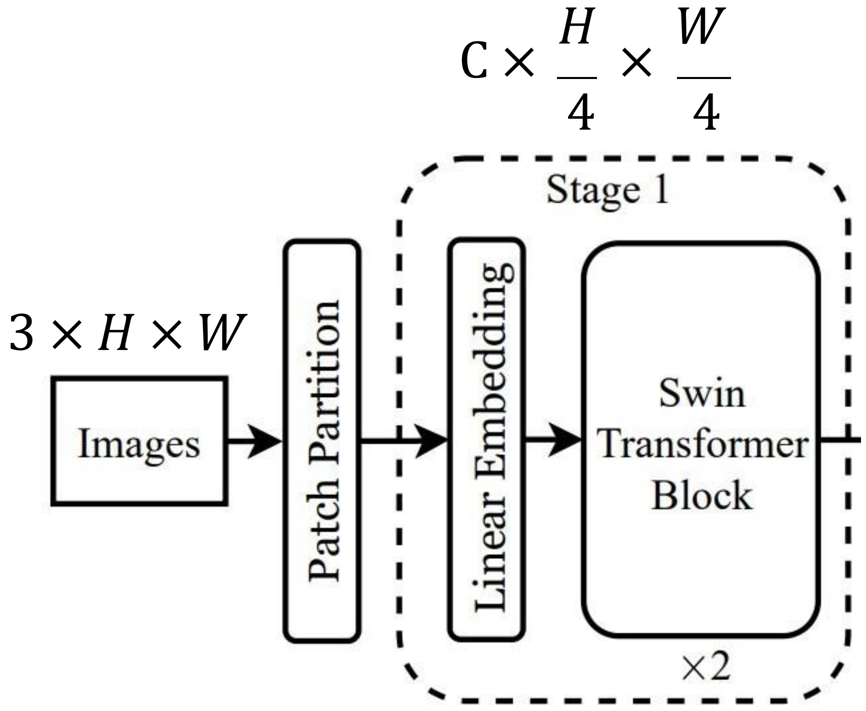
Previously –
Attention and variants
ViT, reg., aug., distill. for improving ViT

Today:
Swin – SOTA, introduces hierarchy
SSL, train with no labels

https://lightning.ai/

Assign – 1 is uploaded

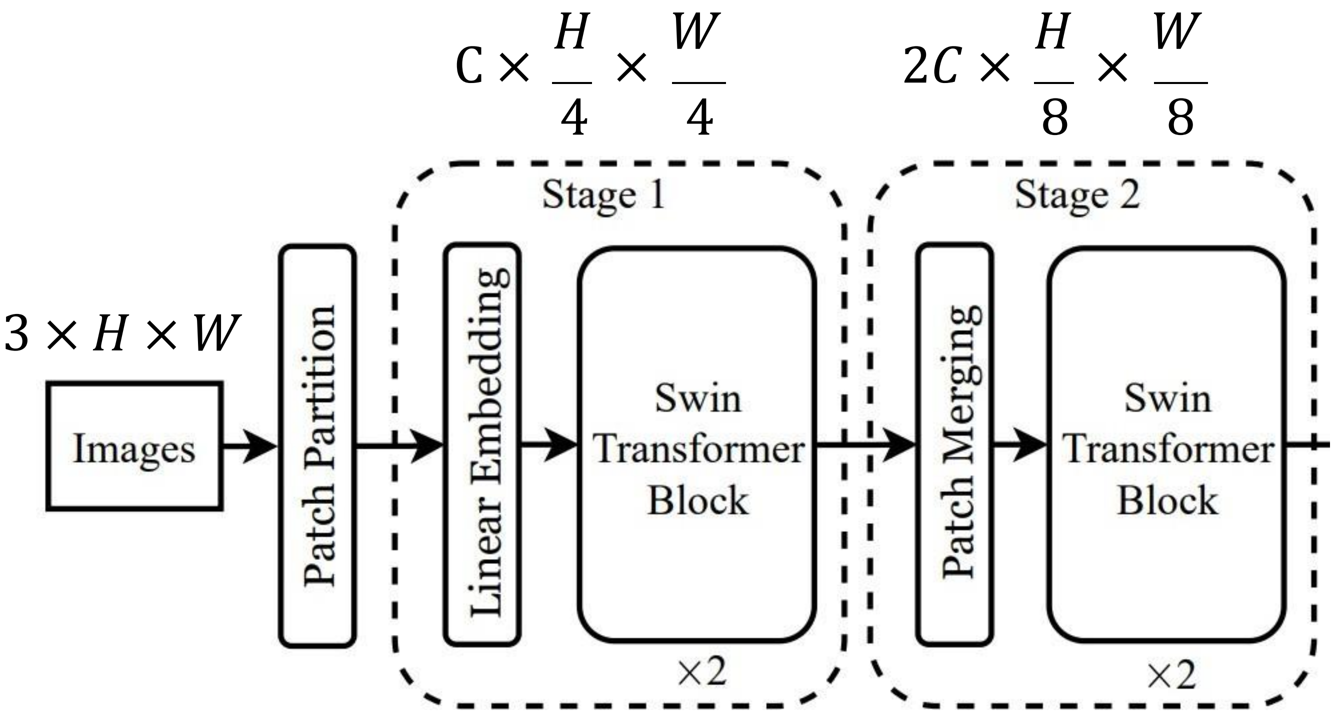
Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

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

Hierarchical ViT: Swin Transformer

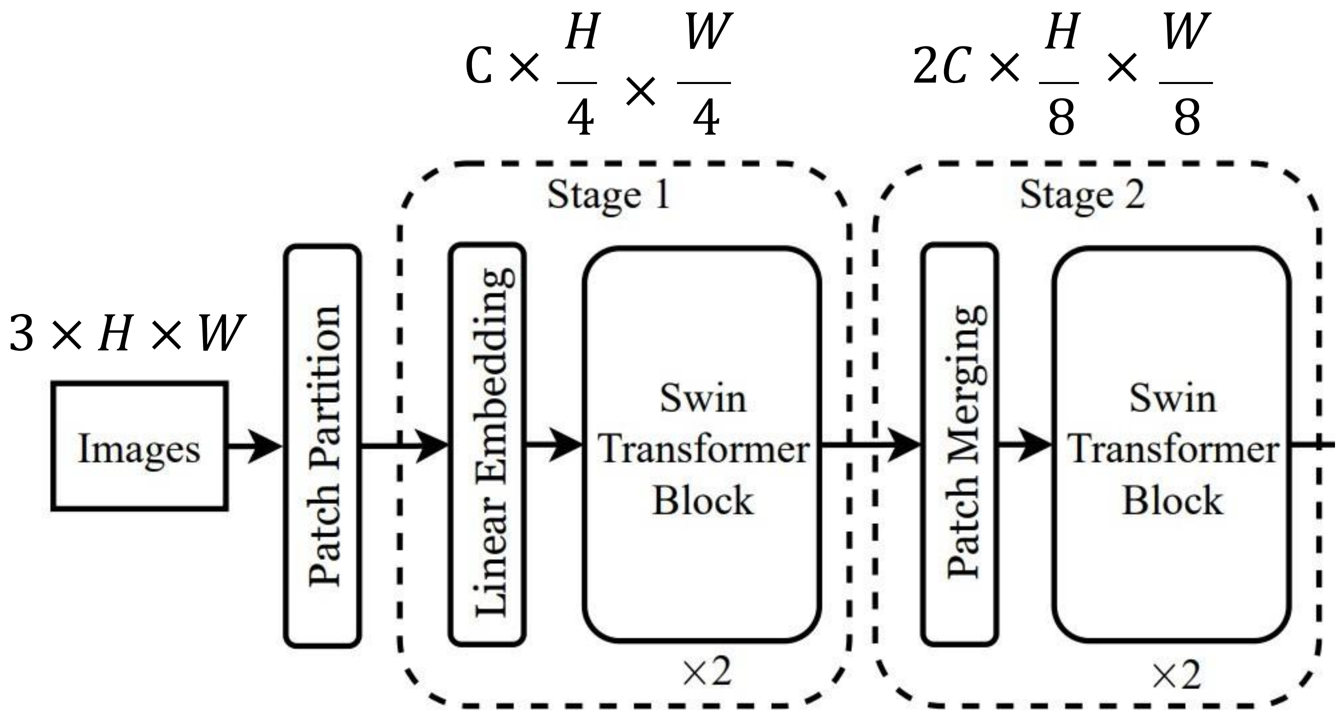


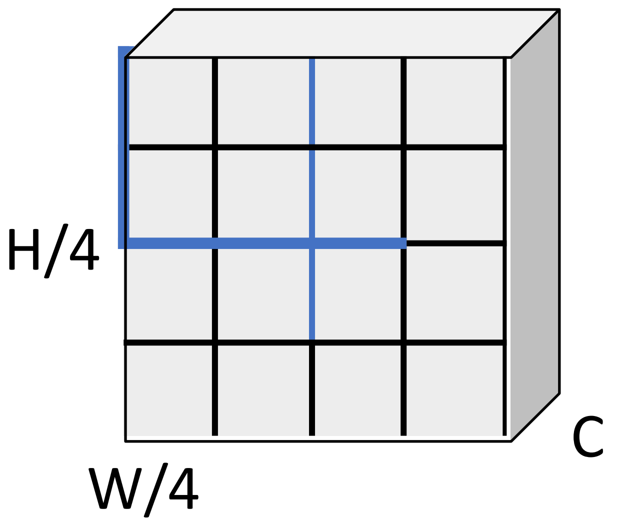
Divide image into 4x4 patches and project to C dimensions

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



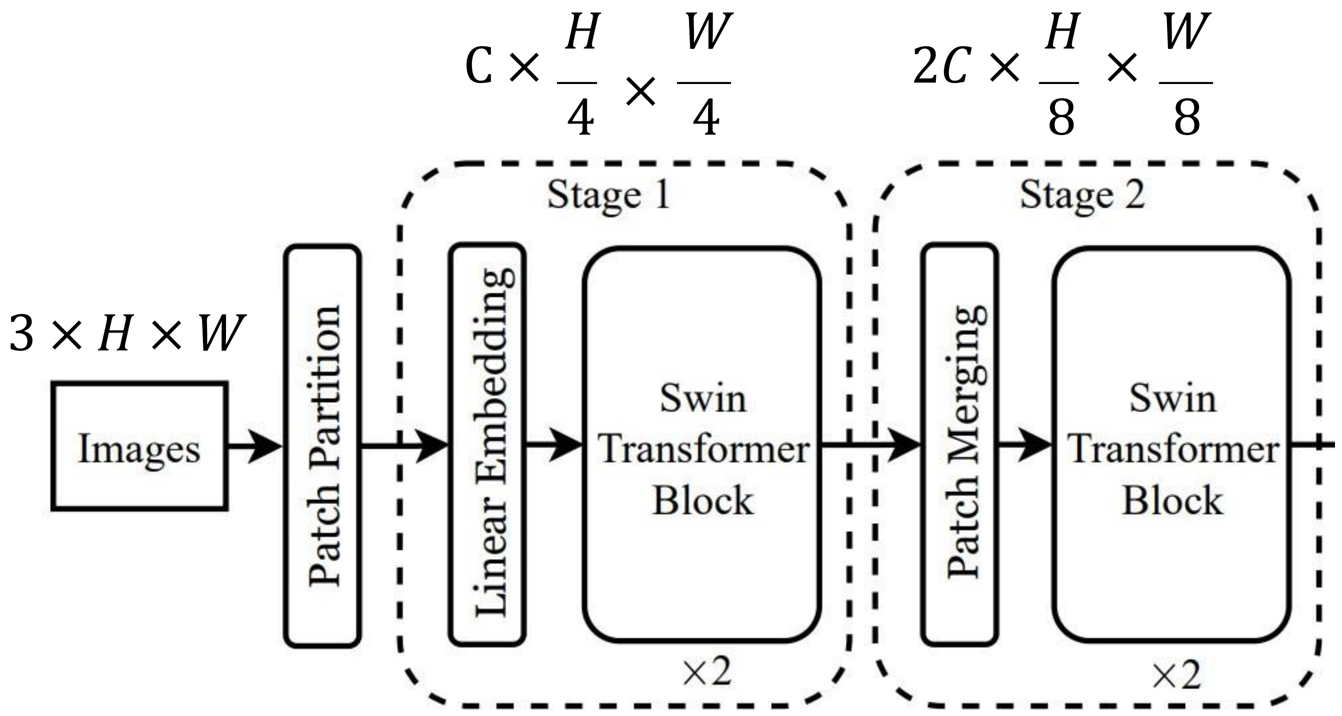


Divide image into 4x4 patches and project to C dimensions

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

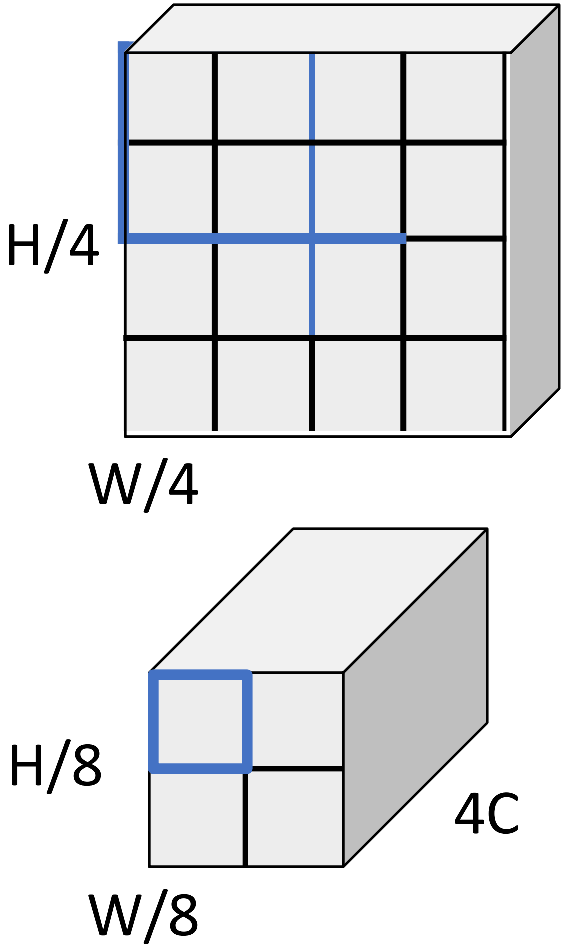
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

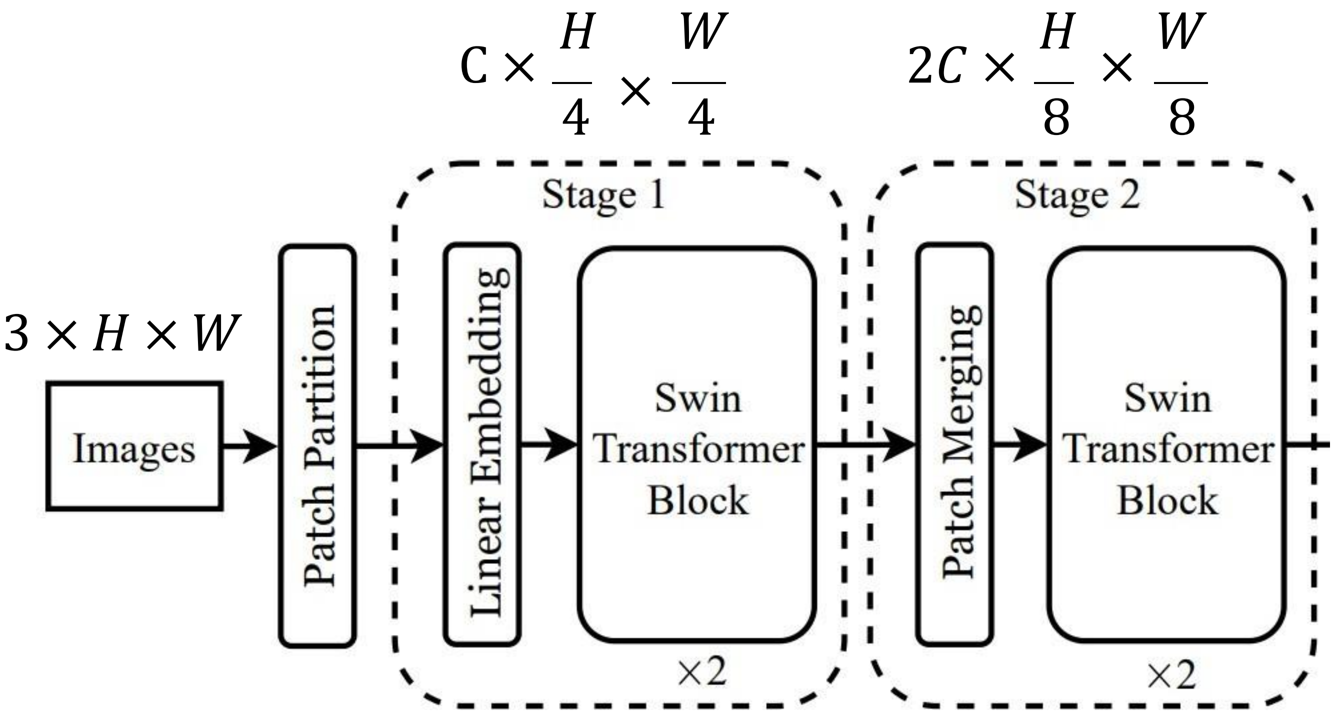
Merge 2x2 neighborhoods; now patches are (effectively) 8x8



Concatenate groups of 2x2 features

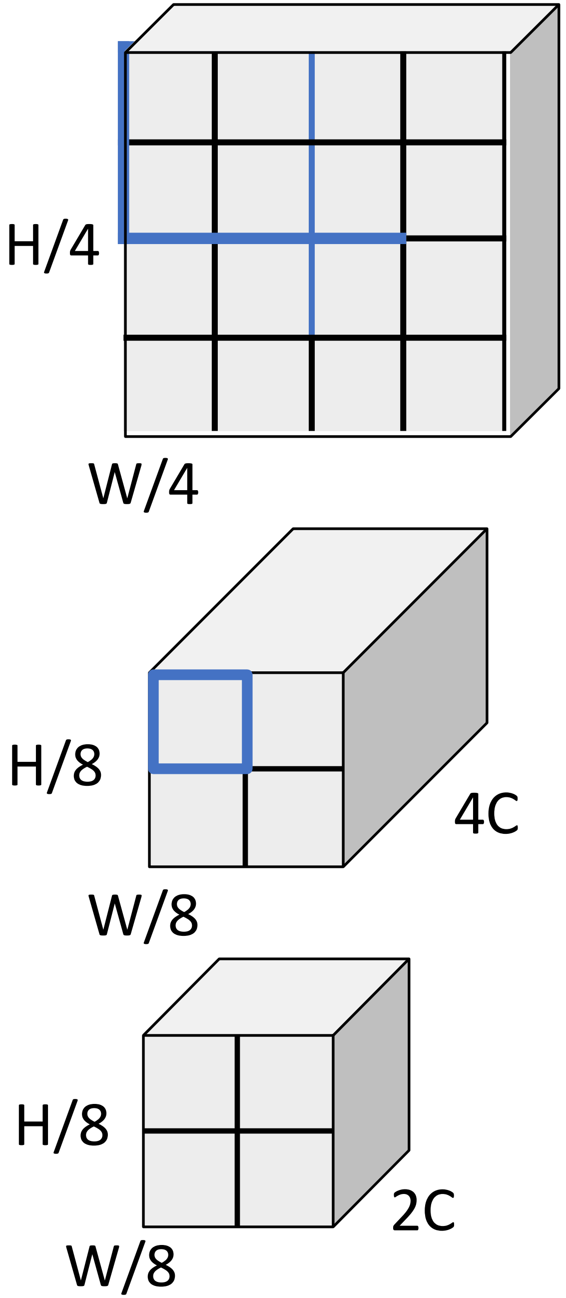
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

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

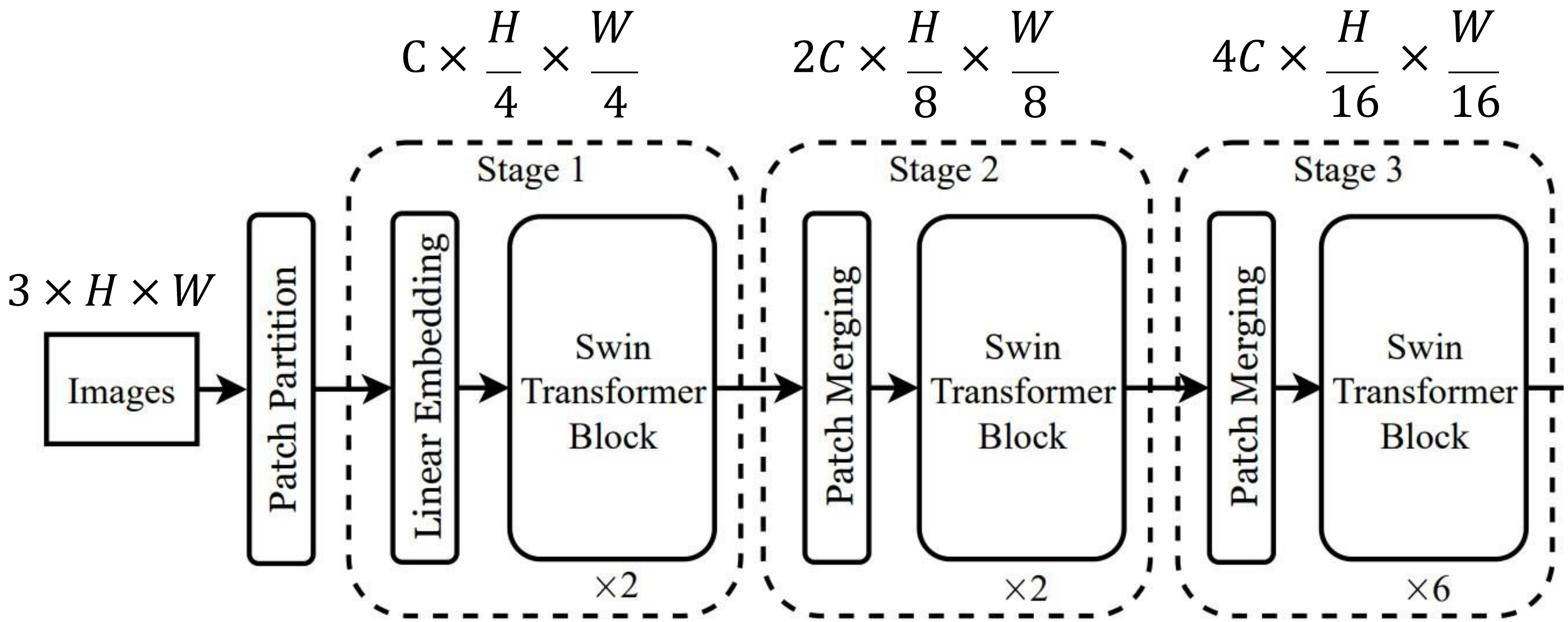


Concatenate groups of 2x2 features

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



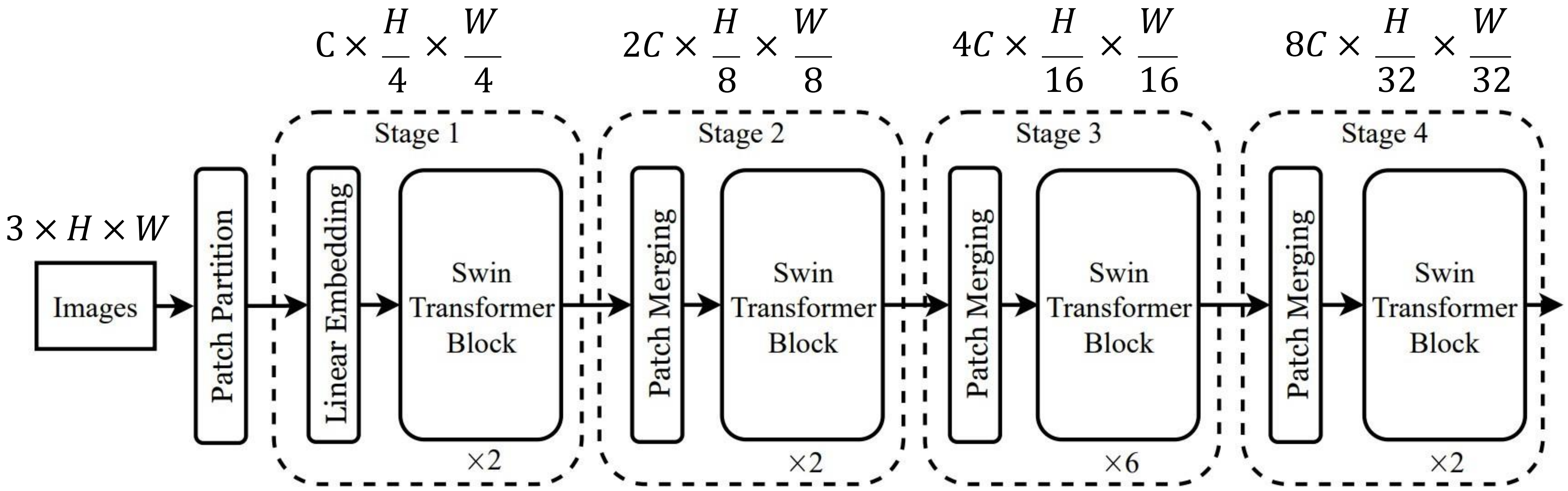
Divide image into 4x4 patches and project to C dimensions

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

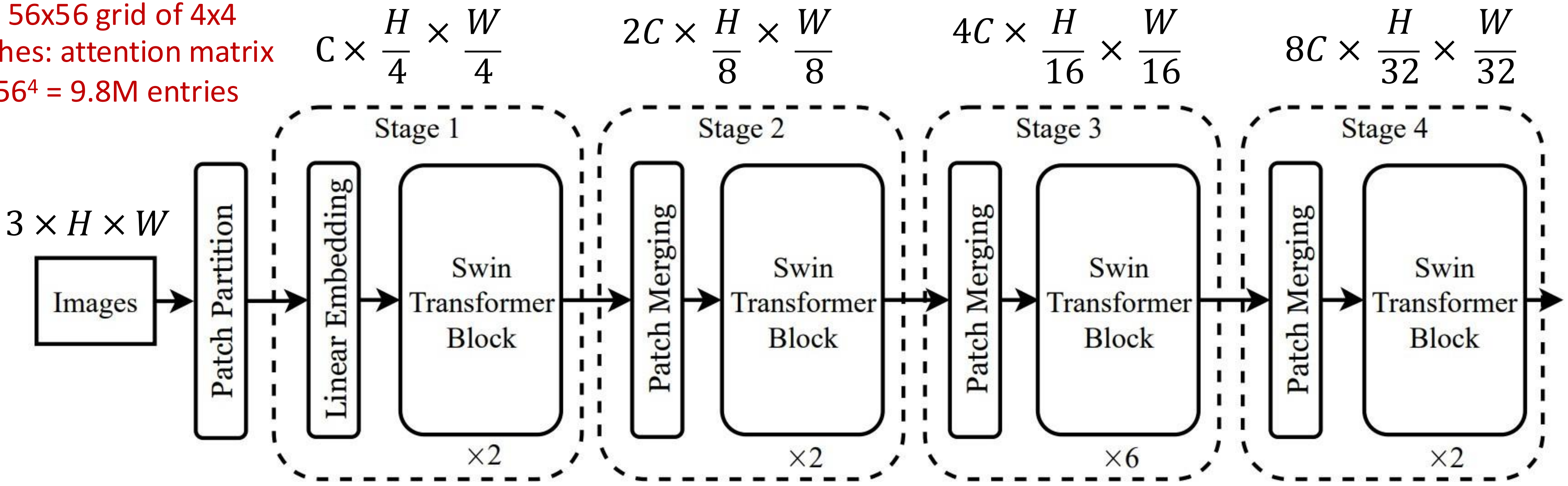
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

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

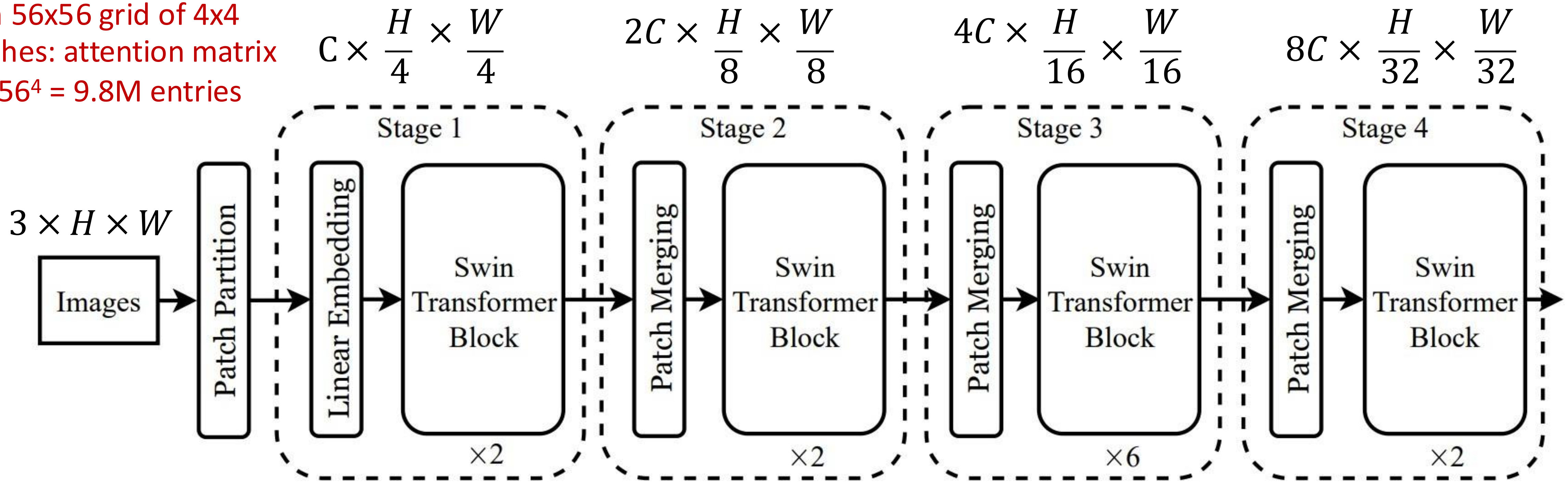
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

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

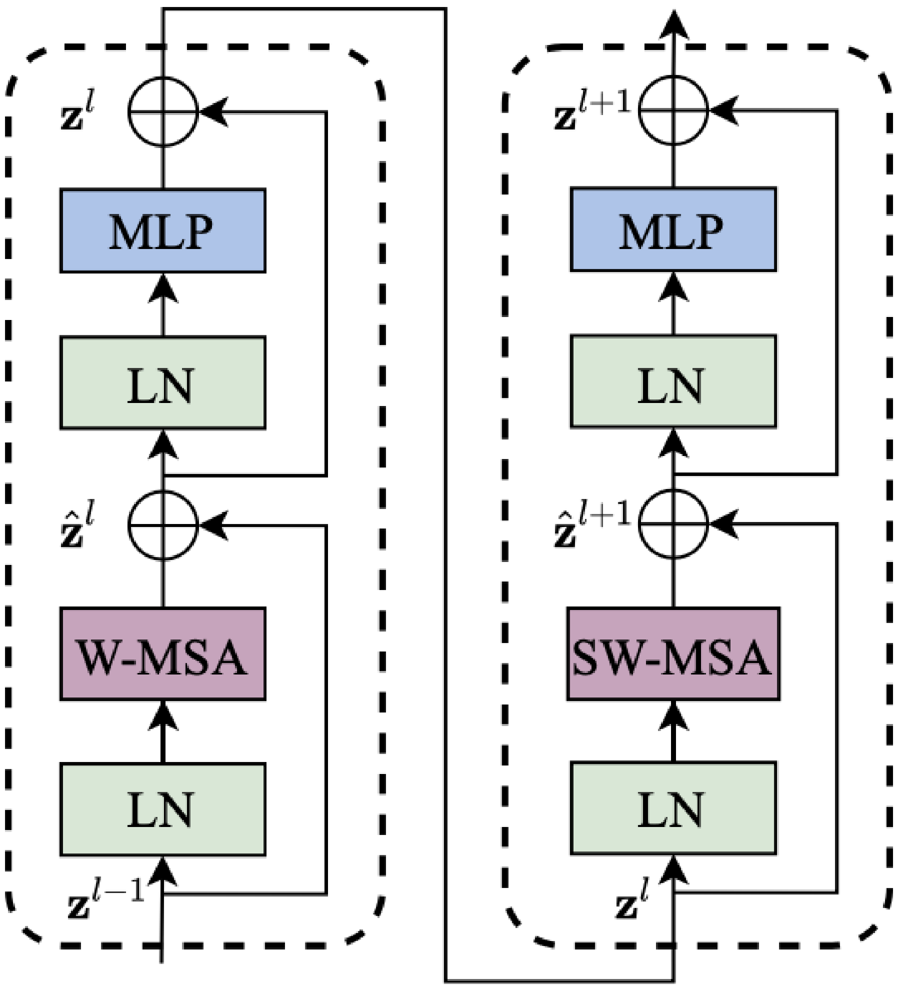
Divide image into 4x4 patches and project to C dimensions

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



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

$$\mathbf{z}^l = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^l\right)\right) + \hat{\mathbf{z}}^l,$$

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

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

Swin 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 (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

Do you see a potential problem?

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

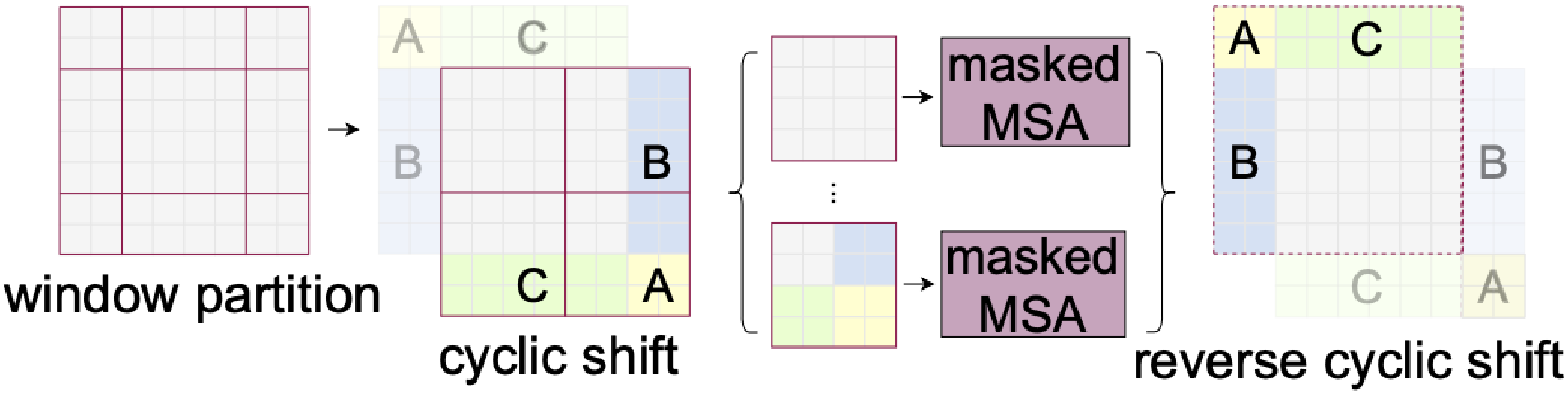
Solution: Alternate between normal windows and shifted windows in successive Transformer blocks

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


A
C
B
window partition
B
C
A
cyclic shift
masked
MSA
masked
MSA
A
C
B
B
C
A
reverse cyclic shift

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:

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

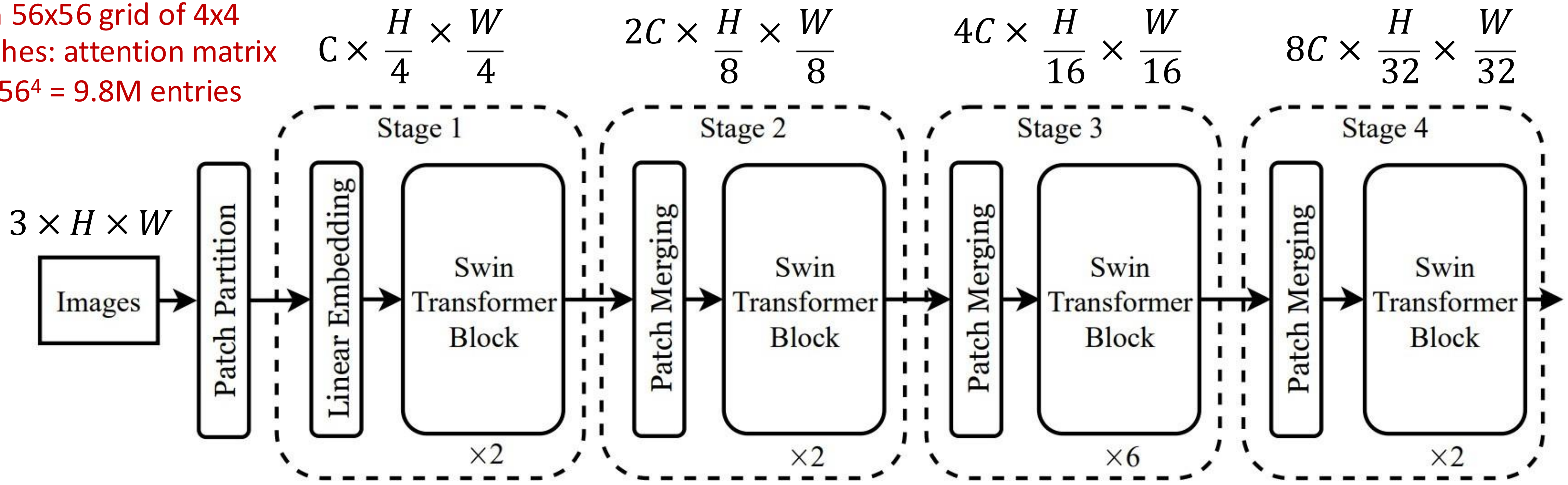
$Q, K, V: M^2 \times D$ (Query, Key, Value)
$B: M^2 \times M^2$ (learned biases)

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

For Swin-T, C = 96, final layer 224/32 x224/32 *8C
Apply GAP, 1x1x768

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

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	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 [34]	384^2	388M	204.6G	-	84.4
R-152x4 [34]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	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

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Swin-T architecture.

w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*;

abs. pos.: absolute position embedding term of ViT;

rel. pos.: the default set- tings with an additional relative position bias term;

app.: the first scaled dot-product term

when window is shifted, attention is applied without masking

How can we introduce global attention? It may have some benefit.
Say image is 224x224. There are 56x56 tokens.

Take 196 random tokens from stage 1: 196xC
Key K: Project it to 2C via a linear layer
Value V: Project it to 2C via another linear layer

Take query Q as output tokens from stage 2: 784x2C

Perform Softmax(QK^T/sqrt(2C))V : attention matrix 784x196 still relatively small

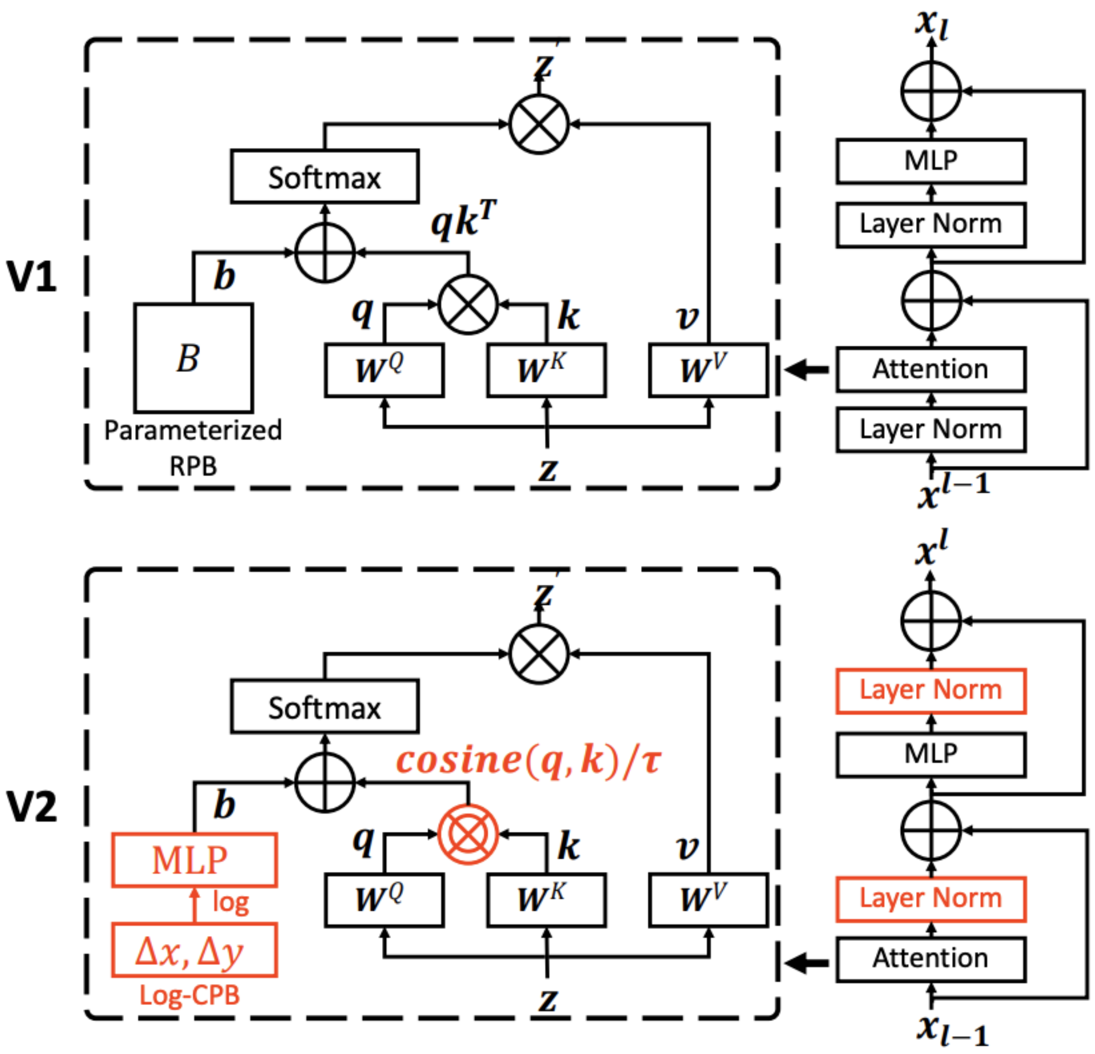
You can fuse both outputs after stage 2: max pool, add, concatenate to 4C and reproject to 2C

For the next, take 196/4 random tokens from stage 2: 49x2C
Key K: Project it to 4C via a linear layer
Value V: Project it to 4C via another linear layer

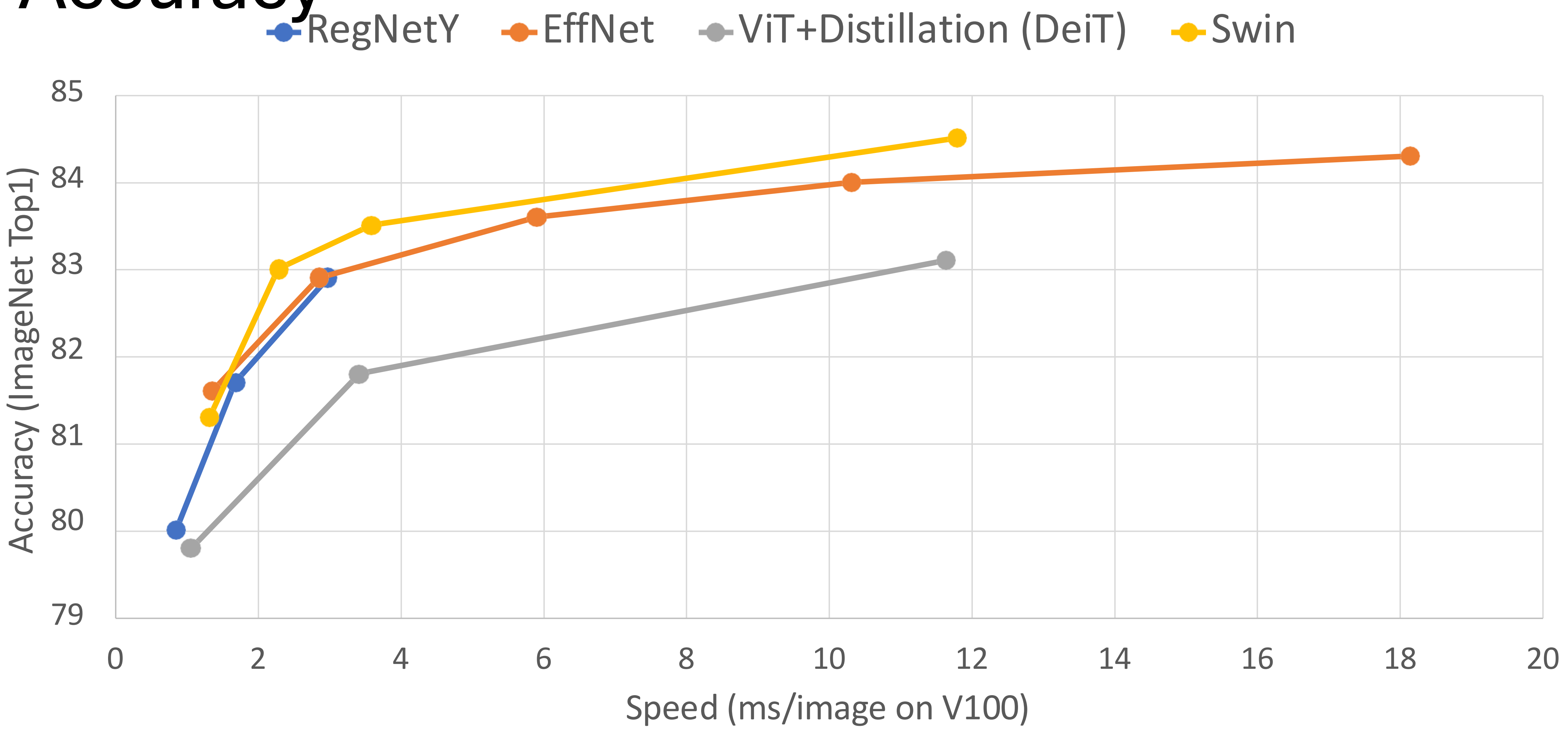
Take query Q as output tokens from stage 3: 196x2C

Repeat attention and pooling

Swin-v2, CVPR22

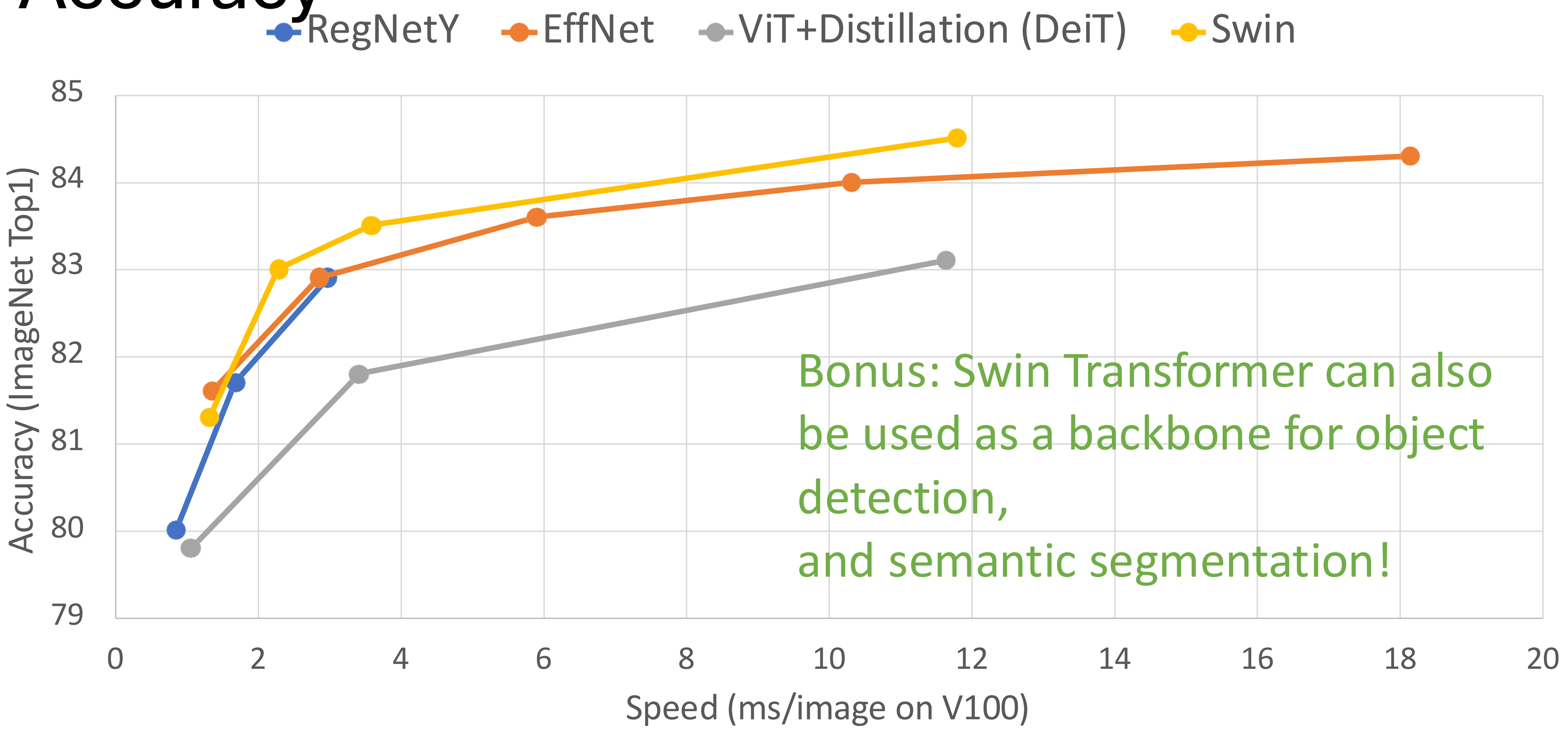


Swin Transformer: Speed vs Accuracy



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

Swin Transformer: Speed vs Accuracy

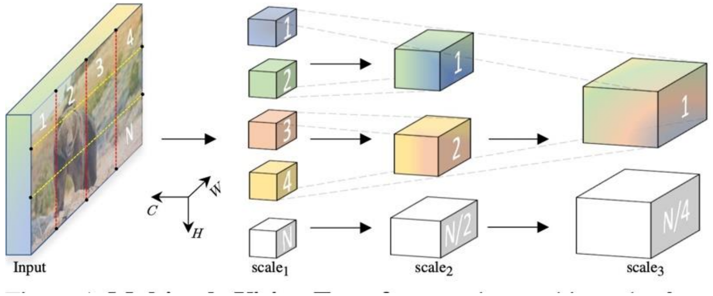


Bonus: Swin Transformer can also be used as a backbone for object detection,
and semantic segmentation!

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

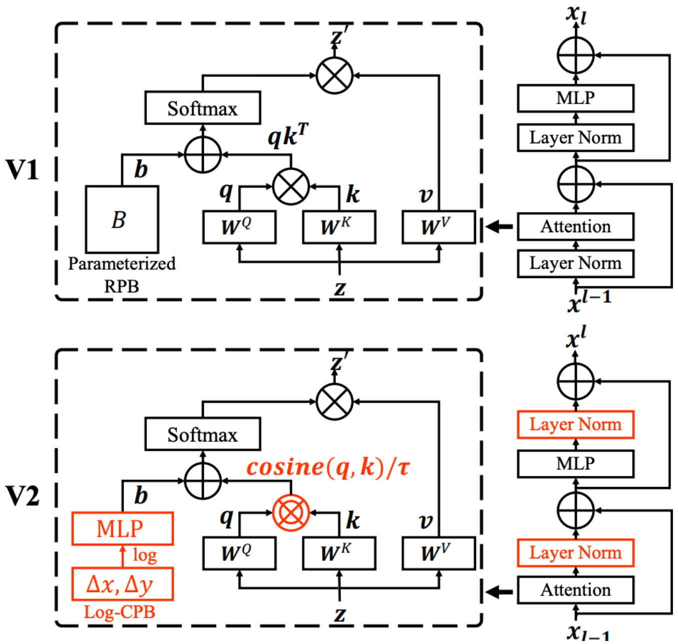
Other Hierarchical Vision Transformers

MViT



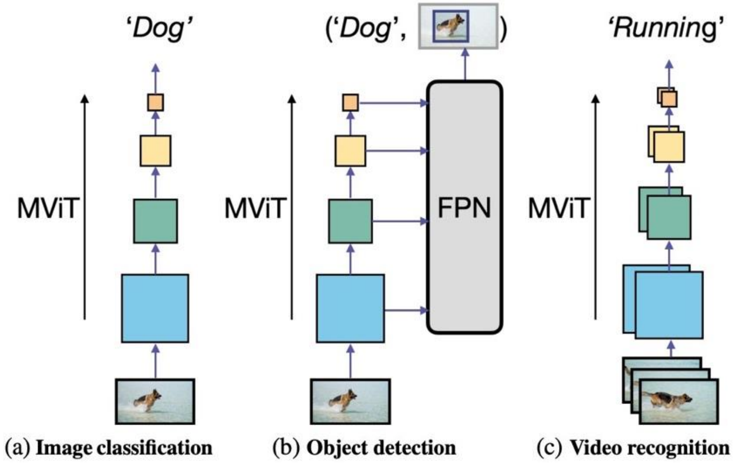
Fan et al, “Multiscale Vision Transformers”, ICCV 2021

Swin-V2



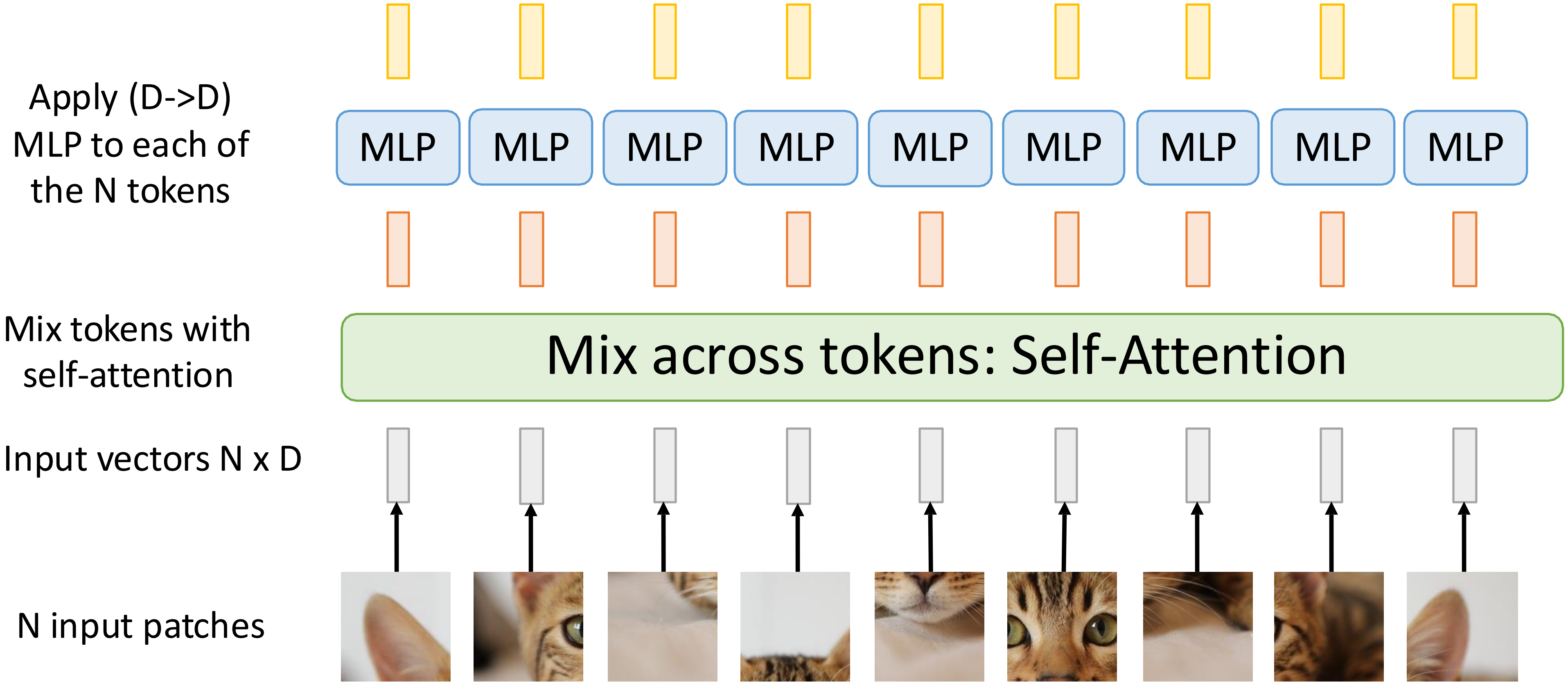
Liu et al, “Swin Transformer V2: Scaling up Capacity and Resolution”, CVPR 2022

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

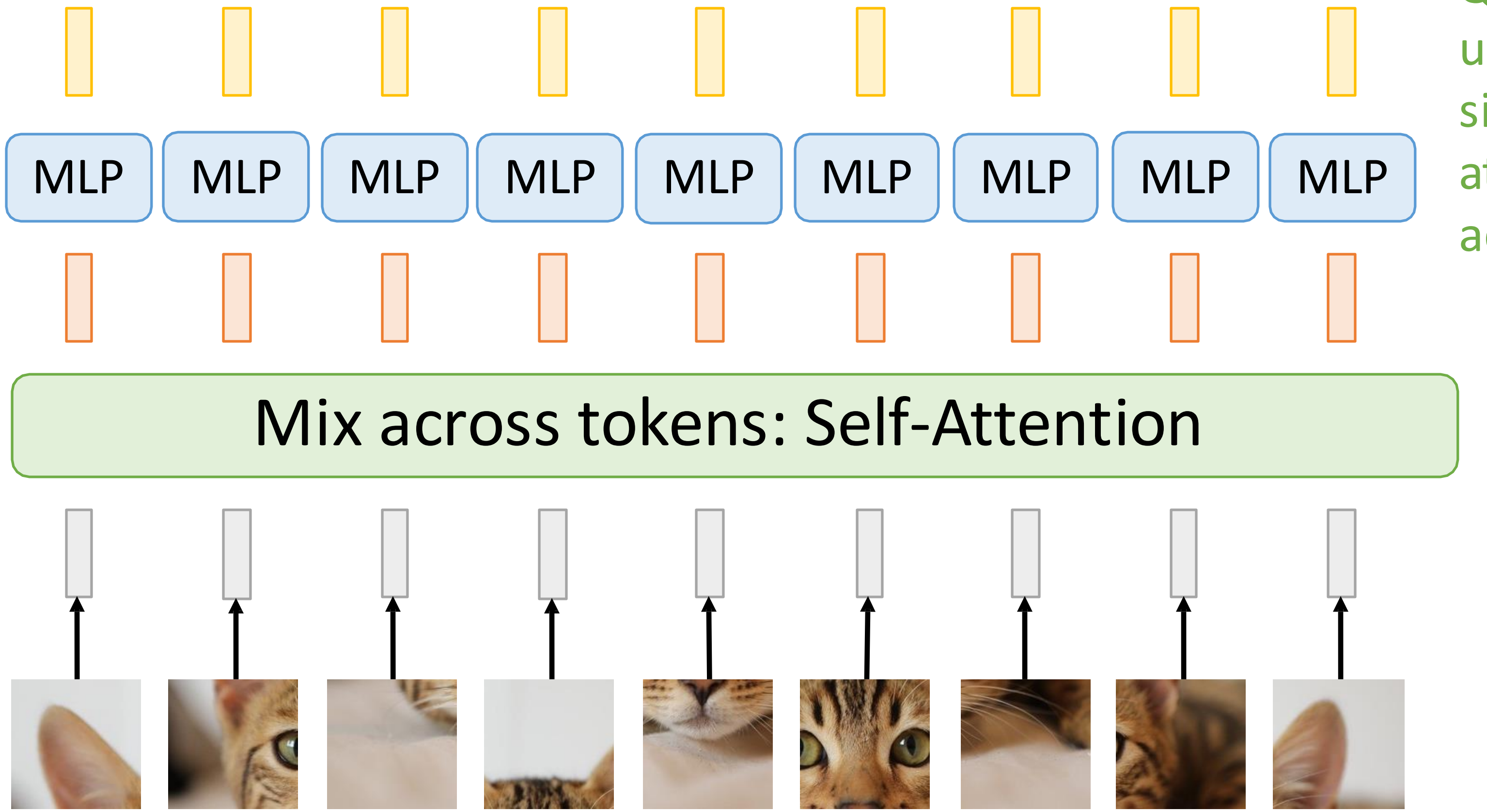
Vision Transformer: Another Look

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

Mix tokens with self-attention

Input vectors N x D

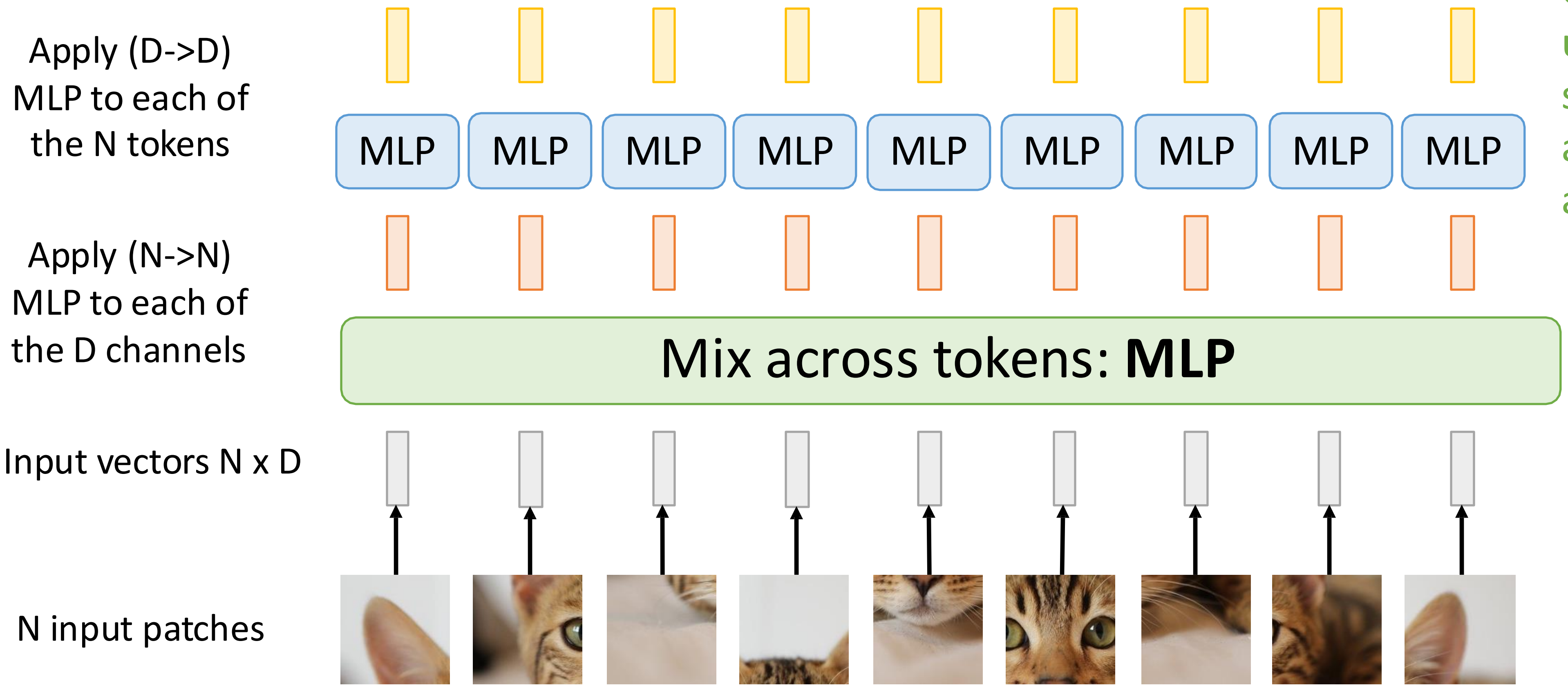
N input patches



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

MLP-Mixer: An All-MLP Architecture



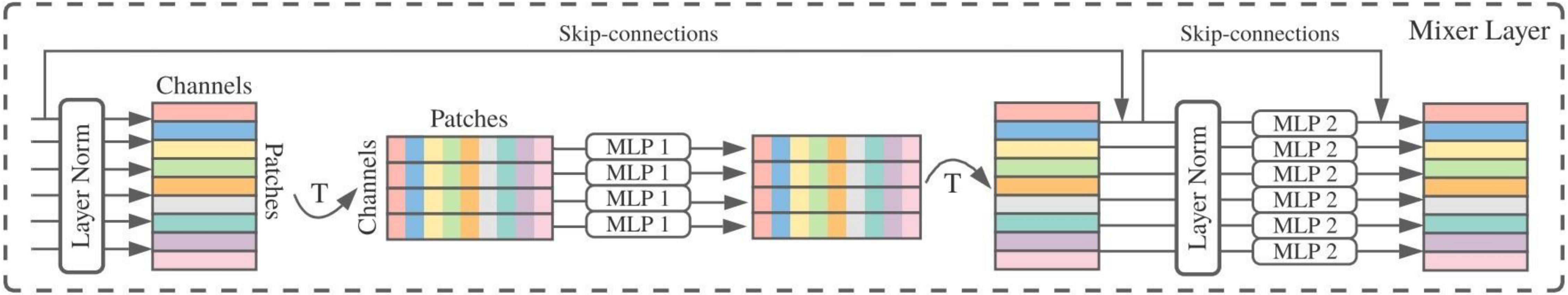
Question: Can we use something simpler than self-attention to mix across tokens?

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

MLP-Mixer: An All-MLP Architecture

Ds: token mixer

Dc: channel mixer



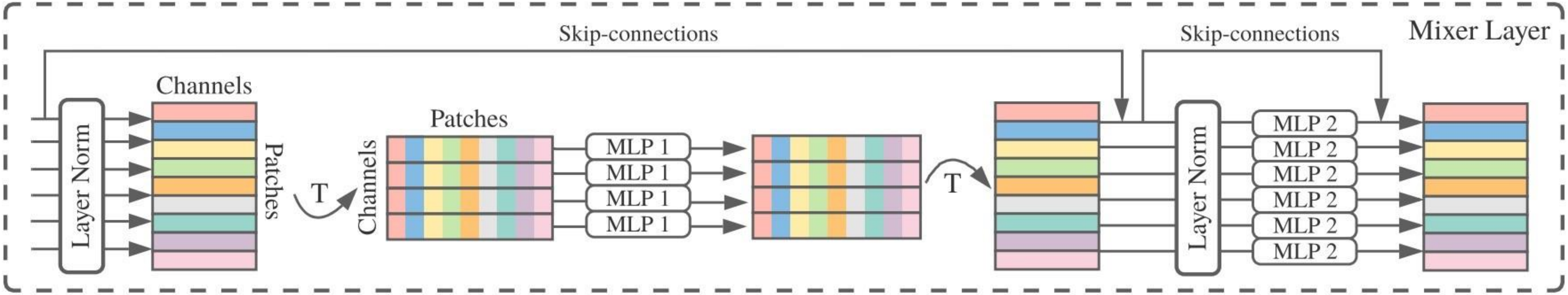
Input: N x C
N patches with
C channels each

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

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

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

MLP-Mixer: An All-MLP Architecture



Input: N x C
N patches with
C channels each

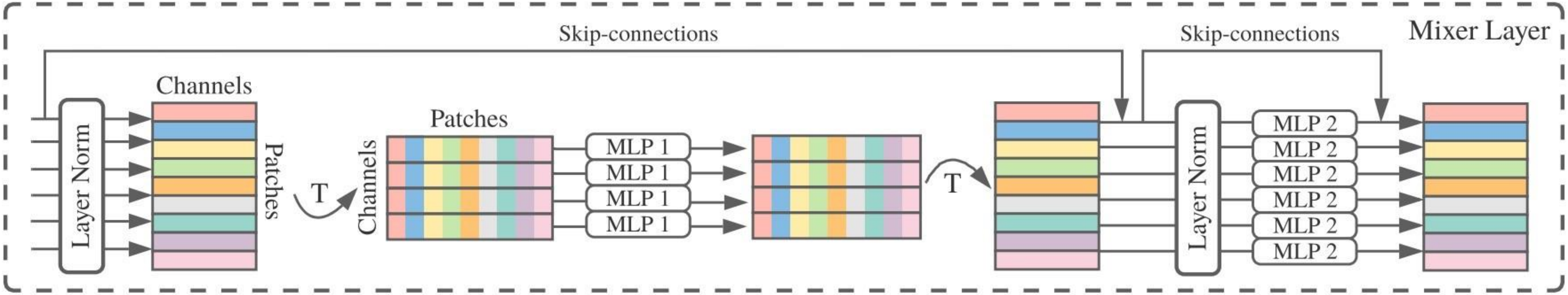
MLP 1: C -> C,
apply to each of
the **N patches**

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

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

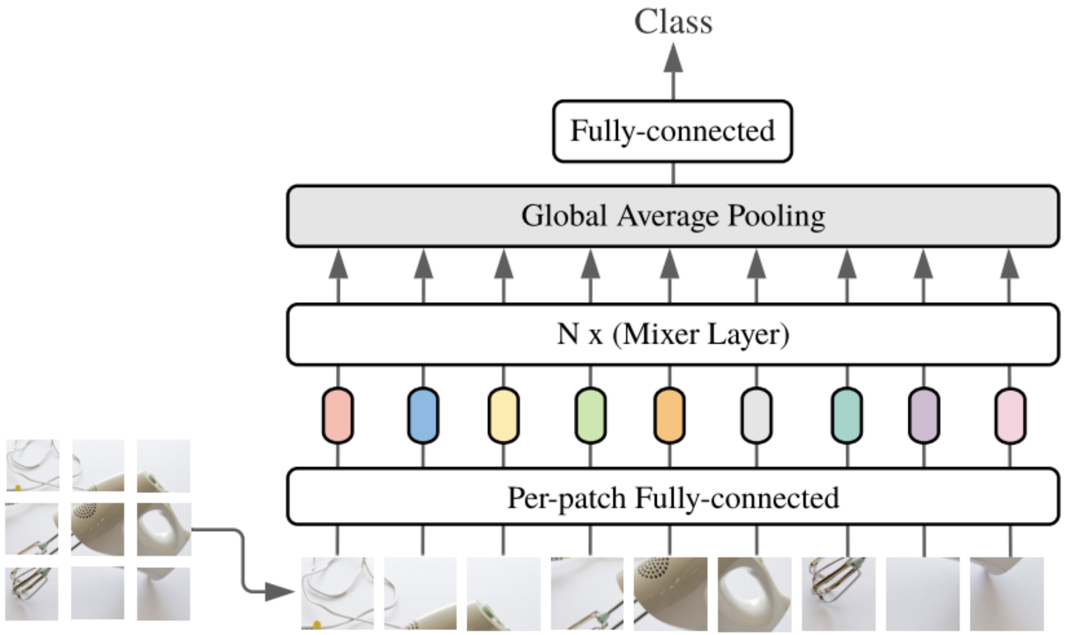


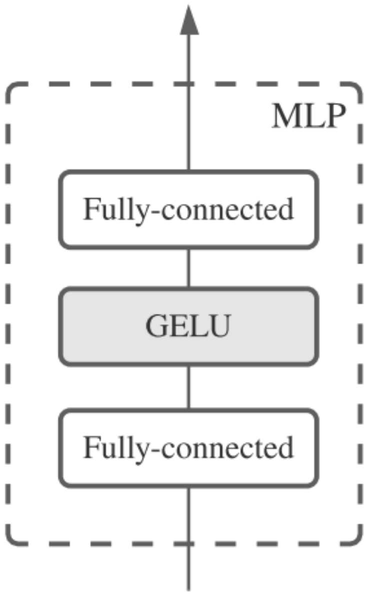
Input: N x C
N patches with
C channels each

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





Specification	S/32	S/16	B/32	B/16	L/32	L/16	H/14
Number of layers	8	8	12	12	24	24	32
Patch resolution $P \times P$	32×32	16×16	32×32	16×16	32×32	16×16	14×14
Hidden size C	512	512	768	768	1024	1024	1280
Sequence length S	49	196	49	196	49	196	256
MLP dimension D_C	2048	2048	3072	3072	4096	4096	5120
MLP dimension D_S	256	256	384	384	512	512	640
Parameters (M)	19	18	60	59	206	207	431

	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
Pre-trained on unlabelled or weakly labelled data (proprietary)						
● MPL [34]	90.0	91.12	—	—	—	20.48k
● ALIGN [21]	88.64	—	—	79.99	15	14.82k

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

Summary

Vision Transformers have been a super hot topic the past ~3 years!

Very different architecture vs traditional CNNs

Applications to all tasks: classification, detection, segmentation, etc

Main benefit is probably speed: Matrix multiply is more hardware-friendly than convolution, so ViTs with same FLOPs as CNNs can train and run much faster