Swin Transformers

Hierarchical Vision Transformer using Shifted Windows

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei
Zheng Zhang, Stephen Lin, Baining Guo

Microsoft Research Asia

Deep Learning and its applications to Signal and Image Processing and Analysis

Paper Presentation

Under the supervision of Prof. Tammy Riklin Raviv

Students:

Daniel Manor, Lior Yaacov

Agenda

	Introduction	Attention, Transformers and ViTs explained	1
	Challenges	Transformers challenges in vision tasks	2
	Method	Swin Transformer method explained	3
	Experiments	Validating Swin Transformer’s Performance	4
	Conclusions	Summarizing and Future Directions	5

Opening

Motivation

Leverage the advantages of transformer architecture to tackle a variety of computer vision tasks, addressing complex and non-trivial challenges effectively.

Problem Statement

Previous attempts to adapt transformers for vision tasks, like in ViT*, struggle with high computational costs and scalability issues, making them inefficient for handling high-resolution images and capturing both local and global features.

Solution

A hierarchical architecture with shifted windows efficiently captures both local and global features while significantly reducing computational costs for high-resolution vision tasks.

* AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, Alexey Dosovitskiy et al.

What are SWIN Transformers?

- SWIN (Shifted Window) Transformers are a type of Vision Transformer (ViT) designed for **computer vision tasks**.
- Developed by **Microsoft Research**.
- Addresses limitations of standard ViTs by introducing **hierarchical feature maps** and **shifted windows**.

CNNs: Foundations of Modern Vision Tasks

- **Early Approaches**: Handcrafted features and classical machine learning methods.
- **Rise of CNNs**: Revolutionized image processing with layers mimicking human visual perception.
- **Key Vision Tasks**:
  - Image Classification: ResNet, VGG..
  - Object Detection: YOLO, Faster R-CNN..
  - Semantic Segmentation: U-Net, DeepLab...
- **Advantages of CNNs:** Spatial hierarchies, local receptive fields, parameter efficiency.

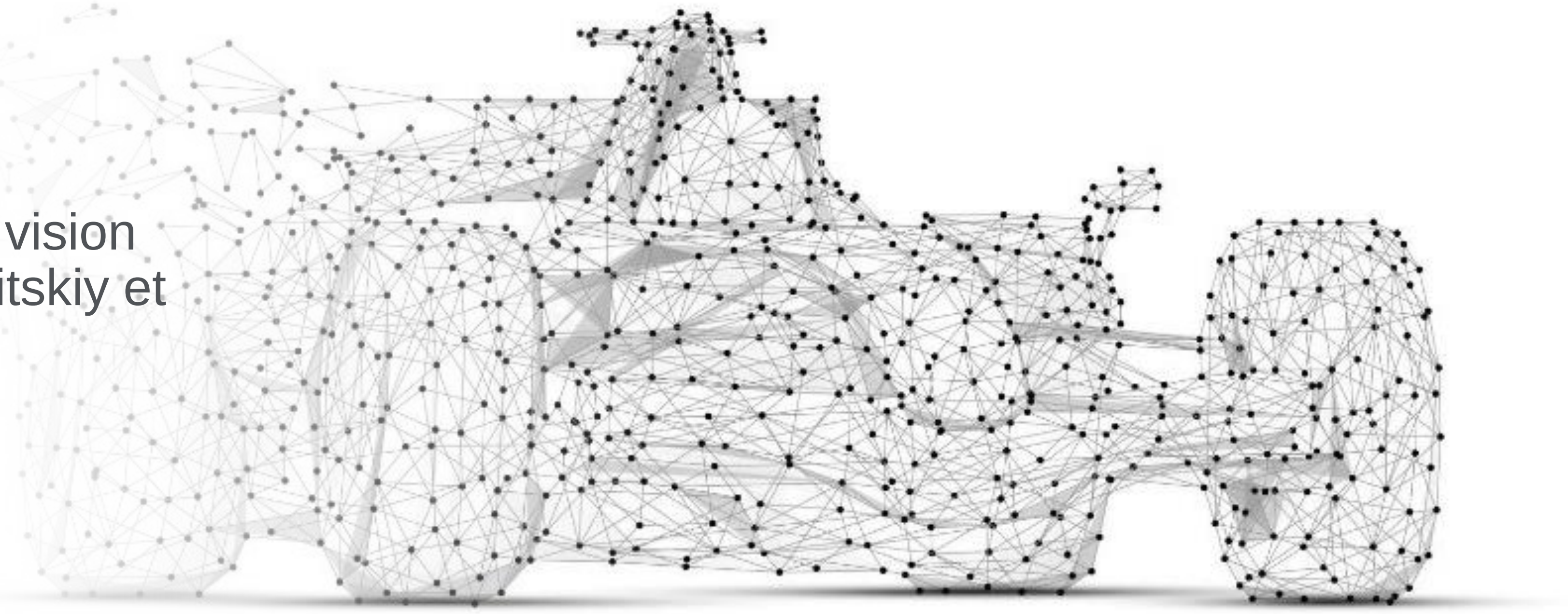
Transformers in Natural Language Processing

- Transformers introduced in "**Attention is All You Need**" paper by Google Brain (2017).
- **Key Features**:
  - Encoder-Decoder Architecture: Processes input to output sequences
  - Self-Attention: Each token attends to all other tokens in a sequence, capturing contextual relationships.
  - Multi-Head Attention: Multiple attention heads allow the model to focus on different parts of the sequence simultaneously.

Evolution from NLP to Vision Tasks

- **Inspiration from NLP Success**: Transformers' ability to handle long-range dependencies and parallelize training.

- **Initial attempts**: Directly applying Transformers to vision tasks (e.g., ViT by Dosovitskiy et al. 2020*).

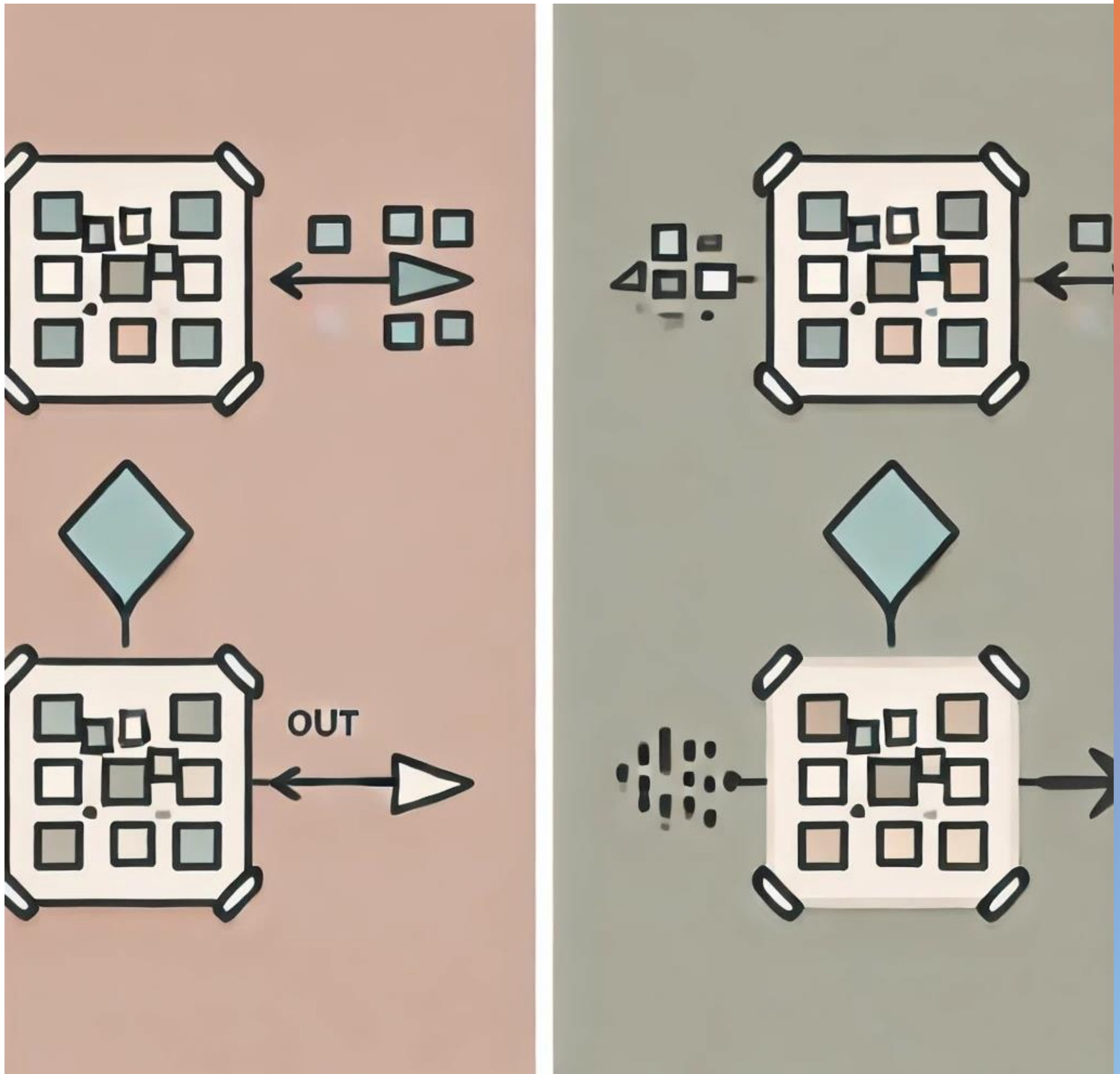


* AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, Alexey Dosovitskiy et al.

Vision Transformers (ViT)

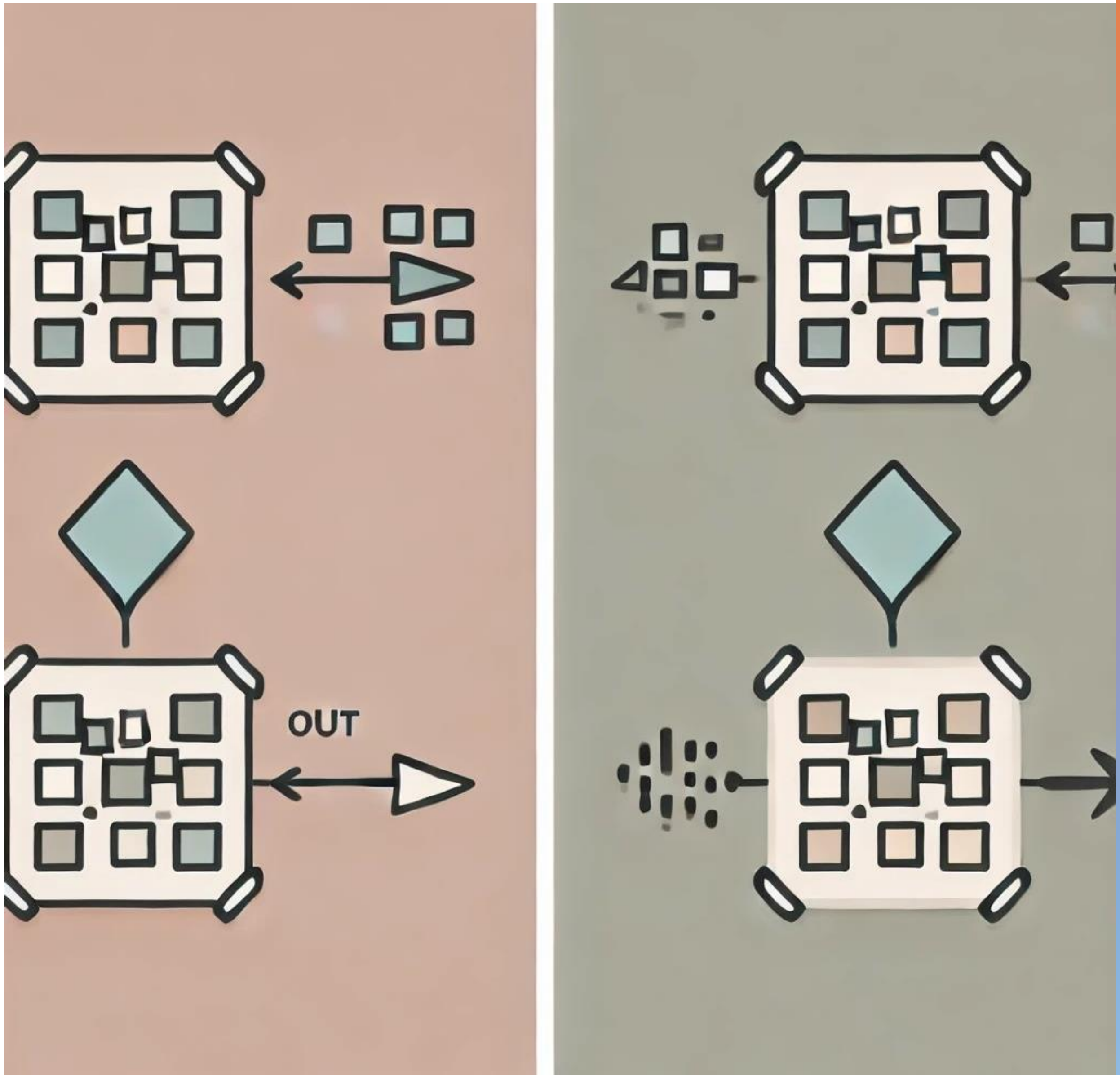
- **Concept**: Treats image patches as sequence tokens, similar to words in a text sequence.

- **How ViT Works**:
  - Image Patches
  - Embedding
  - Model
  - Output

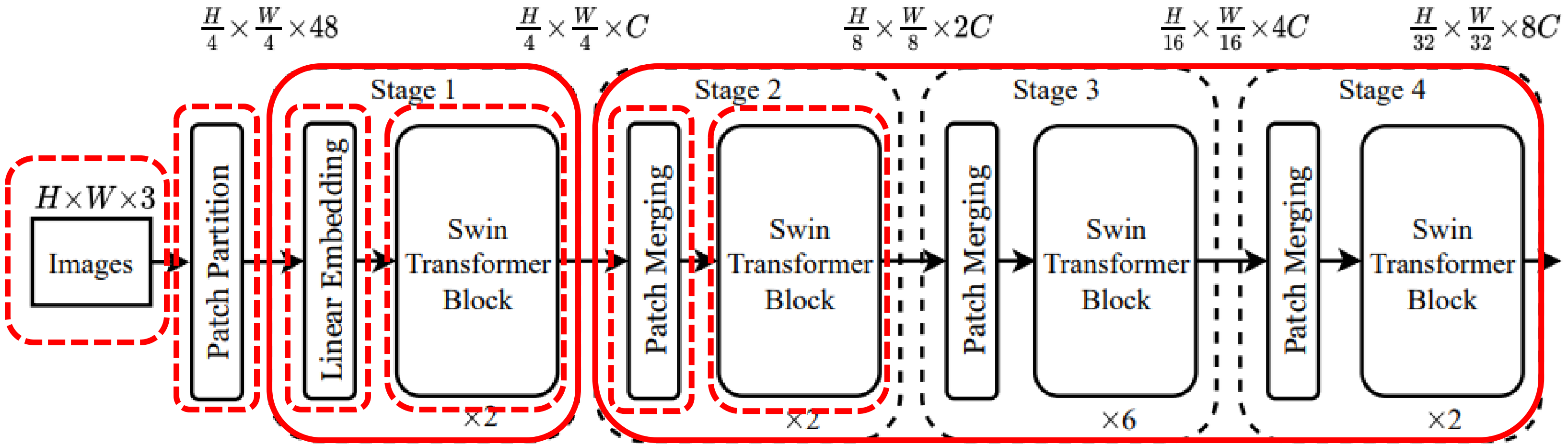


Vision Transformers (ViT)

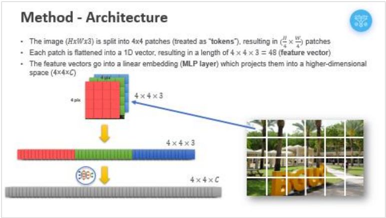
- Challenges of ViT:
  - High Computational Cost
  - Lack of Inductive Biases
  - Data Efficiency

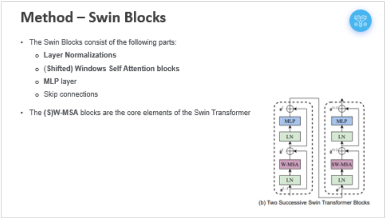


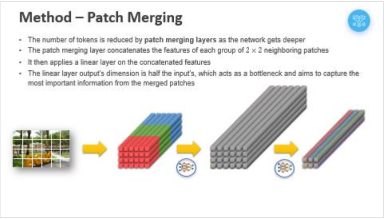
Method - Architecture

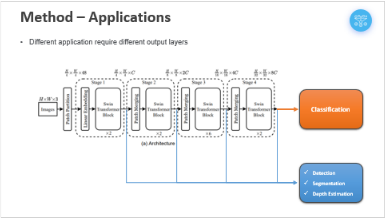


(a) Architecture





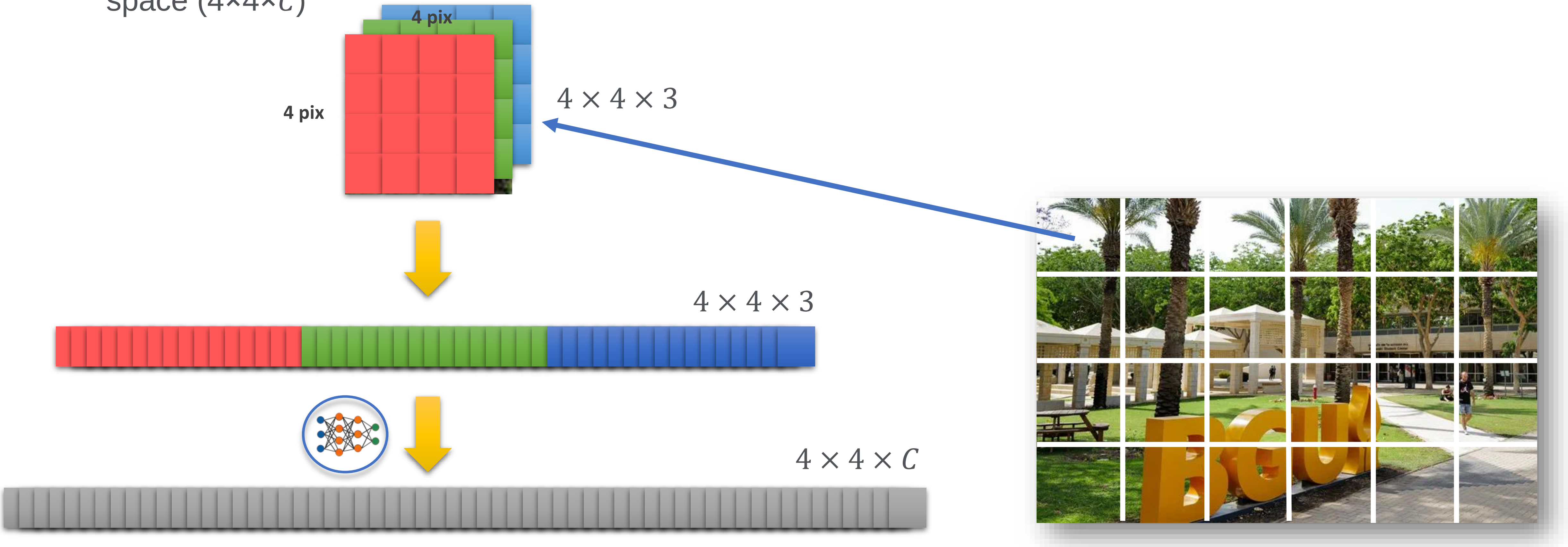




Model	C	Layer Numbers
Swin-T	96	2,2,6,2
Swin-S	96	2,2,18,2
Swin-B	128	2,2,18,2
Swin-L	192	2,2,18,2

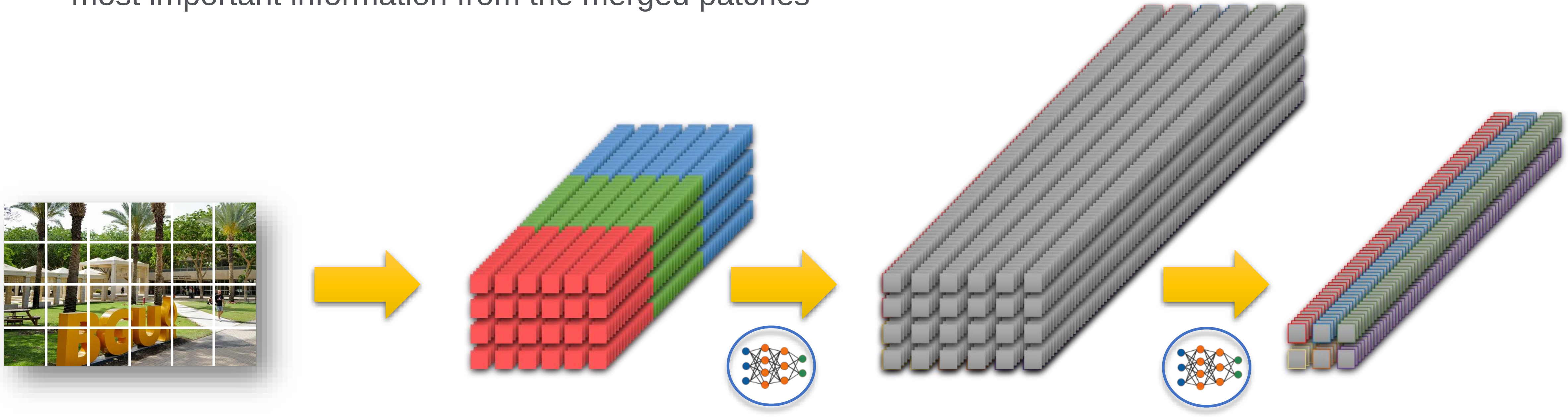
Method - Architecture

- The image ($HxWx3$) is split into 4x4 patches (treated as "**tokens**"), resulting in $(\frac{H}{4} \times \frac{W}{4})$ patches
- Each patch is flattened into a 1D vector, resulting in a length of $4 \times 4 \times 3 = 48$ (**feature vector**)
- The feature vectors go into a linear embedding (**MLP layer**) which projects them into a higher-dimensional space ($4{\times}4{\times}C$)



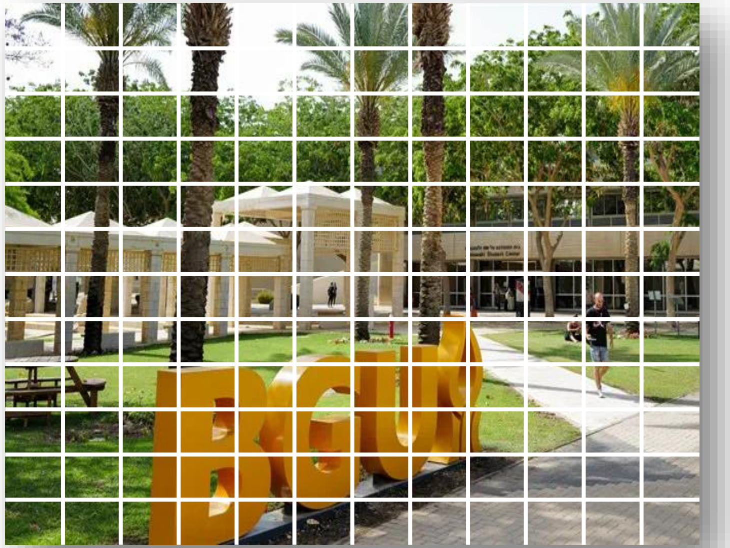
Method – Patch Merging

- The number of tokens is reduced by **patch merging layers** as the network gets deeper
- The patch merging layer concatenates the features of each group of 2×2 neighboring patches
- It then applies a linear layer on the concatenated features
- The linear layer output's dimension is half the input's, which acts as a bottleneck and aims to capture the most important information from the merged patches



Method – Patch Merging

$\frac{H}{4} \times \frac{W}{4} \times C$

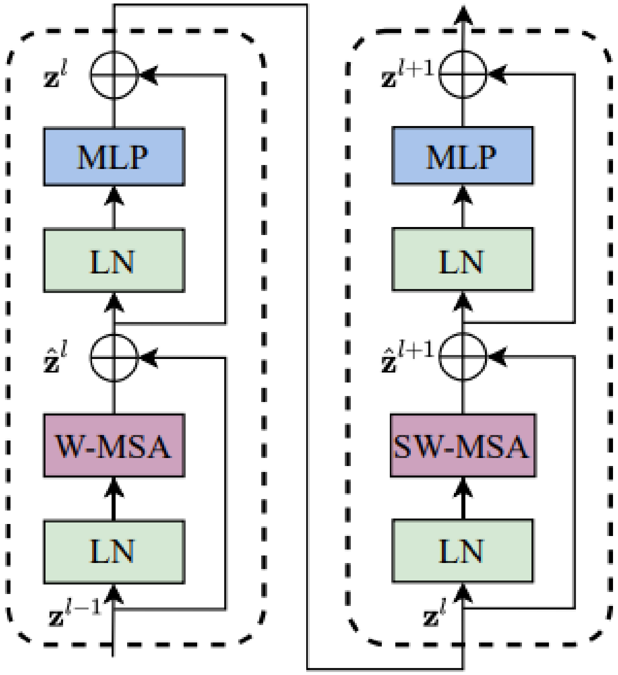


$\frac{H}{8} \times \frac{W}{8} \times 2C$

$\frac{H}{16} \times \frac{W}{16} \times 4C$

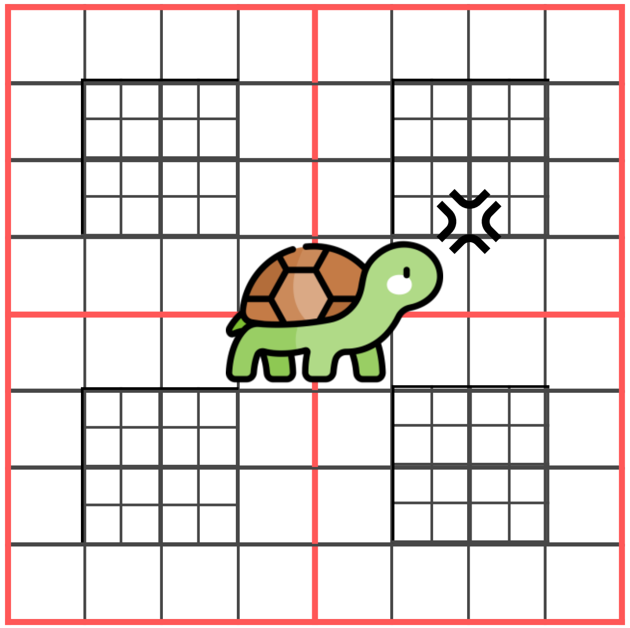
Method – Swin Blocks

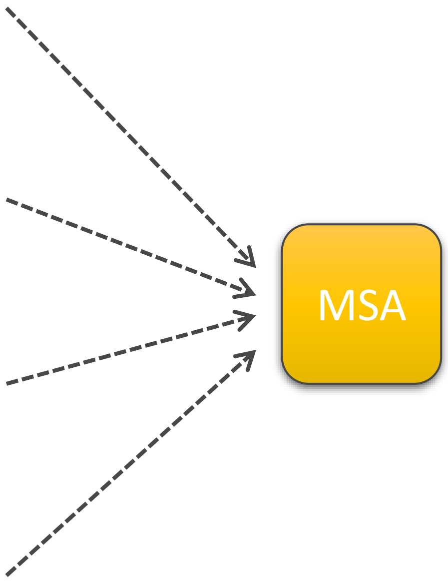
- The Swin Blocks consist of the following parts:
  - **Layer Normalizations**
  - **(Shifted) Windows Self Attention blocks**
  - **MLP** layer
  - Skip connections
- The **(S)W-MSA** blocks are the core elements of the Swin Transformer

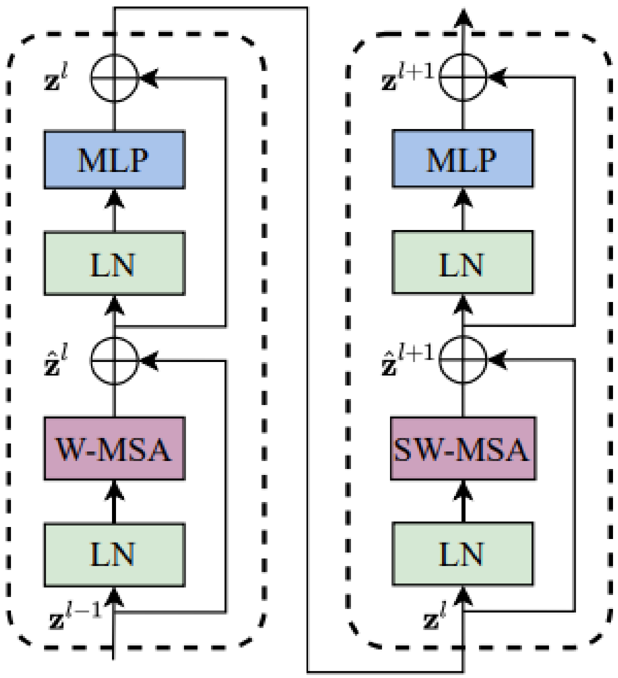


(b) Two Successive Swin Transformer Blocks

Method – Swin Blocks – (S)W-MSA

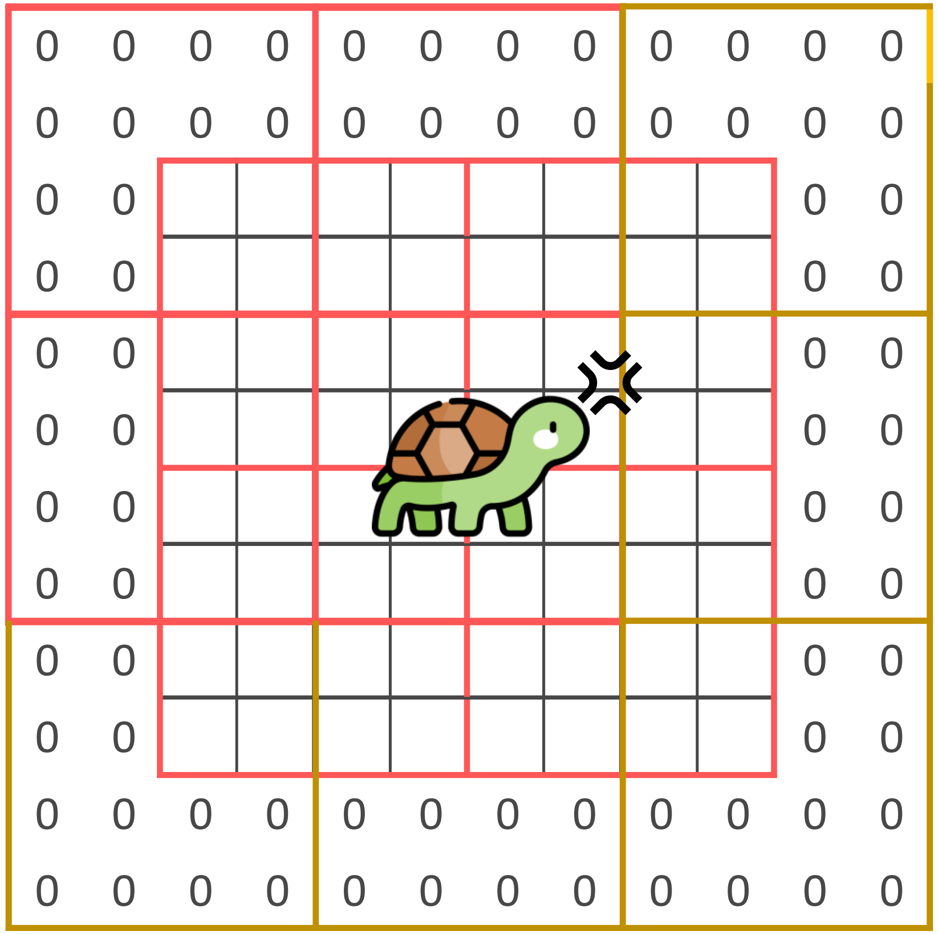




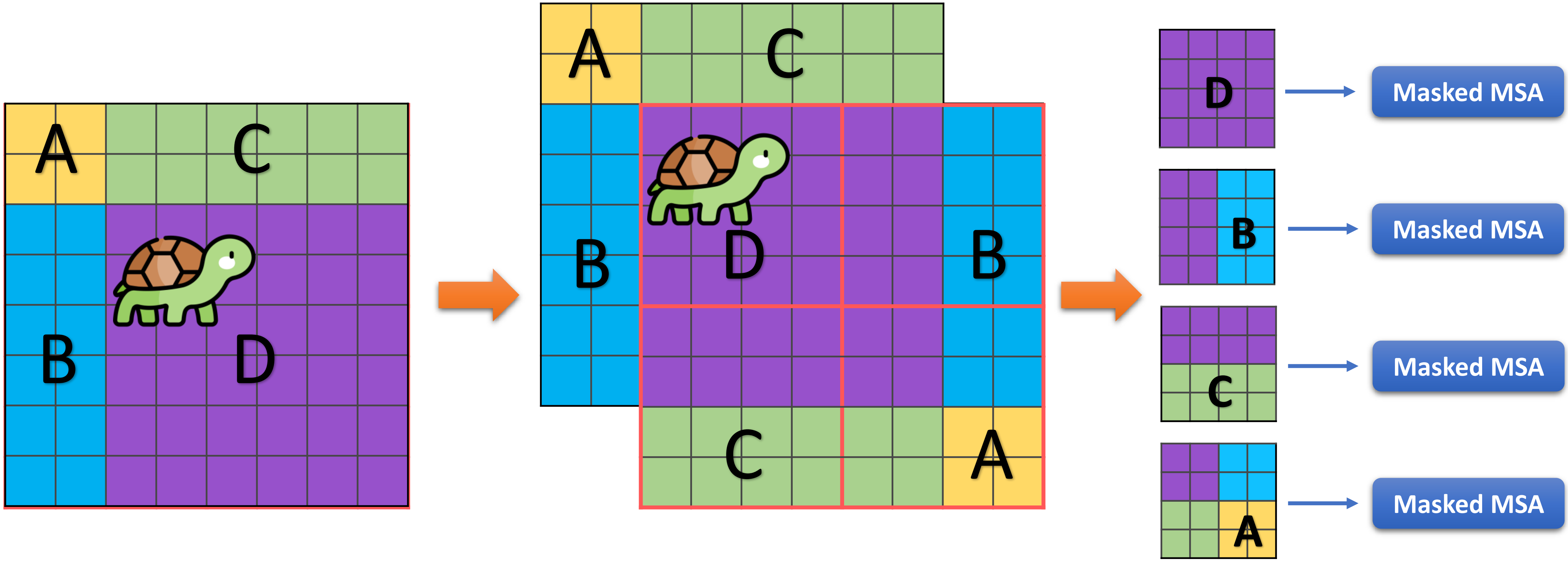


(b) Two Successive Swin Transformer Blocks

Method – Swin Blocks – (S)W-MSA



Method – Swin Blocks – (S)W-MSA



Experiments and Results

- **Objective**:
  - Validate the effectiveness of Swin Transformers across various computer vision tasks.
  - Understand the impact of key design elements of Swin Transformers.

- **Tasks Evaluated**:
  - Image Classification on ImageNet-1K and ImageNet-22K
  - Object Detection on COCO
  - Semantic Segmentation on ADE20K

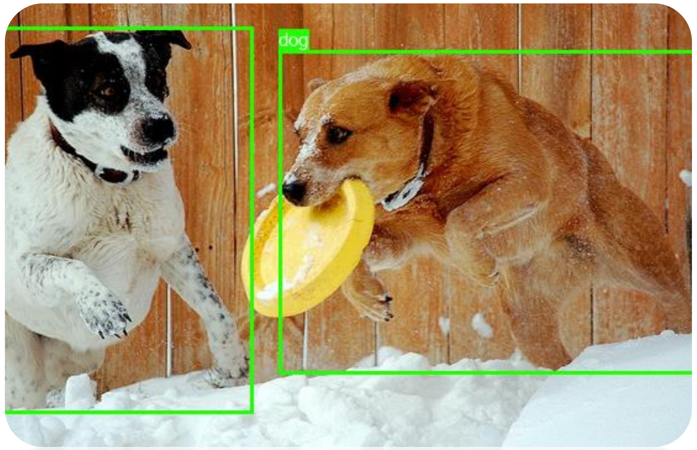
MiDaS Datasets

Dataset	Description	Vision Task	Train	Validation	Test	Link
ImageNet-1k	The ImageNet dataset is a large, diverse collection of annotated images used for image classification tasks	Classification	1.28M	50k	100k	Link
COCO	COCO is a large-scale object detection, segmentation, and captioning dataset	Object Detection	118k	5k	20k	Link
ADE20K	Comprehensive dataset for semantic segmentation and scene parsing tasks	Semantic Segmentation	20k	2k	3k	Link

ImageNet-1k

n01443537_**goldfish**.JPEG

COCO



ADE20k

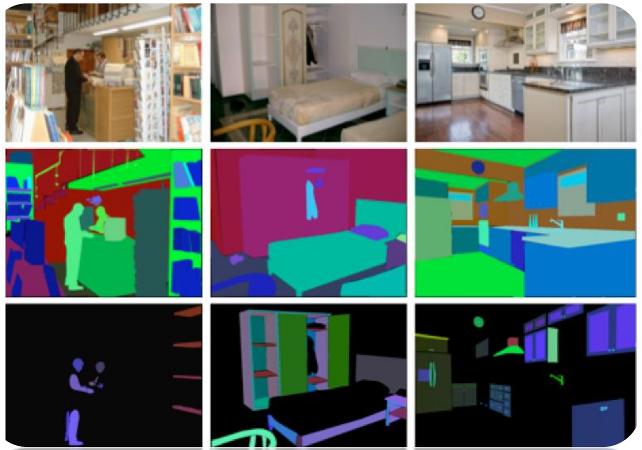


Image Classification on ImageNet-1K

- **Settings:**
  - **Training on ImageNet-1K:**
    - **Optimizer**: AdamW
    - **Epochs**: 300 with cosine decay learning rate
    - **Warm-up**: 20 epochs linear
    - **Batch size**: 1024
    - **Initial learning rate**: 0.001
    - **Weight decay**: 0.05

  - **Pre-training on ImageNet-22K and fine-tuning on ImageNet-1K:**
    - **Optimizer**: AdamW
    - **Pre-training**: 90 epochs, fine-tuning: 30 epochs
    - **Batch size**: Pre-training 4096, fine-tuning 1024
    - **Learning rate**: 0.001 for pre-training, 10^-5 for fine-tuning
    - **Weight decay**: 0.01 for pre-training, 10^-8 for fine-tuning

Image Classification on ImageNet-1K

- **Results:**
  - Comparisons with other backbones:
    - **Swin-T vs. DeiT-S:** +1.5 top-1 acc. (81.3 vs. 79.8 top-1 acc.) using 224x224 input
    - **Swin-B vs. DeiT-B:** +1.5 top-1 acc. /1.4 top-1 acc. (83.3/84.5 top-1 acc. vs. 81.8/83.1 top-1 acc.) using 224x224/384x384 input
    - **Swin-T vs. RegNetY-8G:** -0.4 top-1 acc. (81.3 vs. 81.7 top-1 acc.) using 224x224 input
    - **Swin-T vs. EffNet-B3:** -0.3 top-1 acc. (81.3 vs. 81.6 top-1 acc.) using 224x224 input
  - ImageNet-22K Pre-training Results:
    - **Swin-B:** 86.4 top-1 accuracy (+2.4 top-1 acc. higher than ViT-B/16 and +2 top-1 acc. higher than R-101x3)
    - **Swin-L:** 87.3 top-1 accuracy (+2.1 top-1 acc. better than ViT-L/16 and +2.9 top-1 acc. higher than R-101x3)

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5
(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

Object Detection on COCO

- **Settings:**
  - Models Evaluated:
    - **Faster R-CNN**
    - **Mask R-CNN**
    - **Cascade Mask R-CNN**

  - Training Configuration:
    - **Backbone**: Swin-T, Swin-B, Swin-L
    - **Optimizer**: AdamW
    - **Learning Rate Schedule**: Cosine decay
    - **Batch Size**: 16
    - **Epochs**: 36 (3x schedule)

Object Detection on COCO

- **Results:**
  - **Swin-T vs. ResNet-50**:
    - Achieves +3.4 to 4.2 box AP gains over ResNet-50.
  - **Comparison with Other Backbones**:
    - **ResNet-50 (R-50):**
      - Swin-T achieves +4.2 box AP over ResNet-50.
    - **DeiT-S:**
      - Swin-T achieves +2.5 box AP over DeiT-S.
    - **ResNeXt101-32:**
      - Swin-S shows +3.7 box AP over ResNeXt101-32.
    - **ResNeXt101-64:**
      - Swin-B shows +3.6 box AP over ResNeXt101-64.

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse	R-50	44.5	63.4	48.2	106M	166G	21.0
R-CNN	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

(c) System-level Comparison

Method	mini-val AP^{box}	mini-val AP^{mask}	test-dev AP^{box}	test-dev AP^{mask}	#param.	FLOPs
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

Semantic Segmentation on ADE20K

- **Settings:**
  - **Dataset:** ADE20K
  - **Base Framework:** UperNet

- **Results:**
  - **Swin-S**
    - Achieves 49.3 mIoU, which is +5.3 mIoU higher than DeiT-S (44.0) with similar computation cost.
    - Outperforms ResNet-101 by +4.4 mIoU.
    - Outperforms ResNeSt-101 by +2.4 mIoU.
  - **Swin-L:**
    - With ImageNet-22K pre-training, achieves 53.5 mIoU on the validation set.
    - Surpasses the previous best model (SETR) by +3.2 mIoU, which had 50.3 mIoU with a larger model size.

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. $\ddagger$ indicates that the model is pre-trained on ImageNet-22K.

Ablation Studies

- **Importance of Shifted Windows:**
  - **Comparison with single window partitioning shows:**
    - +1.1% top-1 accuracy on ImageNet-1K
    - +2.8 box AP and +2.2 mask AP on COCO
    - +2.8 mIoU on ADE20K

- **Effect of Relative Position Bias:**
  - **Improves performance across all tasks:**
    - +1.2%/+0.8% top-1 accuracy on ImageNet-1K
    - +1.3/+1.5 box AP and +1.1/+1.3 mask AP on COCO
    - +2.3/+2.9 mIoU on ADE20K

	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**

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the 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 settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

Conclusions

- **Summary**:
  - Introduced a hierarchical vision Transformer using shifted windows.
  - Addressed limitations of standard ViTs by incorporating hierarchical feature maps and shifted windows.
  - Achieves **linear** computational complexity with respect to input image size.
  - Outperforms previous methods in COCO object detection and ADE20K semantic segmentation.

- **Key Innovations**:
  - **Shifted Window Mechanism:** Effective and efficient for vision tasks.
  - **Hierarchical Feature Maps:** Enables efficient computation and better handling of high-resolution images.
  - **Relative Position Bias:** Enhances spatial understanding and performance.

Future Directions

Optimization: Further research into more efficient training methods and architectures.

Applications: Explore the use of shifted window-based self-attention in NLP.

Integration: Combining Swin Transformers with other models and techniques for enhanced performance.

End

Thanks for listening!