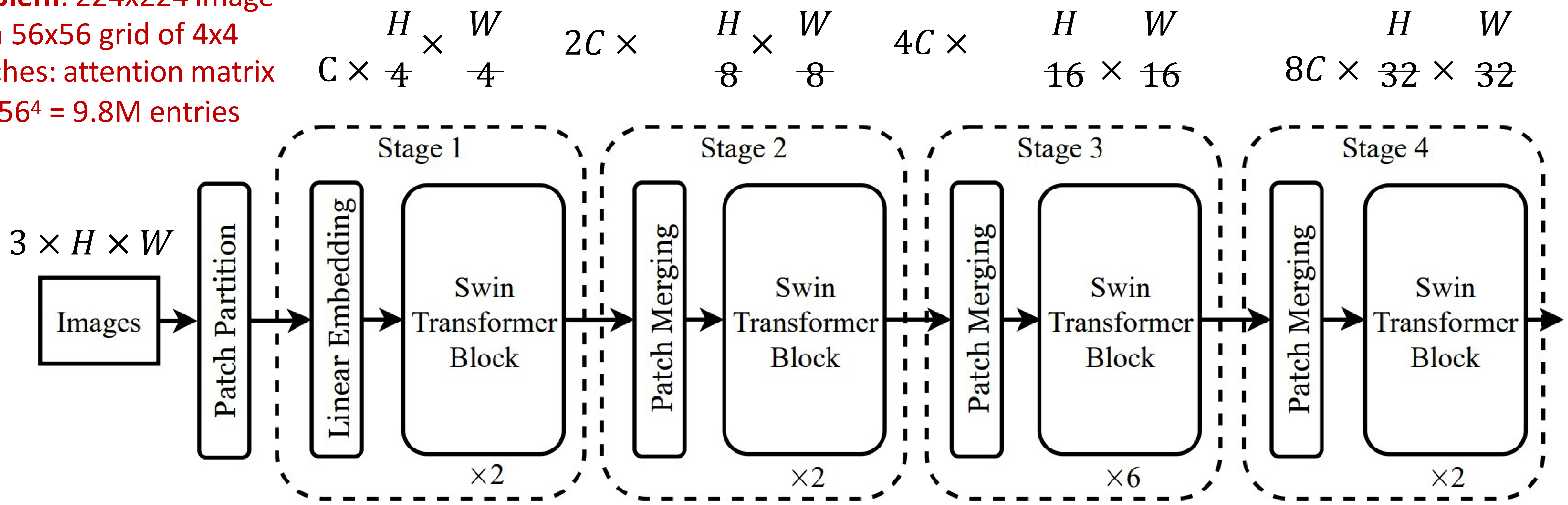
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer

Problem: 224x224 image with 56x56 grid of 4x4 patches: attention matrix has 56^4 = 9.8M entries



Solution: don't use full attention, instead use attention over patches

Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

With H x W grid of **tokens**, each attention matrix is H^2W^2 – **quadratic** in image size

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

With H x W grid of **tokens**, each attention matrix is H^2W^2 – **quadratic** in image size

Rather than allowing each **token** to attend to all other tokens, instead divide into **windows** of M x M tokens (here M=4); only compute attention within each window

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

With H x W grid of **tokens**, each attention matrix is H^2W^2 – **quadratic** in image size

Rather than allowing each **token** to attend to all other tokens, instead divide into **windows** of M x M tokens (here M=4); only compute attention within each window

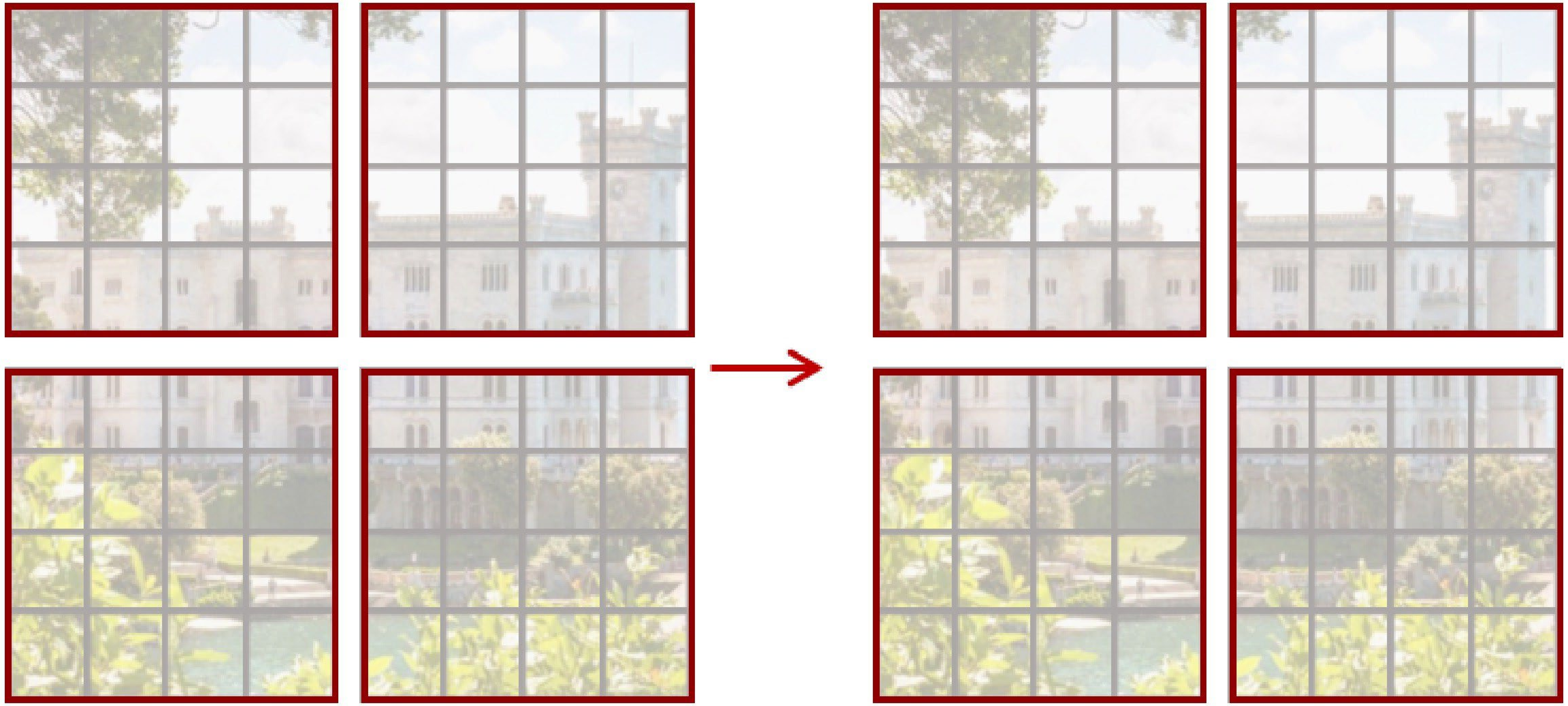
Total size of all attention matrices is now: $M^4(H/M)(W/M) = M^2HW$

Linear in image size for fixed M!
Swin uses M=7 throughout the network

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

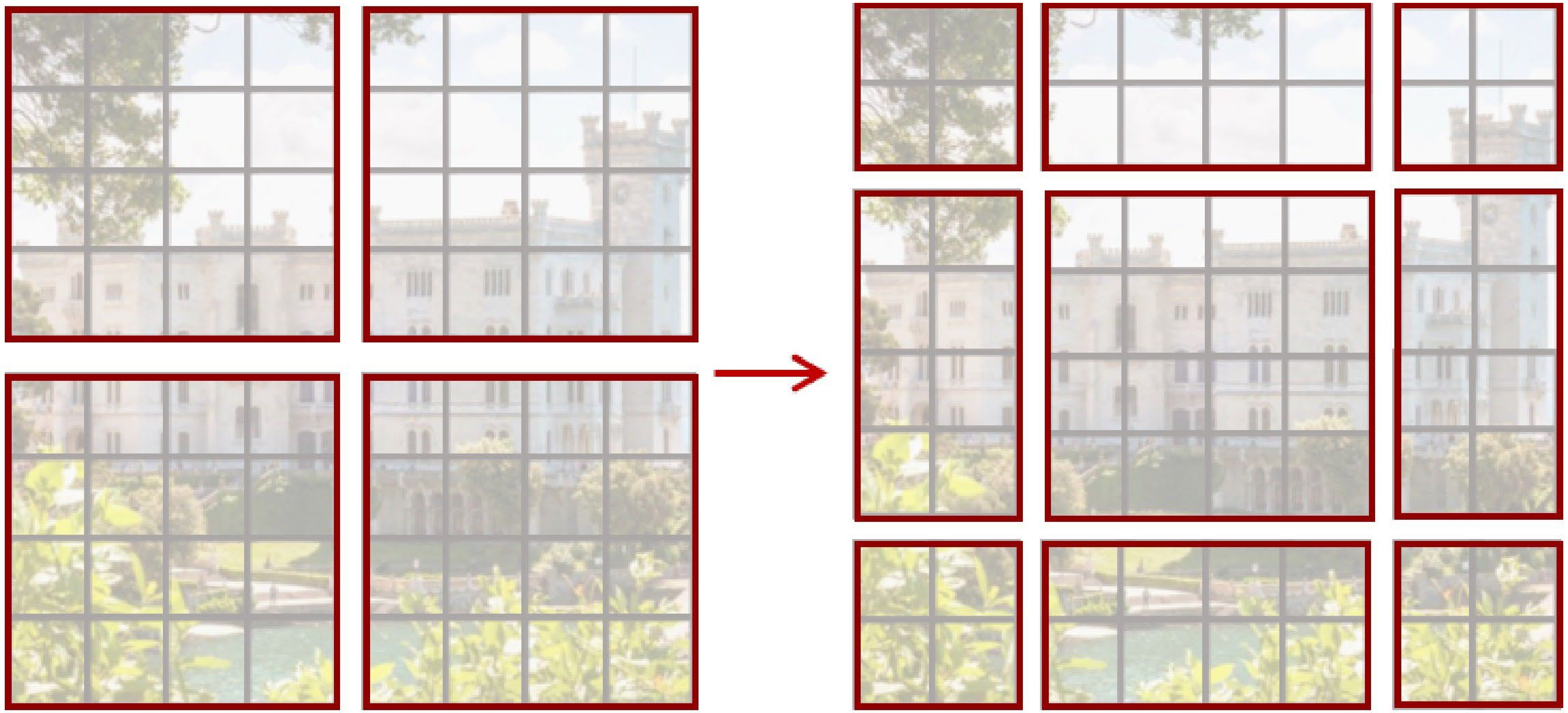
Problem: tokens only interact with other tokens within the same window; no communication across windows



Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Ugly detail: Non-square windows at edges and corners

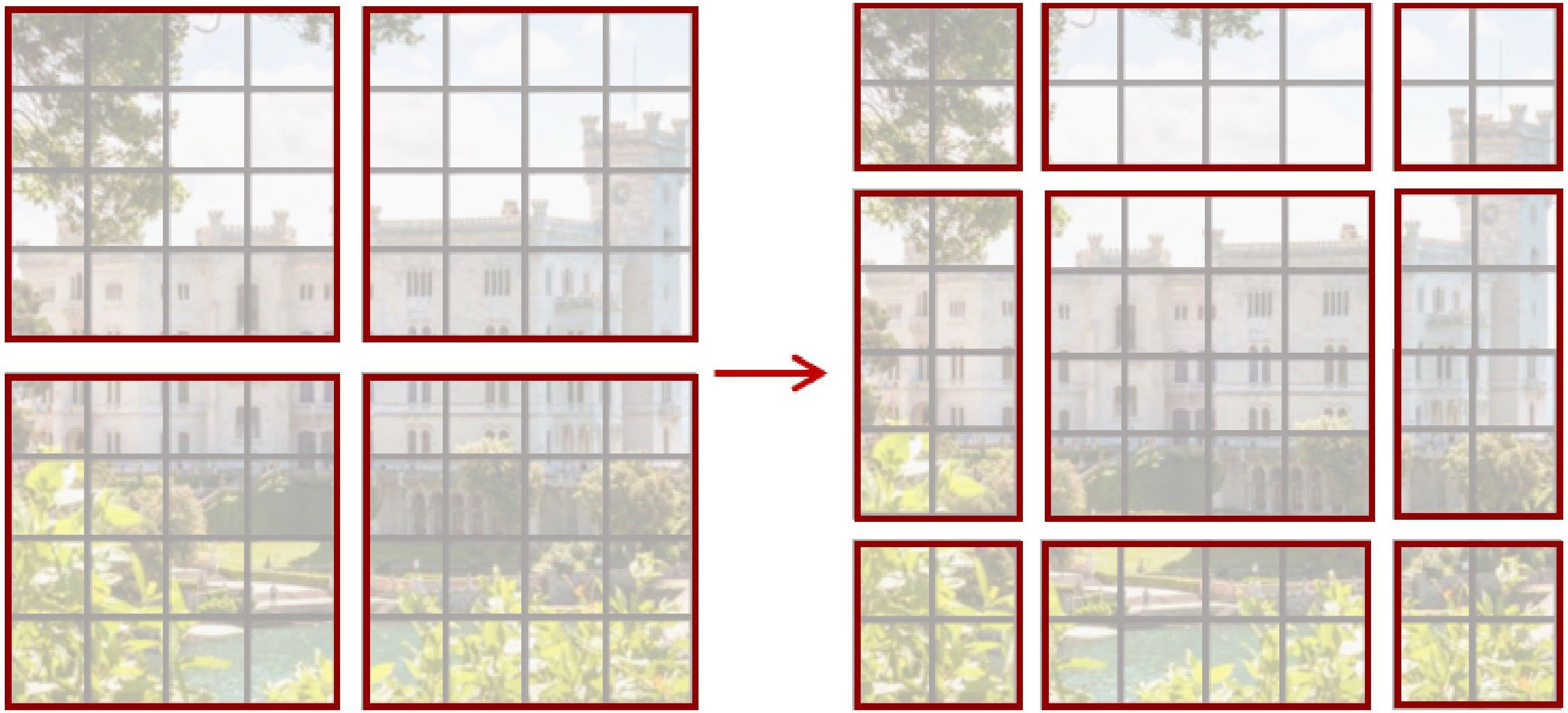
Block L: Normal windows

Block L+1: Shifted Windows

Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Block L: Normal windows

Block L+1: Shifted Windows

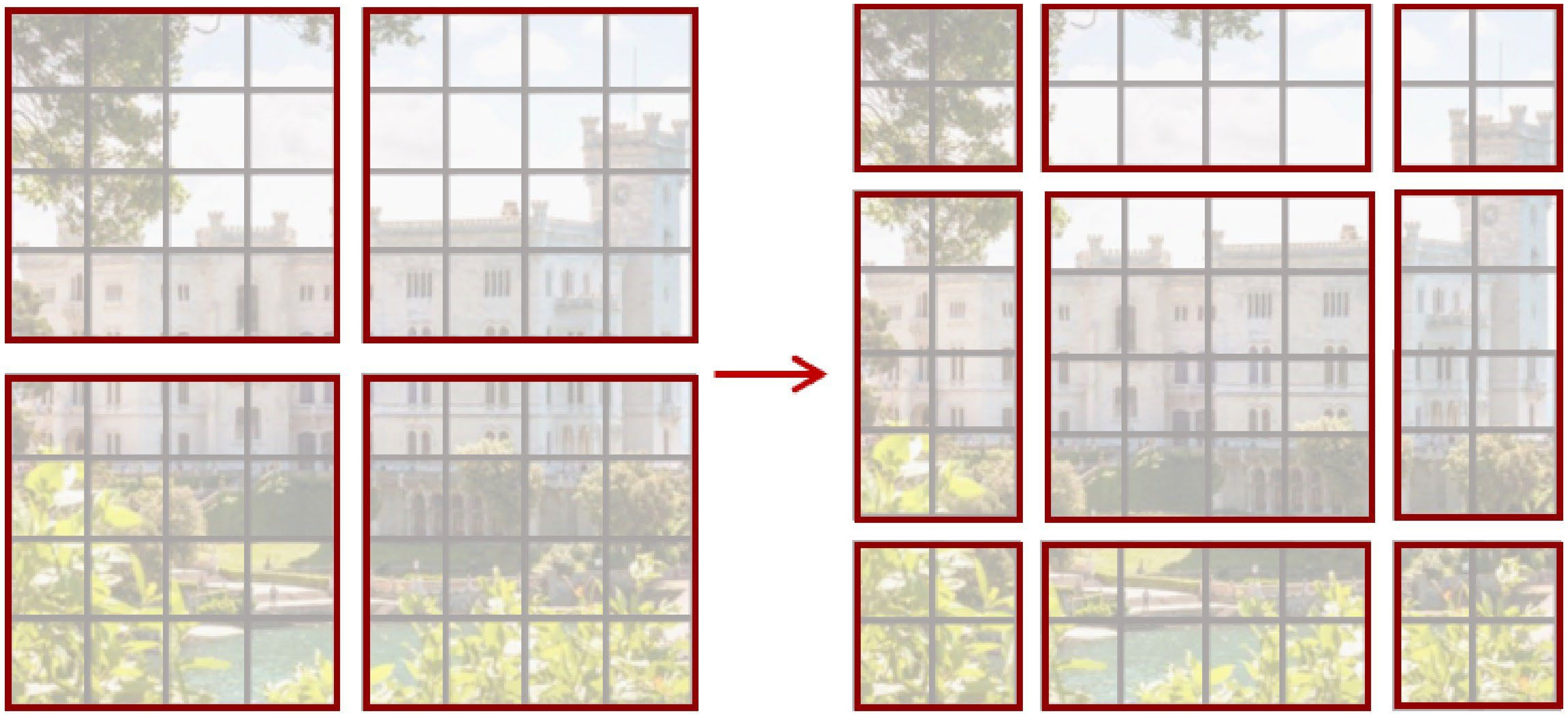
Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Block L: Normal windows

Block L+1: Shifted Windows

Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

Standard Attention:

$$A = Softmax\left(\frac{QK^T}{\sqrt{D}}\right)V$$

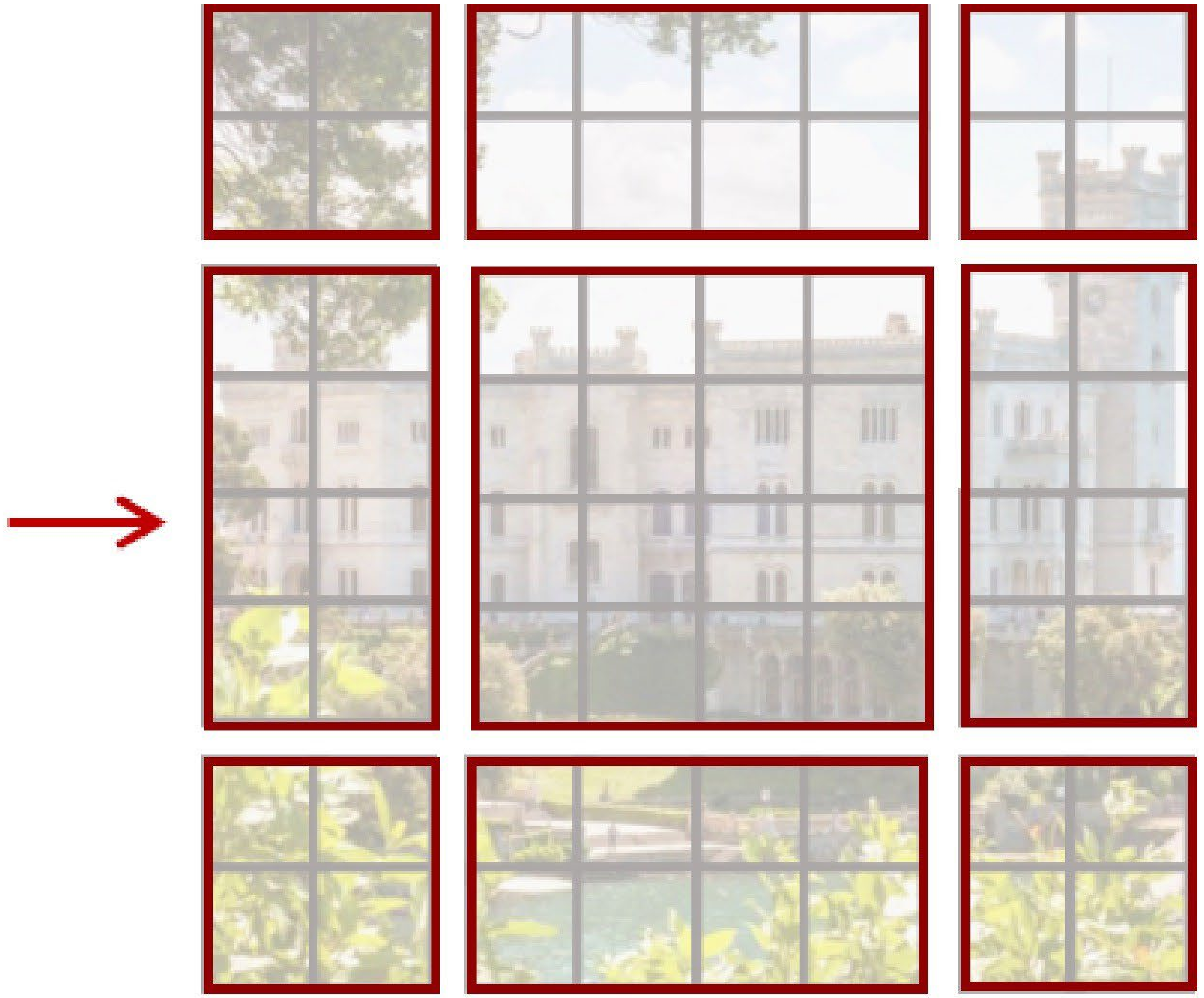
$Q, K, V: M^2 \times D$ (Query, Key, Value)

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks

Block L: Normal windows



Block L+1: Shifted Windows

Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

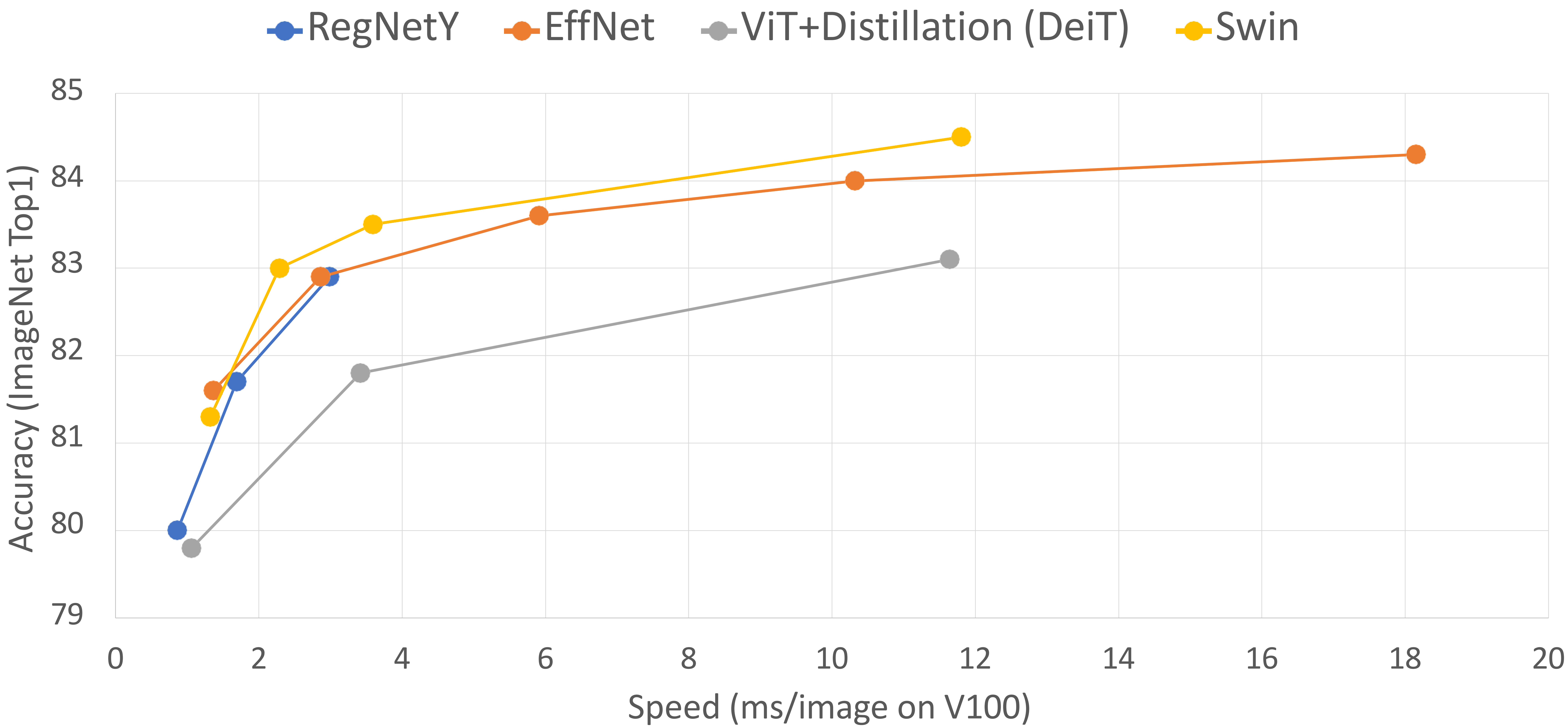
Attention with relative bias:

$$A = Softmax\left(\frac{QK^T}{\sqrt{D}} + B\right)V$$

$Q, K, V: M^2 \times D$ (Query, Key, Value)
$B: M^2 \times M^2$ (learned biases)

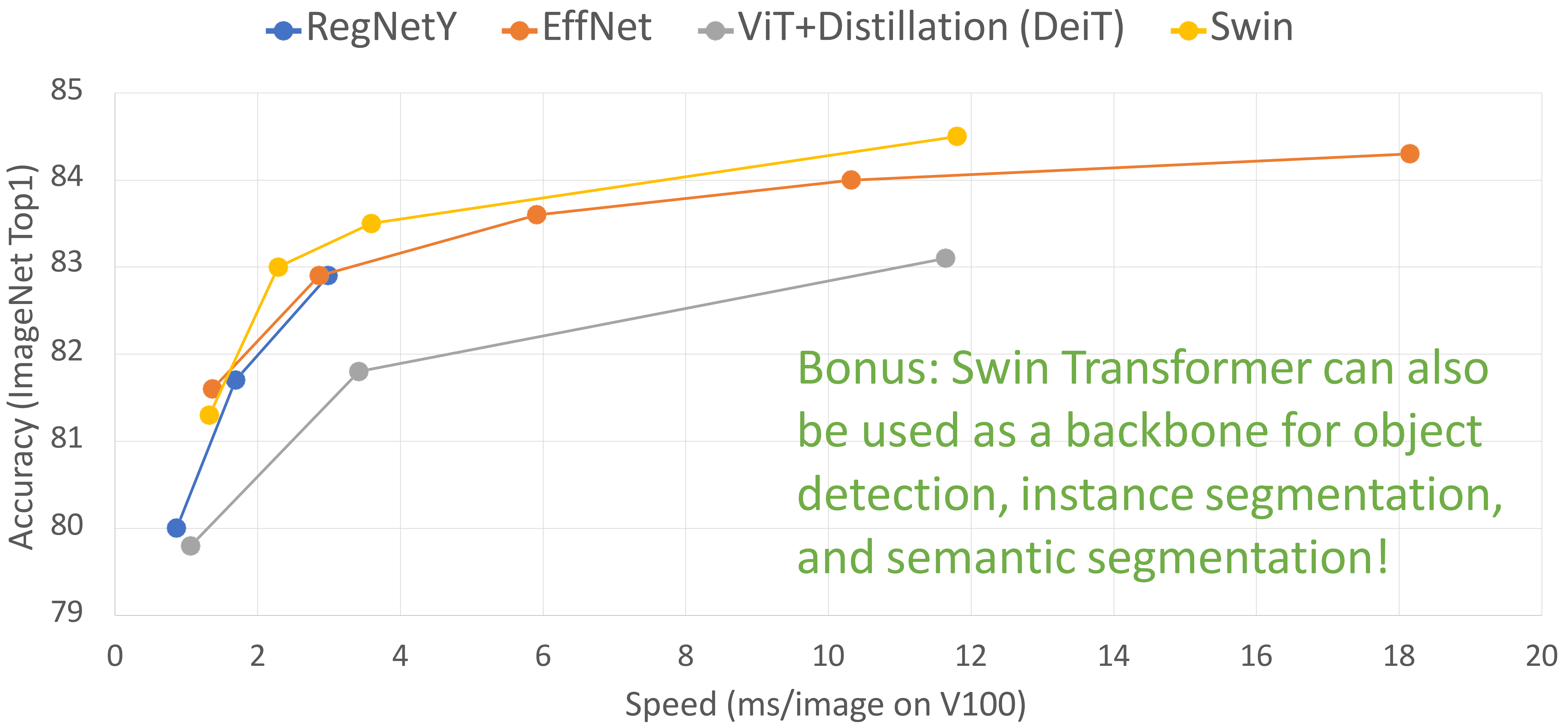
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Speed vs Accuracy



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

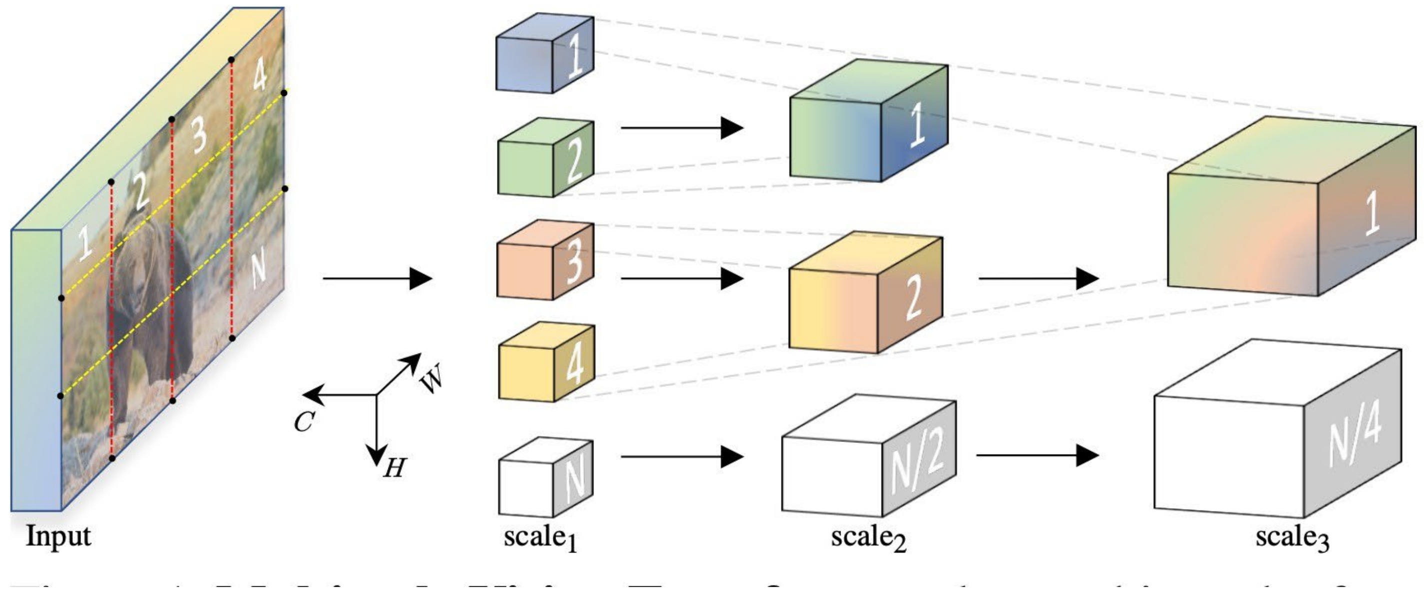
Swin Transformer: Speed vs Accuracy



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

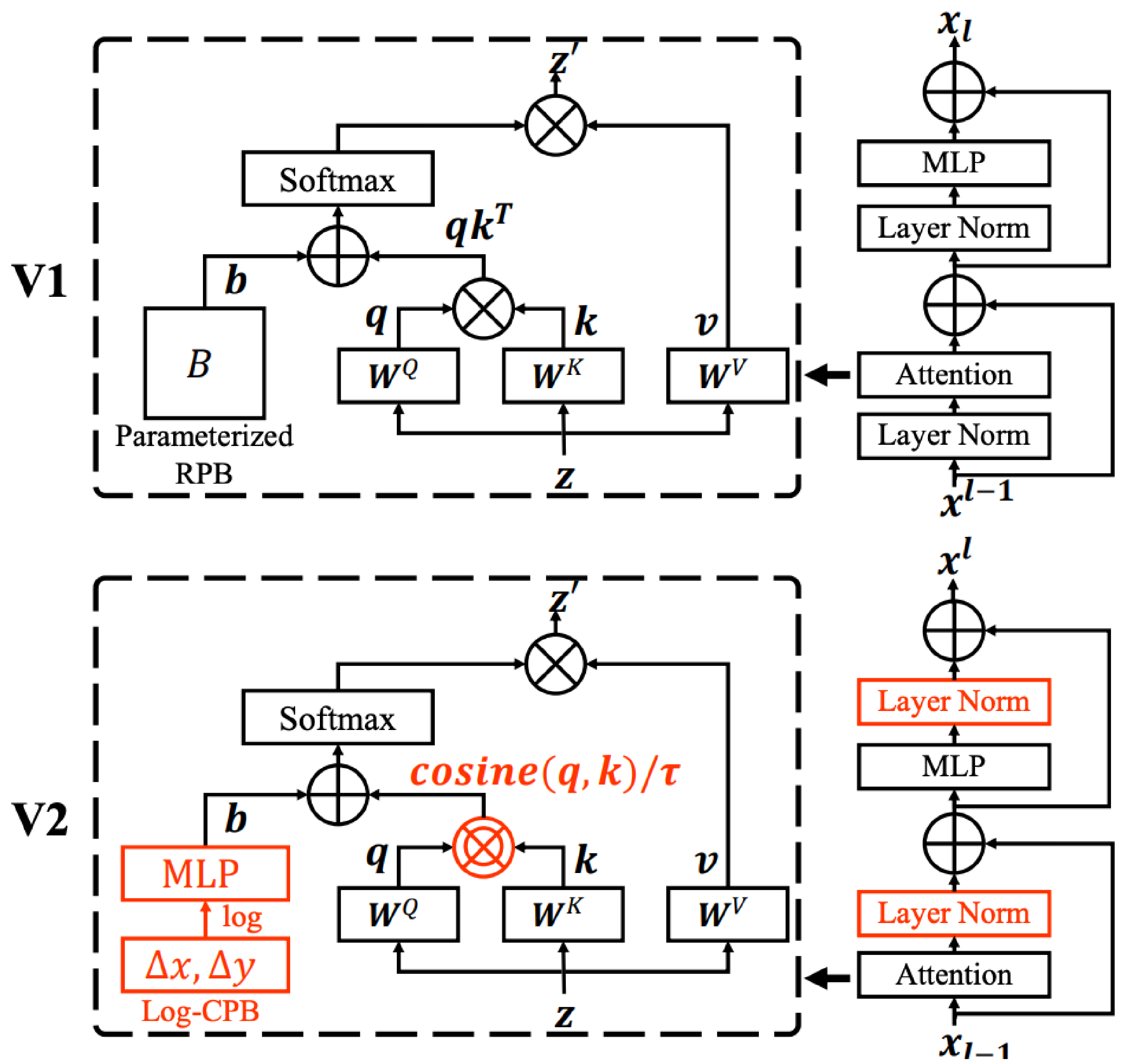
Other Hierarchical Vision Transformers

MViT



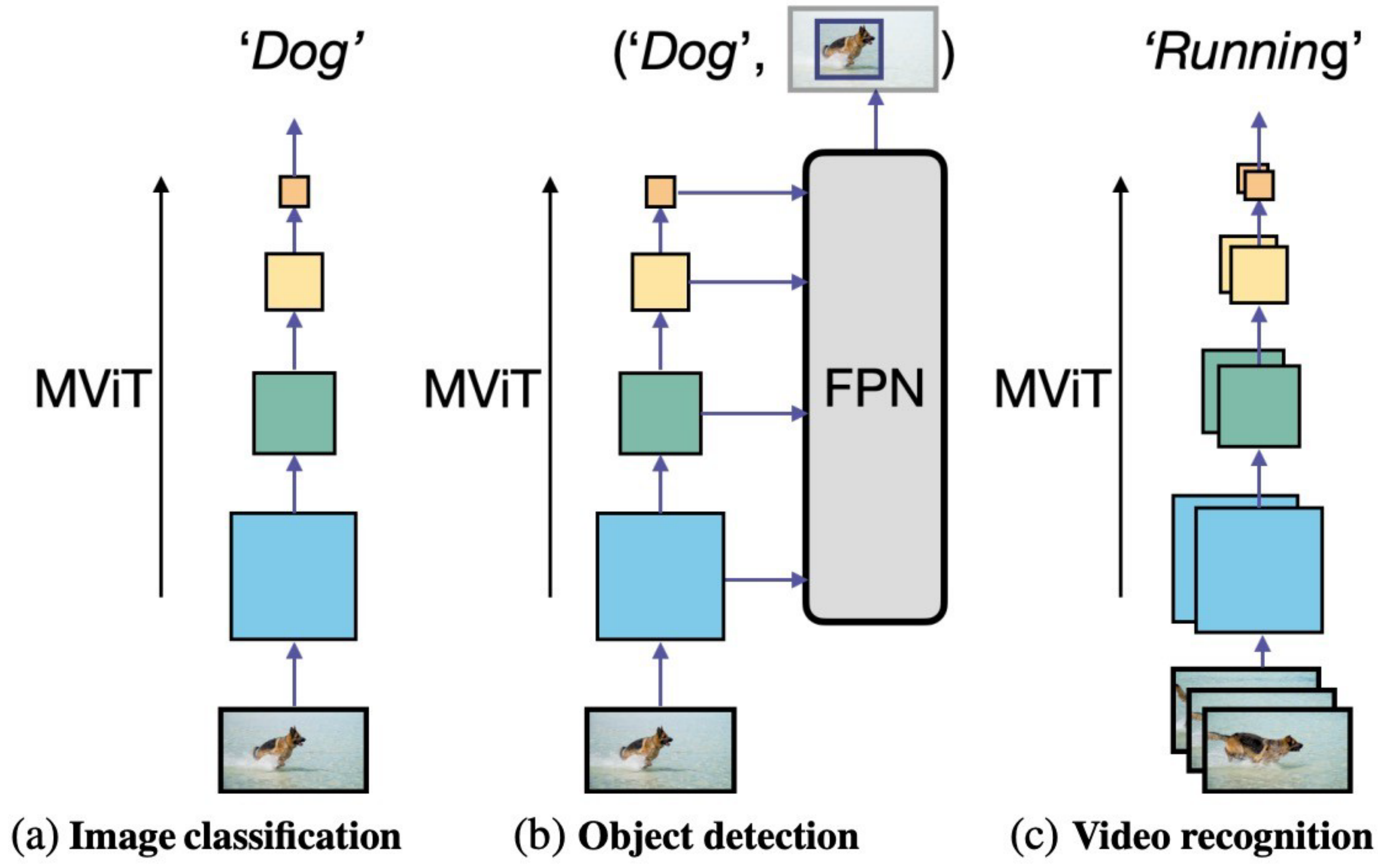
Fan et al, “Multiscale Vision Transformers”, ICCV 2021

Swin-V2



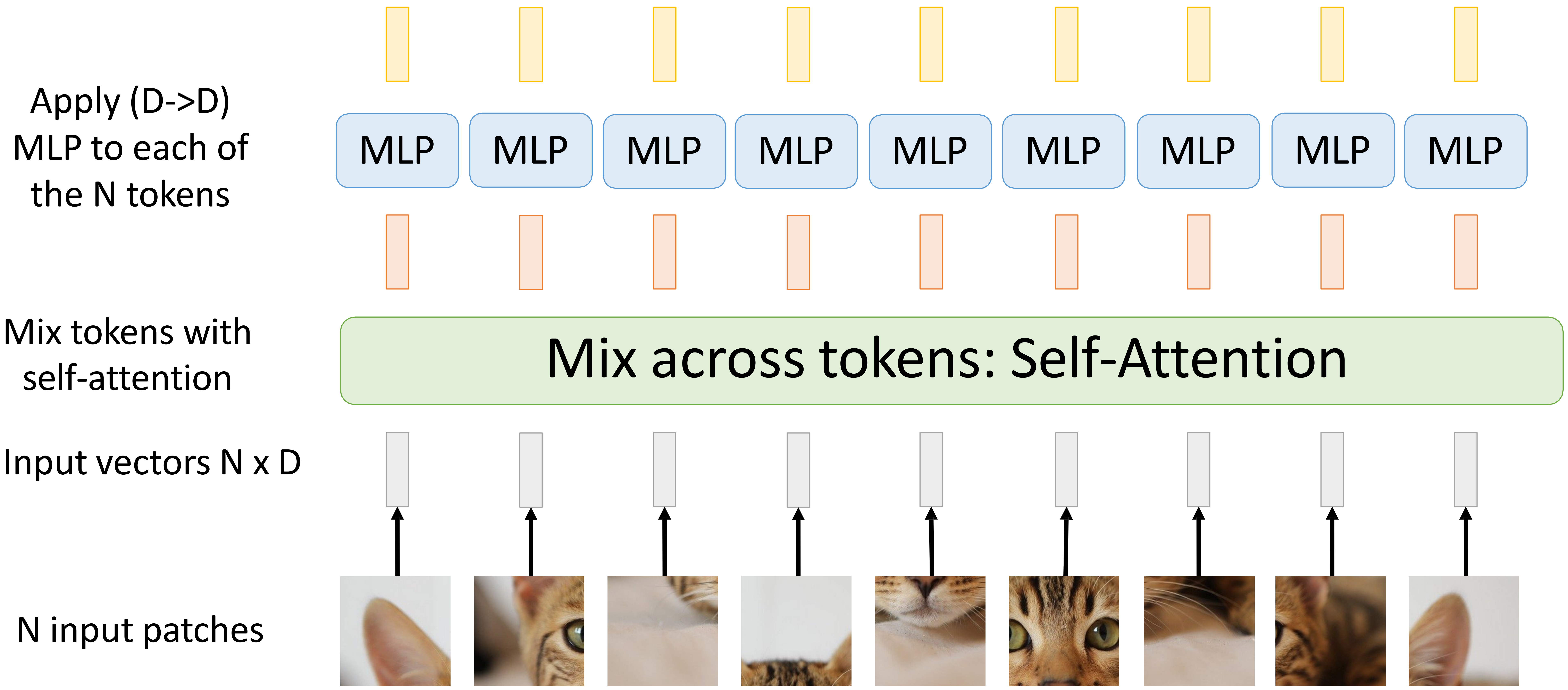
Liu et al, “Swin Transformer V2: Scaling up Capacity and Resolution”, CVPR 2022

Improved MViT



Li et al, “Improved Multiscale Vision Transformers for Classification and Detection”, arXiv 2021

Vision Transformer: Another Look



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

Cat image is free for commercial use under a Pixabay license

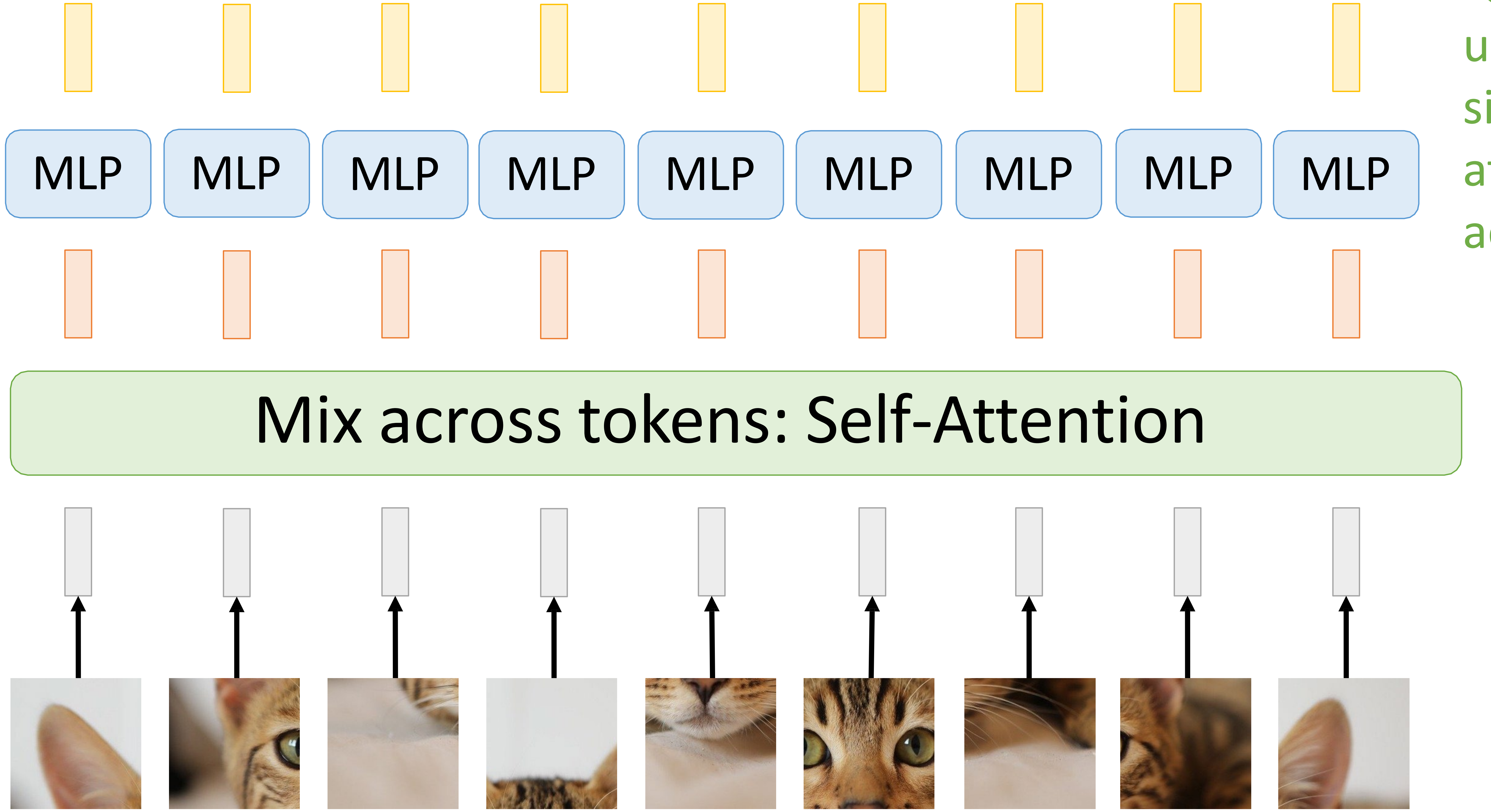
Vision Transformer: Another Look

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

Mix tokens with self-attention

Input vectors N x D

N input patches

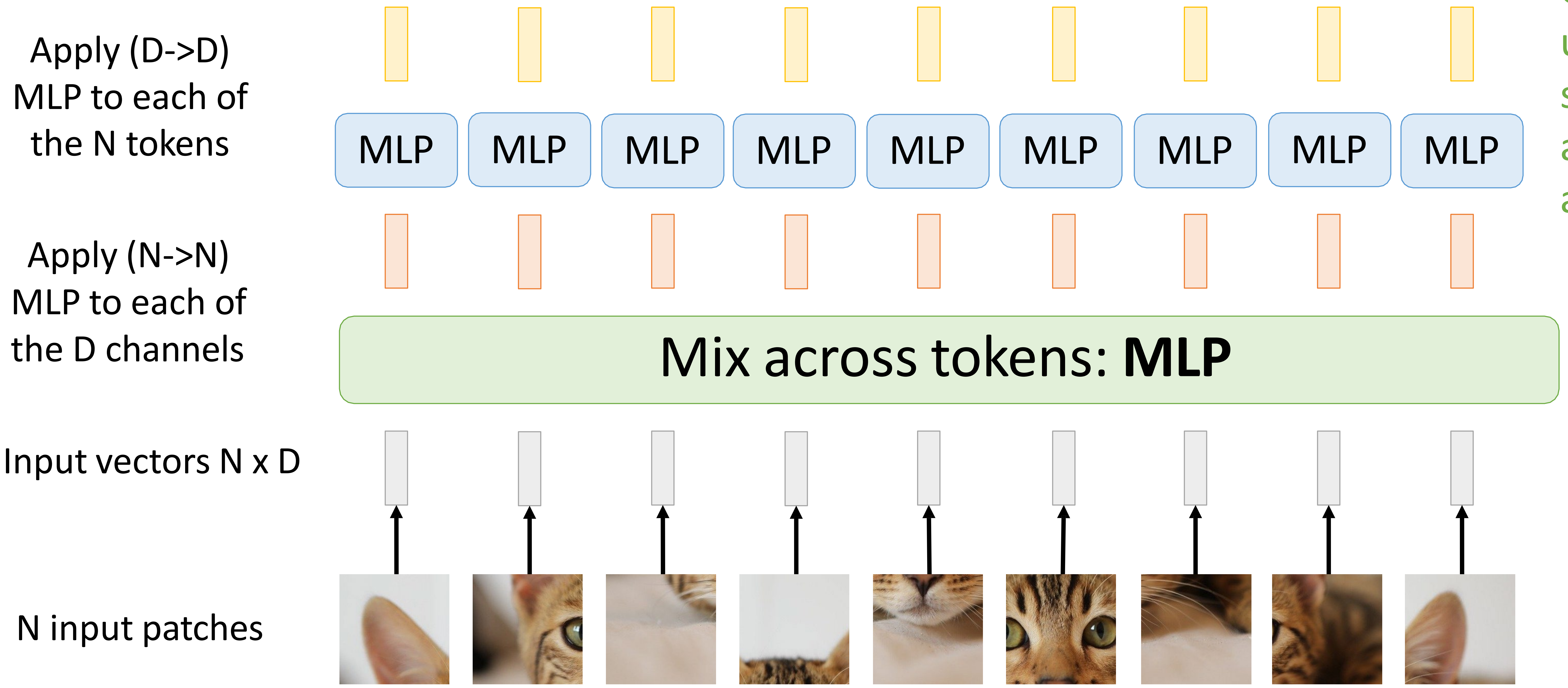


Question: Can we use something simpler than self-attention to mix across tokens?

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

Cat image is free for commercial use under a Pixabay license

MLP-Mixer: An All-MLP Architecture

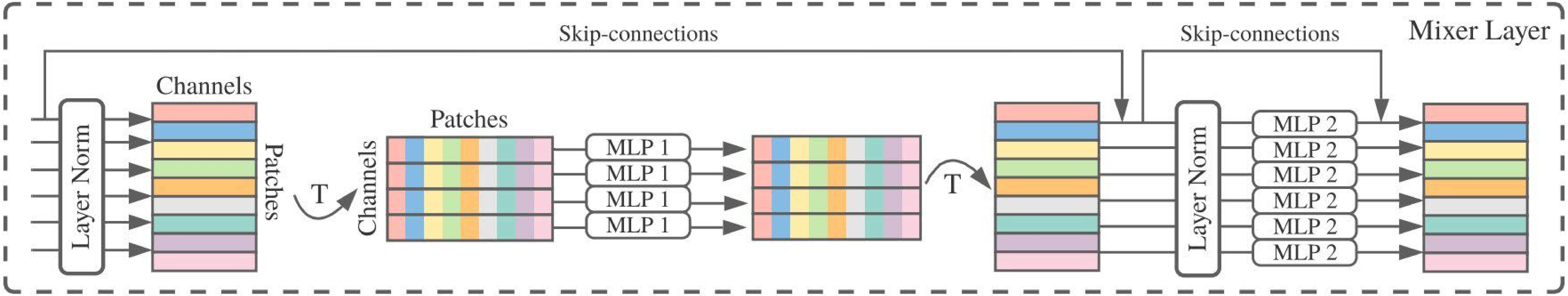


Question: Can we use something simpler than self-attention to mix across tokens?

Tolstikhin et al, "MLP-Mixer: An all-MLP architecture for vision", NeurIPS 2021

Cat image is free for commercial use under a Pixabay license

MLP-Mixer: An All-MLP Architecture



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**

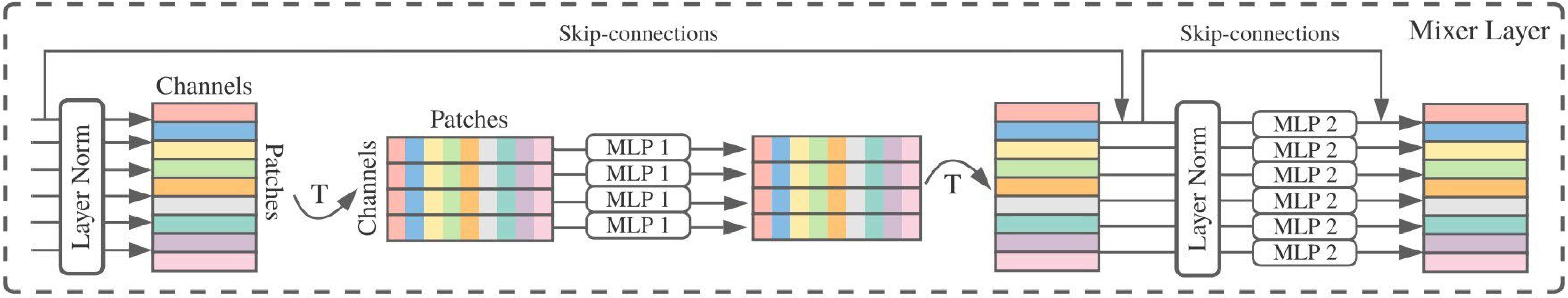
Tolstikhin et al, “MLP-Mixer: An all-MLP architecture for vision”, NeurIPS 2021

MLP-Mixer: An All-MLP Architecture

MLP-Mixer is actually just a weird CNN???

Equivalent to Conv(1x1, C->C, stride=1)

Equivalent to Conv($N^{1/2}$ x $N^{1/2}$, C->C, groups=C)



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**

Tolstikhin et al, “MLP-Mixer: An all-MLP architecture for vision”, NeurIPS 2021

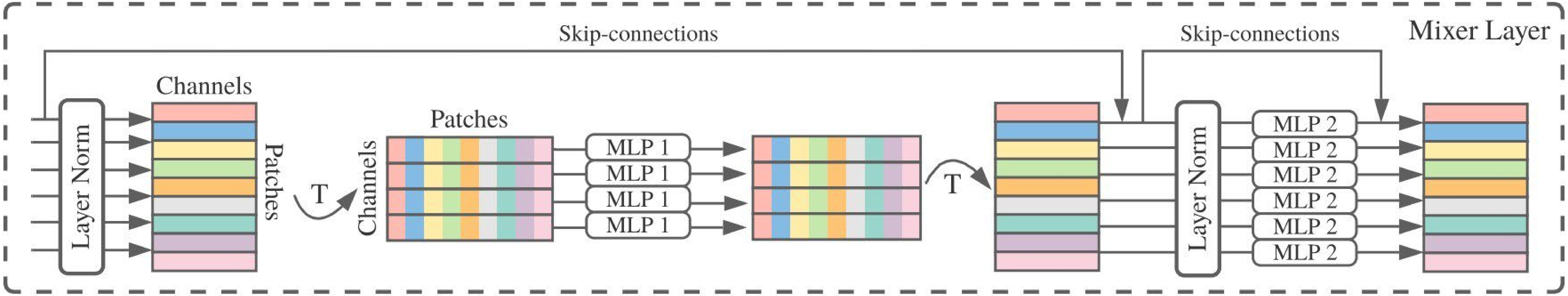
MLP-Mixer: An All-MLP Architecture

Cool idea; but initial ImageNet results not very compelling (but better with JFT pretraining)

MLP-Mixer is actually just a weird CNN???

Equivalent to
Conv(1x1, C->C, stride=1)

Equivalent to
Conv($N^{1/2}$ x $N^{1/2}$, C->C, groups=C)



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**

Tolstikhin et al, “MLP-Mixer: An all-MLP architecture for vision”, NeurIPS 2021

MLP-Mixer: Many concurrent and followups

Touvron et al, “ResMLP: Feedforward Networks for Image Classification with Data-Efficient Training”, arXiv 2021, https://arxiv.org/abs/2105.03404

Tolstikhin et al, “MLP-Mixer: An all-MLP architecture for vision”, NeurIPS 2021, https://arxiv.org/abs/2105.01601

Liu et al, “Pay Attention to MLPs”, NeurIPS 2021, https://arxiv.org/abs/2105.08050

Yu et al, “S2-MLP: Spatial-Shift MLP Architecture for Vision”, WACV 2022, https://arxiv.org/abs/2106.07477

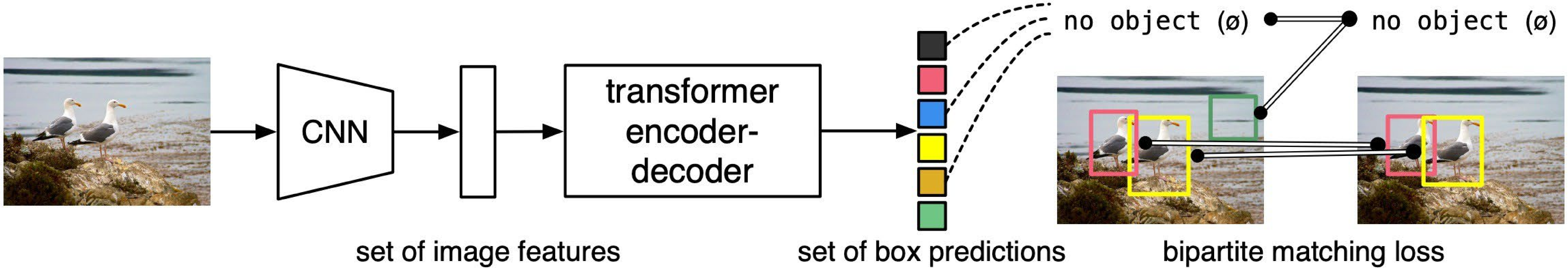
Chen et al, “CycleMLP: A MLP-like Architecture for Dense Prediction”, ICLR 2022, https://arxiv.org/abs/2107.10224

Object Detection with Transformers: DETR

Simple object detection pipeline: directly output a set of boxes from a Transformer

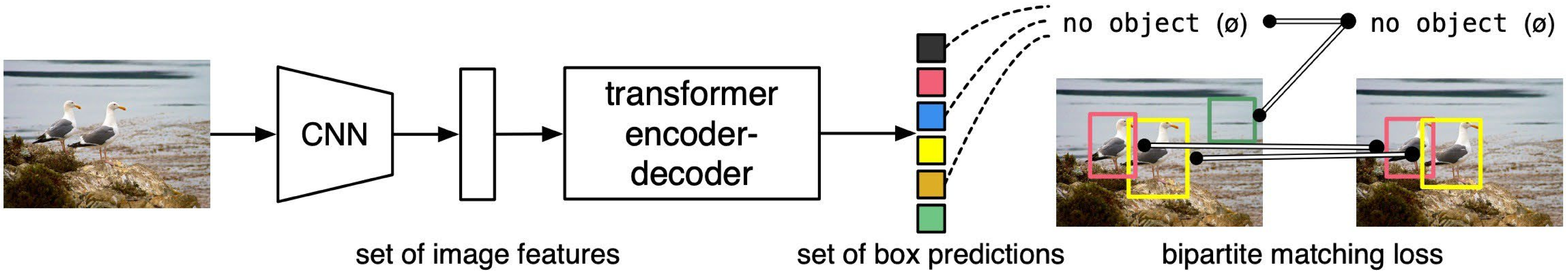
No anchors, no regression of box transforms

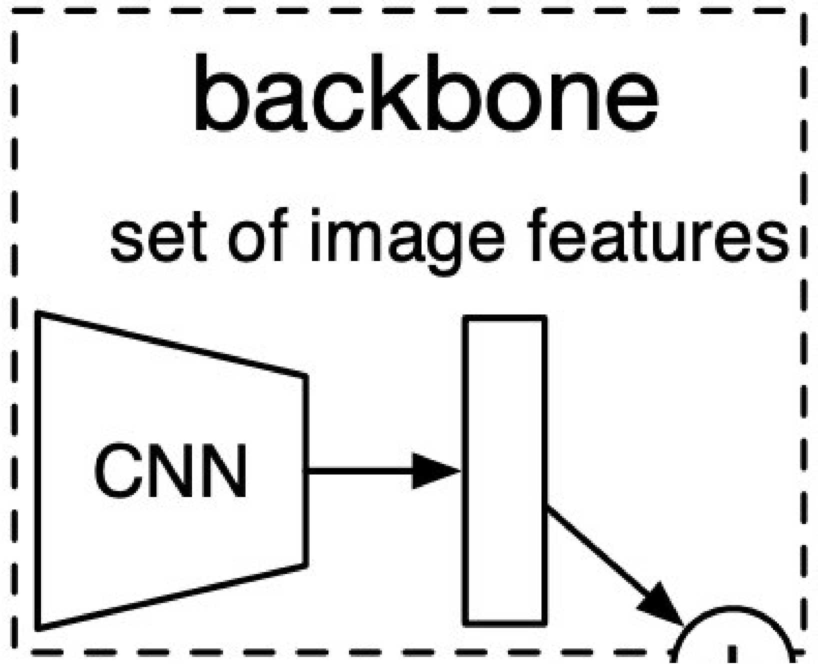
Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates



Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

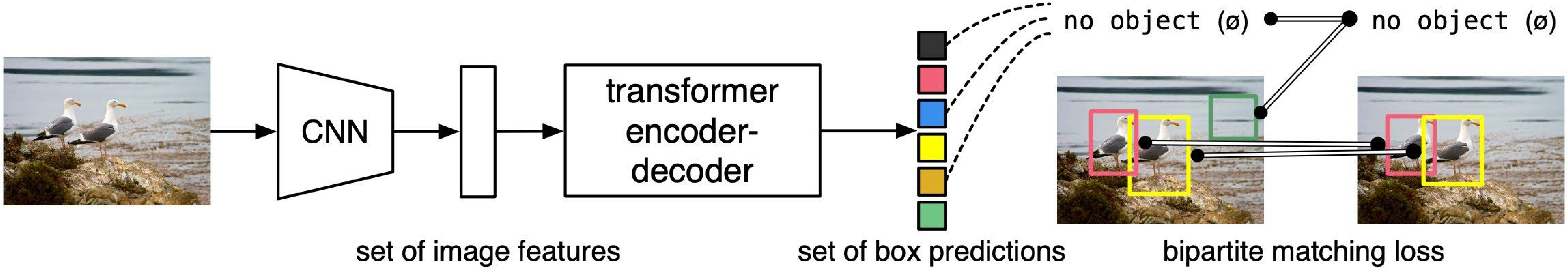
Object Detection with Transformers: DETR

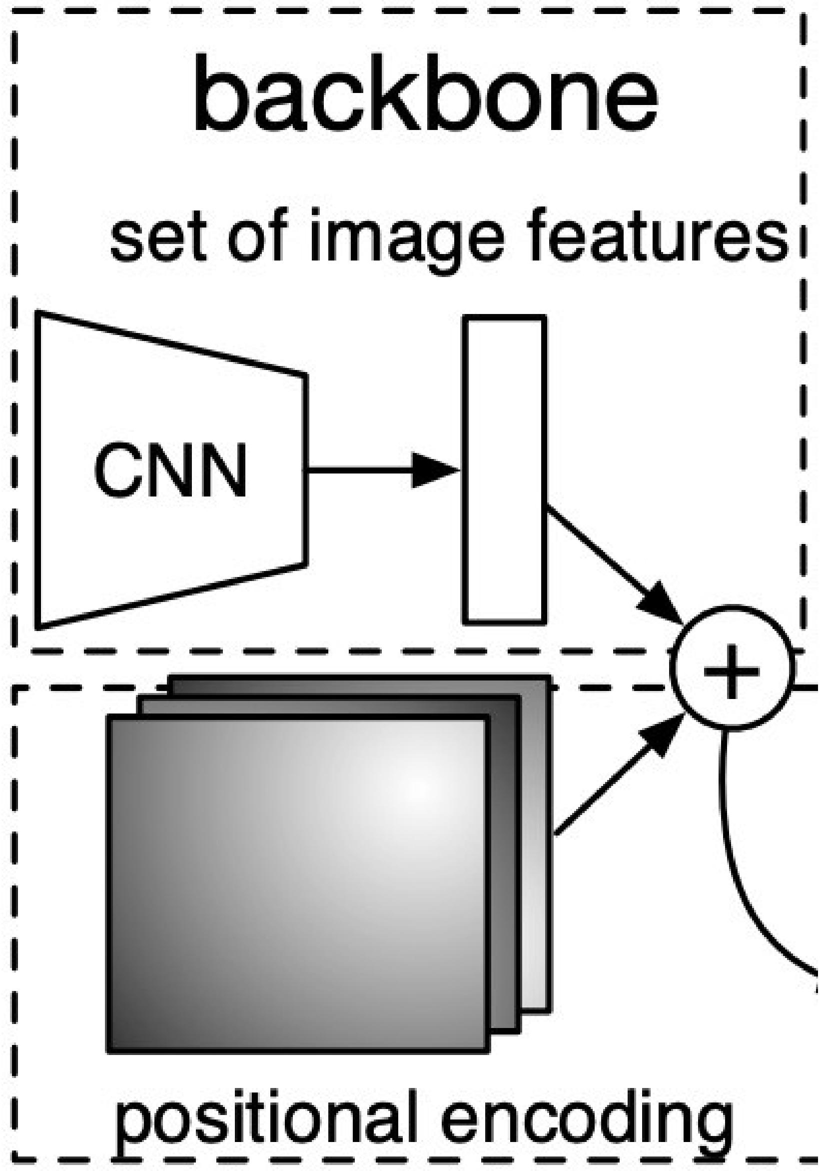




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

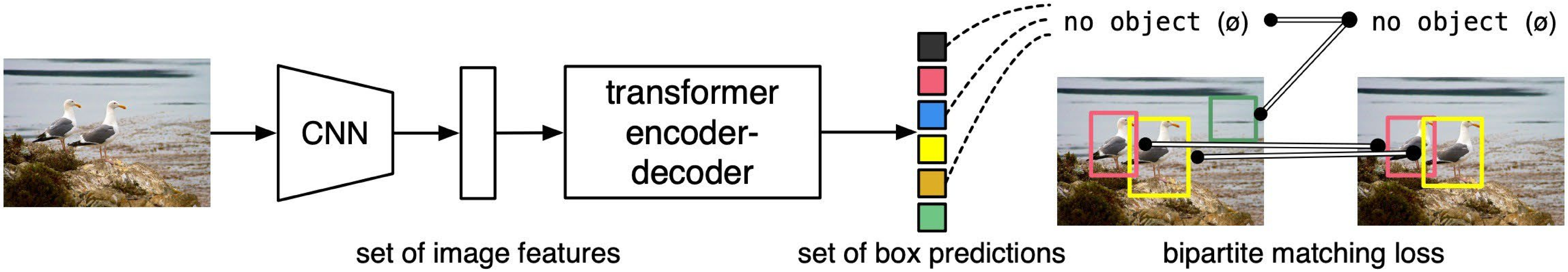
Object Detection with Transformers: DETR

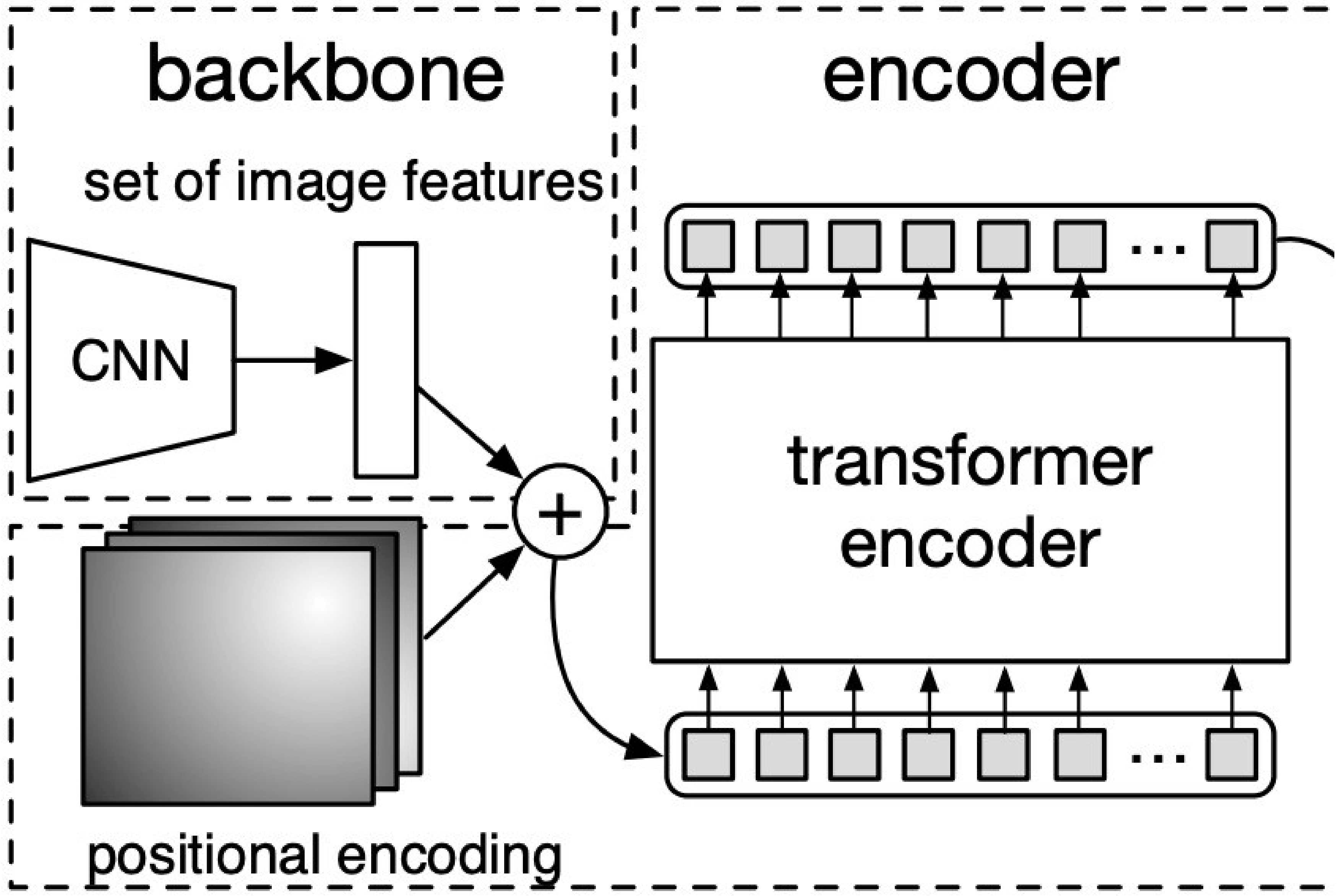




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

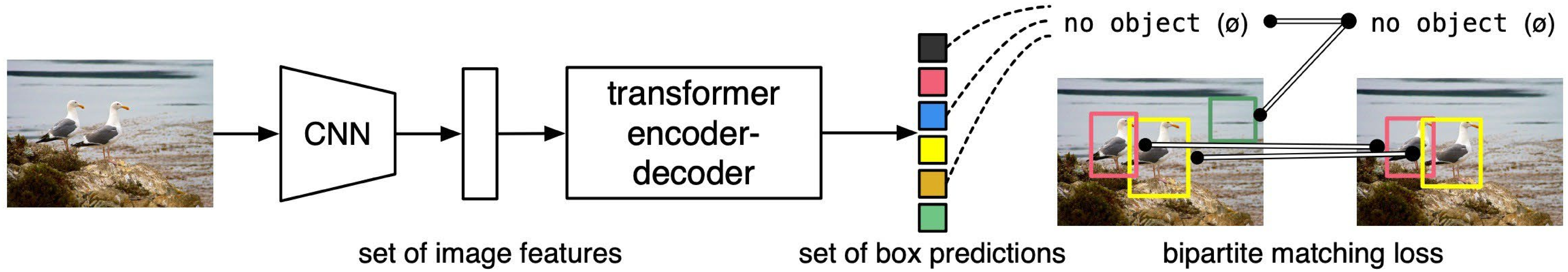
Object Detection with Transformers: DETR

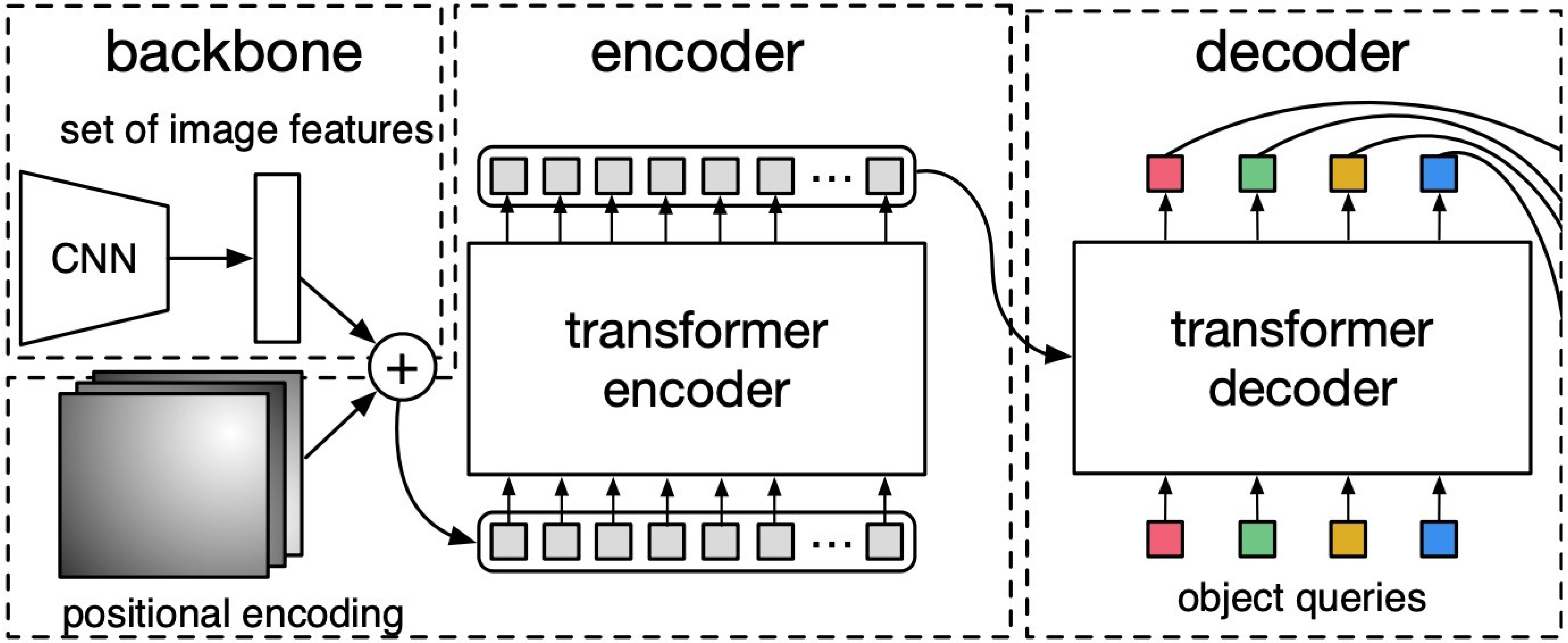




Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

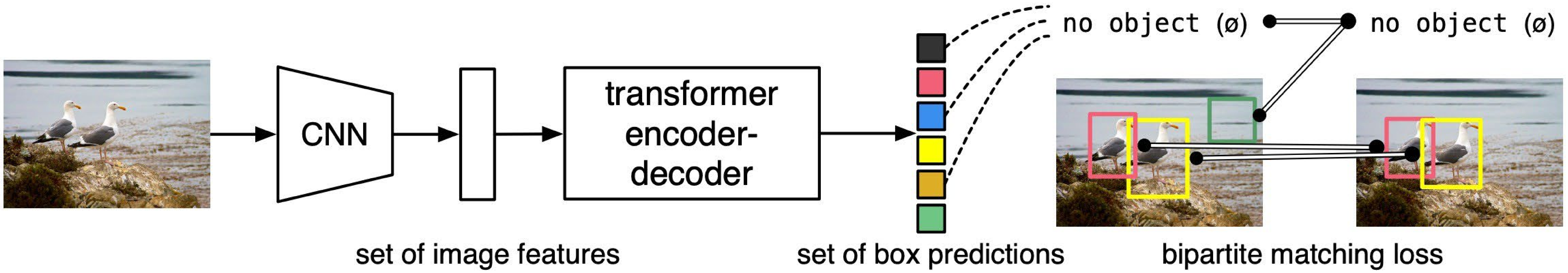
Object Detection with Transformers: DETR

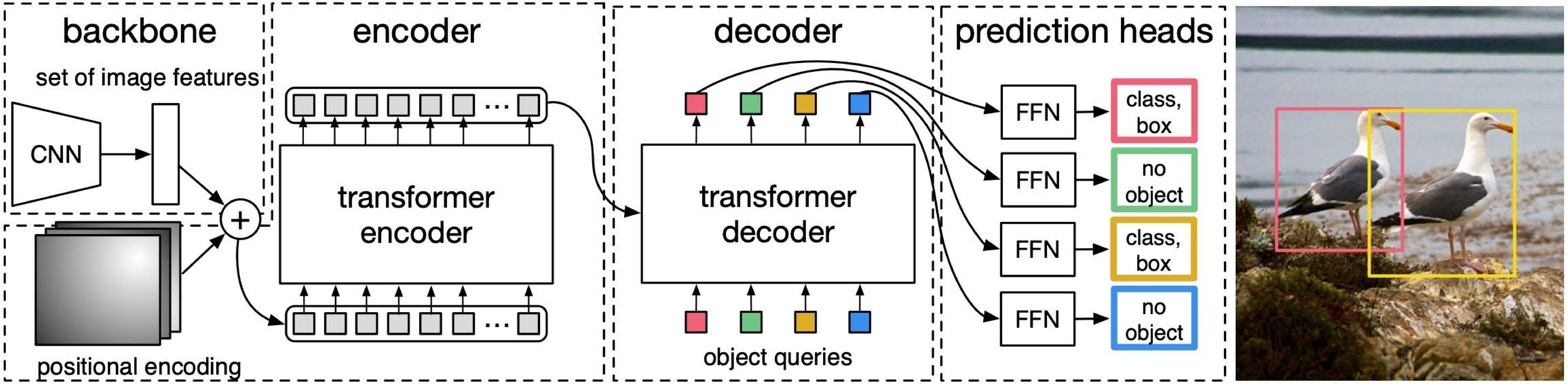




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

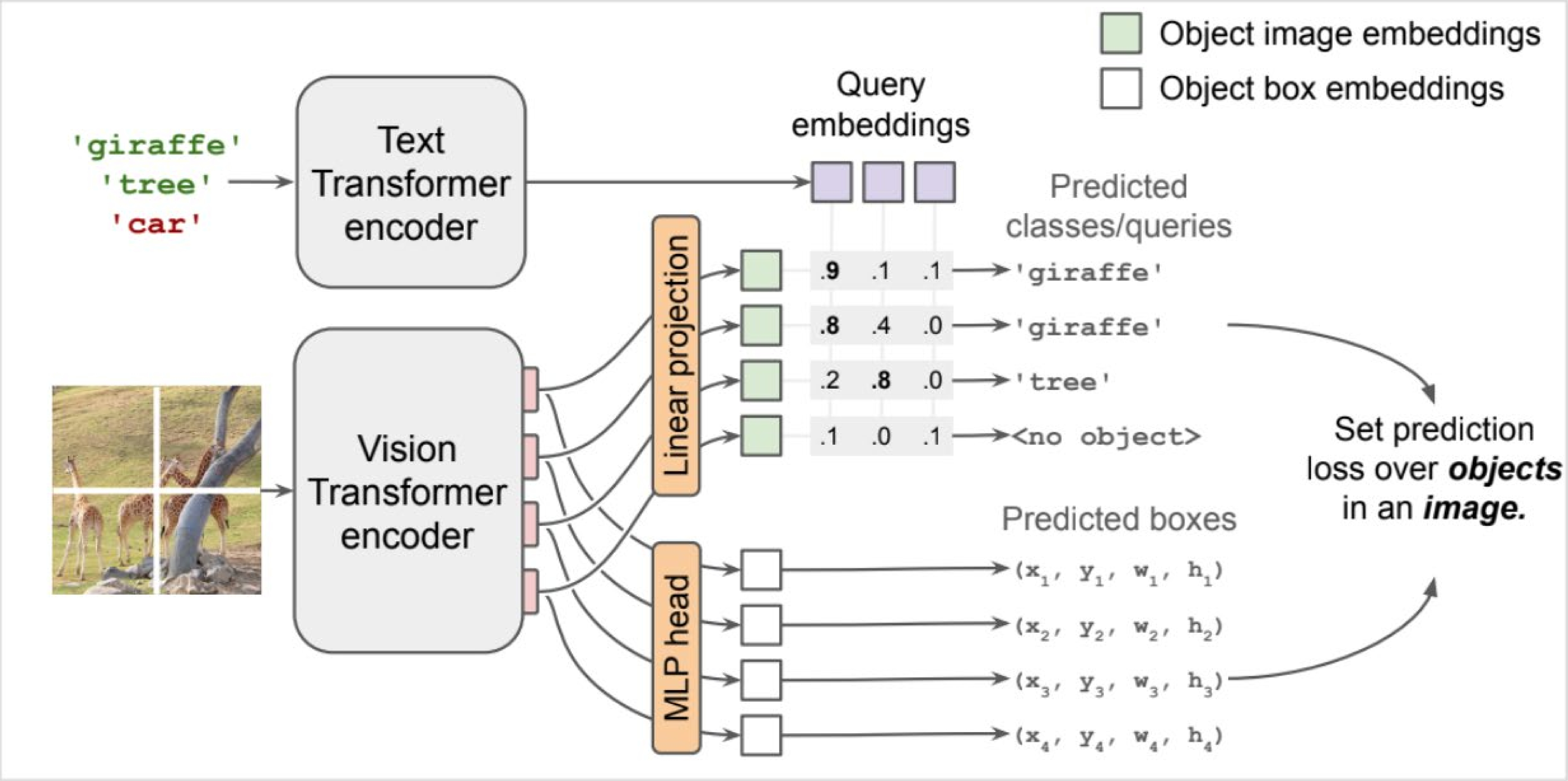
Object Detection with Transformers: DETR





Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

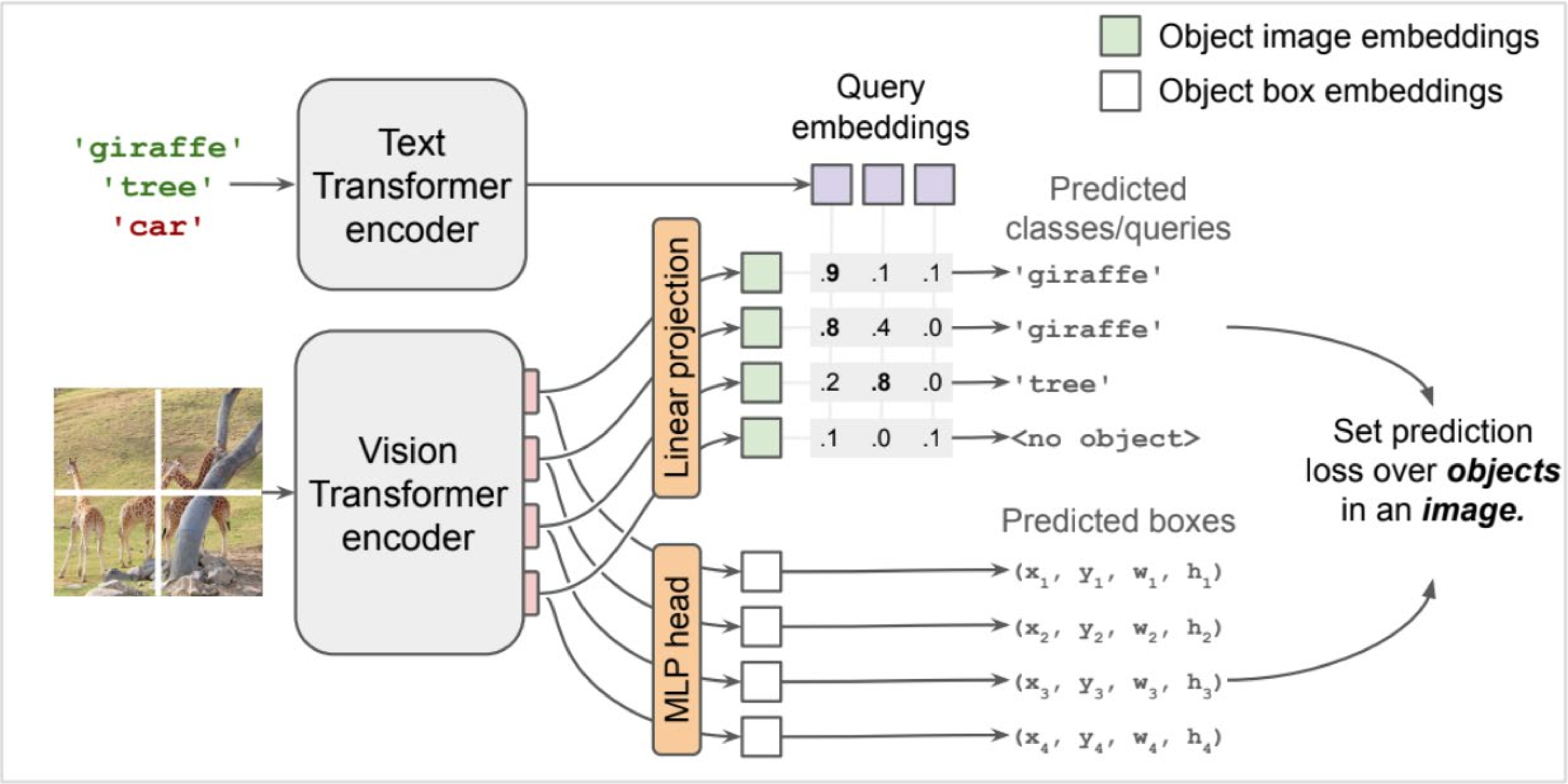
OWL-ViT:


Object image embeddings
Object box embeddings
'giraffe'
'tree'
'car'
Text Transformer encoder
Query embeddings
Predicted classes/queries
Linear projection
.9 .1 .1
'giraffe'
.8 .4 .0
'giraffe'
.2 .8 .0
'tree'
.1 .0 .1
<no object>
Vision Transformer encoder
Set prediction loss over objects in an image.
Predicted boxes
MLP head
(x1, y1, w1, h1)
(x2, y2, w2, h2)
(x3, y3, w3, h3)
(x4, y4, w4, h4)

OWL-ViT:

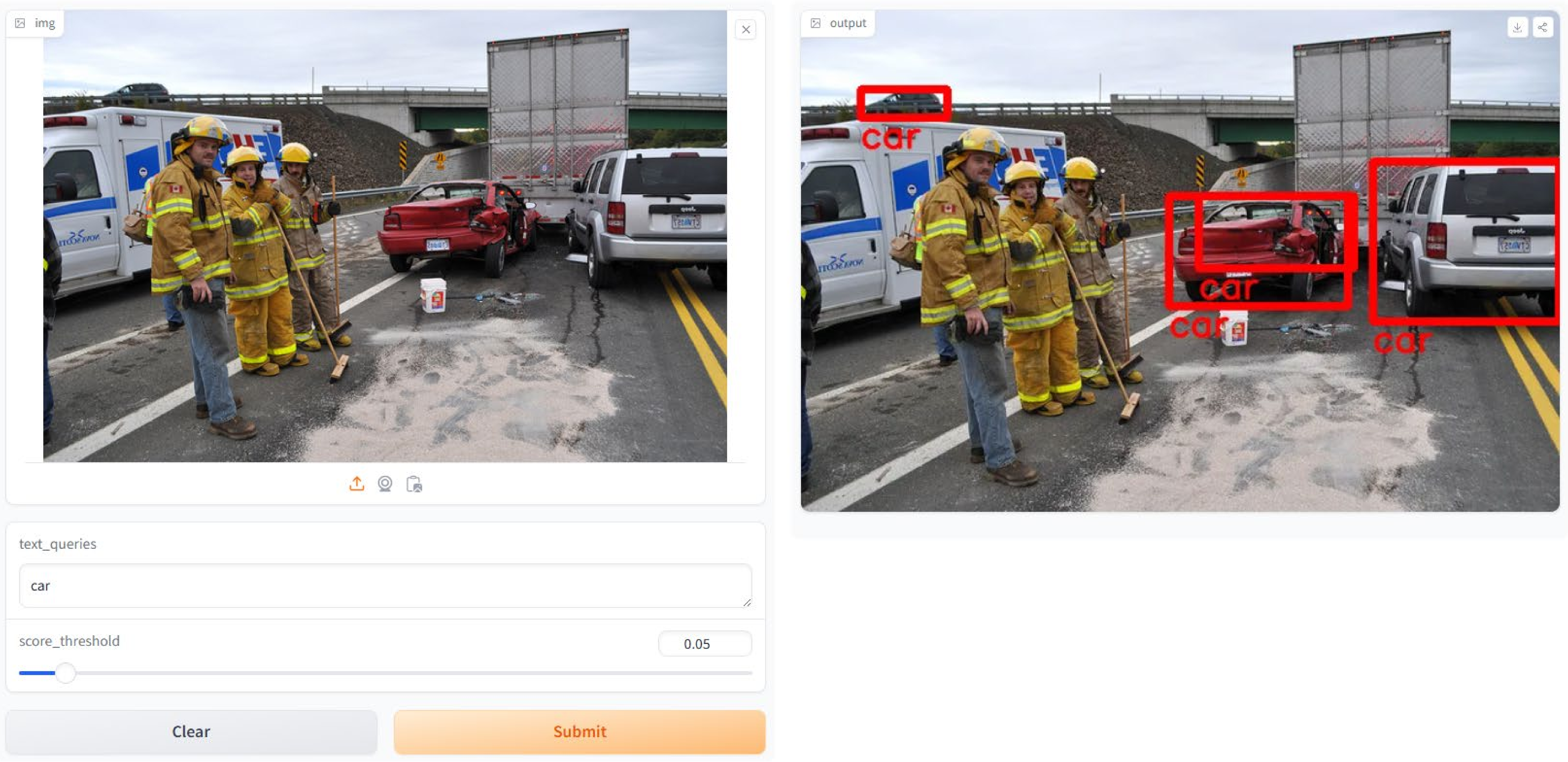
Given: image + ***free-text queries***

Unlike traditional object detection models,
- not trained on labeled object datasets
- leverages multi-modal representations
- performs open-vocabulary detection.

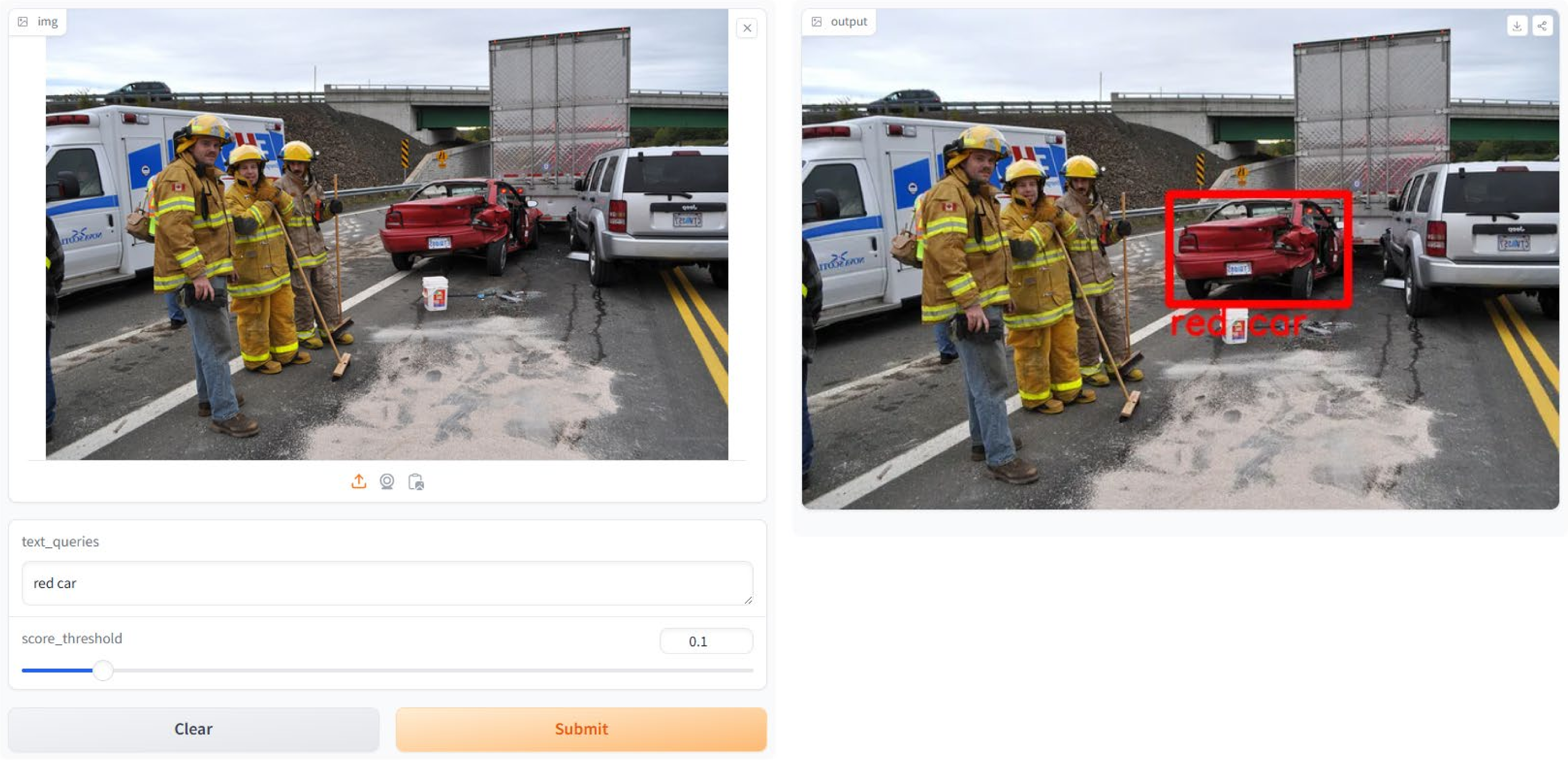


CLIP with a ViT-like Transformer as its backbone to get visual and text features.
- To use CLIP for object detection, OWL-ViT removes the final token pooling layer of the vision model and attaches a lightweight classification and box head to each transformer output token.
- Open-vocabulary classification is enabled by replacing the fixed classification layer weights with the class-name embeddings obtained from the text model.

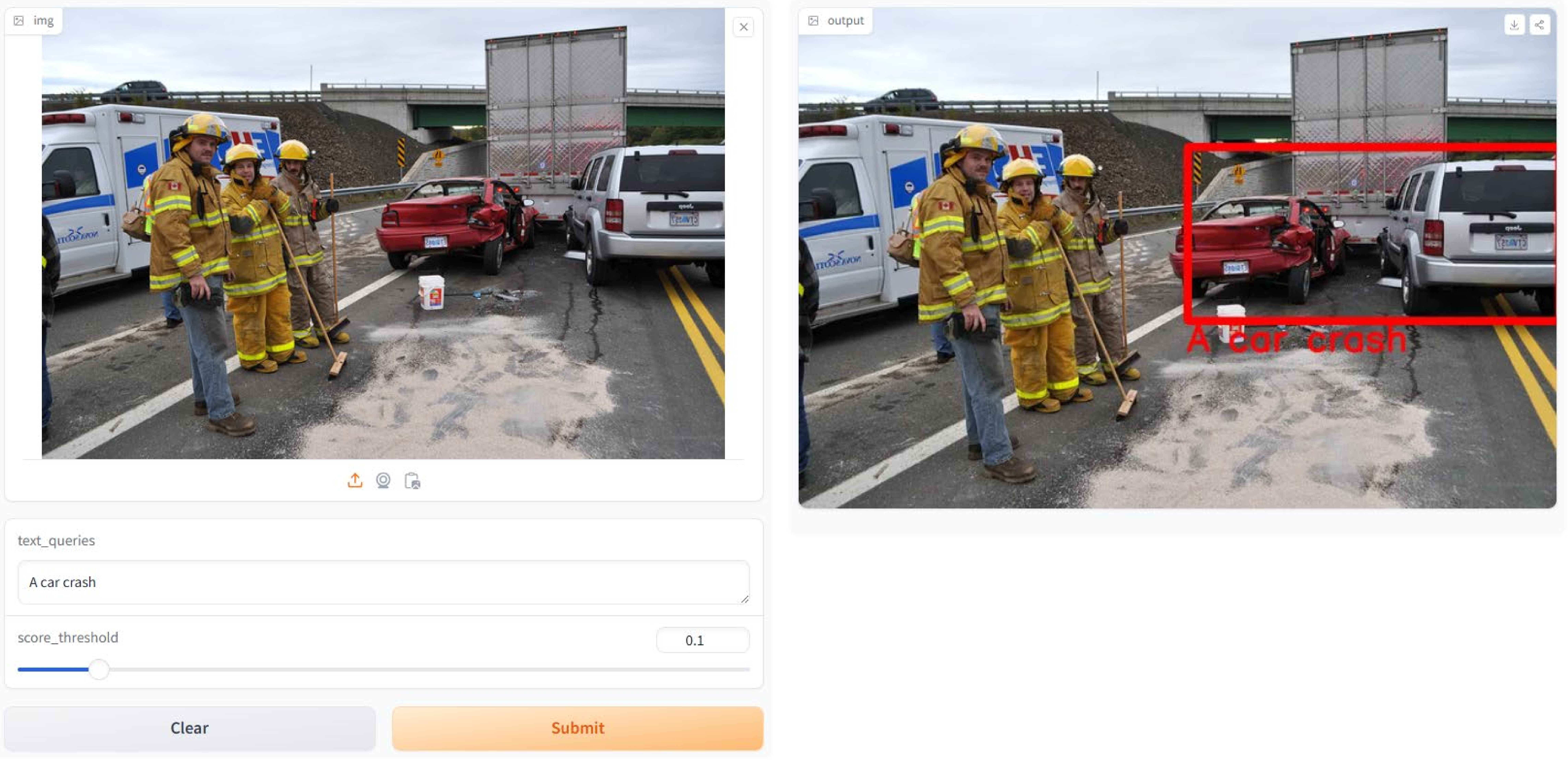
Examples:


img
text_queries
car
score_threshold
0.05
Clear
Submit
output
car
car
car
car

Examples:


img
output
text_queries
red car
score_threshold
0.1
Clear
Submit
red car

Examples:


img
output
A car crash
text_queries
A car crash
score_threshold
0.1
Clear
Submit

Examples:

img

text_queries

sand on the highway

score_threshold

0.04

Clear

Submit

output


img
output
empty bottle
green napkin
green napkin
empty bottl
text_queries
empty bottle, green napkin, red plate