Efficient Vision Transformers for Object Recognition

Xuan-Thuy Vo
xthuy@islab.ulsan.ac.kr

November 17, 2023

Publications

- **Journals:**
  - **Xuan-Thuy Vo** and Kang-Hyun Jo, *Accurate Bounding Box Prediction for Single-Shot Object Detection*, IEEE Transactions on Industrial Informatics, Vol.18, No.9, pp.5961-5971, 12 2021. **(IF 12.3)**
  - **Xuan-Thuy Vo** and Kang-Hyun Jo, *A Review on Anchor Assignment and Sampling Heuristics in Deep Learning-based Object Detection*, Neurocomputing, Vol.506, pp. 96-116, 7 2022. **(IF 6.0)**
  - Van-Dung Hoang, **Xuan-Thuy Vo** and Kang-Hyun Jo, *Categorical Weighting Domination for Imbalanced Classification With Skin Cancer in Intelligent Healthcare Systems*, IEEE Access, Vol.11, pp. 105170 - 105181, 9 2023. **(IF 3.9)**

- **Conferences:**
  - 34 papers (3 Awards)

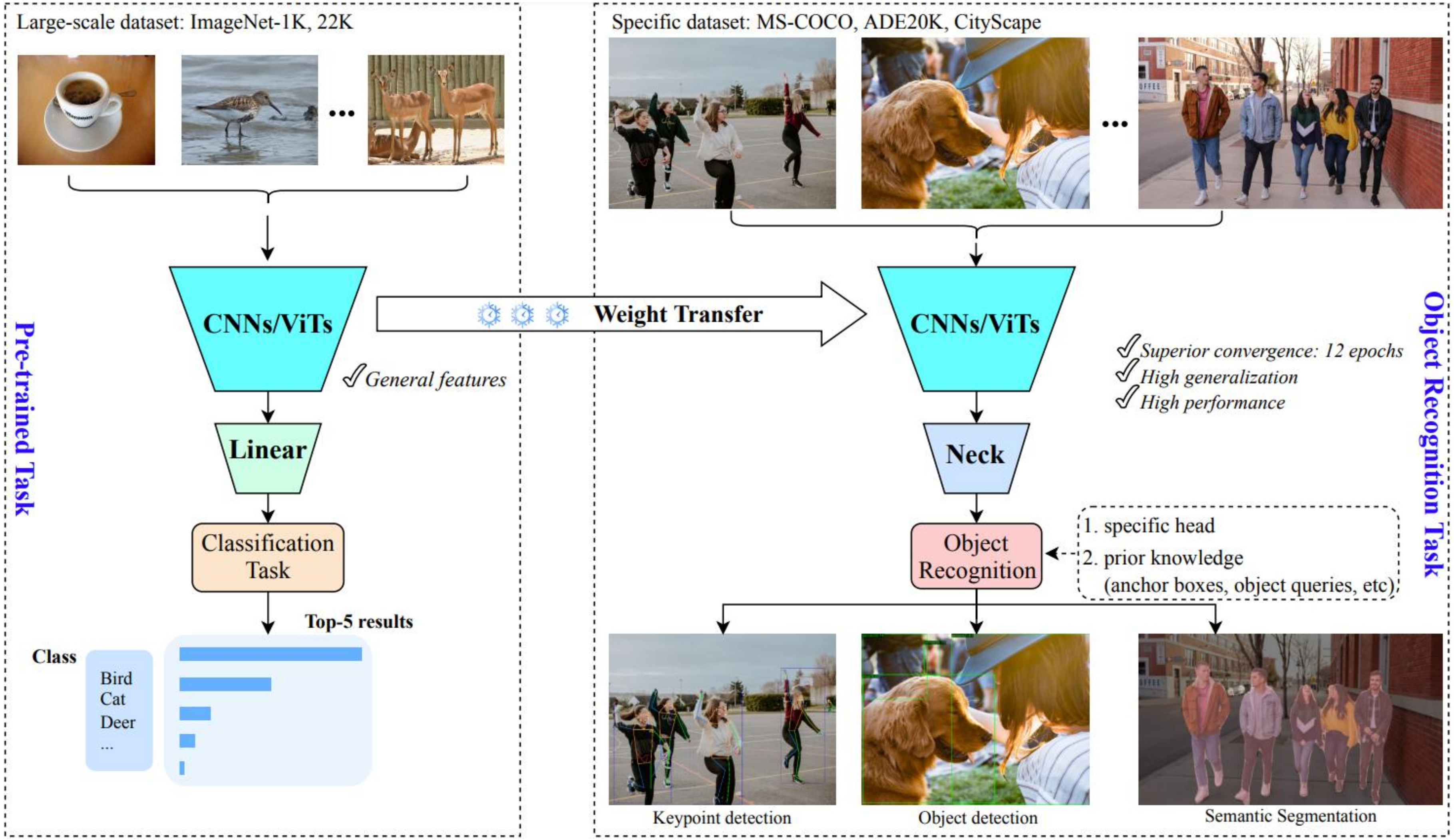
Publications

- **Under-Review Works:**
  - **Xuan-Thuy Vo**, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, *Exchange Information across Non-overlapped Local Self-Attentions via Mixing Abstract Tokens*, AAAI Conference on Artificial Intelligence, (*Second Round*) (**h5-index: 212**)
  - **Xuan-Thuy Vo**, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Efficient Vision Transformers with Partial Attention, IEEE/CVF Computer Vision and Pattern Recognition Conference-CVPR, (*First Round*) (**h5-index: 422**)
  - **Xuan-Thuy Vo**, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, *Efficient Multi-scale Spatial Interactions for Object Recognition*, IEEE Transactions on Industrial Informatics, (*To Submit*) (**IF: 12.3, h5-index: 162**)

Contents

Introduction

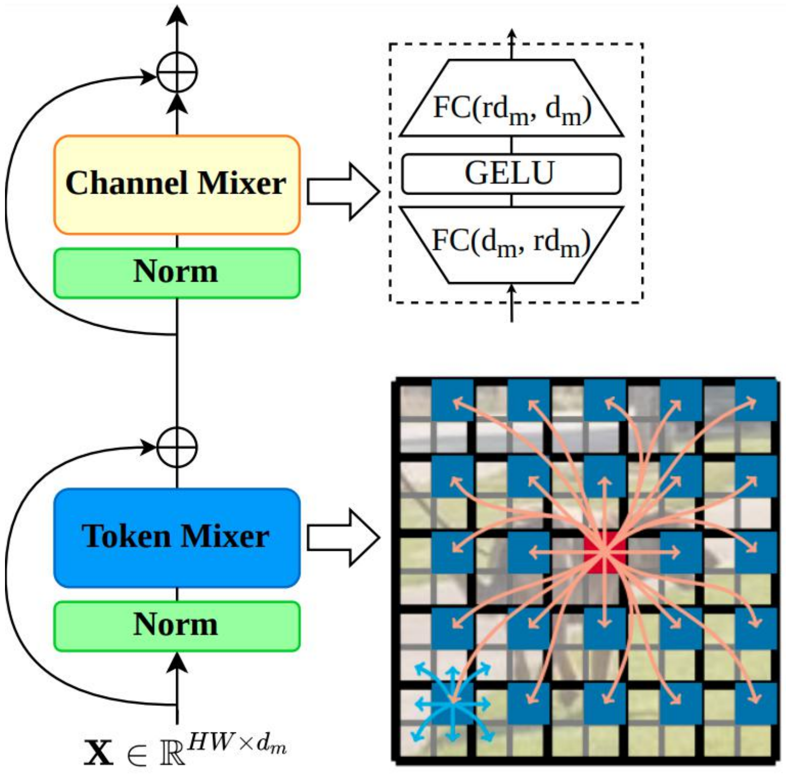
General Pipeline of Object Recognition:



CNNs: Convolutional Neural Networks; ViTs: Vision Transformers

Introduction

▶ Modern networks: focus on the improvements of the token mixer

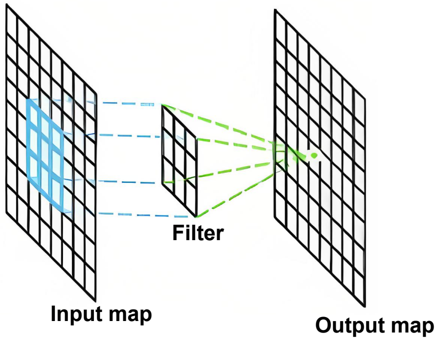


2×2 patch ~ 1 token
Red point ~ a query
Blue point ~ keys/values

1) Convolution:

$$\mathbf{Y}_i \leftarrow Conv(\mathbf{X}_i, \mathbf{M}_i^{conv})$$

$$\mathbf{M}_i^{conv} \in \mathbb{R}^{k \times k}$$

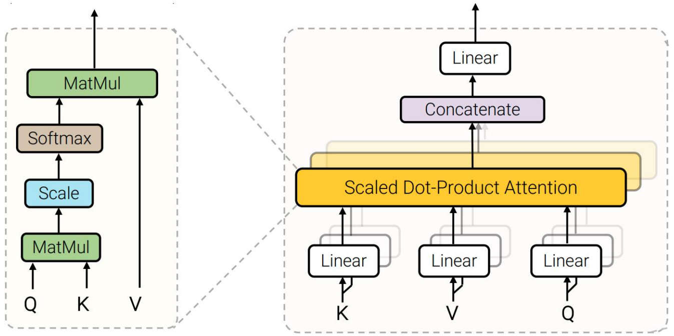


2) Self-attention operation:

$$\mathbf{M}^{att} = \delta\left(\frac{\mathbf{X}\mathbf{W}^Q(\mathbf{X}\mathbf{W}^K)^T}{\sqrt{d_m}}\right)$$

$$\mathbf{Y} \leftarrow \mathbf{M}^{att}(\mathbf{X}, \mathbf{W}^V)$$

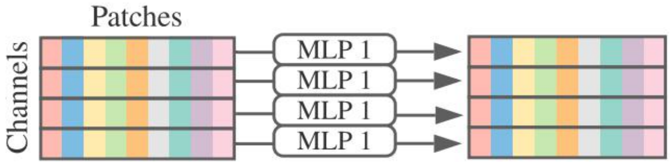
$$\mathbf{M}_i^{att} \in \mathbb{R}^{HW \times HW}, \mathbf{W} \in \mathbb{R}^{d_m \times d_m}$$



3) MLP operation:

$$\mathbf{Y} \leftarrow \mathbf{M}^{mlp}\mathbf{X}$$

$$\mathbf{M}_i^{mlp} \in \mathbb{R}^{HW \times HW}$$



Comparisons of Token Mixers:

Token Mixers	Computational Cost *O*()	#Params	Properties		
			Input dependent weight	Global receptive field	Relative positions
Depthwise Convolution	k^2HWd_m *linear*	k^2d_m	✗	✗	✓
Self-Attention	$H^2W^2d_m + HWd^2_m$ ***quadratic***	$4d_m^2$	✓	✓	✗
Spatial MLP	$H^2W^2d_m$ ***quadratic***	H^2W^2	✗	✓	✗
Window Self-Attention	$HWw^2d_m + HWd^2_m$ ***linear***	$4d_m^2$	✓	✗	✓

2D input

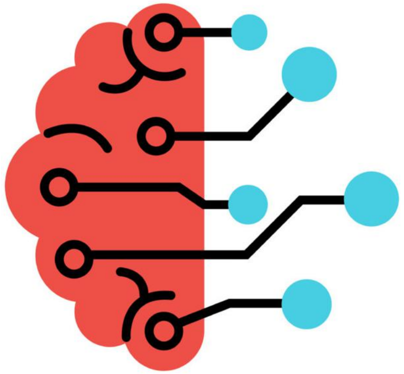
Flatten input

High resolution
→ Huge costs

w: window size, k: kernel size
H, W, d_m: Height, Width, and #channels

Unified all properties → better performance

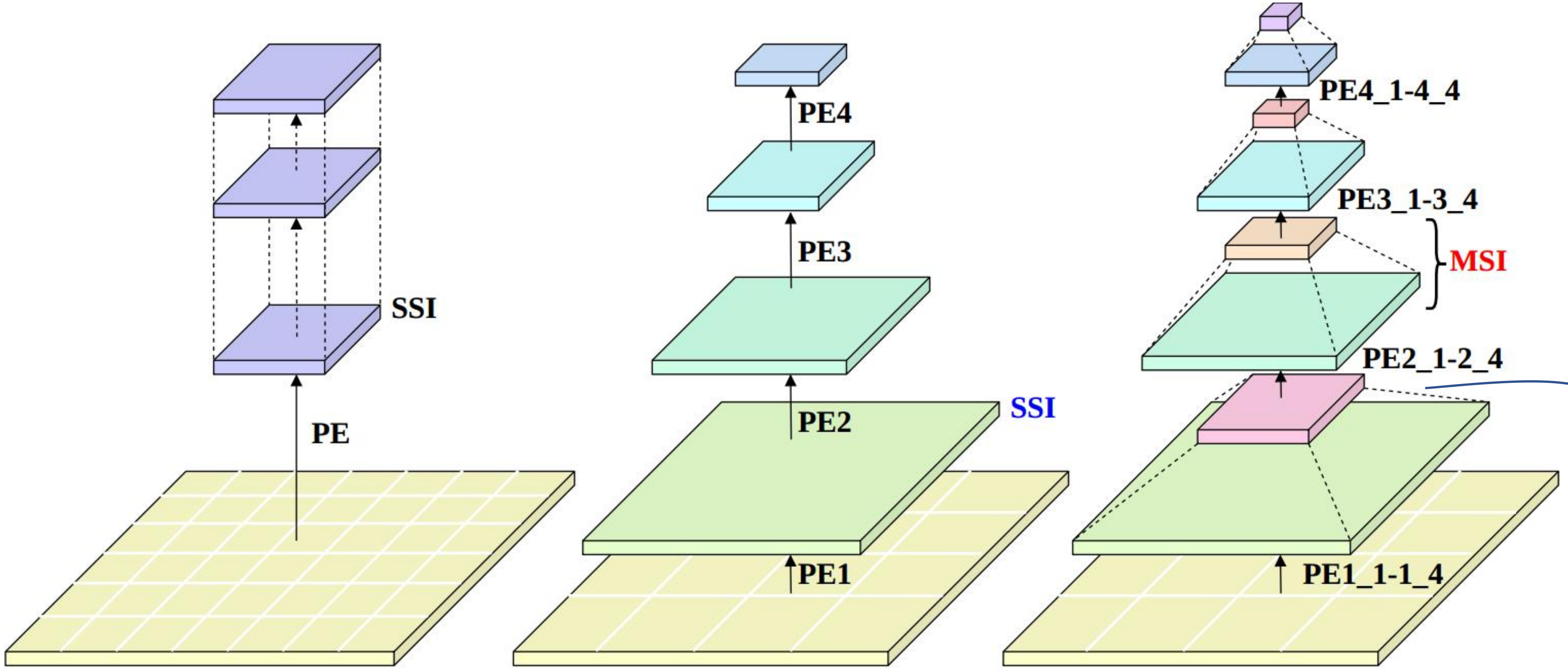
Efficient Multi-Scale Spatial Interactions (EMSNet)



- Motivation
- Proposed EMSNet
- Experimental Results

Motivation

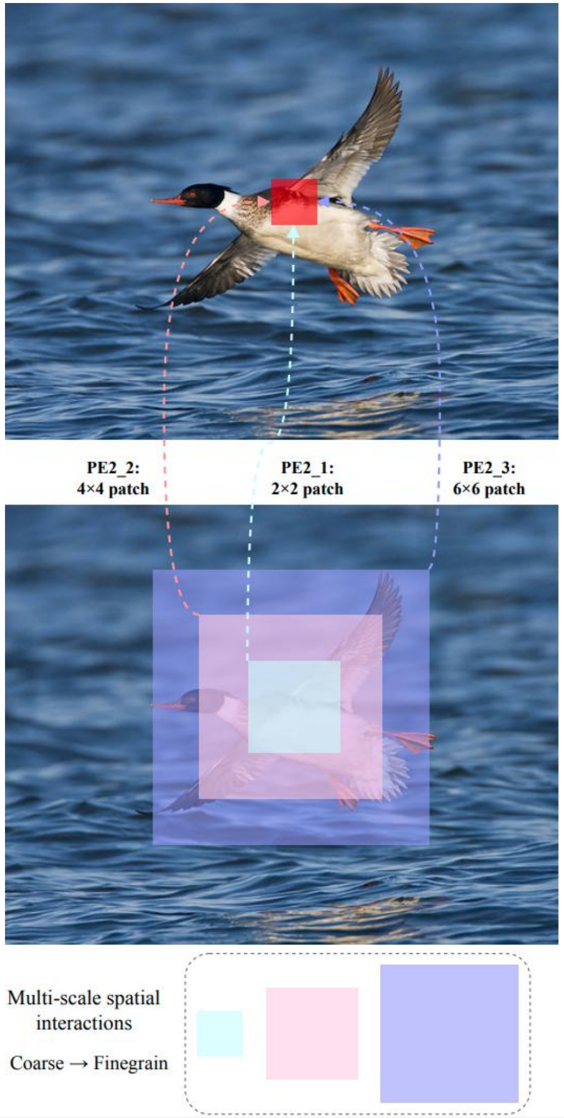
▶ Single-scale Spatial Interaction (SSI) *versus* Multi-scale Spatial Interactions (MSI):



(a) Columnar model with SSI
- Vision Transformer

(b) Pyramid model with SSI
- ResNet, ConvNext, PvT, Swin

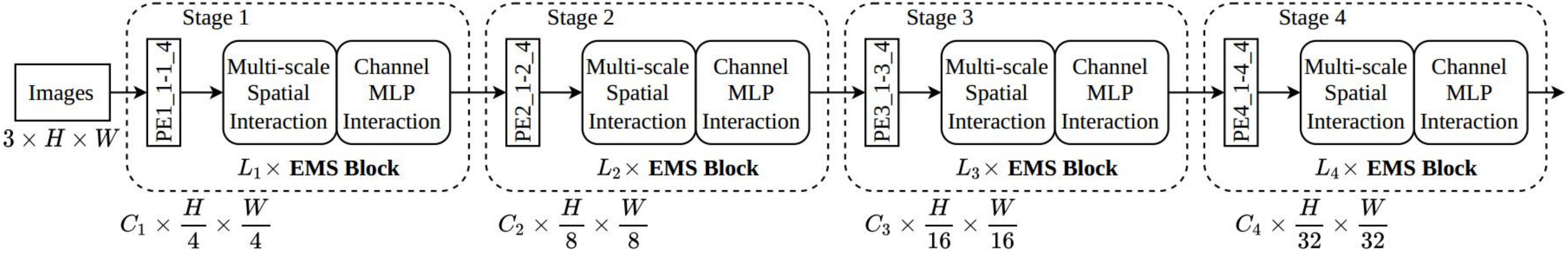
(c) Pyramid model with MSI
- **EMS (Ours)**



→ Extend feature learning from single-scale to multi-scale interactions
→ Capture multiple views of the input
PE: Patch Embedding

Proposed Networks

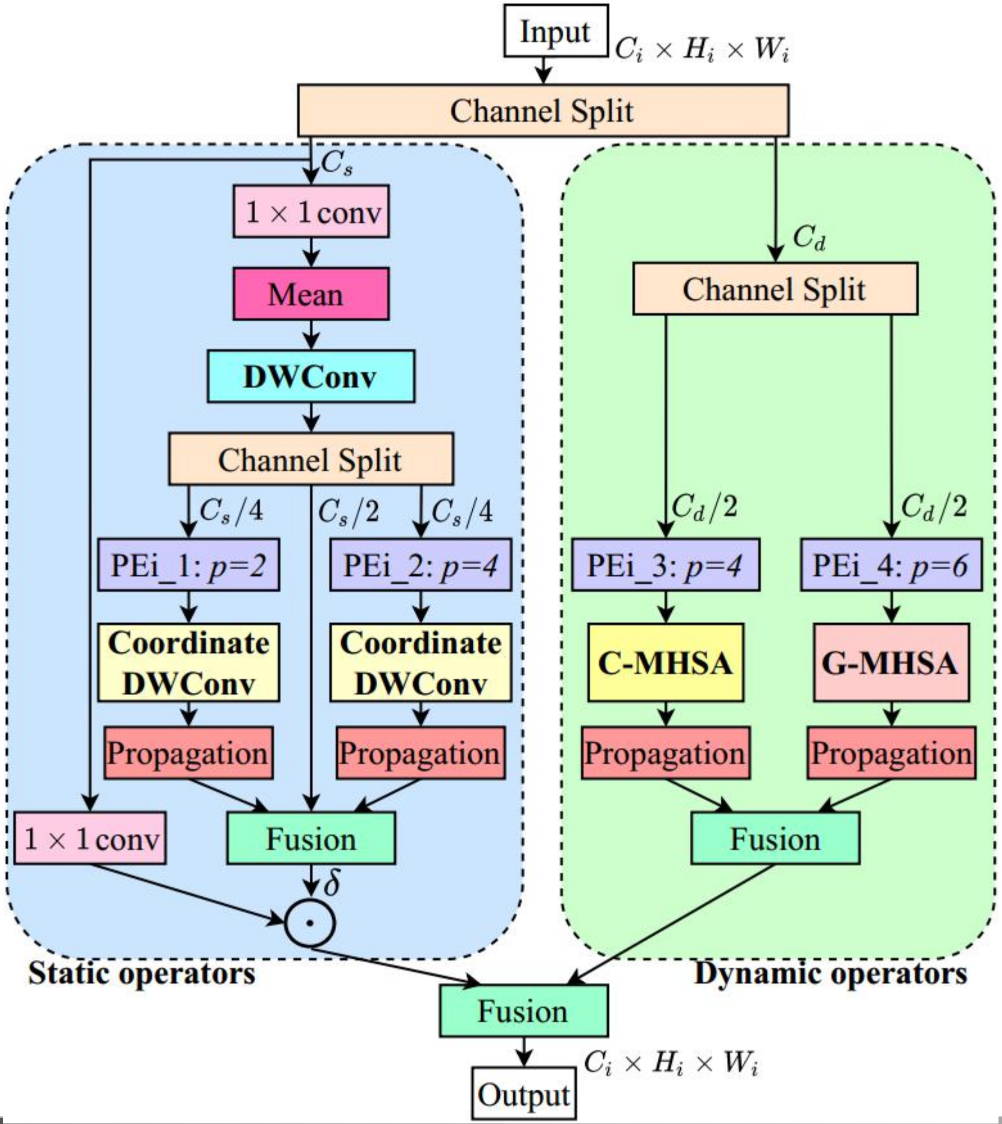
Overview of EMSNet:

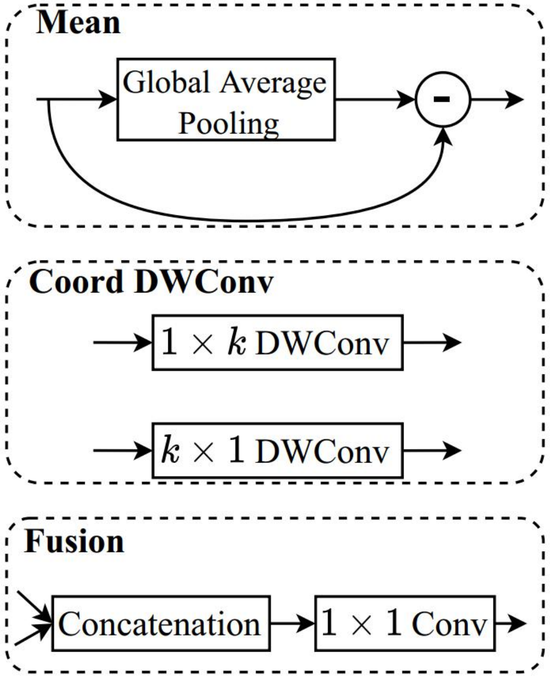


- ✓ PE1_1-1_4: 4 patch embeddings with different scales
- ✓ Channel MLP (Multi-Layer Perceptron) Interaction:
    - 2 Fully Connected Layers

H, W, C: Height, Width, and number of channels

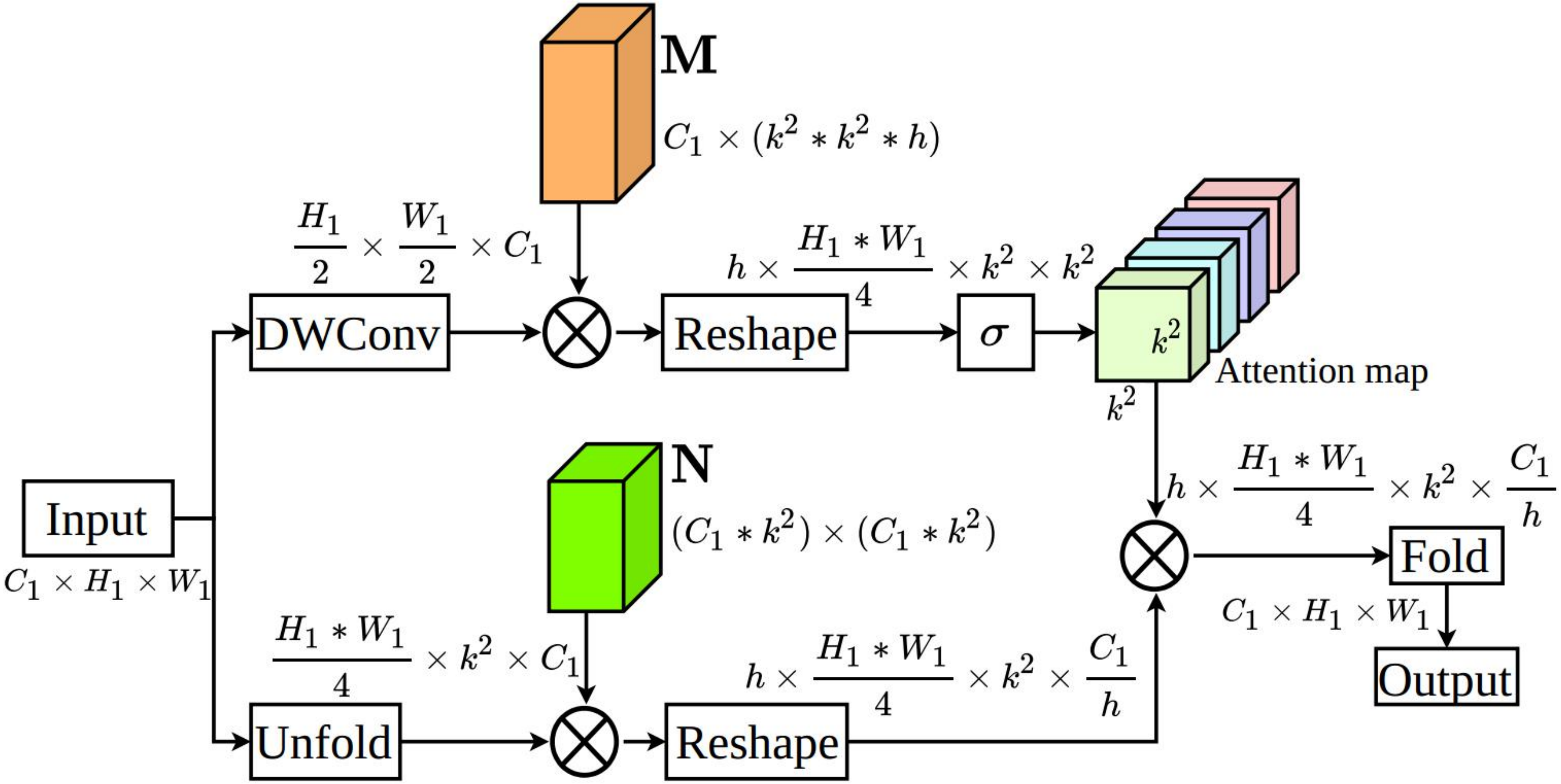
▶ Multi-scale Spatial Interactions:





- ✓ PE: patch embedding with patch size **p**
  - Implemented by depthwise (dw) conv with kernel **p** and stride **p**
- ✓ Propagation: distribute mixed information of represented tokens to its neighborhood
  - Implemented by transposed convolution
- ✓ G-MHSA: Global Multi-Head Self-Attention: capture global features
  - Adopt G-MHSA from ViT model
- ✓ C-MHSA: Convolution-based Multi-Head Self-Attention

Convolution-based Multi-Head Self-Attenion (C-MHSA):



M, N: learnable matrices
k: kernel size
h: number of heads

Proposed Networks

- **EMSNet Variants:**
  - Stack more blocks in stage 3 inspired by EdgeViT, Swin Transformer

Model	$[C_1, C_2, C_3, C_4]$	$[L_1, L_2, L_3, L_4]$	MLP ratio	GFLOPs	#param (M)
EMSNet-XXTiny	[32, 64, 128, 192]	[2, 2, 4, 2]	[8, 8, 4, 4]	0.5	2.5
EMSNet-XTiny	[32, 64, 96, 128]	[3, 3, 10, 2]	[8, 8, 4, 4]	0.7	3.0
EMSNet-Tiny	[64, 96, 128, 256]	[3, 3, 10, 2]	[8, 8, 4, 4]	1.9	5.4

Experimental Setup

- **Image Classification:**
  - Dataset: ImageNet-1K (1.2M training and 50K validation images with 1K classes)

  - Configurations:
    - Epochs: 300, Batch size: 512
    - Optimizer: Adam
    - Learning rate: $1e^{-3}$
    - Image size: 224×224

Image Classification Results

▶ Comparison with lightweight networks:

Method	Type	Image Size	#param (M)	GFLOPs	Top-1 Accuracy (%)
MobileViTv1-XXS	Hybrid	256^2	1.3	0.4	69.0
MobileViTv2-0.5	Hybrid	256^2	1.4	0.5	70.2
PVTv2-B0	Hybrid	224^2	3.7	0.6	70.5
MobileViTv3-0.5	Hybrid	256^2	1.4	0.5	72.3
ResNet-18	Conv	224^2	11.7	1.8	69.8
TNT-Ti	Attn	224^2	6.1	1.4	73.9
EdgeViT-XXS	Hybrid	256^2	4.1	0.6	74.4
Swin-0.7G	Attn	224^2	4.4	0.7	74.4
PVTv1-T	Attn	224^2	13.2	1.9	75.1
PoolFormerS12	Hybrid	224^2	11.9	1.8	77.2
ParC-Net-S	Conv	256^2	5.0	3.5	78.6
EMSNet-XXTiny	**Hybrid**	**224^2**	**2.5**	**0.5**	**73.1**
EMSNet-XTiny	**Hybrid**	**224^2**	**3.0**	**0.7**	**77.1**
EMSNet-Tiny	**Hybrid**	**224^2**	**5.4**	**1.9**	**79.3**

Experimental Setup

- **Object Detection and Instance Segmentation:**
  - Dataset: MS-COCO
    - 115K training images, 5K validation images with 80 categories
  - Baseline detectors: RetinaNet and Mask R-CNN
    - Replace backbone ResNet-50 with pretrained EMSNet
    - Neck, Head is kept same as baseline
  - Configurations:
    - Epochs: 12, Batch size: 4
    - Optimizer: Adam
    - Learning rate: $1e^{-4}$
    - Image size: 1333×800

Object Detection and Instance Segmentation Results

Object Detection with RetinaNet:

Backbone	#param (M)	GFLOPs	AP^{box}	AP^{50}	AP^{75}
ResNet-18	21.3	188.7	31.7	49.6	33.4
ResNet-50 (baseline)	37.7	250.3	36.3	55.4	39.1
PVTv1-T	23.0	183.3	36.6	56.6	38.8
PVTv2-B0	13.0	160.4	37.1	57.2	39.2
EdgeViT-XXS	13.7	162.2	38.7	59.0	41.0
EMSNet-XXTiny	**11.7**	**162.1**	**37.3**	**57.3**	**39.4**
EMSNet-XTiny	**12.4**	**167.9**	**39.0**	**59.1**	**41.4**
EMSNet-Tiny	**14.7**	**190.3**	**41.2**	**61.3**	**44.2**

Instance Segmentation with Mask R-CNN:

Backbone	#param (M)	GFLOPs	AP^{mask}	AP^{50}	AP^{75}
ResNet-18	31	207	31.2	51.0	32.7
ResNet-50 (baseline)	44	260	34.4	55.1	36.7
PVTv1-T	33	208	35.1	57.6	37.3
PVTv2-B0	24	179	36.2	57.8	38.6
EMSNet-XTiny	**23**	**186**	**37.1**	**58.5**	**40.0**
EMSNet-Tiny	**25**	**209**	**39.0**	**62.1**	**41.9**

Ablation Study

▶ Importance of each component in EMS Block:

Component		#param (M)	GFLOPs	Top-1 (%)
Static branch	Baseline	2.09	0.48	70.2
	+Mean	2.24	0.51	70.5
	+CoordDW w/ patch size=2	2.41	0.53	70.9
	+CoordDWw/patch size=4	2.38	0.53	71.1
	+Gated Aggregation	2.68	0.59	71.9
Dynamic branch	+C-MHSA	3.10	0.55	72.7
	+G-MHSA	2.56	0.54	73.1

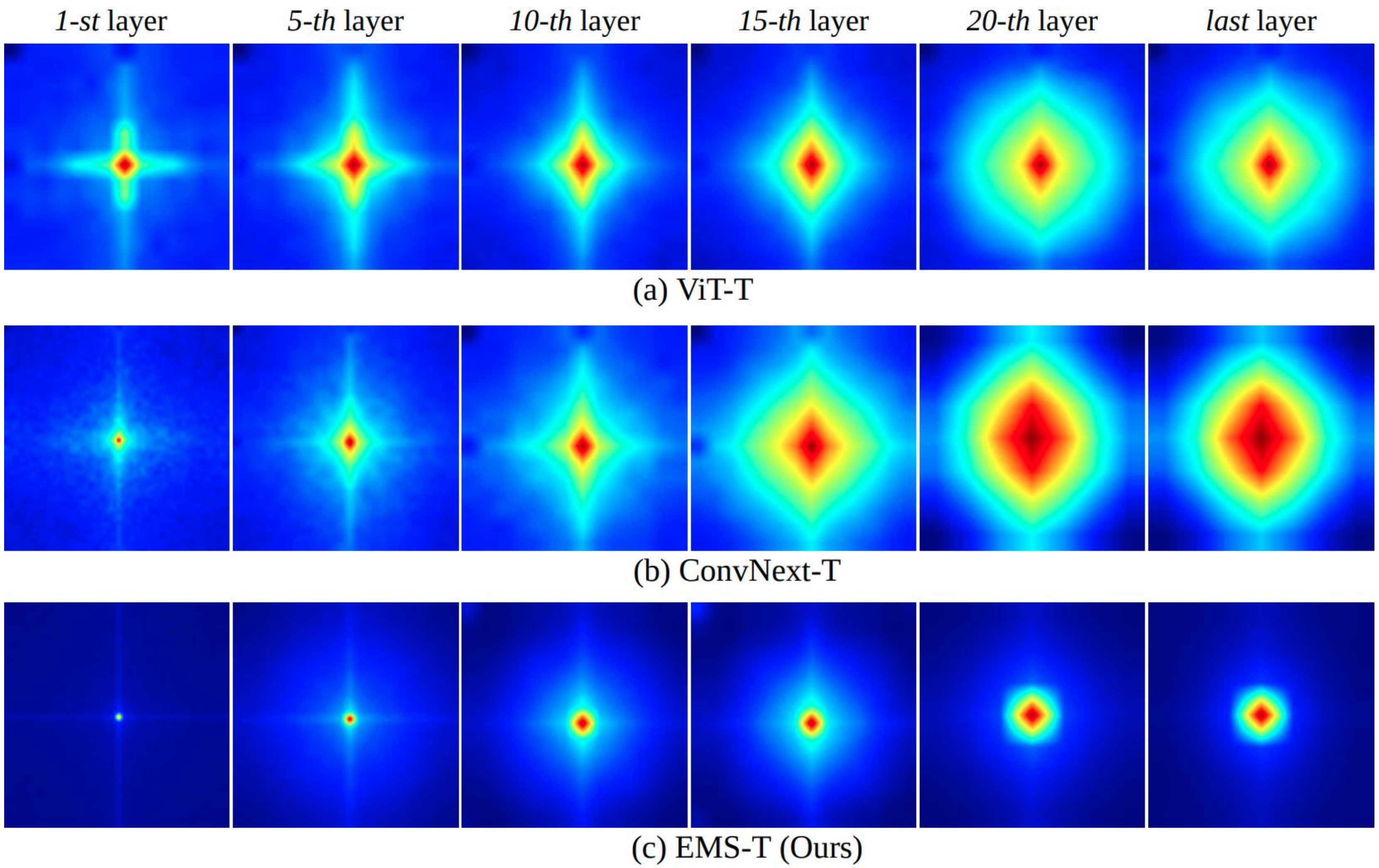
Comparison with Other Token Mixers

- **Latency comparison:**
  - CPU: Intel(R) Xeon(R) Gold 5220R@2.20GHz
  - GPU: Tesla V100 32GB

Method	Token Mixer	#param (M)	GFLOPs	Latency (ms)		Top-1 Accuracy
				CPU	GPU	
PVTv2-B0	Spatial Reduction Attention	3.7	0.6	67.3	0.46	70.5
Swin-0.7G	Window+Shifted Attention	4.4	0.7	67.3	0.76	74.4
ConvNeXt-XT	7×7 Depthwise Convolution	4.4	0.7	37.6	0.78	75.1
HaloNet	Local Attention	4.4	0.7	83.7	1.03	75.8
EMSNet-XTiny	Ours	3.0	0.7	70.3	0.57	77.1

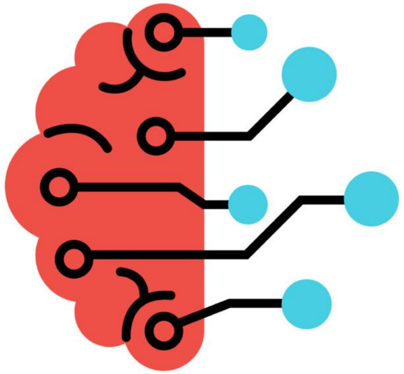
Investigation of EMSNet

Amplitude Spectrum:



- ✓ Tend to weaken high-frequency component
- ✓ Tend to enhance high-frequency component
- ✓ Tend to balance the range of frequencies

Mixing Abstract Tokens (MAT Transformers)

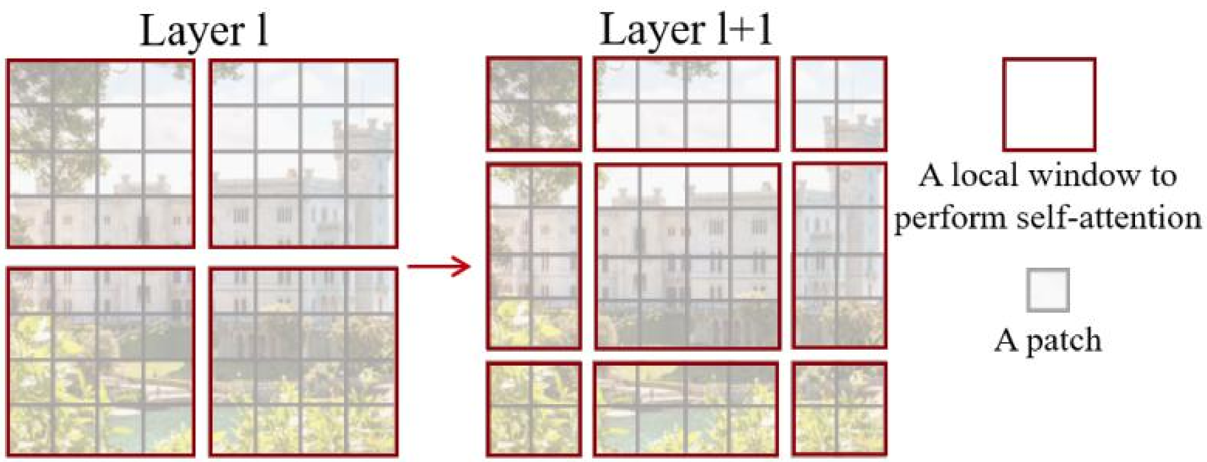


Bottleneck of Swin Transformer

Proposed MAT

Experimental Results

Drawback of Swin Transformer:

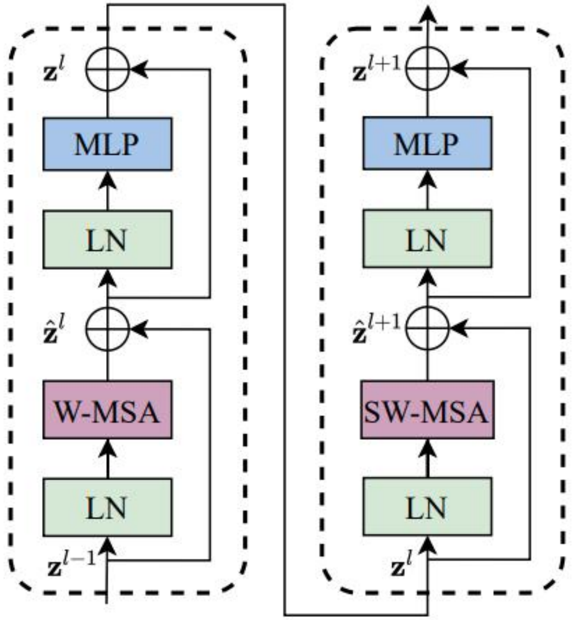


Non-overlapped windows

Shifted windows

- ☑ Locality
- ☑ Added relative positions
- ☒ Limited receptive fields
- ☒ Weak modelling capability

- ☑ Cross-window connections
- ☒ Slightly expanding receptive field
- ☒ Inefficient implementation: **torch.roll()**
- ① High memory access and extra latency
- ② Not well supported and optimized in modern deep learning frameworks: ONNX, NVIDIA Tensor RT, Torchscript



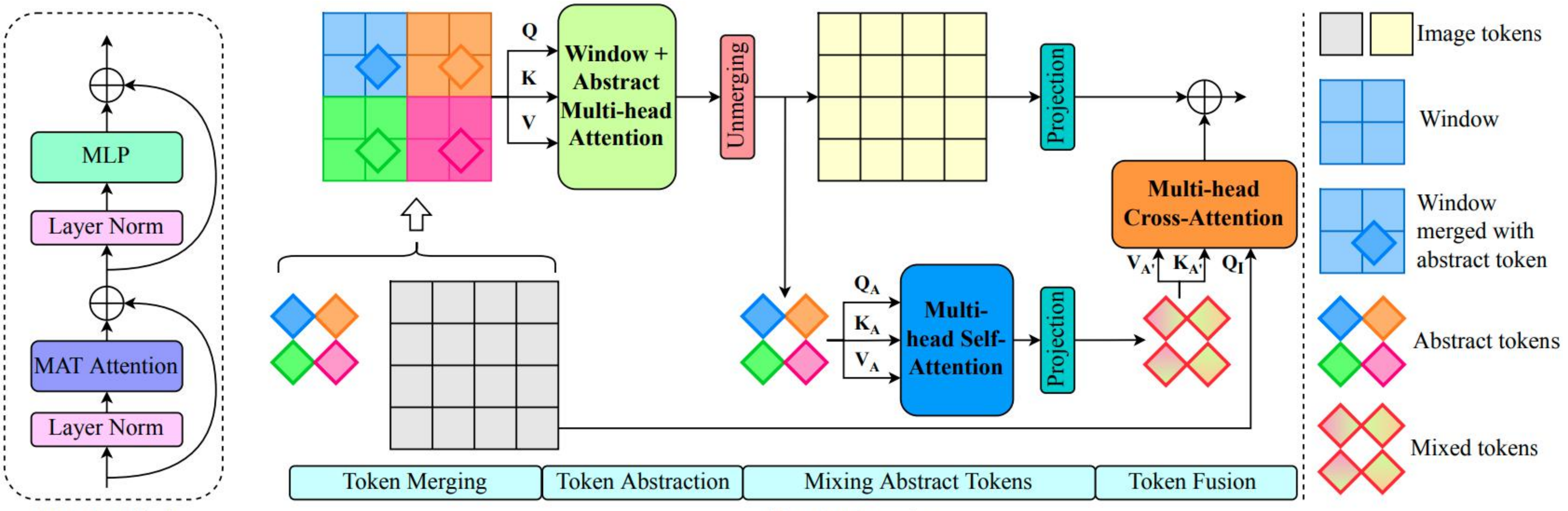
Two Successive Swin Transformer Block

This work efficiently exchanges information across non-overlapped windows → **Mixing Abstract Tokens (MAT)**

✓ Contain **matrix multiplication**

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2019

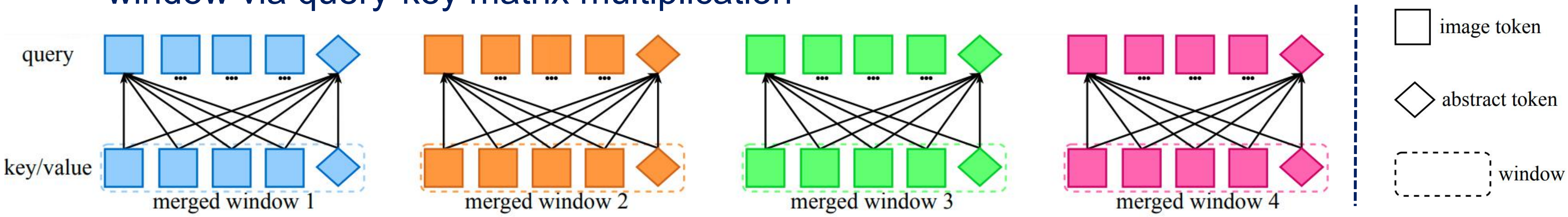
Mixing Abstract Tokens (MAT):



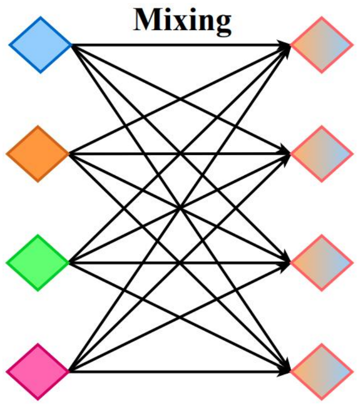
(a) MAT Block (b) MAT Attention

Proposed MAT Block

- **The role of abtract token:**
  - Learnable and a bridge between non-overlapped windows
  - Capturing the information of each window by a weighted sum of all tokens in each window via query-key matrix multiplication



(c) Token Abstraction

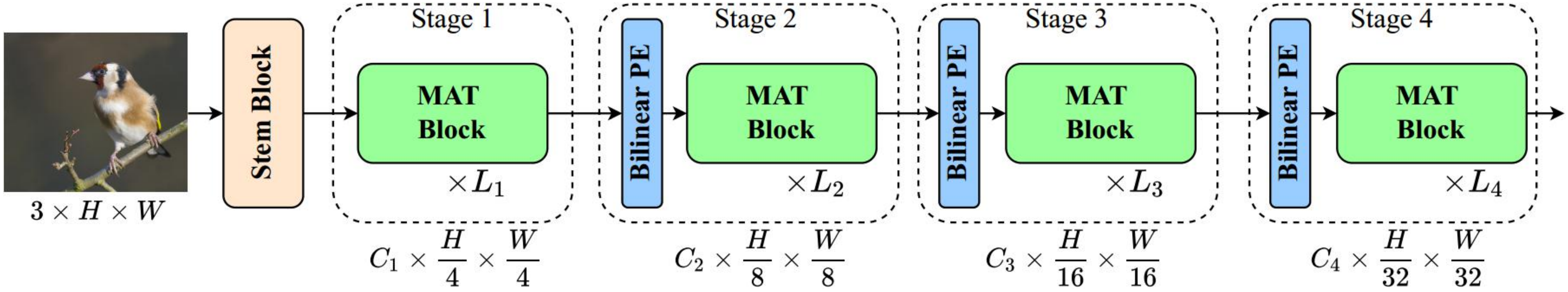


(d) Mixing Abstract Tokens

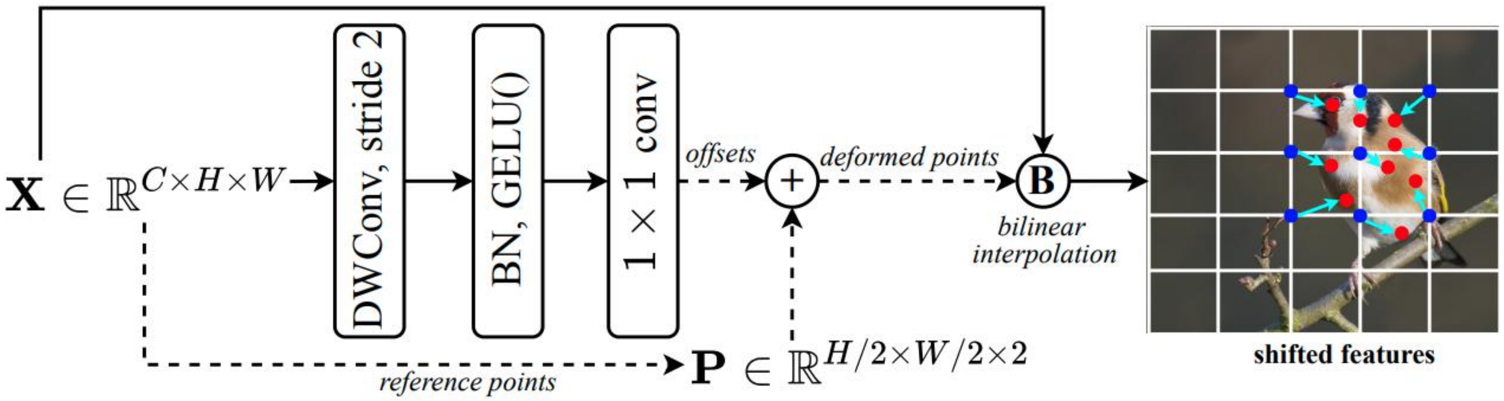
- ✓ Help to exchange information across windows
- ✓ Directly result in global information
- ✓ #windows are small → Low cost

Overall architecture of MAT Transformer:

- Stem Block: two successive 3×3 convolution with stride 2



- Bilinear PE (Patch Embedding): sample relevant regions of input feature based on learned offsets and grid of pixel locations



MAT Transformer Configuration

MAT block:

- Capture long-range dependencies from the input token
- Insert MAT blocks into stages (3, 4) → better trade-offs between accuracy and costs

Model	Stage				Top-1 (%)	GFLOPs	#param(M)
	1	2	3	4			
Model 1	✓	✓	✓	✓	78.9	0.783	10.874
Model 2	✗	✓	✓	✓	78.8	0.707	10.852
Model 3	✗	✗	✓	✓	79.0	0.666	10.767
Model 4	✗	✗	✗	✓	78.5	0.568	9.767

Positions of MAT blocks

✓ MAT blocks used in this stage

✗ Only MLP is used

Variant	#dim	#blocks	#heads	GFLOPs	#param(M)
MAT-1	24	2, 2, 6, 6	12, 24	0.389	6.714
MAT-2	32	2, 2, 6, 6	8, 16	0.666	10.767
MAT-3	36	2, 2, 8, 8	8, 16	1.042	17.008
MAT-4	48	3, 3, 8, 8	12, 24	1.933	29.057
MAT-5	64	2, 2, 8, 8	16, 32	3.156	50.108

Detailed configurations of 5 MAT Transformers

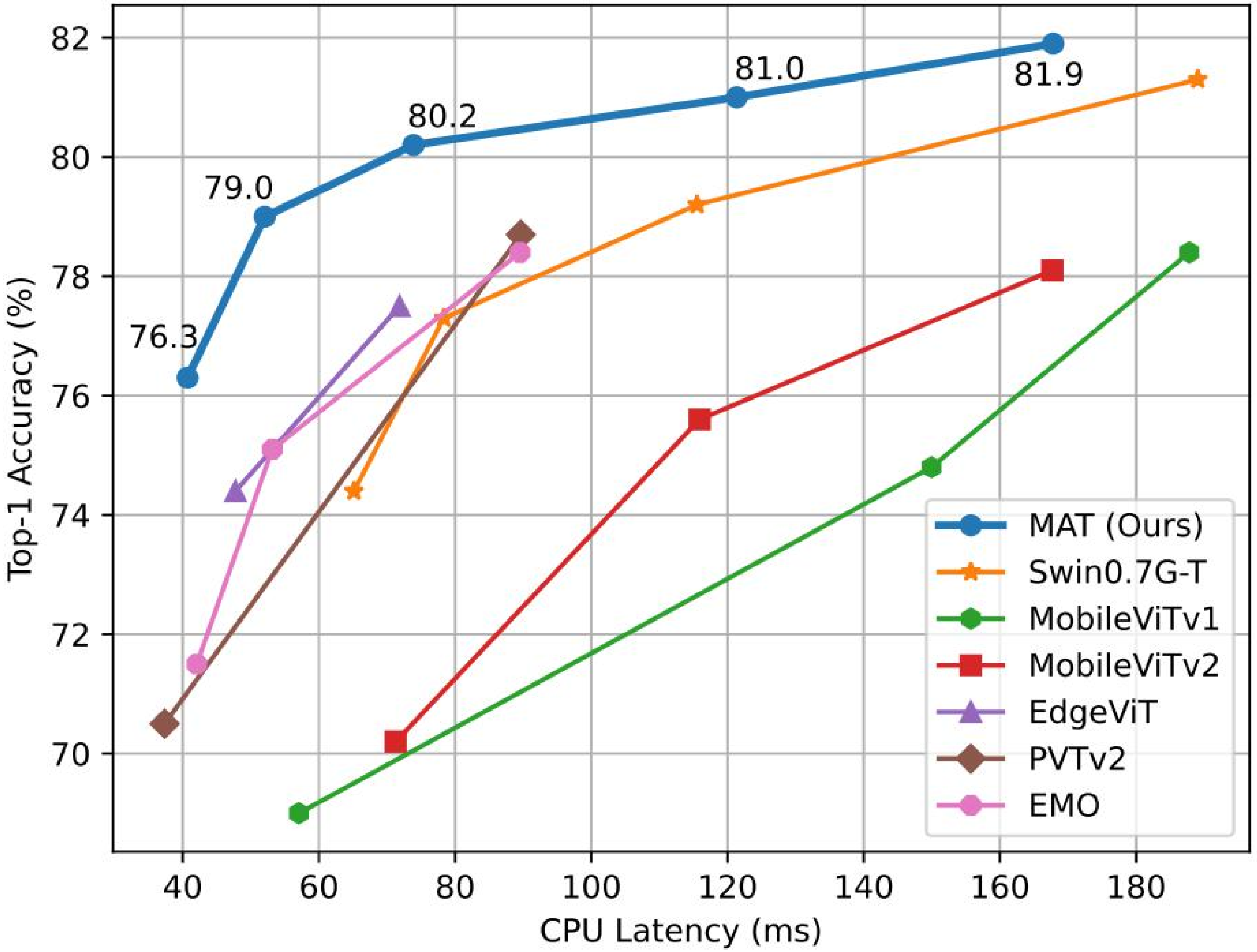
- #dim: number of base channels and duplicated in the next stage
- #blocks: number of stacked MAT blocks
- #heads: number of heads

Experimental Setup - Image Classification

- **Image Classification:**
  - Dataset: ImageNet-1K
    - 1.2M training images, 50K validation images with 1K categories
  - Configurations:
    - Epochs: 300, Batch size: 4096
    - Optimizer: Adam
    - Learning rate: $1e^{-3}$
    - Image size: 224×224

CPU Latency:

- Device: CPU-Intel(R) Xeon(R) Gold 5220R@2.20GHz



Experimental Setup - Object Detection, Instance Segmentation

- **Object Detection and Instance Segmentation:**
  - Dataset: MS-COCO
    - 115K training images, 5K validation images with 80 categories
  - Baseline detectors: RetinaNet and Mask R-CNN
    - Replace original backbone with pretrained MAT Transformers
    - Neck, Head is kept same as baseline
  - Configurations:
    - Epochs: 12
    - Batch size: 16 (RetinaNet, Mask R-CNN)
    - Optimizer: Adam
    - Learning rate: $1e^{-4}$
    - Image size: 1333×800 (RetinaNet, Mask R-CNN)

Object Detection and Instance Segmentation Results

Baseline: RetinaNet

Backbone	#param (M)	GFLOPs	AP^{box}
ResNet-18	21	189	31.8
ResNet-50	38	250	36.3
PVT-T	23	183	36.7
PVTv2-B0	13	160	37.1
PoolFormer-S12	22	207	36.2
EMO-2M	12	167	36.2
EMO-5M	15	207	38.9
PVT-S	34	167	40.4
LIT-S	39	178	41.6
Swin-T	38	273	41.5
MAT-2 (Ours)	**18**	**164**	**38.1**
MAT-3 (Ours)	**25**	**172**	**39.6**
MAT-4 (Ours)	**37**	**191**	**41.9**
MAT-5 (Ours)	**58**	**217**	**42.8**

Baseline: Mask R-CNN

Backbone	#param (M)	GFLOPs	AP^{box}	AP^{mask}
ResNet-18	31	207	34.0	31.2
ResNet-50	44	260	38.0	34.4
ResNet-101	63	336	40.4	36.4
PVTv2-B0	23	196	38.2	36.2
PVT-T	33	208	36.7	35.1
PVT-S	44	245	40.4	37.8
PVT-M	64	302	42.0	39.0
PVT-L	81	364	42.9	39.5
LIT-S	48	324	42.0	39.1
Swin-T	48	264	42.2	39.1
MAT-2 (Ours)	**29**	**182**	**39.4**	**36.7**
MAT-3 (Ours)	**35**	**190**	**41.2**	**38.1**
MAT-4 (Ours)	**47**	**209**	**43.2**	**39.6**
MAT-5 (Ours)	**68**	**235**	**43.8**	**40.0**

Experimental Setup - Semantic Segmentation

- **Semantic Segmentation:**
  - Dataset: ADE20K
    - 20K training images, 2K validation images
  - Baseline segmentor: Semantic FPN
    - Replace original backbone with pretrained MAT Transformers
    - Neck, Head is kept same as baseline
  - Configurations:
    - Iterations: 80K
    - Batch size: 16
    - Optimizer: Adam
    - Learning rate: $2e^{-4}$
    - Image size: 512×512

Semantic Segmentation Results

▶ Baseline: Semantic FPN

Backbone	#param (M)	GFLOPs	mIoU (%)
ResNet-50 (original)	29	183	36.7
ResNet-101	48	260	38.8
PVT-S	28	161	39.8
PVT-M	48	219	41.6
PVT-L	65	283	42.1
Swin-T	32	182	41.5
MAT-2 (Ours)	**13**	**98**	**40.0**
MAT-3 (Ours)	**19**	**107**	**41.9**
MAT-4 (Ours)	**31**	**127**	**43.3**
MAT-5 (Ours)	**52**	**154**	**44.1**

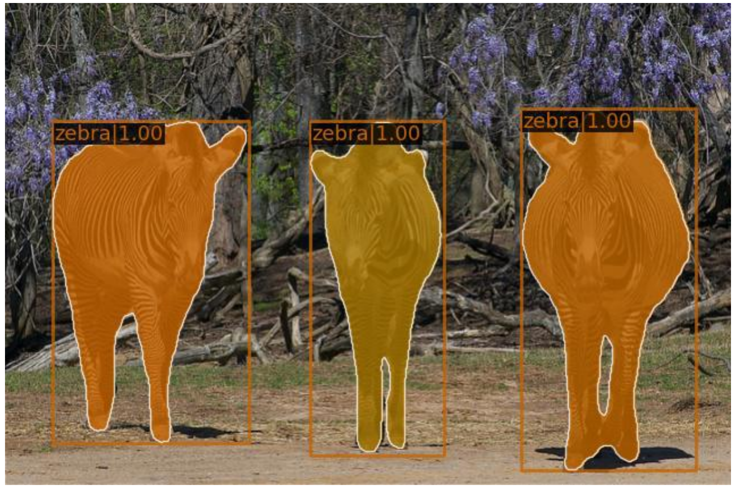
Ablation Study

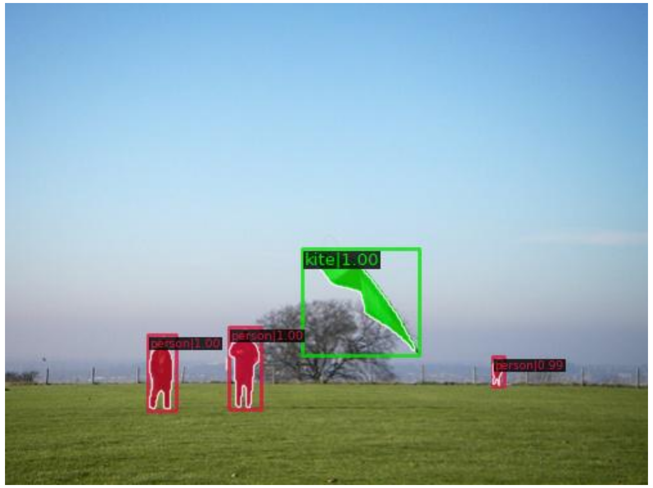
▶ MAT Attention and Bilinear Patch Embedding:

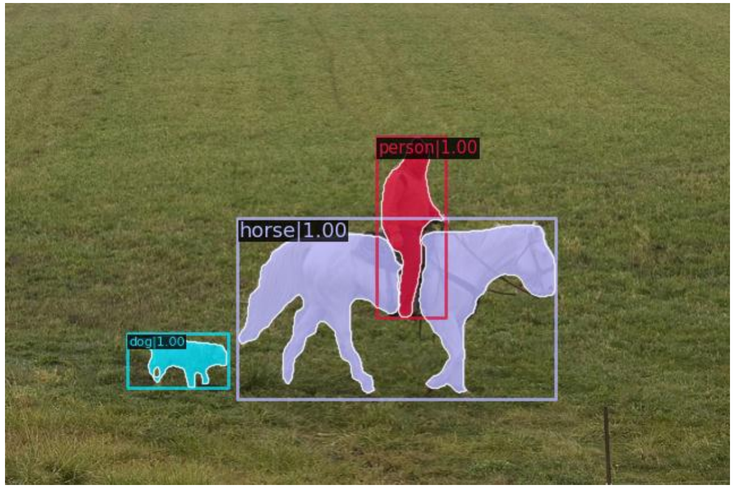
- ▶ Throughput is measured on one GPU Tesla V100 32GB

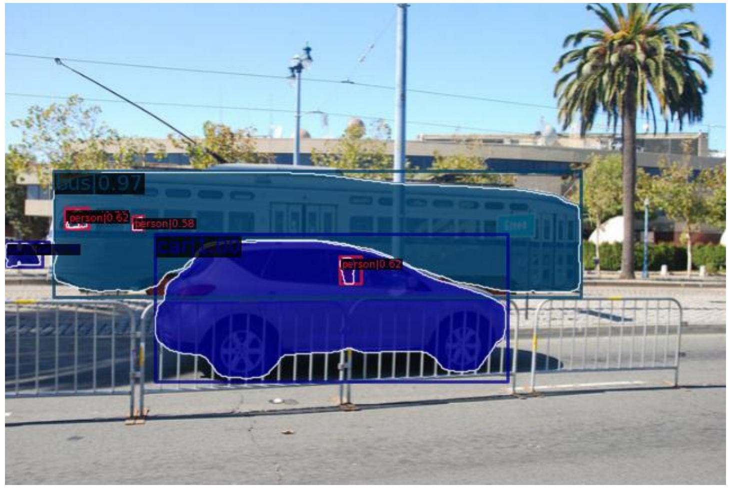
Module		Top-1 Accuracy	#param (M)	GFLOPs	Throughput (images/second)
Pure MLP		58.4	5.699	0.461	10303
Token Mixer	+Window Attention	76.9	7.802	0.659	4353
	+MAT Attention (Ours)	79.0	10.767	0.666	4333
Patch Embed	3×3 Conv, stride 2	78.7	11.107	0.703	4432
	Patch Merging	78.5	10.892	0.679	4944
	Bilinear PE (Ours)	79.0	10.767	0.666	4333

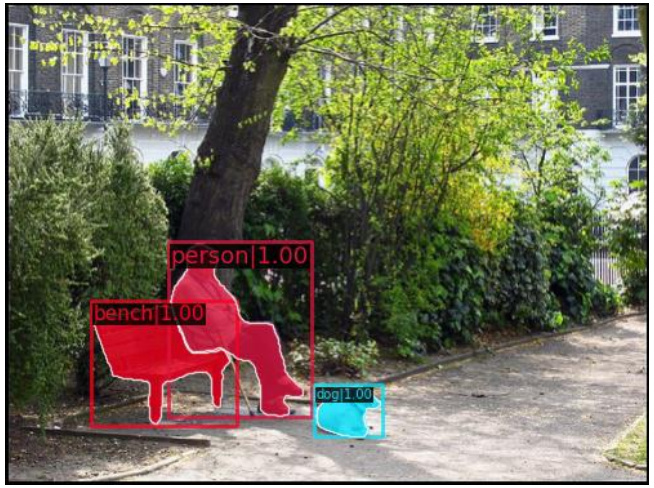
Qualitative Results: Mask R-CNN with MAT-2

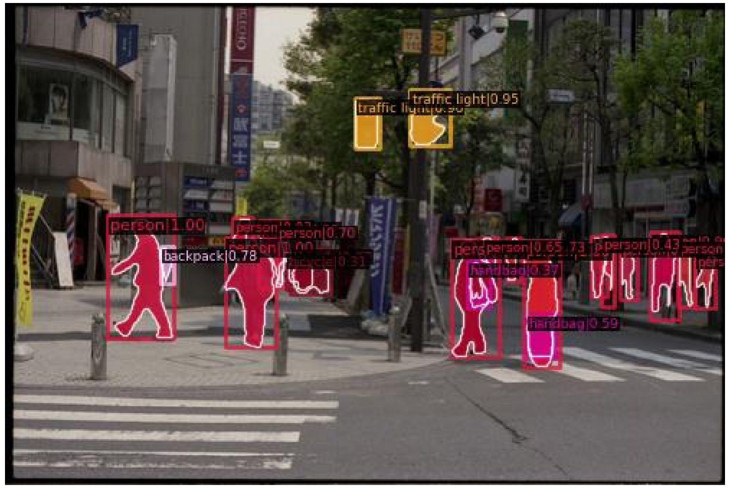










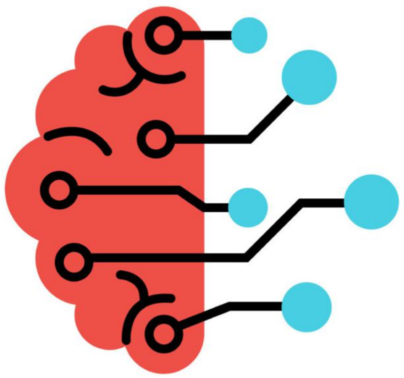


Partial Transformers (PartialFormer)

Computation Redundancy

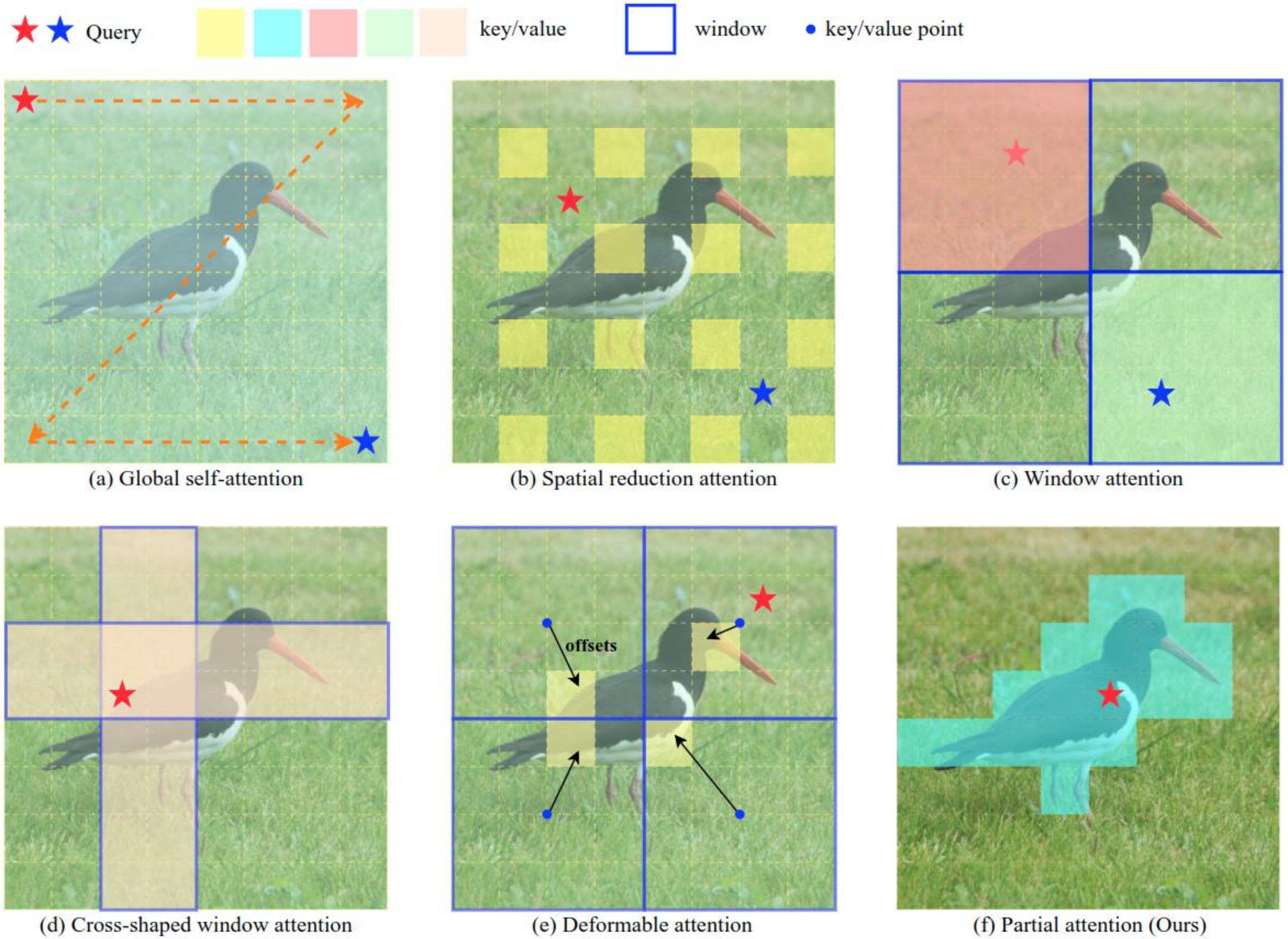
Proposed Partial Attention

Experimental Results



Computation Redundancy

Improvement of self-attention:



- ✓ These self-attentions: ***all queries*** attend to:
  - all regions (a)
  - down-sampled regions (b)
  - local windows (c, d)
  - shifted windows (e)

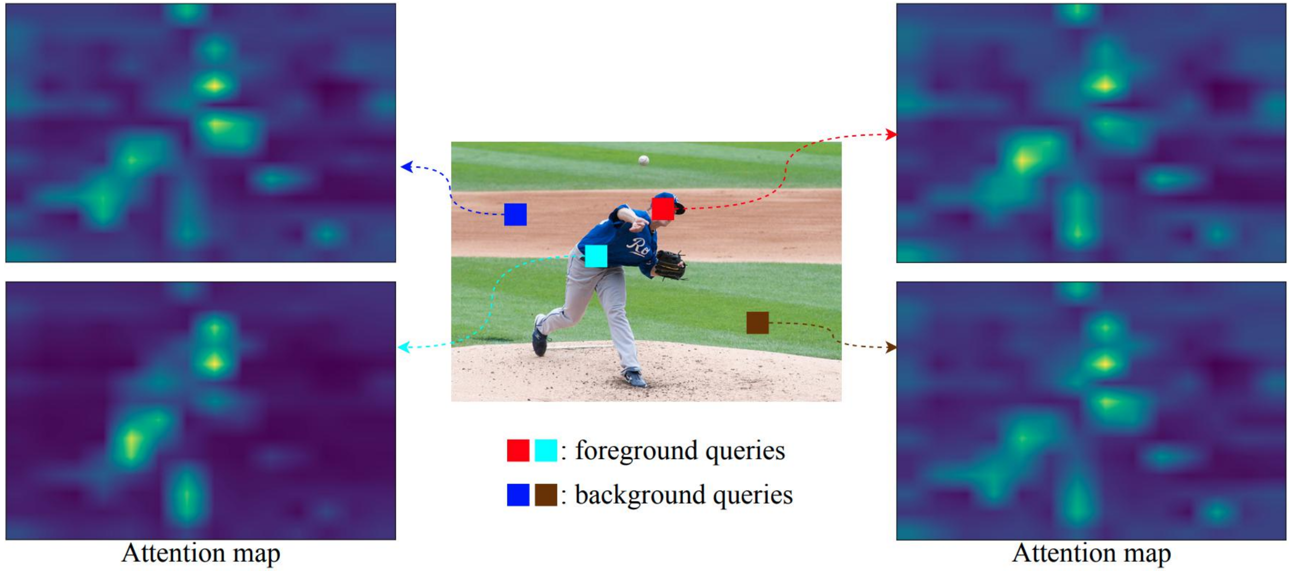
→ **attention patterns have high similarities**

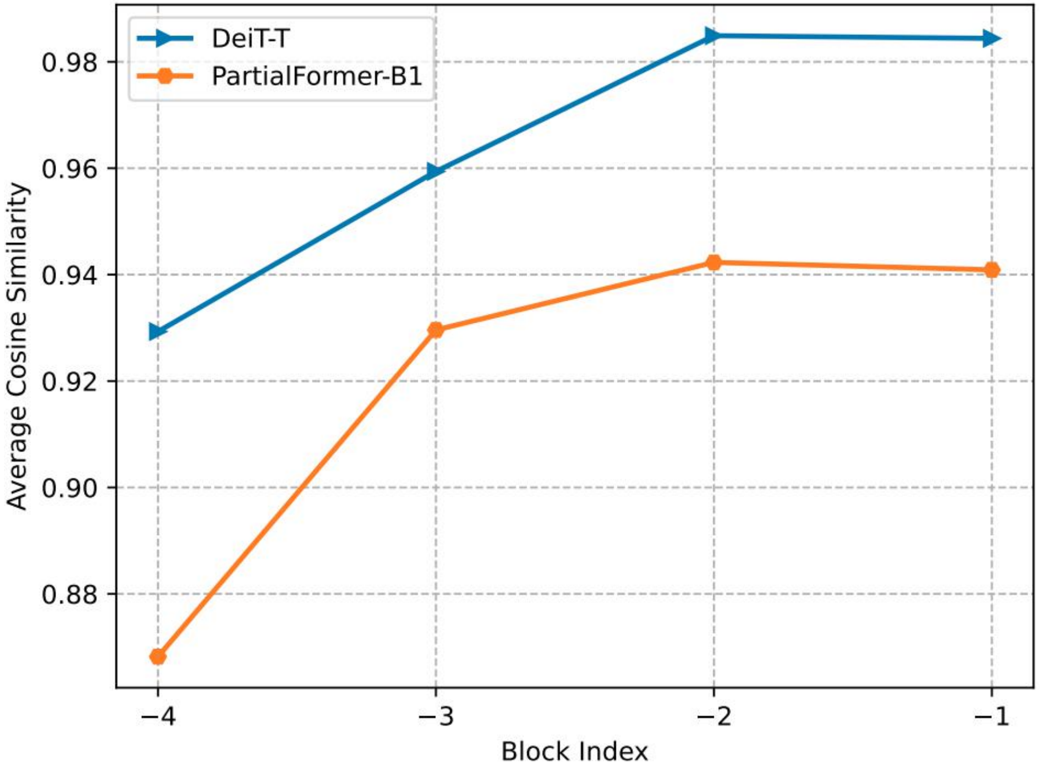
- ✓ Partial attention:
  - only foreground queries attend to relevant regions

→ **reduce computation redundancy of query-key interactions**

Computation Redundancy

- **Observation** - Visualization of trained DeiT model on ImageNet-1K
  - Attention maps for foreground/background queries are almost the same



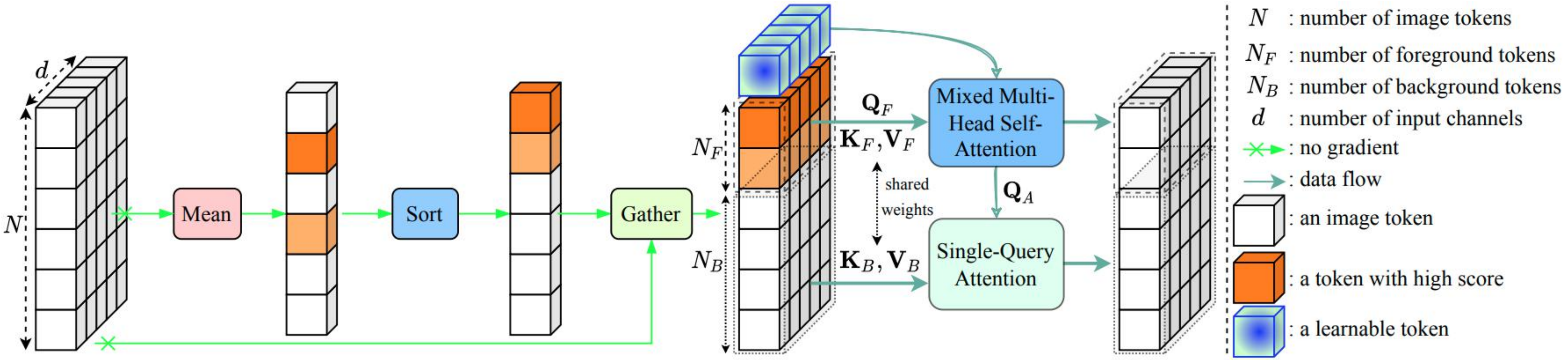


✓ Interacting each query with the full set of keys/values → suboptimal, computation redundancy

DeiT: Training data-efficient image transformers & distillation through attention, ICML`2021

Proposed Partial Attention

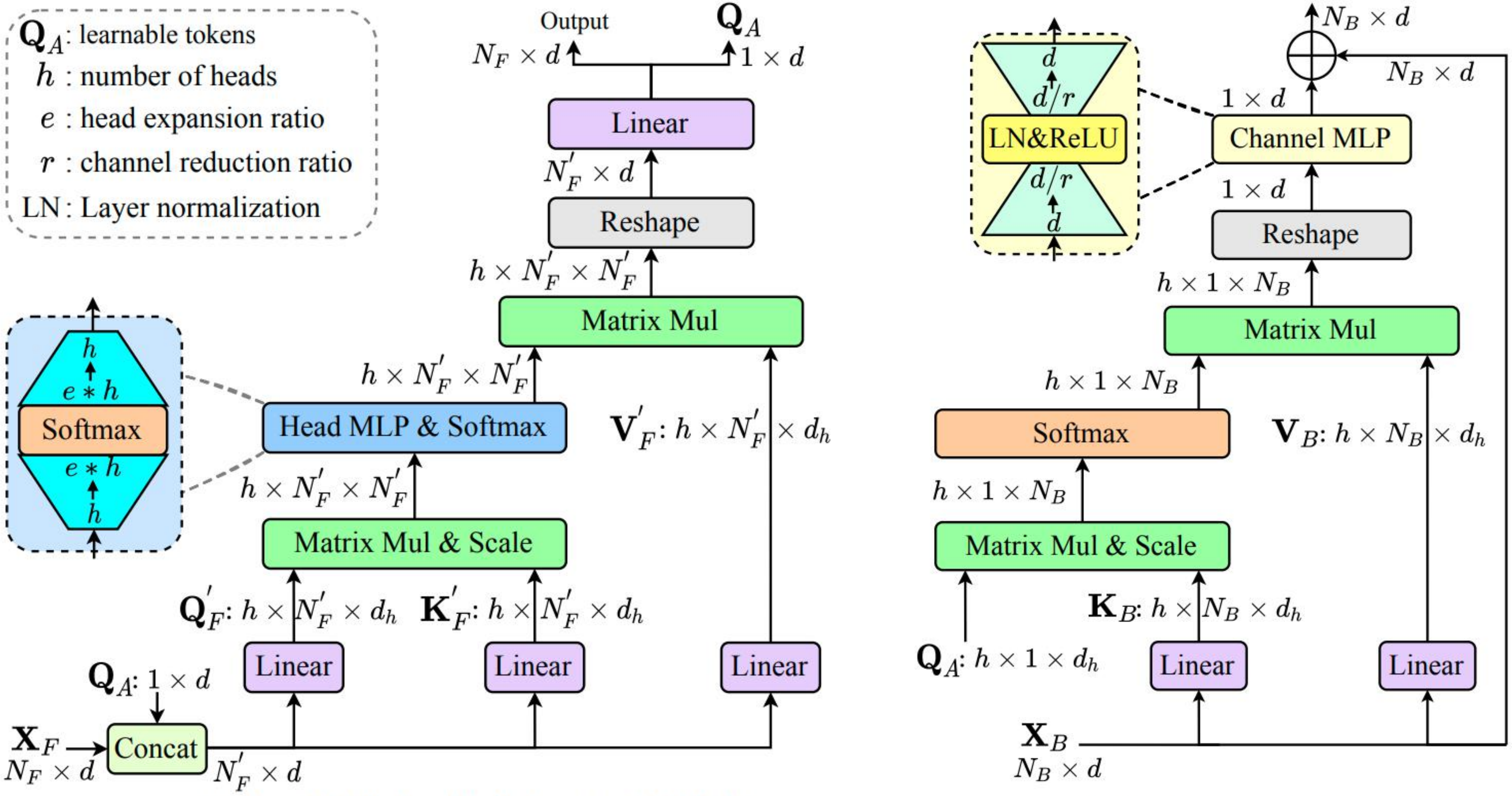
- **Token separation:** seperate image tokens into foreground and background sets based on context scores
  - Mean() + Sort() + Gather()
- **Mixed Multi-Head Self-Attention (MMSA):** fully capture informative features from foreground set
- **Single-Query Attention (SQA):** squeeze the information of the most background tokens
- **Learnable token Q_A:** a bridge between to two sets



Input tokens

Output

Detailed structure of MMSA and SQA:

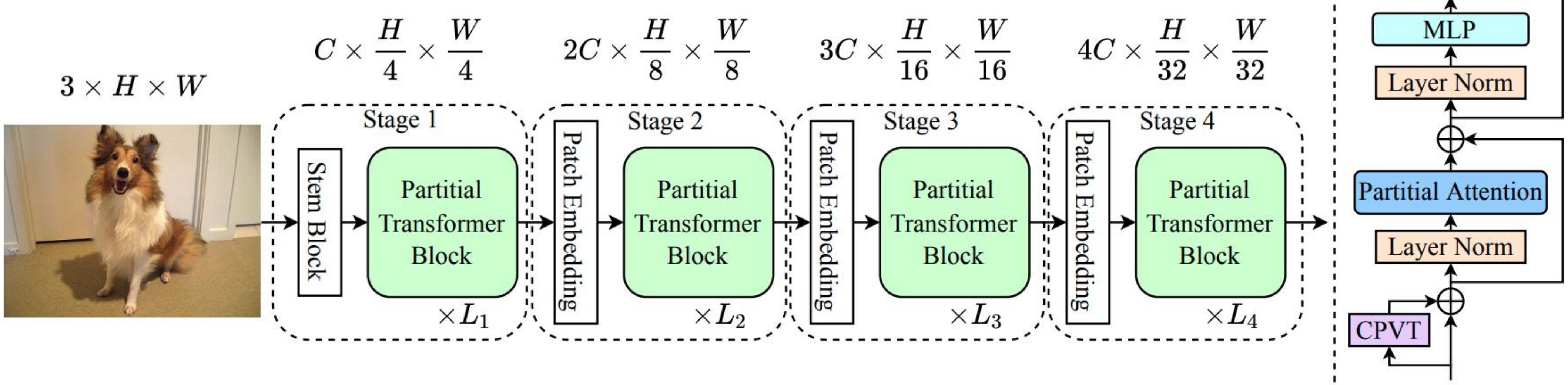


(a) Mixed Multi-head Self-Attention (MMSA)

(b) Single-Query Attention (SQA)

Model Configuration

▶ Overall architecture of PartialFormer (*Partial* Trans*Former*):



CPVT: learn local features implemented by 3×3 depthwise convolution

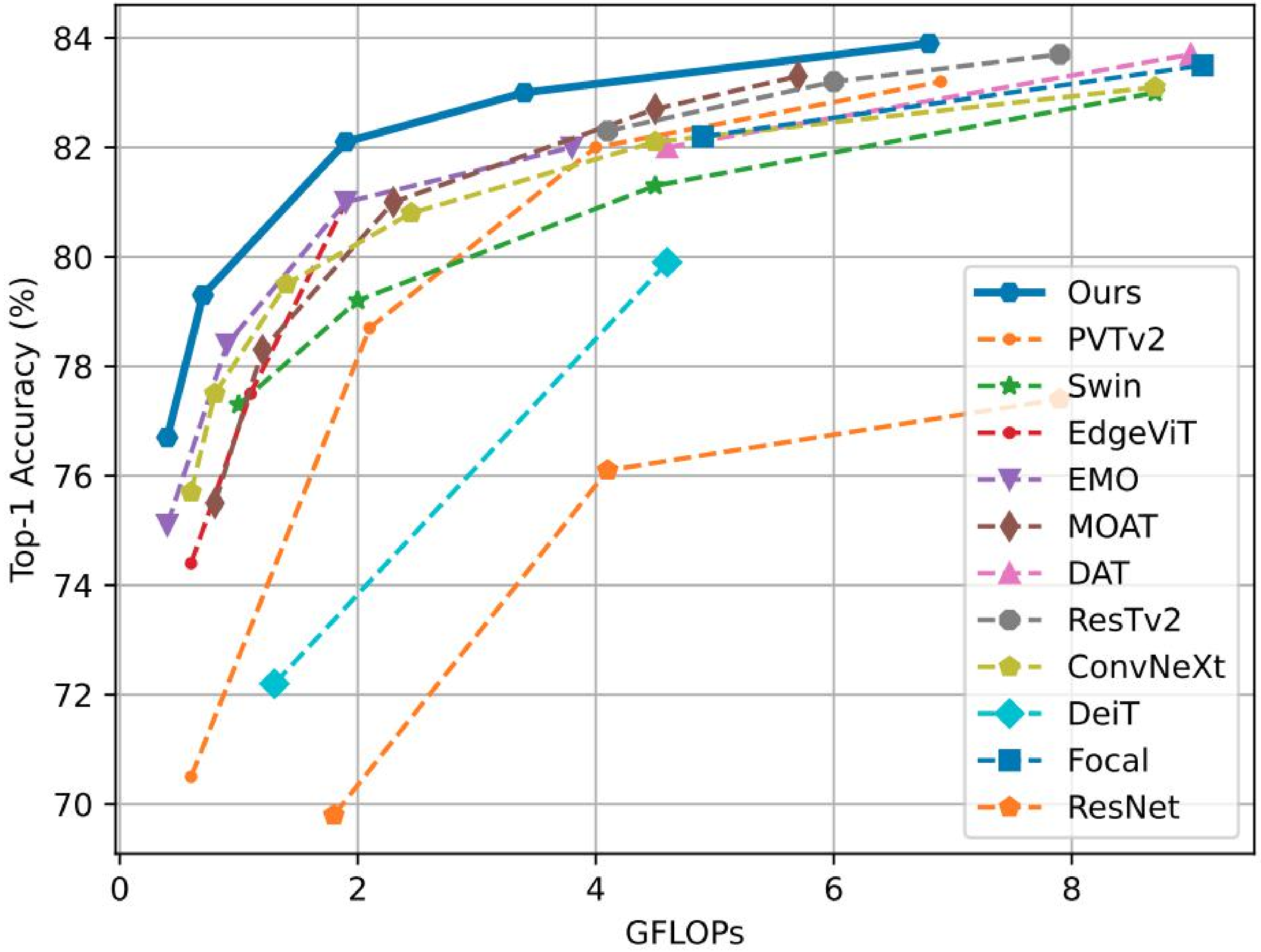
▶ Detailed configurations of five PartialFormers:

Model	*C*	*L*	#heads	#param (M)	GFLOPs
PartialFormer-B0	24	2, 2, 6, 6	2, 4, 8, 16	5.3	0.4
PartialFormer-B1	32	2, 2, 6, 6	2, 4, 8, 16	8.2	0.7
PartialFormer-B2	48	2, 2, 8, 8	3, 6, 12, 24	21.1	1.9
PartialFormer-B3	64	2, 2, 8, 8	4, 8, 16, 32	36.1	3.4
PartialFormer-B4	96	2, 2, 8, 6	6, 12, 24, 48	64.5	6.8

Experimental Setup - Image Classification

- **Image Classification:**
  - Dataset: ImageNet-1K
    - 1.2M training images, 50K validation images with 1K categories
  - Configurations:
    - Epochs: 300, Batch size: 4096
    - Optimizer: Adam
    - Learning rate: $1e^{-3}$
    - Image size: 224×224

- **Trade-off between accuracy and cost:**



Semantic Segmentation Results

▶ Comparison with other backbones:

Backbone	Semantic FPN 80K			UperNet 160K		
	#param(M)	GFLOPs	mIoU	#param(M)	GFLOPs	mIoU
ResNet-50 (Original)	25.8	183	36.7	66.5	951	42.0
ResNet-101 (Original)	47.5	260	38.8	86.0	1029	43.8
PVT-S	28.2	161	39.8	—	—	—
PVT-M	48.0	219	41.6	—	—	—
Swin-T	31.9	182	41.5	59.9	945	44.5
Focal-T	—	—	—	62.0	998	45.8
MixFormer-B3	—	—	—	44.0	880	44.5
DAT-T	32.0	198	42.6	60.0	957	45.5
Swin-S	53.2	274	45.2	81.0	1038	47.6
PartialFormer-B1 (Ours)	**10.8**	**101**	**40.2**	**34.8**	**856**	**43.3**
PartialFormer-B2 (Ours)	**23.5**	**131**	**42.3**	**48.6**	**887**	**45.9**
PartialFormer-B3 (Ours)	**38.4**	**166**	**43.5**	**64.5**	**923**	**47.0**
PartialFormer-B4 (Ours)	**66.6**	**246**	**45.0**	**94.7**	**1005**	**48.3**

Object Detection and Instance Segmentation Results

Baseline: RetinaNet and Mask R-CNN

Backbone	RetinaNet 1×			Mask R-CNN 1×			
	#param(M)	GFLOPs	AP^{box}	#param(M)	GFLOPs	AP^{box}	AP^{mask}
ResNet-18	21	189	31.8	31	207	34.0	31.2
ResNet-50	38	250	36.3	44	260	38.0	34.4
ResNet-101	57	315	38.5	63	336	40.4	36.4
PVT-T	23	183	36.7	33	208	36.7	35.1
PVT-S	34	273	40.4	44	245	40.4	37.8
PVT-M	54	384	41.9	64	367	42.0	39.0
LIT-S	39	305	41.6	48	324	42.9	39.6
Swin-T	38	251	41.5	48	270	42.2	39.1
Twin-S	34	225	43.0	44	244	43.4	40.3
DAT-T	38	253	42.8	48	272	44.4	40.4
PartialFormer-B1 (Ours)	**16**	**167**	**40.2**	**26**	**185**	**41.2**	**38.2**
PartialFormer-B2 (Ours)	**29**	**196**	**43.5**	**39**	**214**	**44.1**	**40.4**
PartialFormer-B3 (Ours)	**44**	**230**	**44.1**	**54**	**248**	**45.0**	**40.9**

Object Detection and Keypoint Detection

Object detection: SSD

Backbone	Image size	GFLOPs	#param(M)	AP^{box}
MobileViTv1-XXS	320^2	0.9	1.7	19.9
MobileViTv2-0.5	320^2	0.9	2.0	21.2
MobileNetv3	320^2	0.6	5.0	22.0
MobileNetv2	320^2	0.8	4.3	22.1
MobileNetv1	320^2	1.3	5.1	22.2
MobileViTv2-0.75	320^2	1.8	3.6	24.6
ResNet-50	320^2	20.2	22.9	25.2
PartialFormer-B0	**320^2**	**0.9**	**5.0**	**24.3**
PartialFormer-B1	**320^2**	**1.5**	**8.0**	**27.1**

Keypoint detection: SimpleBaseline

Backbone	Crop size	GFLOPs	#param(M)	$AP^{keypoint}$
RSN-18	256×192	2.3	9.1	70.4
ResNet-50 (original)	256×192	5.5	34.0	71.8
ResNet-101	256×192	9.1	53.0	72.8
PVT-S	256×192	4.1	28.2	71.4
Swin-T	256×192	6.1	32.8	72.4
PartialFormer-B1	**256×192**	**1.7**	**9.8**	**70.6**
PartialFormer-B2	**256×192**	**2.9**	**23.0**	**72.7**
PartialFormer-B3	**256×192**	**4.4**	**38.4**	**73.2**

Conclusion

- Develop efficient vision Transformers for image classification and dense prediction tasks
  - Mitigate computational bottlenecks in Transformer encoder
  - Enhance modeling ability of window self-attention
  - Reduce computation redundancy in global self-attention
- The proposed methods can outperform other state-of-the-art methods in both accuracy and speed
- **Future works**
  - Make models smaller and faster for real-time application and deploying on embedded devices
  - Integrate additional information from other modalities (a pair of image+text) into feature learning

Thank you for your attention!

Appendix

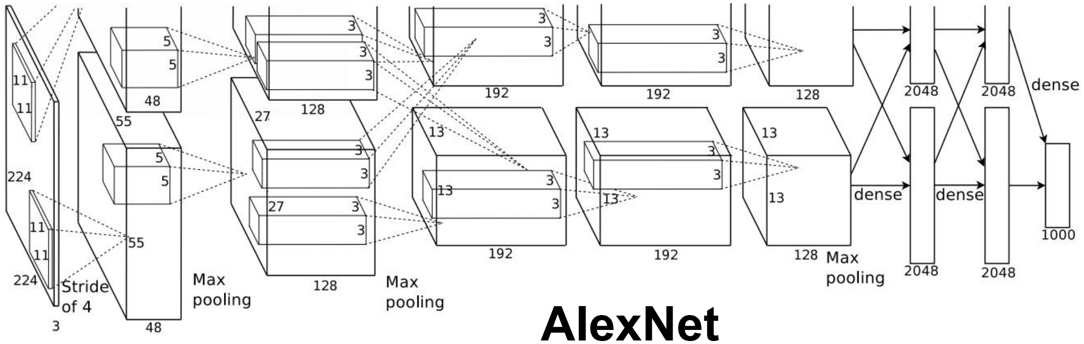
Conferences

- Xuan-Thuy Vo, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Hierarchical Vision Transformers with Shuffled Local Self-Attentions, IWIS, Korea, 2023.
- Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo, Simultaneous Person, Face, and Hand Detector Based on Improved YOLOv5, IWIS, Korea, 2023.
- Adri Priadana, Muhamad Dwisnanto Putro, Jinsu An, Duy-Linh Nguyen, XuanThuy Vo, and Kang-Hyun Jo, Gender Recognizer based on Human Face using CNN and Bottleneck Transformer Encoder, IWIS, Korea, 2023.
- Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo, Vehicle Detector Based on YOLOv5 Architecture for Traffic Management and Control Systems, IECON, Singapore, 2023.
- Adri Priadana, Muhamad Dwisnanto Putro, Jinsu An, Duy-Linh Nguyen, XuanThuy Vo, and Kang-Hyun Jo, Facial Attribute Recognition using Lightweight Multi-Label CNN-Transformer Architecture for Intelligent Advertising, IECON, Singapore, 2023.
- Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo, Car Detector Based on YOLOv5 for Parking Management, CITA, Vietnam, 2023.
- Xuan-Thuy Vo, Jehwan Choi, Duy-Linh Nguyen, Adri Priadana, and KangHyun Jo, Unifying Local and Global Fourier Features for Image Classification, ISIE, Finland, 2023.
- Adri Priadana, Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Xuan-Thuy Vo, and Kang-Hyun Jo, Age Group Recognizer based on Human Face Supporting Smart Digital Advertising Platforms, ISIE, Finland, 2023.
- Adri Priadana, Muhamad Dwisnanto Putro, Duy-Linh Nguyen, Xuan-Thuy Vo, and Kang-Hyun Jo, Human Face Detector with Gender Identification by Split-Based Inception Block and Regulated Attention Module, IW-FCV, 2023.
- Duy-Linh Nguyen, Xuan-Thuy Vo, Adri Priadana, and Kang-Hyun Jo, YOLO5PKLot: A Parking Lot Detection Network Based on Improved YOLOv5 for Smart Parking Management System, IW-FCV, 2023.
- Xuan-Thuy Vo, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo, Dynamic Circular Convolution for Image Classification, IW-FCV, Korea, 2023.
- Tien-Dat Tran, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, Combination of Deep Learner Network and Transformer for 3D Human Pose Estimation, ICCAS, Korea, 2022.
- Adri Priadana, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and Kang-Hyun Jo, A Facial Gender Detector on CPU using Multi-dilated Convolution with Attention Modules, ICCAS, Korea, 2022.
- Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen, Adri Priadana, and KangHyun Jo, Balancing Multiple Object Tracking Objectives based on Learned Weighting Factors, MAPR, Vietnam, 2022.
- Adri Priadana, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and Kang-Hyun Jo, An Efficient Face-based Age Group Detector on a CPU using Two Perspective Convolution with Attention Modules, MAPR, Vietnam, 2022.
- Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, Tien-Dat Tran, and Kang-Hyun Jo, Fire Warning Based on Convolutional Neural Network and Inception Mechanism, MAPR, Vietnam, 2022.

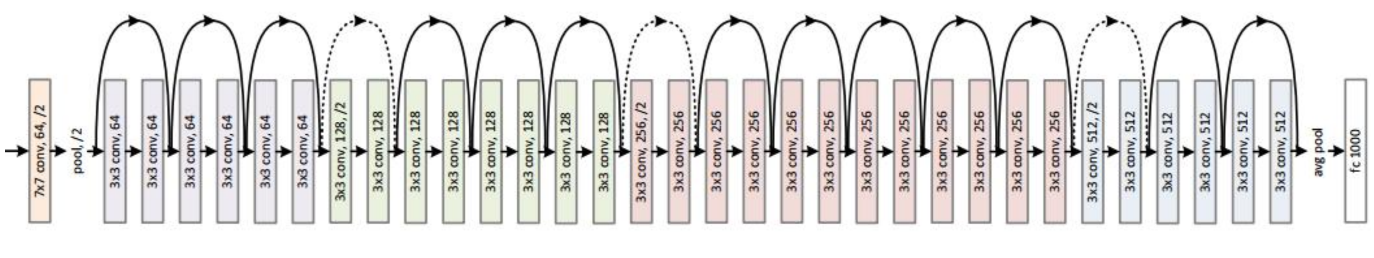
Conferences

- Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen, and Kang-Hyun Jo, A Study on Efficient Multi-task Networks for Multiple Object Tracking, IWIS, Korea, 2022.
- Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, Tien-Dat Tran, and Kang-Hyun Jo, Robust Hand Detection Based on Convolutional Neural Network and Attention Module, IWIS, Korea, 2022.
- Tien-Dat Tran, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, Efficient High-Resolution Network for Human Pose Estimation, IWIS, Korea, 2022.
- Van-Dung Hoang, Xuan-Thuy Vo, Khac-Anh Phu, and Kang-Hyun Jo, Fusion of Segmentation and Classification for Improving Skin Disease Diagnosis, GTSD, Vietnam, 2022.
- Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen, Kang-Hyun Jo, Multi-level Feature Reweighting and Fusion for Instance Segmentation, INDIN, Australia, 2022.
- Tien-Dat Tran, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, HighResolution Network with Attention Module for Human Pose Estimation, ASCC, Korea, 2022.
- Xuan-Thuy Vo, Van-Dung Hoang, Duy-Linh Nguyen, and Kang-Hyun Jo, Pedestrian Head Detection and Tracking via Global Vision Transformer, IW-FCV, Japan, 2022.
- Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and KangHyun Jo, Convolutional Neural Network Design for Eye Detection Under LowIllumination, IW-FCV, Japan, 2022.
- Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and KangHyun Jo, Light-weight Convolutional Neural Network for Distracted Driver Classification, IECON, Canada, 2021.
- Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen, and Kang-Hyun Jo, Dynamic Multi-loss Weighting for Multiple People Tracking in Video Surveillance Systems, INDIN, Spain, 2021.
- Duy-Linh Nguyen, Muhamad Dwisnanto Putro, Xuan-Thuy Vo, and KangHyun Jo, Triple Detector based on Feature Pyramid Network for License Plate Detection and Recognition System in Unusual Conditions, ISIE, Japan, 2021.
- Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen, and Kang-Hyun Jo, RegressionAware Classification Feature for Pedestrian Detection and Tracking in Video Surveillance Systems, ICIC, China, 2021.
- Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen, and Kang-Hyun Jo, Stairstep Feature Pyramid Networks for Object Detection, IW-FCV, Korea, 2021.
- Tien-Dat Tran, Xuan-Thuy Vo, Duy-Linh Nguyen, and Kang-Hyun Jo, Efficient Spatial-Attention Module for Human Pose Estimation, IW-FCV, Korea, 2021.
- Tien-Dat Tran, Xuan-Thuy Vo, Moahamammad-Ashraf Russo, and Kang-Hyun Jo, Simple Fine-tuning Attention Modules for Human Pose Estimation, ICCCI, Vietnam, 2020.
- Lihua Wen, Xuan-Thuy Vo, and Kang-Hyun Jo, 3D SaccadeNet: A Single-Shot 3D Object Detector for LiDAR Point Clouds, ICCAS, Korea, 2020.
- Xuan-Thuy Vo, and Kang-Hyun Jo, Enhanced Feature Pyramid Networks by Feature Aggregation Module and Refinement Module, HSI, Japan, 2020.
- Xuan-Thuy Vo, Lihua Wen, Tien-Dat Tran, and Kang-Hyun Jo, Bidirectional Non-local Networks for Object Detection, ICCCI, Vietnam, 2020.

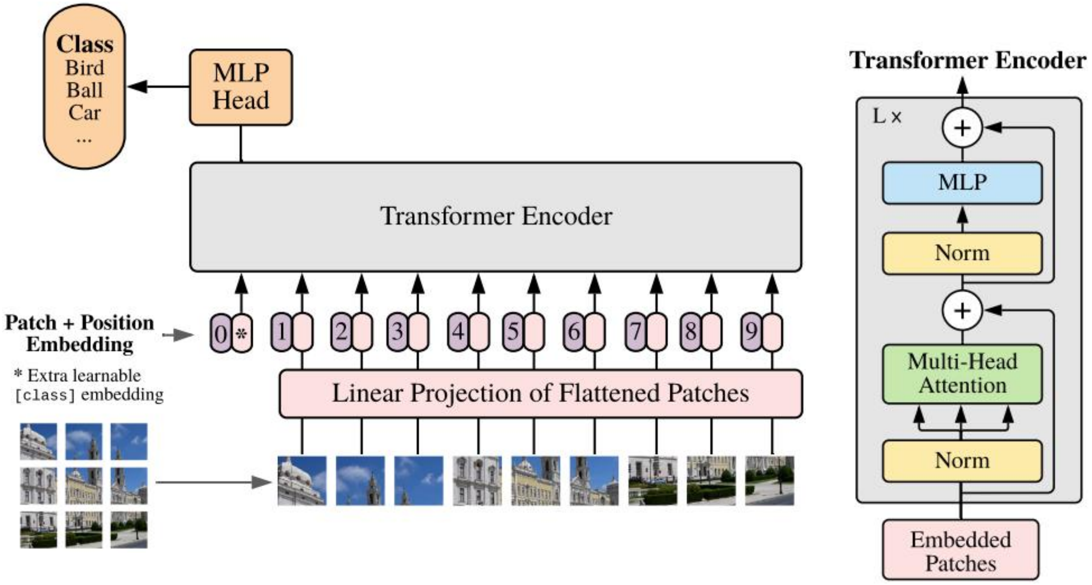
Dominant Networks in Computer Vision: powered architectures ranging from AlexNet, ResNet, ViT, to Swin, and all the modern vision backbones



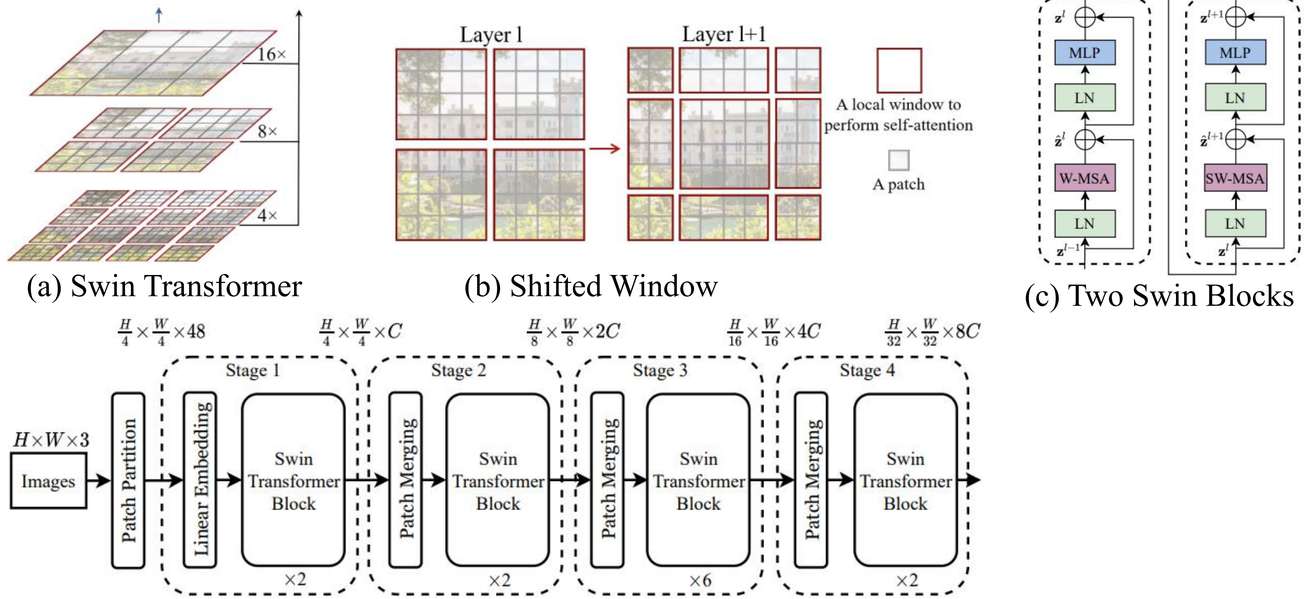
AlexNet



ResNet



Vision Transformer (ViT)



(a) Swin Transformer (b) Shifted Window (c) Two Swin Blocks

(d) Swin Transformer architecture

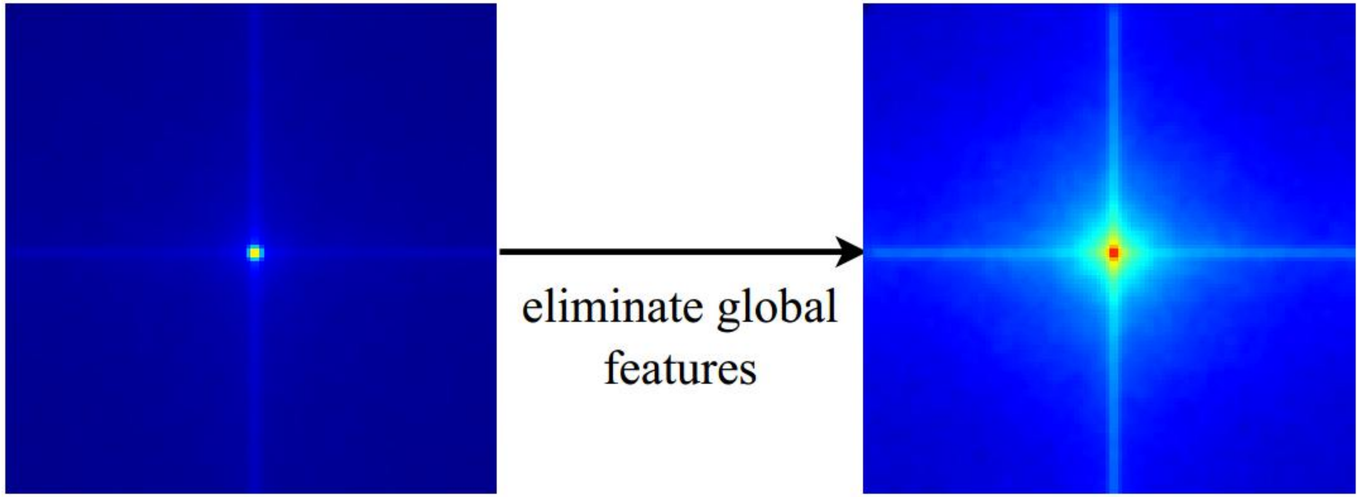
Swin Transformer

EMSNet - Mean()

▶ Elimination of global features for static branch:

- Tend to capture local feature

Amplitude Spectrum:

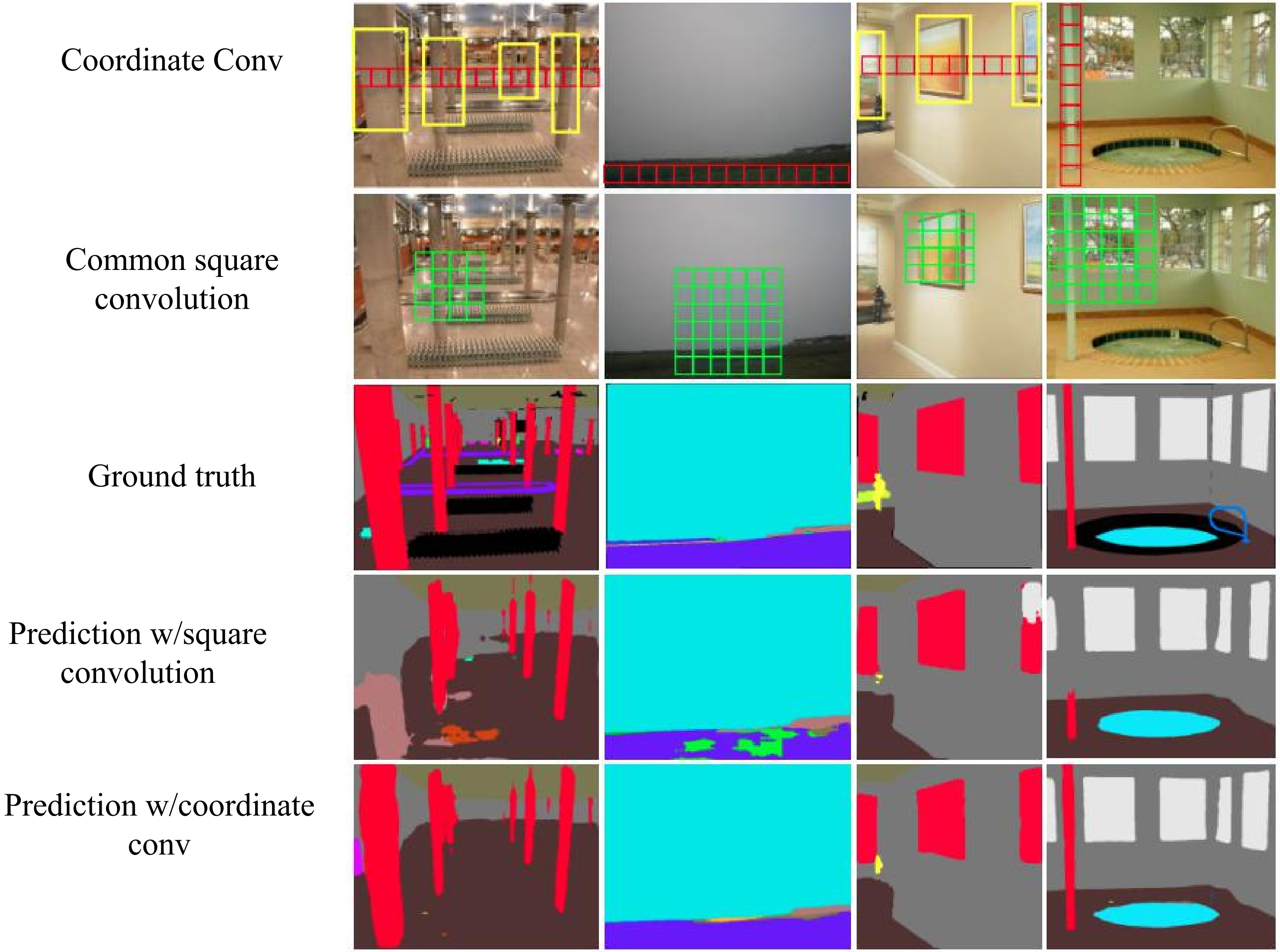


Input feature $\mathbf{X}_s$ $\qquad$ $\mathbf{X}_s - \mathrm{Mean}(\mathbf{X}_s)$

$$\mathrm{Mean}(\mathbf{X}_s) = \frac{1}{H * W} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{X}_s[i, j]$$

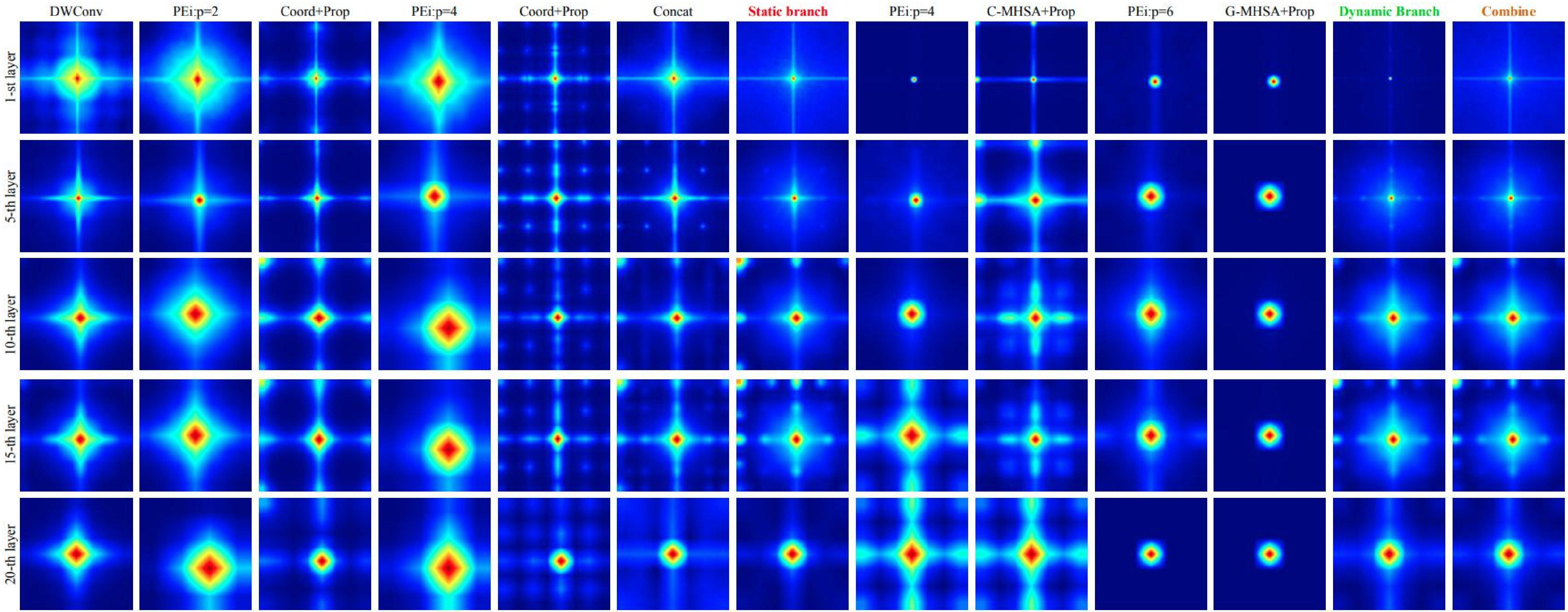
EMSNet - Coordinate Conv

- Full capture semantic information of strip objects:



Type	#params	GFLOPs
Square convolution	$k^2 \times C$	$k^2 \times C \times H \times W$
Coordinate convolution	$2k \times C$	$2 \times k \times C \times H \times W$

Detailed Visualization of EMS Block across Layers



- ✓ Static branch: DWConv, PE, CoordinateDWConv → high frequency components
- ✓ Dynamic branch: G-MHSA → low frequency components
- ✓ Combine → balance the range of frequencies

EMSNet - Addtional Results

TABLE I

ABLATION STUDY ON CHANNEL SPLITTING OF THE STATIC BRANCH

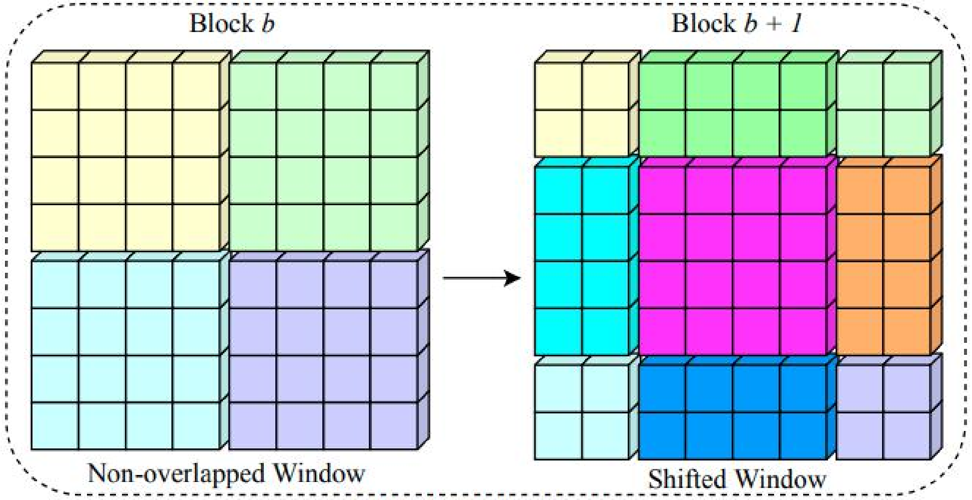
Channel ratios	#params (M)	GFLOPs	Top-1
{1/4:1/2:1/4}	2.56	0.54	73.1
{1/2:1/4:1/4}	2.59	0.55	73.2
{1/4:1/4:1/2}	2.59	0.55	73.2

TABLE III

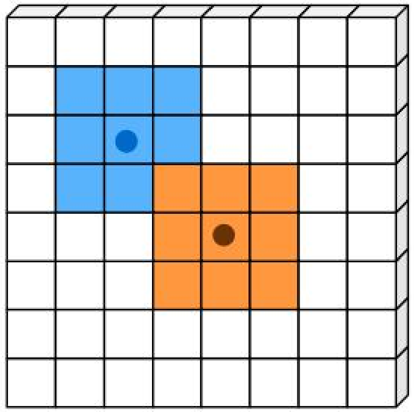
LATENCY OF EACH OPERATOR IN THE EMS BLOCK

	Branch	Operations	#p (M)	G	Latency (ms) CPU	Latency (ms) GPU	Top-1 (%)
Baseline	Identity()		2.09	0.48	33.2	0.24	70.2
Channel Splitting	Static branch	CoordDW	2.68	0.59	43.6	0.41	71.9
	Dynamic branch	+C-MHSA	3.10	0.55	45.4	0.39	72.7
		+G-MHSA	2.56	0.54	50.1	0.43	73.1
w/o fusion	All		2.12	0.52	43.8	0.41	72.2
w/o channel splitting	All		4.45	0.81	90.3	0.82	74.3

Window shifting and sliding



(c) Swin Transformer


(d) Slide Attention

Method	Information exchange across windows	Implementation	Test GPU Mem. (MB)	Latency(ms) CPU	Latency(ms) GPU	P(M)	G	Top-1 (%)
Swin(baseline)	window shifting	torch.roll()	8880	67.3	0.76	4.4	0.7	74.4
HaloNet	window sliding	Unfold() & Padding	16934	83.7	1.03	4.4	0.7	75.8
MAT(Ours)	**mixing abstract token**	**Q,K,V Matrix Mul**	**8482**	52.1(**-15.2↓**)	0.23(**-0.53↓**)	**10.8**	**0.7**	79.0(**+4.4↑**)

Accuracy & Memory Access - MAT

Method	#param ↓ (M)	GFLOPs↓	Test GPU Mem. (MB)↓	Latency(ms) CPU↓	Latency(ms) GPU↓	Top-1 (%) ↑
PVT-T	13.2	1.6	21236	74.79	0.55	75.1
PVT-S	24.5	3.8	21370	214.13	0.93	79.8
PVTv2-B0	3.7	0.6	14774	37.36	0.17	70.5
PVTv2-B1	13.1	2.1	24370	90.10	0.32	78.7
PVTv2-B2	25.4	4.0	24424	164.20	0.55	82.0
Swin-0.7G	4.4	0.7	8880	67.30	0.76	74.4
Swin-1G	7.3	1.0	14926	78.37	0.37	77.3
Swin-2G	12.8	2.0	18060	118.61	0.45	79.2
Swin-T	28.3	4.5	24408	188.68	0.60	81.3
MAT-1	6.7	0.4	6604	40.71	0.19	76.3
MAT-2	10.8	0.7	8482	52.10	0.23	79.0
MAT-3	17.0	1.0	8996	73.91	0.32	80.2
MAT-4	29.1	1.9	10812	120.35	0.47	81.0
MAT-5	50.1	3.2	13801	164.78	0.57	81.9

Image Classification Results - MAT

Comparison of MAT 1-3 and other efficent methods:

Method	Image Size	#param (M)	FLOPs G	Top-1 (%)
MobileViTv1-XXS	256^2	1.3	0.4	69.0
MobileViTv2-0.5	256^2	1.4	0.5	70.2
EMO-1M	224^2	1.3	0.3	71.5
EfficientViT-M4	224^2	8.8	0.3	74.3
EfficientViT-M5	224^2	12.4	0.5	77.1
PVTv2-B0	224^2	3.7	0.6	70.5
Swin-0.7G	224^2	4.4	0.7	74.4
DFvT-T	224^2	4.0	0.3	73.0
MobileViTv1-XS	256^2	2.3	1.0	74.8
MobileViTv2-0.75	256^2	2.9	1.0	75.6
EdgeViT-XXS	256^2	4.1	0.6	74.4
tiny-MOAT-0	224^2	3.4	0.8	75.5
EMO-2M	224^2	2.3	0.4	75.1
ConvNext-XT	224^2	7.4	0.6	77.5

Method	Image Size	#param (M)	FLOPs G	Top-1 (%)
MobileFormer	256^2	1.3	0.4	69.0
VAN-B0	256^2	1.4	0.5	70.2
LVT	224^2	1.3	0.3	71.5
Swin-1G	224^2	8.8	0.3	74.3
EMO-5M	224^2	12.4	0.5	77.1
DFvT-S	224^2	4.4	0.7	74.4
MAT-1 (Ours)	**224^2**	**6.7**	**0.4**	**76.3**
MAT-2 (Ours)	**224^2**	**10.8**	**0.7**	**79.0**
MAT-3 (Ours)	**224^2**	**17.0**	**1.0**	**80.2**

Image Classification Results - MAT

Comparison of MAT 4-5 and recent methods:

Method	Image Size	#param (M)	FLOPs G	Top-1 (%)
PVT-T	224^2	13.2	1.6	75.1
tiny-MOAT-1	224^2	5.1	1.2	78.3
ResT-Lite	224^2	10.5	1.4	77.2
ResT-Small	224^2	13.7	1.9	79.6
EdgeViT-XS	256^2	6.7	1.1	77.5
MobileViTv1-S	256^2	5.6	2.0	78.4
MobileViTv2-1.0	256^2	4.9	1.9	78.1
PoolFormer-S12	224^2	11.9	1.8	77.2
Slide-PVT-T	224^2	12.2	2.0	78.0
PVTv2-B1	224^2	13.1	2.1	78.7
Slide-PVTv2-B1	224^2	13.0	2.2	79.5
Swin-2G	224^2	12.8	2.0	79.2
MAT-4 (Ours)	**224^2**	**29.1**	**1.9**	**81.0**

Method	Image Size	#param (M)	FLOPs G	Top-1 (%)
PoolFormer-S24	224^2	21.3	3.4	80.3
ParC-Net-S	256^2	5.0	3.5	78.6
PVT-S	224^2	24.5	3.8	79.8
ResT-Base	224^2	30.3	4.3	81.6
LITv1-Ti	224^2	19.0	3.6	81.1
LITv1-S	224^2	27.0	4.1	81.5
LITv2-S	224^2	28.0	3.7	82.0
ConvNeXt-T	224^2	28.0	4.5	82.1
Swin-T	224^2	28.3	4.5	81.3
MAT-5 (Ours)	**224^2**	**50.1**	**3.2**	**81.9**

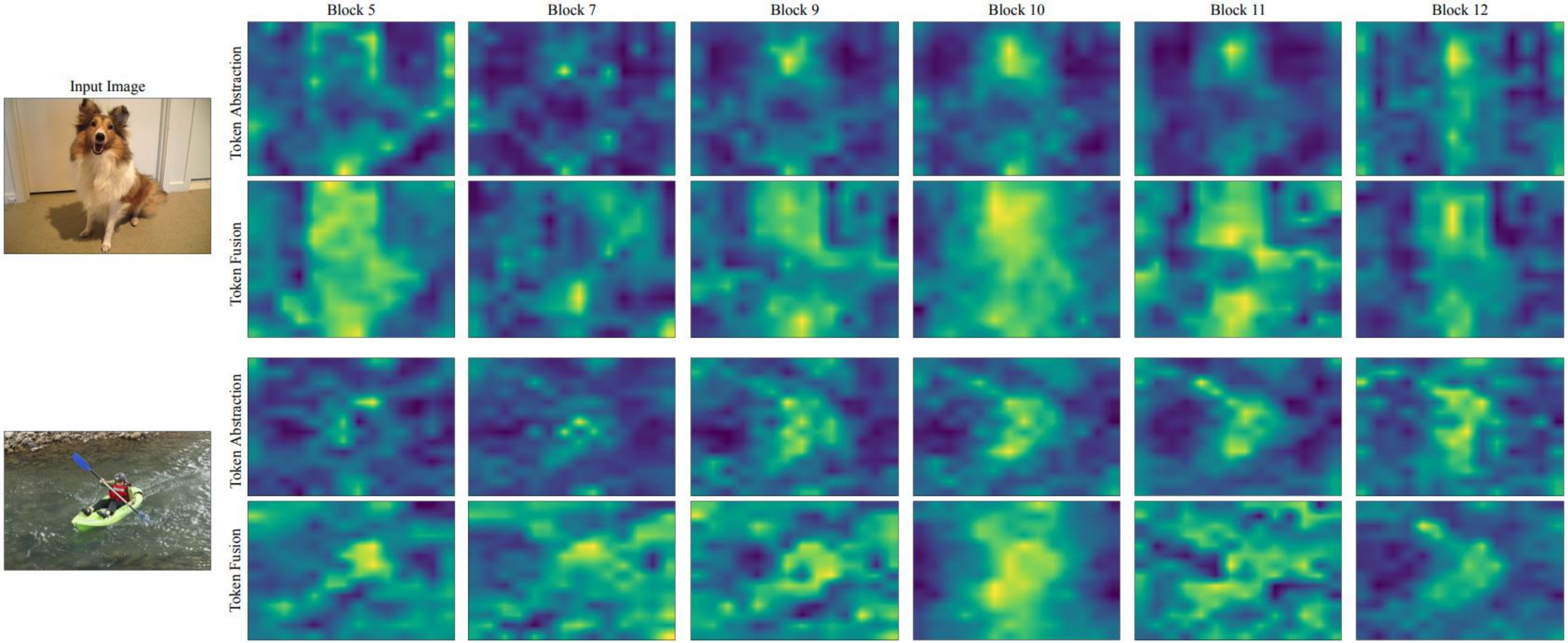
Object Detection and Instance Segmentation Results - MAT

- **Baseline: SSD [11]**

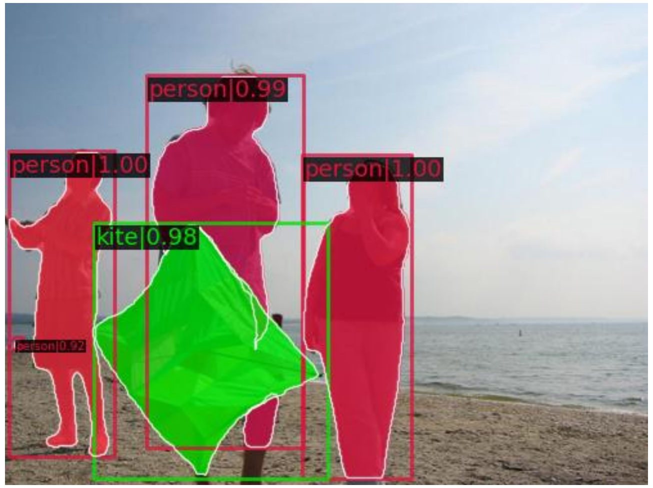
Backbone	#params (M)	GFLOPs	AP^{box}
MobileViTv1-XXS	1.7	0.9	19.9
MobileViTv2-0.5	2.0	0.9	21.2
MobileNetv2	4.3	0.8	22.1
EMO-1M	2.3	0.6	22.0
EMO-2M	3.3	0.9	25.2
MobileViTv2-0.75	3.6	1.8	24.6
MobileViTv1-S	5.7	3.4	27.7
MobileViTv2-1.25	8.2	4.7	27.8
EMO-5M	6.0	1.8	27.9
MobileViTv2-1.75	14.9	9.0	29.5
ResNet-50	26.6	8.8	25.2
MAT-1 (Ours)	**6.4**	**0.8**	**23.3**
MAT-2 (Ours)	**10.5**	**1.5**	**26.3**
MAT-3 (Ours)	**16.7**	**2.3**	**28.2**

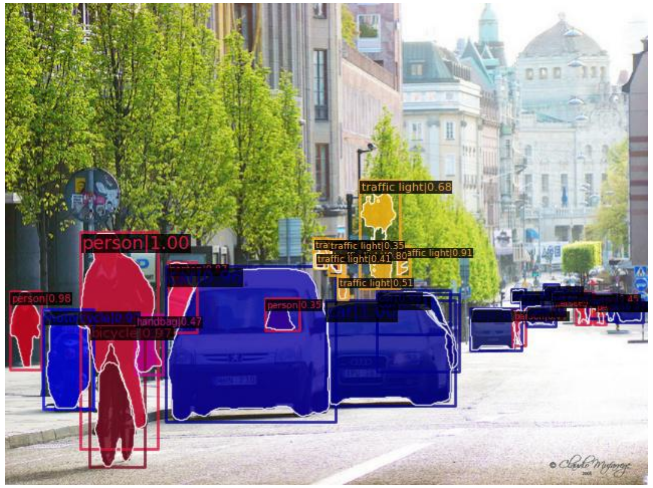
Visualization of Attention Maps - MAT

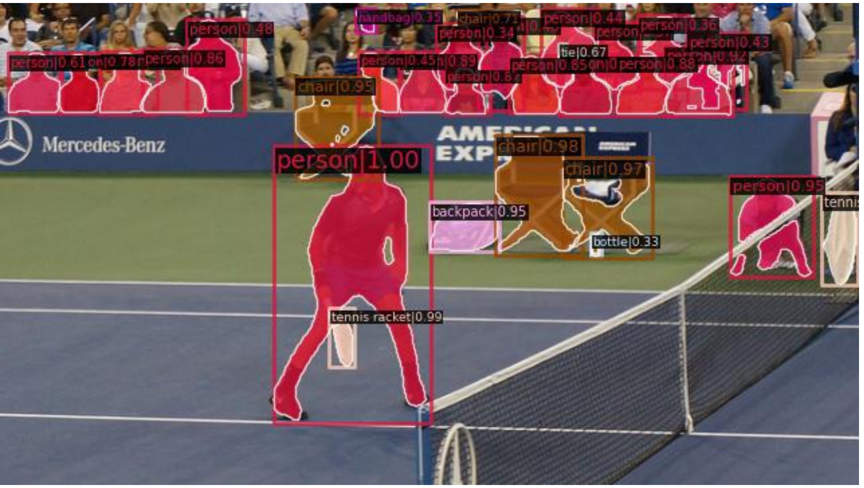
- **Earlier blocks:**
  - abstract tokens → learn object boundaries
  - mixing abstract tokens → larger regions
- **Later blocks:**
  - abstract tokens → capture key parts of objects
  - mixing abstract tokens → focus on target regions



Qualitative Results: Mask R-CNN with MAT-2







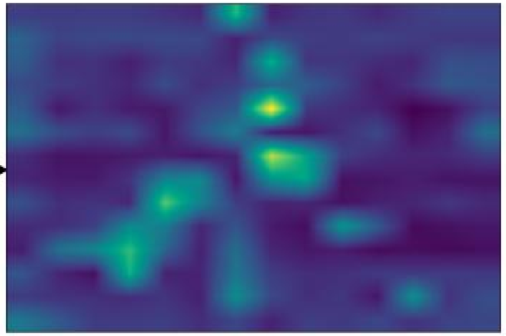
Detailed Configurations of Five MAT Models

Stage	Out	Layer Name	MAT-1	MAT-2	MAT-3	MAT-4	MAT-5
Stem	56^2	Patch Embed	3×3 conv, stride 2, 12 3×3 DWconv, stride 1, 12 3×3 conv, stride 2, 24	3×3 conv, stride 2, 16 3×3 DWconv, stride 1, 16 3×3 conv, stride 2, 32	3×3 conv, stride 2, 18 3×3 DWconv, stride 1, 18 3×3 conv, stride 2, 36	3×3 conv, stride 2, 24 3×3 DWconv, stride 1, 24 3×3 conv, stride 2, 48	3×3 conv, stride 2, 32 3×3 DWconv, stride 1, 32 3×3 conv, stride 2, 64
Stage 1	56^2	Pure MLP	[MLP, exp=4] ×2	[MLP, exp=4] ×2	[MLP, exp=4] ×2	[MLP, exp=4] ×3	[MLP, exp=4] ×2
Stage 2	28^2	Bilinear PE	3×3 DWconv, stride 2, 24 1×1 conv, stride 1, 48 bilinear interpolation	3×3 DWconv, stride 2, 32 1×1 conv, stride 1, 64 bilinear interpolation	3×3 DWconv, stride 2, 36 1×1 conv, stride 1, 72 bilinear interpolation	3×3 DWconv, stride 2, 48 1×1 conv, stride 1, 96 bilinear interpolation	3×3 DWconv, stride 2, 64 1×1 conv, stride 1, 128 bilinear interpolation
		Pure MLP	[MLP, exp=4] ×2	[MLP, exp=4] ×2	[MLP, exp=4] ×2	[MLP, exp=4] ×3	[MLP, exp=4] ×2
Stage 3	14^2	Bilinear PE	3×3 DWconv, stride 2, 48 1×1 conv, stride 1, 96 bilinear interpolation	3×3 DWconv, stride 2, 64 1×1 conv, stride 1, 128 bilinear interpolation	3×3 DWconv, stride 2, 72 1×1 conv, stride 1, 144 bilinear interpolation	3×3 DWconv, stride 2, 96 1×1 conv, stride 1, 192 bilinear interpolation	3×3 DWconv, stride 2, 128 1×1 conv, stride 1, 256 bilinear interpolation
		Transformer	$\begin{bmatrix} \text{MATAttn} & h=12 \\ \text{MLP} & exp=4 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{MATAttn} & h=8 \\ \text{MLP} & exp=4 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{MATAttn} & h=8 \\ \text{MLP} & exp=4 \end{bmatrix} \times 8$	$\begin{bmatrix} \text{MATAttn} & h=12 \\ \text{MLP} & exp=4 \end{bmatrix} \times 8$	$\begin{bmatrix} \text{MATAttn} & h=16 \\ \text{MLP} & exp=4 \end{bmatrix} \times 8$
Stage 4	7^2	Bilinear PE	3×3 DWconv, stride 2, 96 1×1 conv, stride 1, 192 bilinear interpolation	3×3 DWconv, stride 2, 128 1×1 conv, stride 1, 256 bilinear interpolation	3×3 DWconv, stride 2, 144 1×1 conv, stride 1, 288 bilinear interpolation	3×3 DWconv, stride 2, 192 1×1 conv, stride 1, 384 bilinear interpolation	3×3 DWconv, stride 2, 256 1×1 conv, stride 1, 512 bilinear interpolation
		Transformer	$\begin{bmatrix} \text{MATAttn} & h=24 \\ \text{MLP} & exp=4 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{MATAttn} & h=16 \\ \text{MLP} & exp=4 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{MATAttn} & h=16 \\ \text{MLP} & exp=4 \end{bmatrix} \times 8$	$\begin{bmatrix} \text{MATAttn} & h=24 \\ \text{MLP} & exp=4 \end{bmatrix} \times 8$	$\begin{bmatrix} \text{MATAttn} & h=32 \\ \text{MLP} & exp=4 \end{bmatrix} \times 8$

$$Attention(Q', K') = softmax\left(\frac{Q' K'^{T}}{\sqrt{d_m/h}}\right)$$

$$A = \begin{array}{c} a_1 \\ \\ \\ \\ \end{array} \begin{bmatrix} q_1 k_1^T & q_1 k_2^T & \dots & q_1 k_N^T \\ q_2 k_1^T & q_2 k_2^T & \dots & q_2 k_N^T \\ q_3 k_1^T & q_3 k_2^T & \dots & q_3 k_N^T \\ q_4 k_1^T & q_4 k_2^T & \dots & q_4 k_N^T \end{bmatrix}$$

$$\boxed{similarity = \frac{\langle a_i, a_j \rangle}{\|a_i\|_2 * \|a_j\|_2}}, i \neq j$$



Attention map

PartialFormer - Cosine Similarity

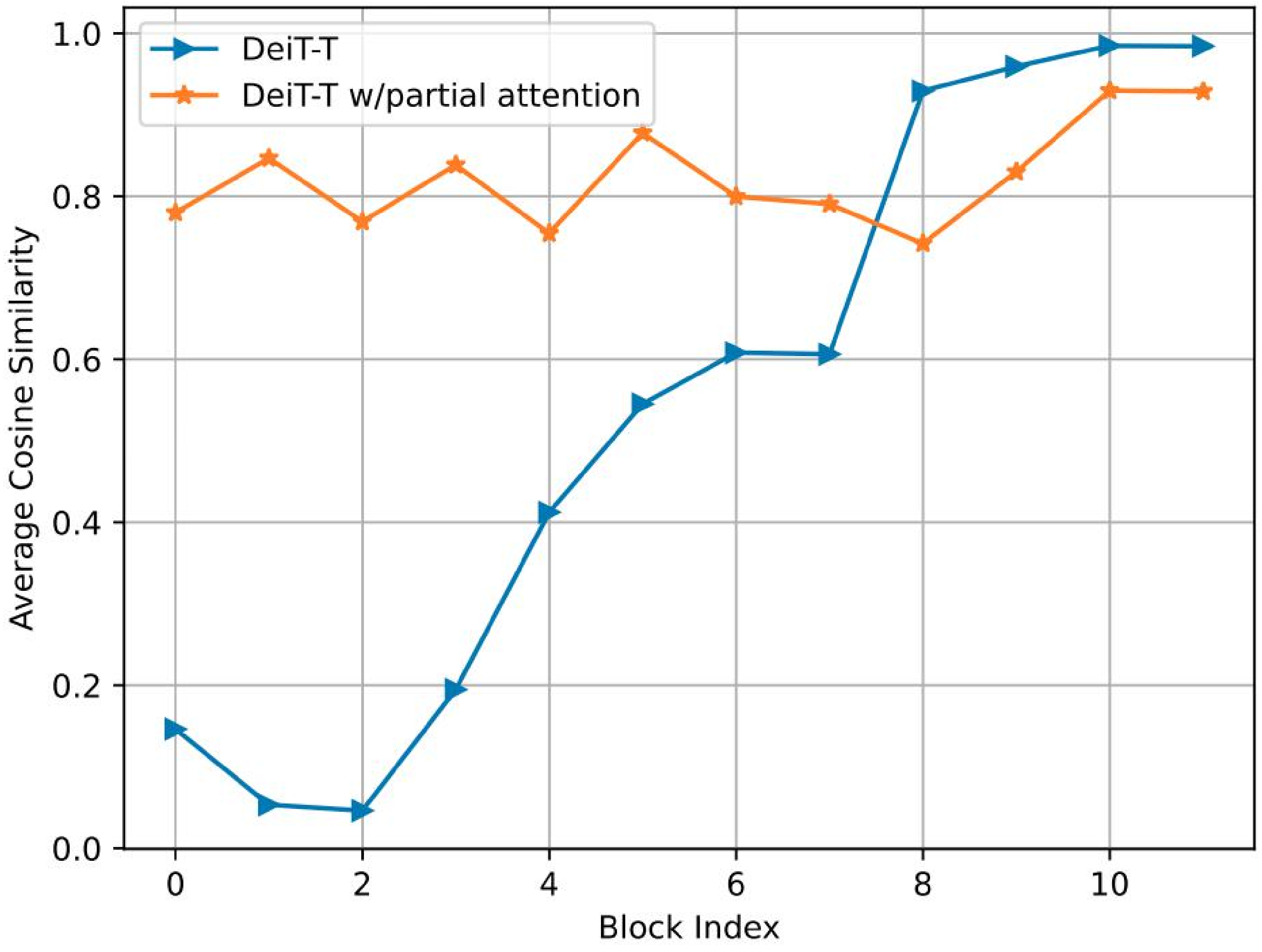


Image Classification Results - PartialFormer

▶ PartialFormer B0-B2 and recent methods:

Method	Im. Size	GFLOPs	Params	Top-1(%)
MobileViTv1-XXS	256^2	0.4	1.3M	69.0
MobileViTv2-0.5	256^2	0.5	1.4M	70.2
PVTv2-B0	224^2	0.6	3.7M	70.5
DFvT-T	224^2	0.3	4.0M	73.0
EfficientViT-M4	224^2	0.4	8.8M	74.3
EdgeViT-XXS	256^2	0.6	4.1M	74.4
SwiftFormer-XS	224^2	0.6	3.5M	75.7
PartialFormer-B0	**224^2**	**0.4**	**5.3M**	**76.7**
DeiT-T	224^2	1.3	6.0M	72.2
LVT	224^2	0.9	3.4M	74.8
ConvNeXtV1-A	224^2	0.6	3.7M	75.7
ConvNeXtV2-A	224^2	0.6	3.7M	76.2
ResT-Lite	224^2	1.4	10.5M	77.2
SwiftFormer-S	224^2	1.0	6.1M	78.5
PartialFormer-B1	**224^2**	**0.7**	**8.2M**	**79.3**

Method	Im. Size	GFLOPs	Params	Top-1(%)
PVT-T	224^2	1.8	13.0	75.1
Slide-PVT-T	224^2	2.0	12.2	78.0
PVTv2-B1	224^2	2.1	13.1	78.7
Swin-2G	224^2	2.0	12.8	79.2
ResT-Small	224^2	1.9	13.7	79.6
Shunted-T	224^2	2.1	11.5	79.8
GC ViT-XXT	224^2	2.1	12.0	79.9
QuadTree-B1	224^2	2.3	13.6	80.0
ConvNeXtV1-N	224^2	2.5	15.6	80.8
SwiftFormer-T	224^2	1.6	12.1	80.9
tiny-MOAT-2	224^2	2.3	9.8	81.0
EdgeViT-S	256^2	1.9	11.1	81.0
ConvNeXtV2-N	224^2	2.5	15.6	81.2
BiFormer-T	224^2	2.2	13.1	81.4
PartialFormer-B2	**224^2**	**1.9**	**21.1**	**82.0**

Image Classification Results - PartialFormer

PartialFormer B3-B4 and recent methods:

Method	Im. Size	GFLOPs	Params	Top-1(%)
ResNet-50	224^2	4.1	26	76.1
PVT-S	224^2	3.8	25	79.8
DeiT-S	224^2	4.6	22	79.9
PaCa-Tiny	224^2	3.2	12	80.9
Swin-T	224^2	4.5	29	81.3
LIT-S	224^2	4.1	27	81.5
ResT-Base	224^2	4.3	30	81.6
Slide-PVT-S	224^2	4.0	23	81.7
PVTv2-B2	224^2	4.0	25	82.0
DAT-T	224^2	4.5	29	82.0
LITv2-S	224^2	3.7	28	82.0
ConvNeXt-T	224^2	4.5	29	82.1
Focal-T	224^2	4.9	29	82.2
ResTv2-T	224^2	4.1	30	82.3
PartialFormer-B3	**224^2**	**3.4**	**36**	**83.0**

Method	Im. Size	GFLOPs	Params	Top-1(%)
PVT-M	224^2	6.7	44	81.2
PVT-L	224^2	9.8	61	81.7
Swin-S	224^2	8.7	50	83.0
Twins-SVT-B	224^2	8.6	56	83.2
PVTv2-B3	224^2	6.9	45	83.2
LITv2-M	224^2	7.5	49	83.3
Focal-S	224^2	9.1	51	83.6
CSWin-S	224^2	6.9	35	83.6
DAT-S	224^2	9.0	50	83.6
PVTv2-B4	224^2	10.1	63	83.6
ResTv2-B	224^2	7.9	56	83.7
PartialFormer-B4	**224^2**	**6.8**	**64**	**83.9**

Experimental Setup - Semantic Segmentation

- **Semantic Segmentation:**
  - Dataset: ADE20K
    - 20K training images, 2K validation images
  - Baseline segmentors: Semantic FPN, UperNet
    - Replace original backbone with pretrained MAT Transformers
    - Neck, Head is kept same as baseline
  - Configurations:
    - Iterations: 80K (Semantic FPN), 160K (UPerNet)
    - Batch size: 16
    - Optimizer: Adam
    - Learning rate: $2e^{-4}$
    - Image size: 512×512

Experimental Setup - Object Detection, Instance Segmentation

- **Object Detection and Instance Segmentation:**
  - Dataset: MS-COCO
    - 115K training images, 5K validation images with 80 categories
  - Baseline detectors: SSD, RetinaNet, Mask R-CNN, SimpleBaseline (keypoint detection)
    - Replace original backbone with pretrained MAT Transformers
    - Neck, Head is kept same as baseline
  - Configurations:
    - Epochs: 12 (SSD, RetinaNet, Mask R-CNN), 210 (SimpleBaseline)
    - Batch size: 16 (RetinaNet, Mask R-CNN), 192 (SSD)
    - Optimizer: Adam
    - Learning rate: $1e^{-4}$
    - Image size: 1333×800 (RetinaNet, Mask R-CNN), 320×320 (SSD), 256×192 (SimpleBaseline)

Ablation Study - PartialFormer

▶ Importance of each component in Partial Transformer Block:

Model	Module	#params(M)	GFLOPs	Top-1 Acc
PartialFormer-B1	Full	8.264	0.734	79.3
	-No exchange	8.262	0.734	78.8
	-MMSA	8.261	0.706	77.2
	-SQA	6.213	0.536	76.5

▶ Ratio between N/N_F across 4 stages: only MMSA used in partial attention

Model	N/N_F	#params(M)	GFLOPs	Top-1 Acc
PartialFormer-B1 with MMSA	**[64, 16, 4, 1]**	8.232	0.653	**78.5**
	[4, 4, 4, 4]	8.232	0.699	77.3
	[8, 8, 8, 8]	8.232	0.589	77.1
	[16, 16, 16, 16]	8.232	0.556	77.0
	[64, 32, 16, 8]	8.232	0.553	77.0

Ablation Study - PartialFormer

▶ **Head MLP in MMSA:** only MMSA used in partial attention

Model	Head MLP	#params	GFLOPs	Top-1 Acc
PartialFormer-B1 with MMSA	No	8.230	0.647	78.3
	e=1	8.232	0.653	**78.5**
	e=2	8.239	0.668	78.6

▶ **Channel Reduction in SQA:** only MMSA used in partial attention

Model	Head MLP	#params	GFLOPs	Top-1 Acc
PartialFormer-B1 with SQA	r=1	8.344	0.707	77.4
	r=4	8.289	0.706	77.2
	r=8	8.261	0.706	**77.2**
	r=16	8.248	0.706	77.1

Application of Partial Attention

- **Replace existing attentions with our partial attention:**

Method	#params	GFLOPs	Top-1 Acc
DeiT-T	6.0M	1.3	72.2
DeiT-T with partial attention	5.7M (-0.3M)	0.9 (-0.4)	74.2 **(+2.0)**
PVT-T	13.0M	1.8	75.1
PVT-T with partial attention	11.0M (-2M)	1.6 (-0.2)	77.3 **(+2.2)**

Throughput Comparison - PartialFormer

- **Device:** CPU-Intel(R) Xeon(R) Gold 5220R@2.20GHz; GPU-Tesla V100

Method	Attention	Type	FLOPs G	Params M	Top-1 Acc %	Throughput (images/second)	
						CPU	GPU
PVT	Spatial reduction attention	M	6.7	44	81.2	6.9	1071
		L	9.8	61	81.7	4.6	781
Swin	Window attention	T	4.5	29	81.3	5.2	1665
		S	8.7	50	83.0	3.2	710
Focal	Multi-scale attention	T	4.9	29	82.1	3.4	515
		S	9.1	51	83.6	2.3	316
CSWin	Cross-shaped window attention	T	4.5	23	82.7	7.4	1464
		S	6.9	35	83.6	4.6	907
DAT	Deformable attention	T	4.5	29	82.0	5.6	1176
		S	9.0	50	83.6	3.1	686
PartialFormer(Ours)	**Partial attention**	B3	3.4	36	83.0	5.5	1353
		B4	6.8	64	83.9	3.7	847

References

[1] AlexNet: ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS`2015

[2] ResNet: Deep Residual Learning for Image Recognition, CVPR`2016

[3] ViT: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR`2021

[4] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV`2021

[5] MobileNetV2: Inverted Residuals and Linear Bottlenecks, CVPR`2018

[6] ConvNeXt: A ConvNet for the 2020s, CVPR`2022

[7] PVT: Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions, ICCV`2021

[8] EdgeViTs: Competing Light-weight CNNs on Mobile Devices with Vision Transformers, ECCV`2022

[9] RetinaNet: Focal Loss for Dense Object Detection, ICCV`2017

[10] Mask R-CNN, ICCV`2017

[11] SSD: Single Shot MultiBox Detector, ECCV`2016

[12] Semantic FPN: Panoptic Feature Pyramid Networks, CVPR`2019

[13] CSWin Transformer: A General Vision Transformer Backbone With Cross-Shaped Windows, CVPR`2022

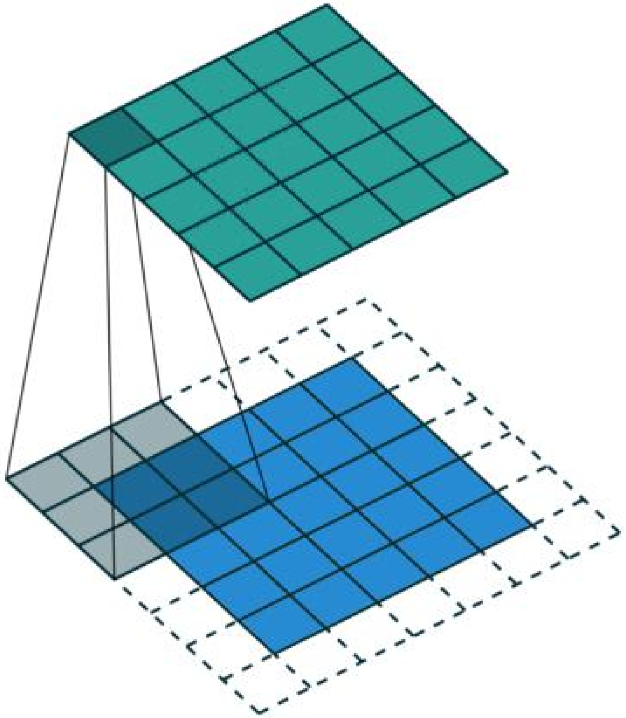
[14] Vision Transformer with Deformable Attention, CVPR`2022

[15] Unified Perceptual Parsing for Scene Understanding, ECCV`2018

[16] Simple Baselines for Human Pose Estimation and Tracking, ECCV`2018

2D Convolution

- **Formula:** $$(f * h)(x, y) = \sum_m \sum_n f(m, n) * h(x - m, y - n)$$
  - f(m, n): the pixel value of the input image at position (m, n)
  - h(x-m, y-n): the value of convolution kernel at shifted positions (x-m, y-n)
  - (f*h)(x, y): the result of convolution at the point (x, y)



https://github.com/vdumoulin/conv_arithmetic/tree/master

2D Convolution

▶ Example:

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	1	1	1	2	1	0
0	2	0	0	2	1	0
0	0	2	2	0	2	0
0	2	2	2	2	0	0
0	0	0	0	2	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	2	1	1	1	0
0	1	0	1	1	0	0
0	1	1	2	0	0	0
0	0	2	2	2	1	0
0	0	0	2	1	0	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	0	1	1	2	1	0
0	2	0	2	2	2	0
0	2	2	2	1	0	0
0	0	1	1	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

1	0	-1
1	-1	1
1	0	1

w0[:,:,1]

-1	-1	0
1	1	-1
1	-1	0

w0[:,:,2]

1	1	-1
1	1	-1
-1	-1	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	0	-1
0	1	0
1	-1	-1

w1[:,:,1]

0	0	-1
-1	-1	1
0	0	-1

w1[:,:,2]

0	-1	-1
0	-1	0
1	-1	1

Bias b1 (1x1x1)

b1[:,:,0]

0

toggle movement

Output Volume (3x3x2)

o[:,:,0]

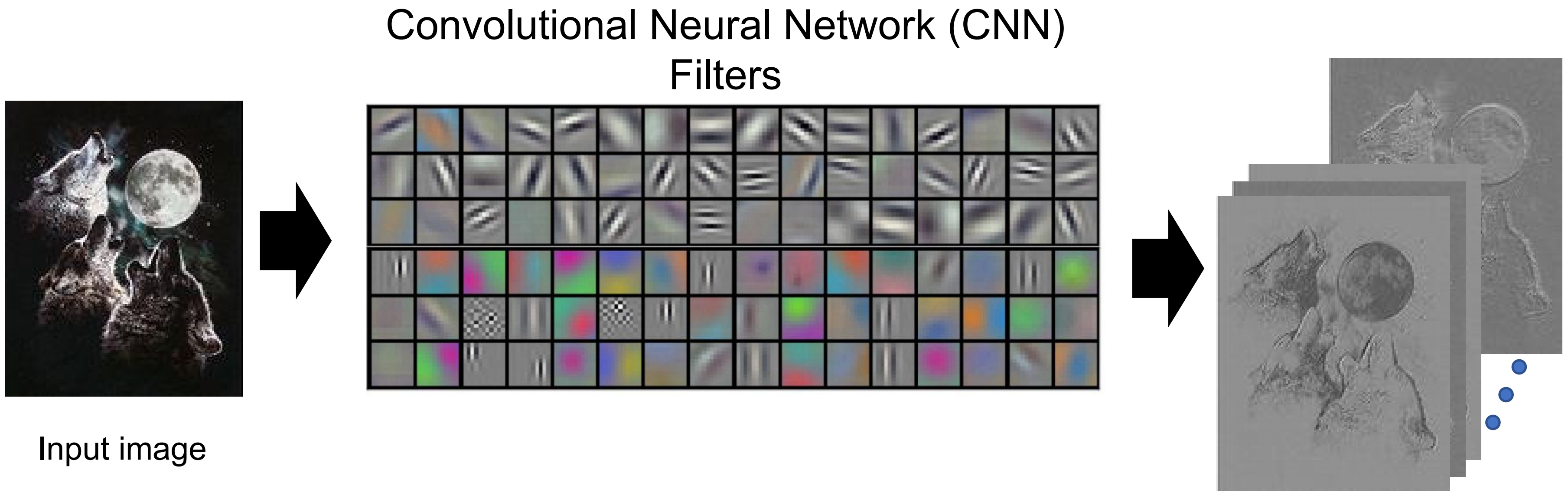
-2	3	6
1	0	9
-2	3	7

o[:,:,1]

0	-3	0
-9	-12	2
-8	-9	-2

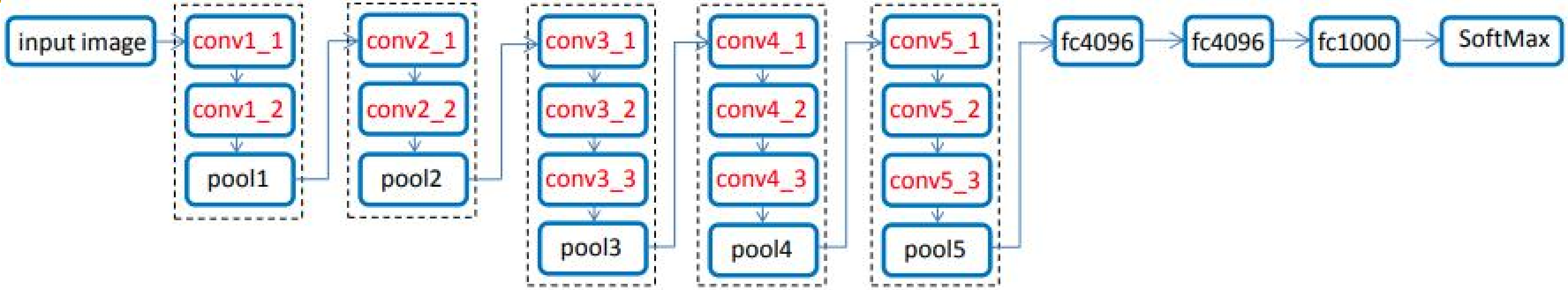
https://cs231n.github.io/convolutional-networks/

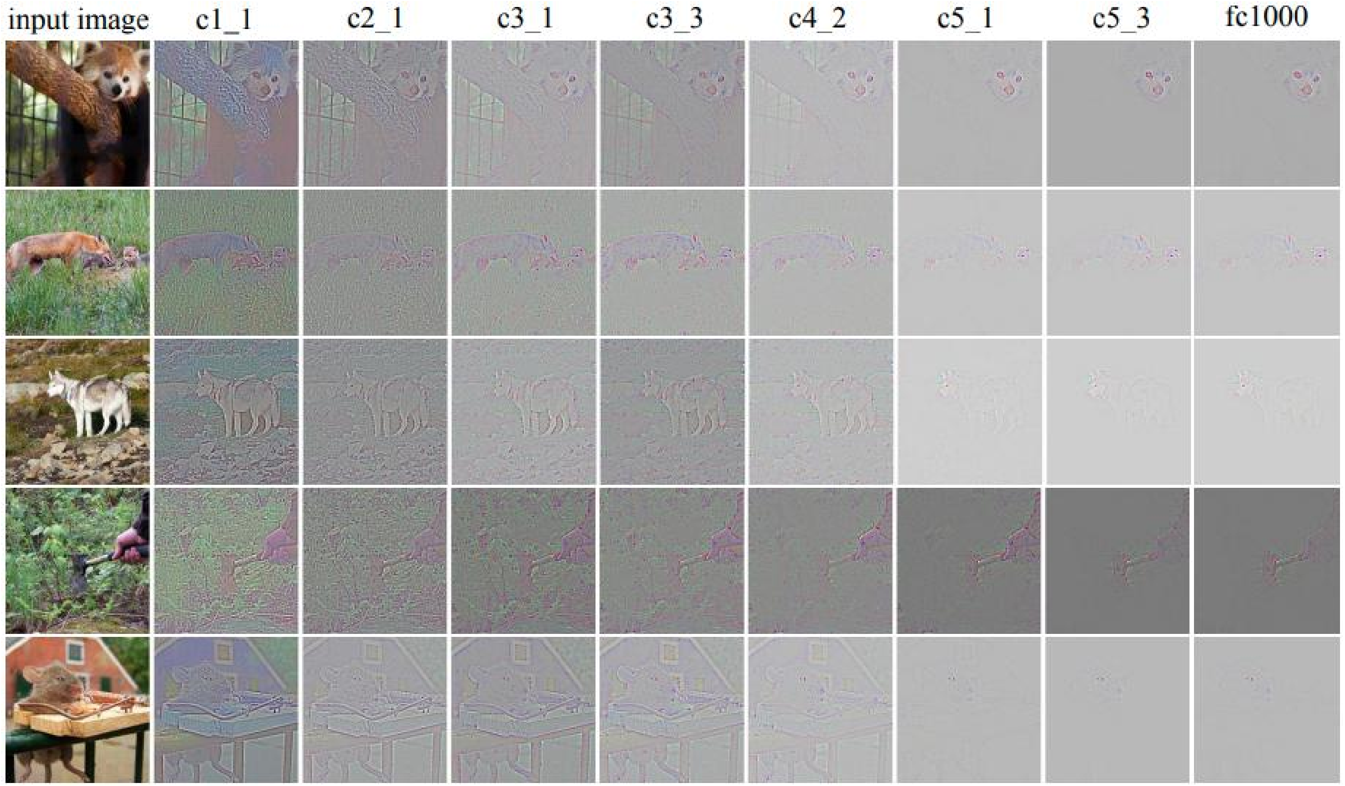
Example of the Convolution Filters



Visualization of the CNN filters→ Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *NIPS* 2012.

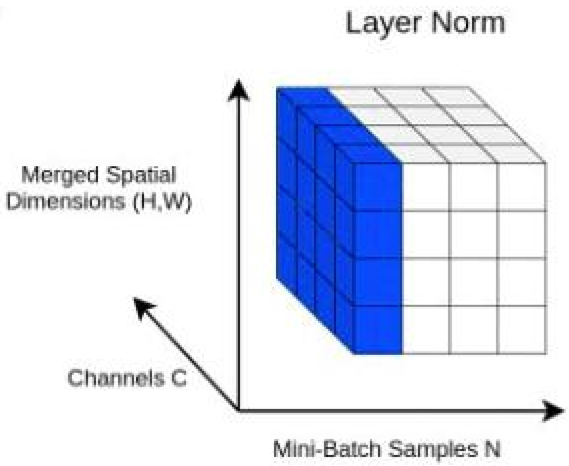
VGG Feature Maps

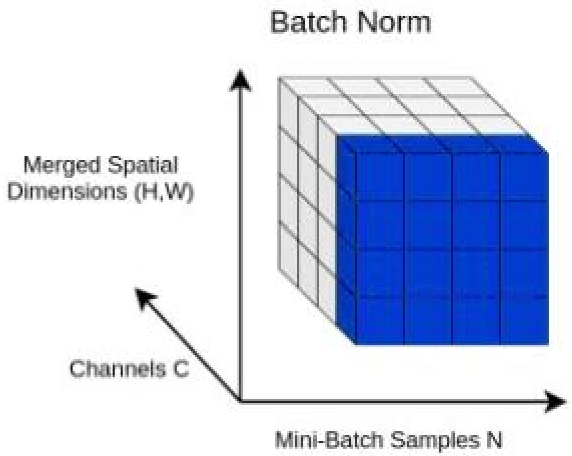




Visualizing and Comparing Convolutional Neural Networks, ICLR`2015

Normalization





Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

https://theaisummer.com/normalization/

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

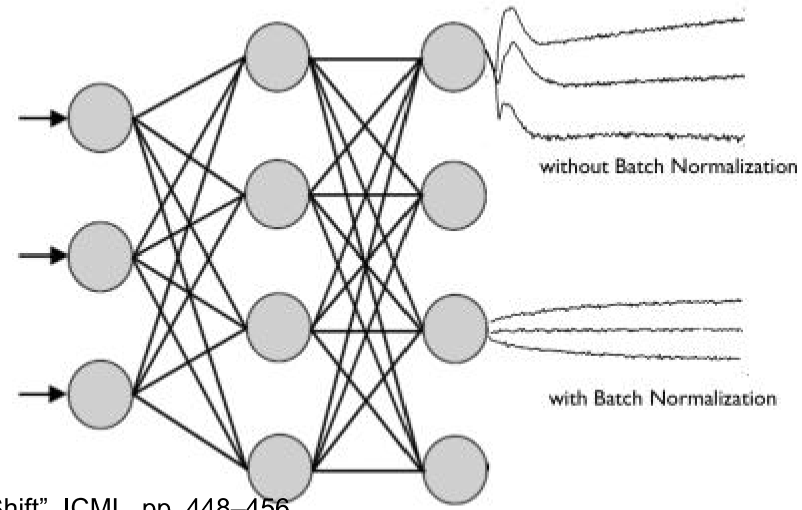
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

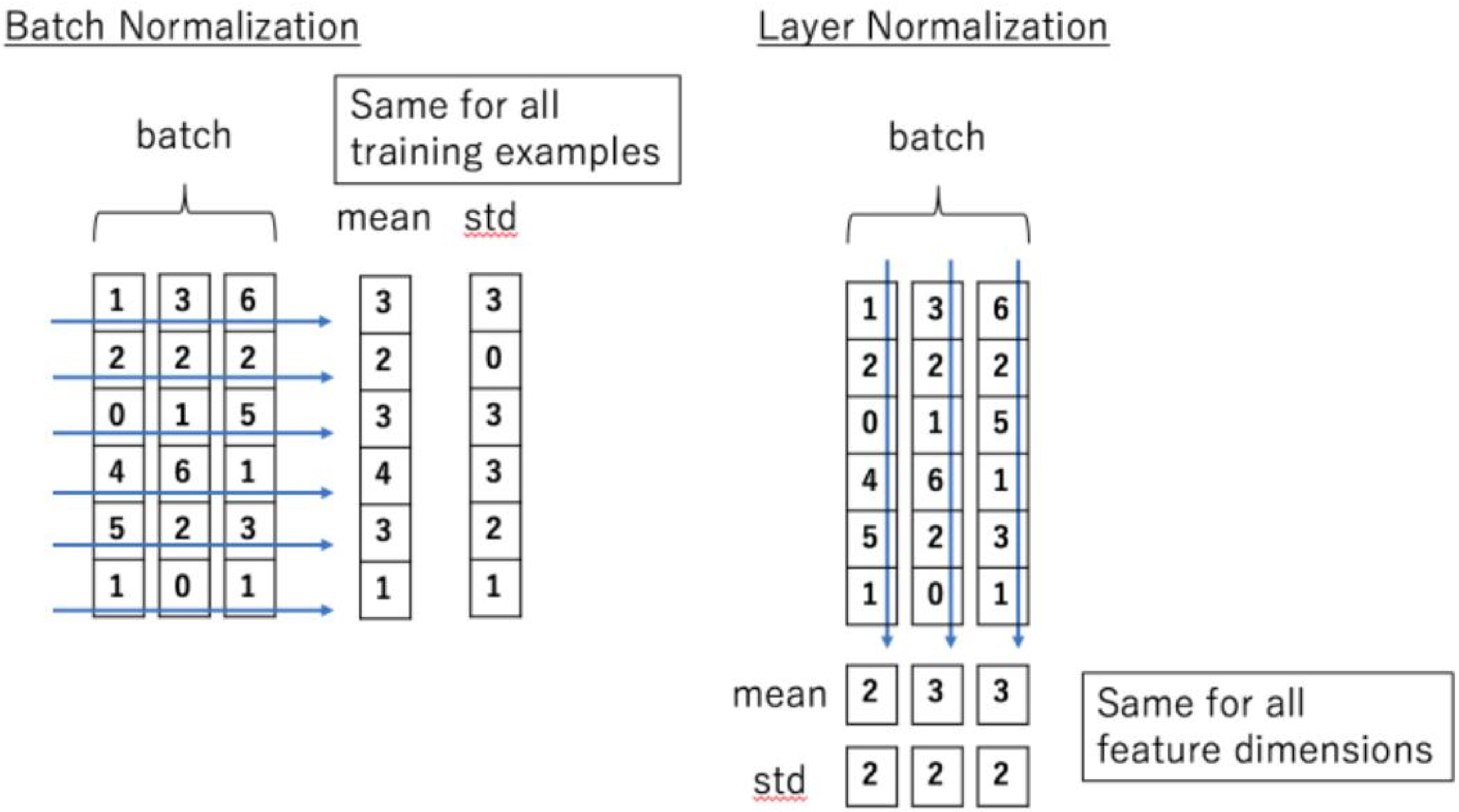
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



http://gradientscience.org/batchnorm/

Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. ICML, pp. 448–456.

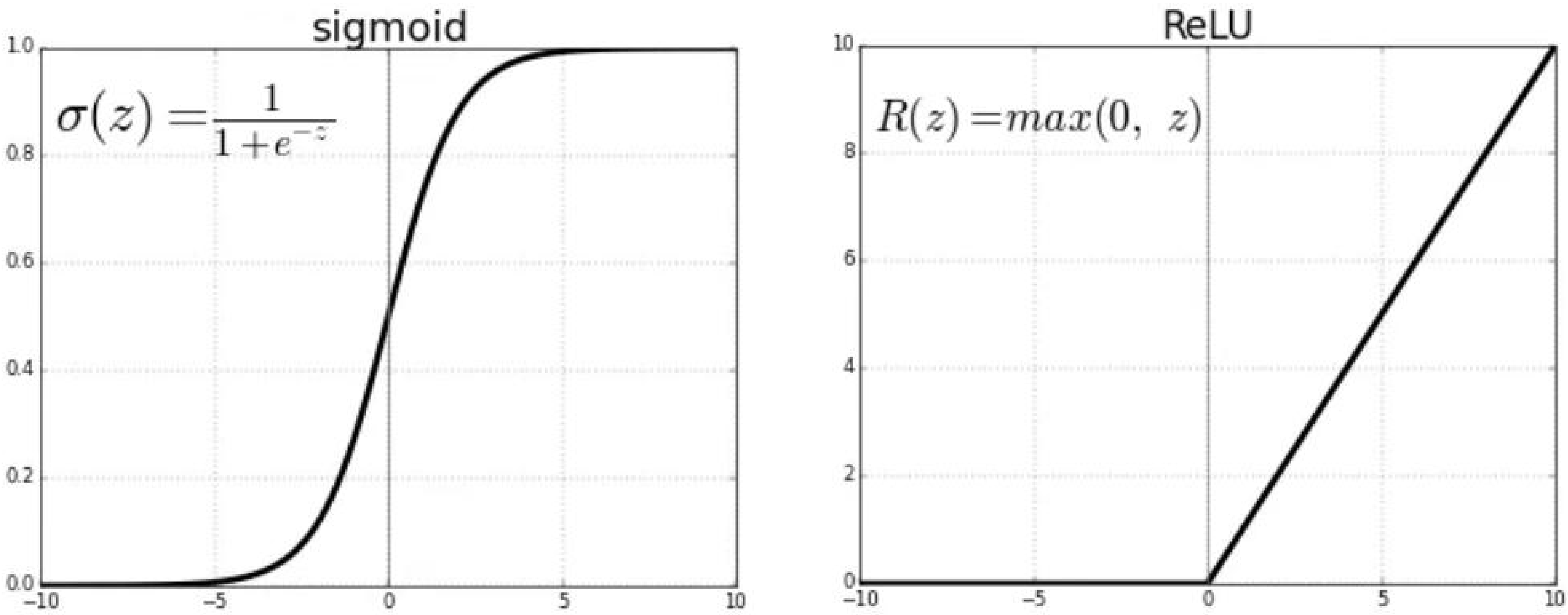
Normalization



https://yonghyuc.wordpress.com/2020/03/04/batch-norm-vs-layer-norm/

Activation Functions

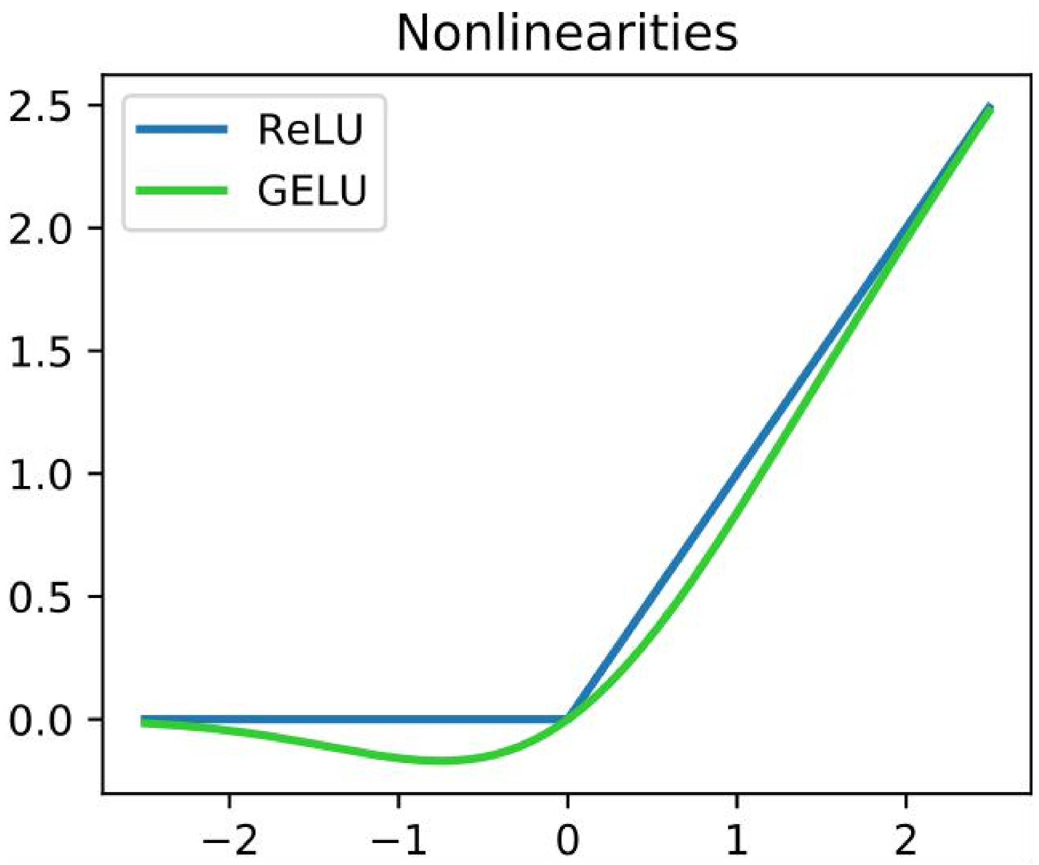
- **Introducing non-linearity into models:**



https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

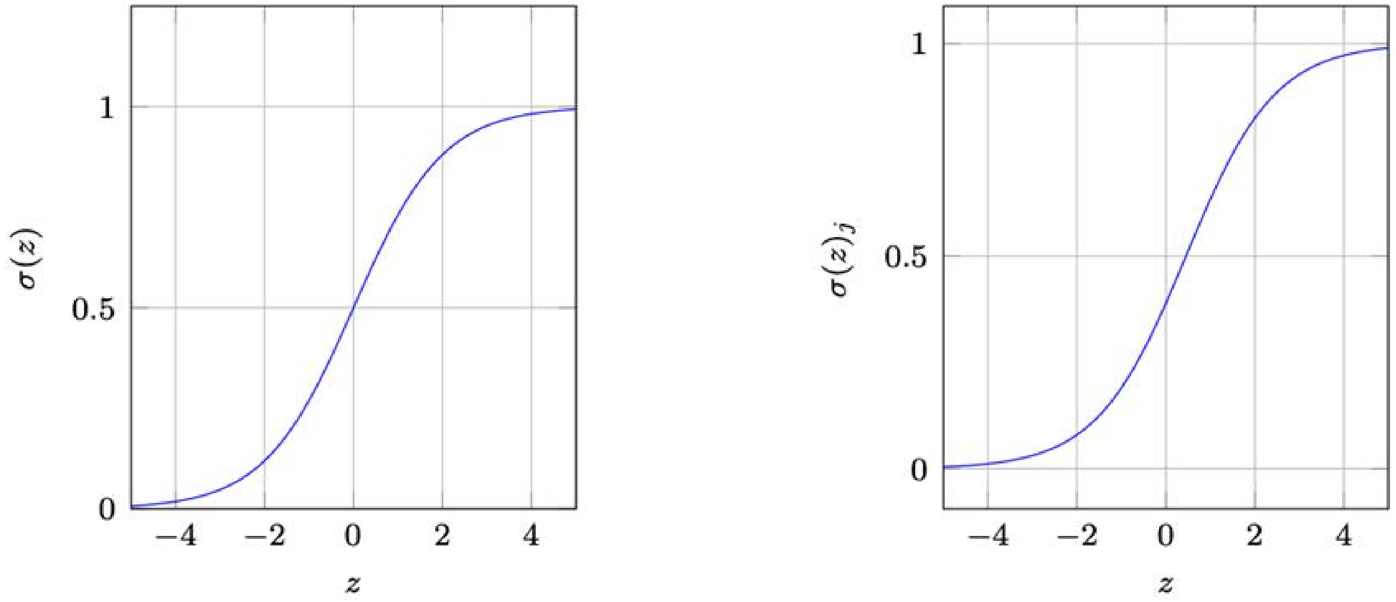
Activation function - GELU

GELU: Gaussian Error Linear Unit



https://en.wikipedia.org/wiki/Rectifier_%28neural_networks%29

$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad \sigma(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

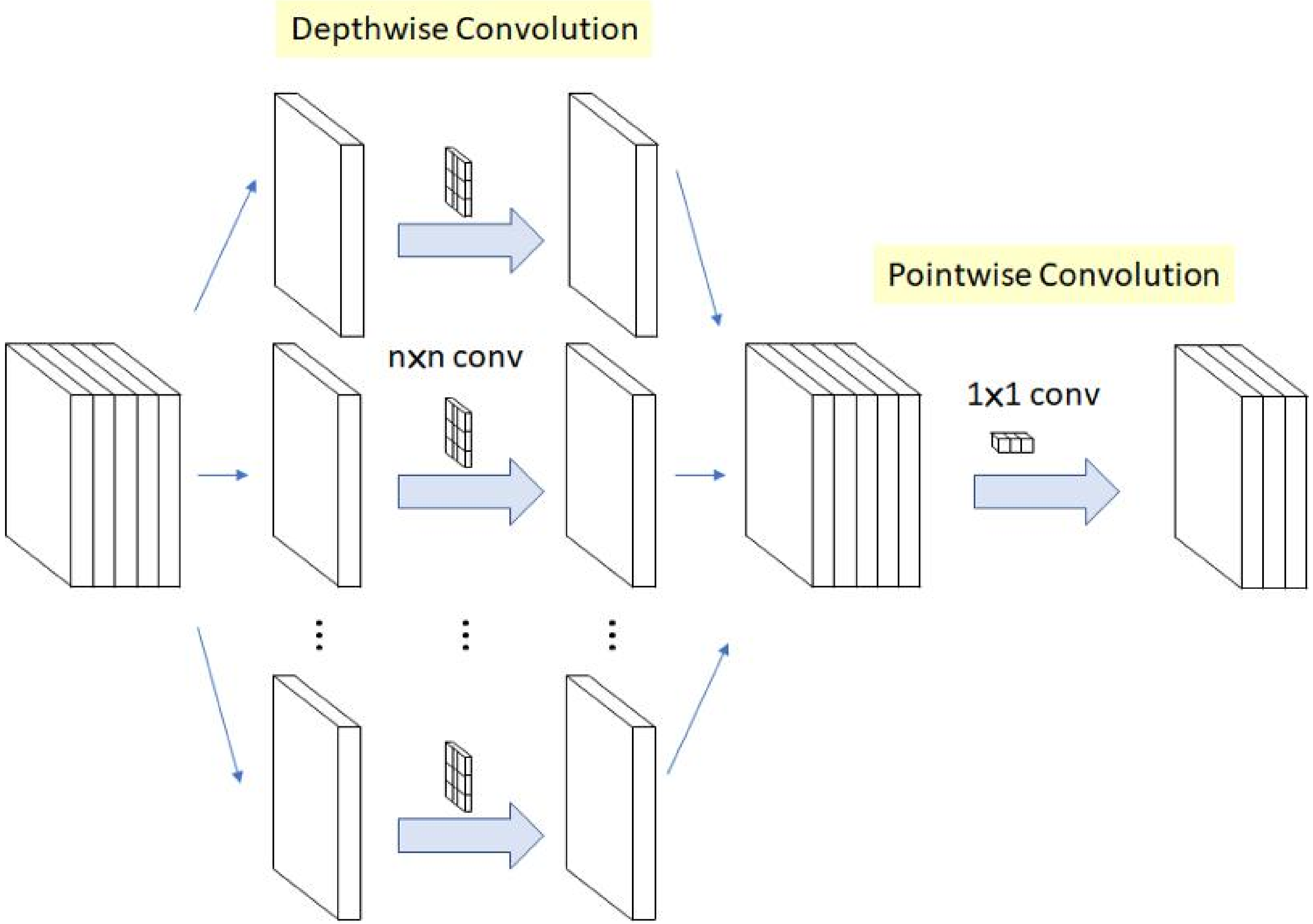


(a) Sigmoid activation function.

(b) Softmax activation function.

Figure 1: Sigmoid and Softmax activation functions

Depth-wise Convolution



https://gaussian37.github.io/dl-concept-dwsconv/

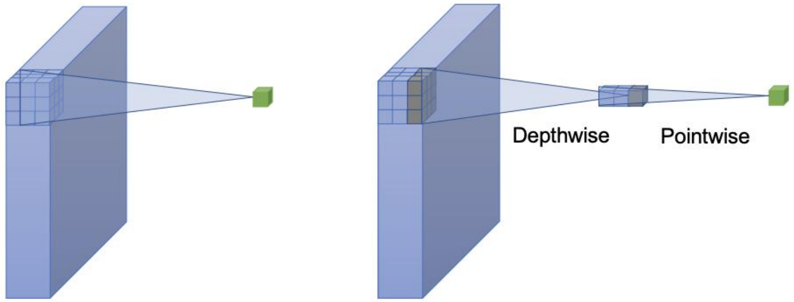
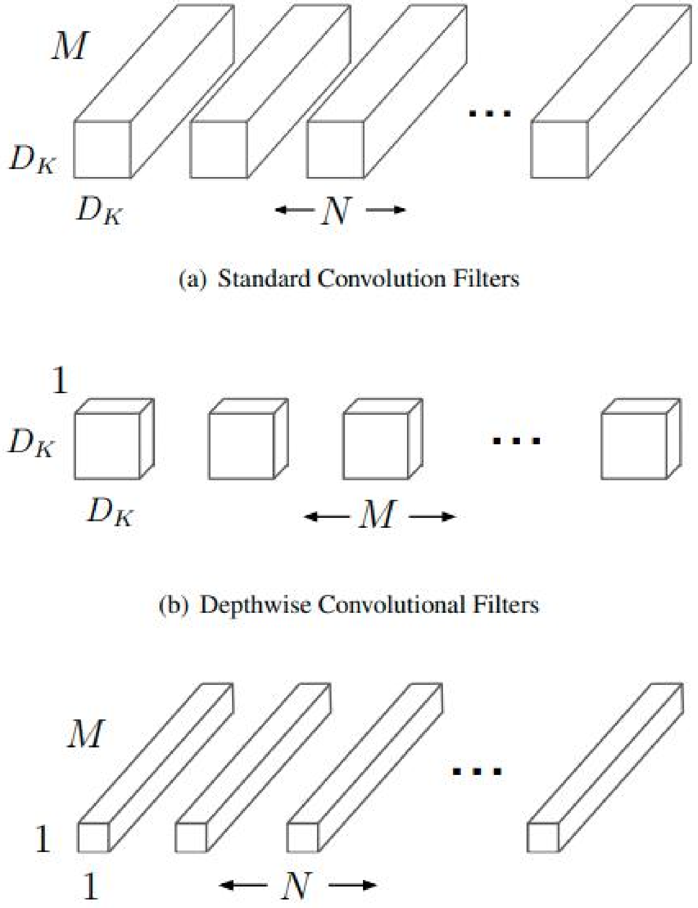


Figure 3: Standard convolution and depthwise separable convolution.



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

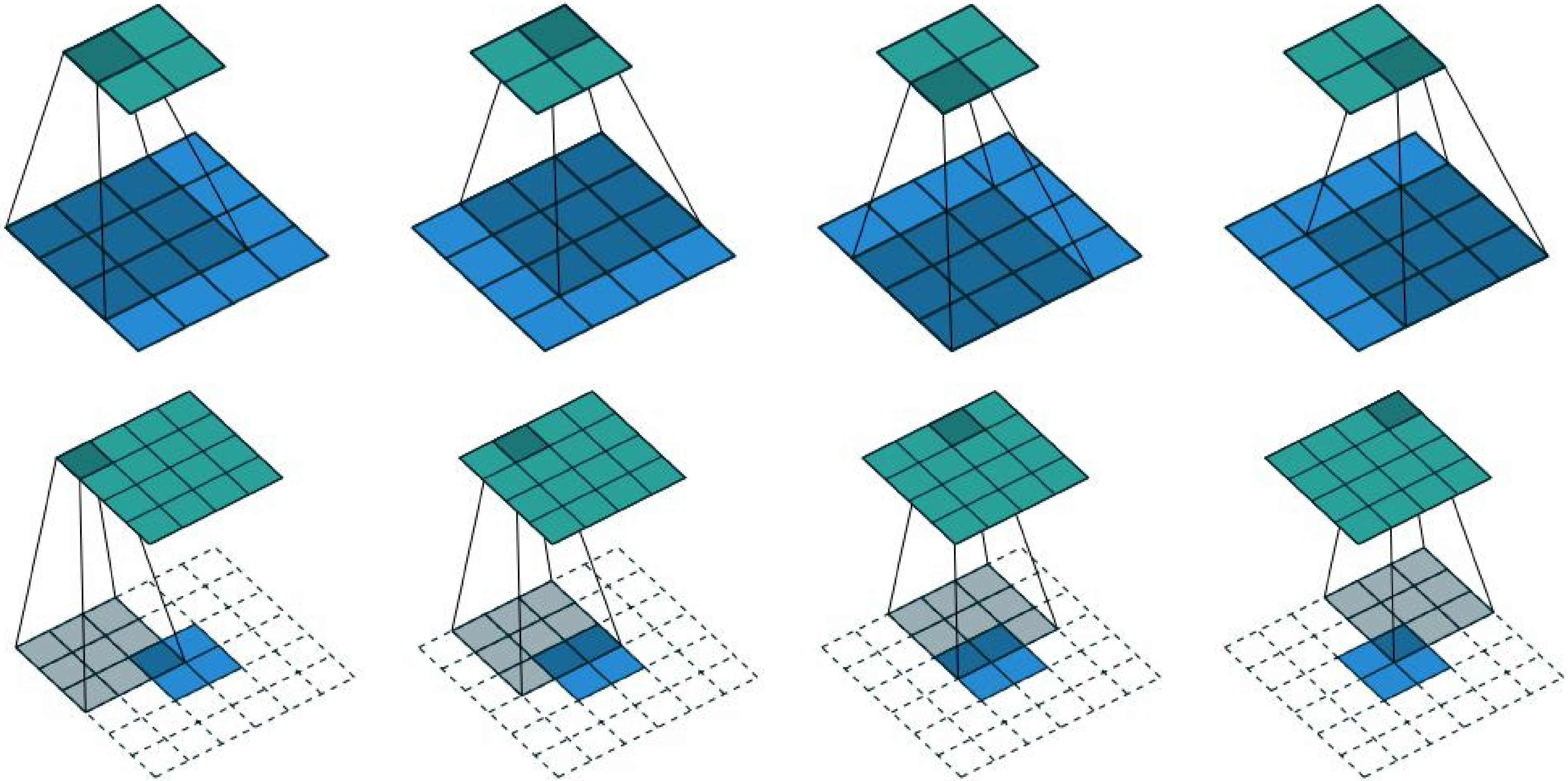
Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

Convolution vs. Transposed Convolution

3×3 convolution

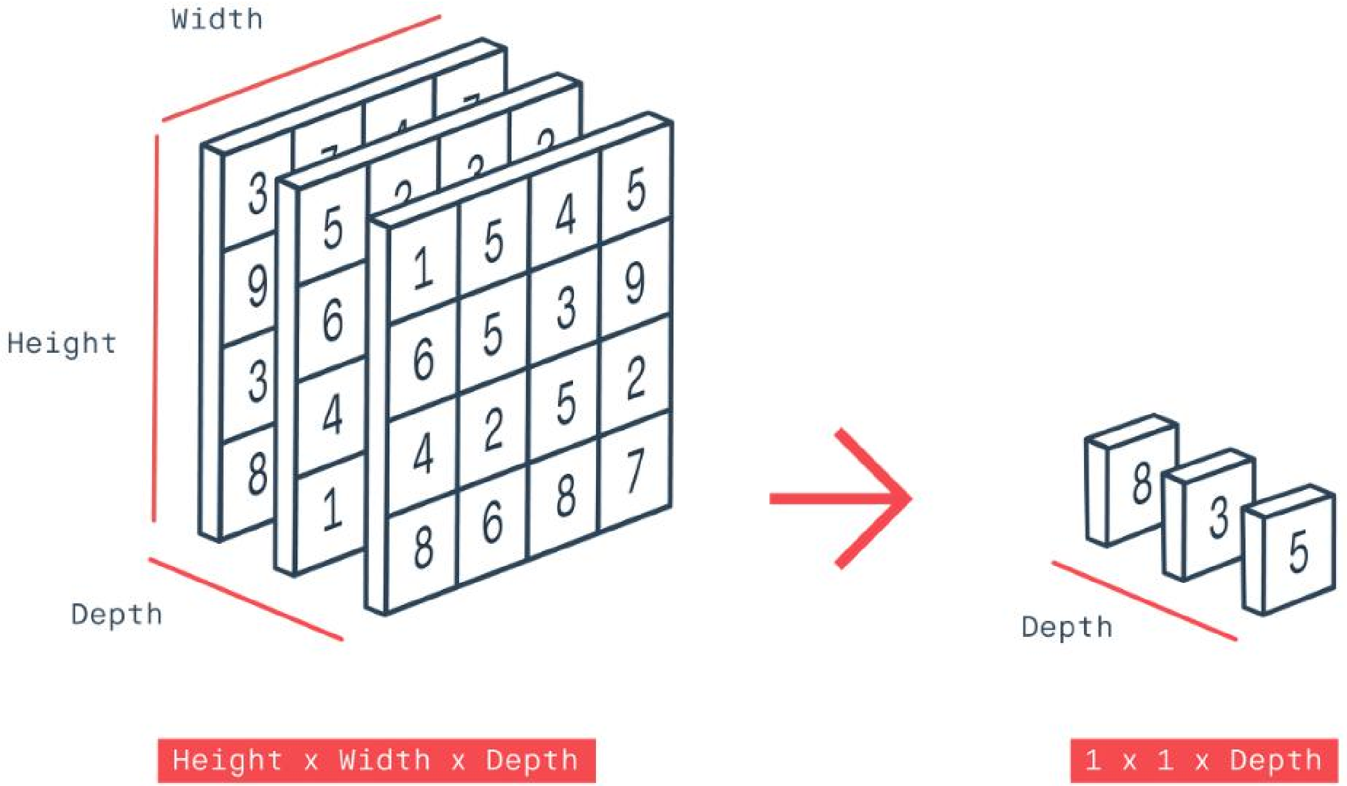
3×3 transposed convolution



✓ Transpose of convolving a 3×3 kernel over a 4×4 input ↔ convolving a 3×3 kernel over a 2×2 input padded with a 2×2 border of zeros.

Dumoulin, Vincent, and Francesco Visin. " A guide to convolution arithmetic for deep learning." arXiv`16

Global Average Pooling



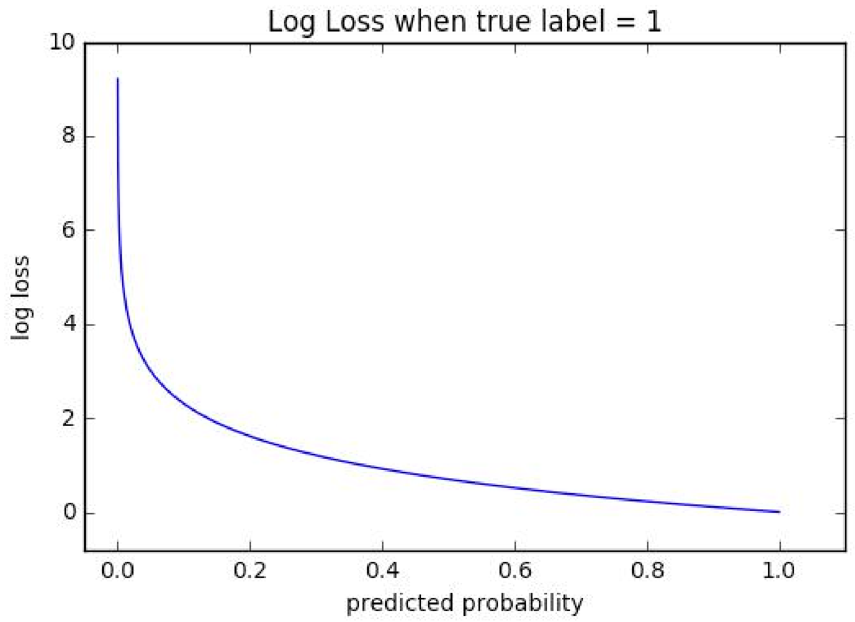
https://underflow101.tistory.com/41

Log-Loss (Cross Entropy)

- Cross entropy of distribution p and q:

$$\mathrm{H}(p,q) = \mathbb{E}_p[-log q]$$

$$\mathrm{H}(p,q) = -\sum_x p(x)\log q(x)$$

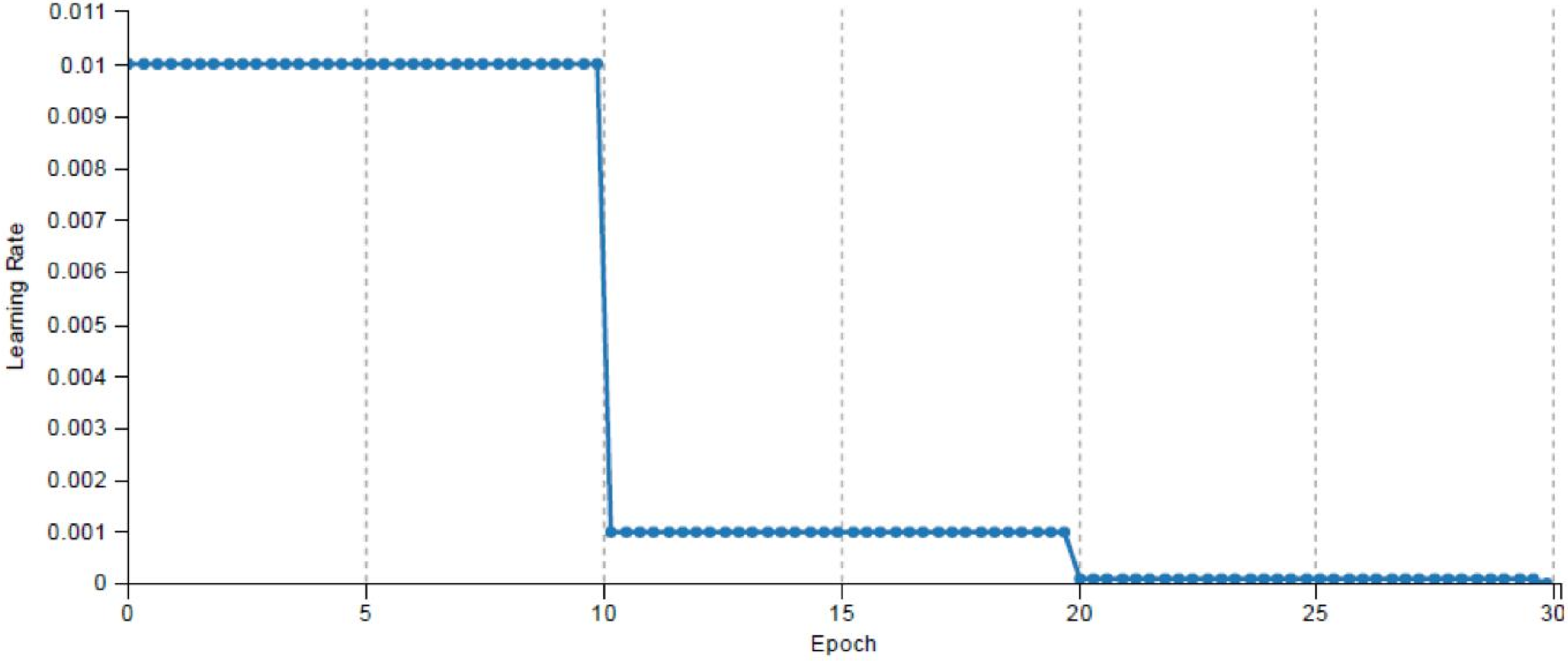


Example :

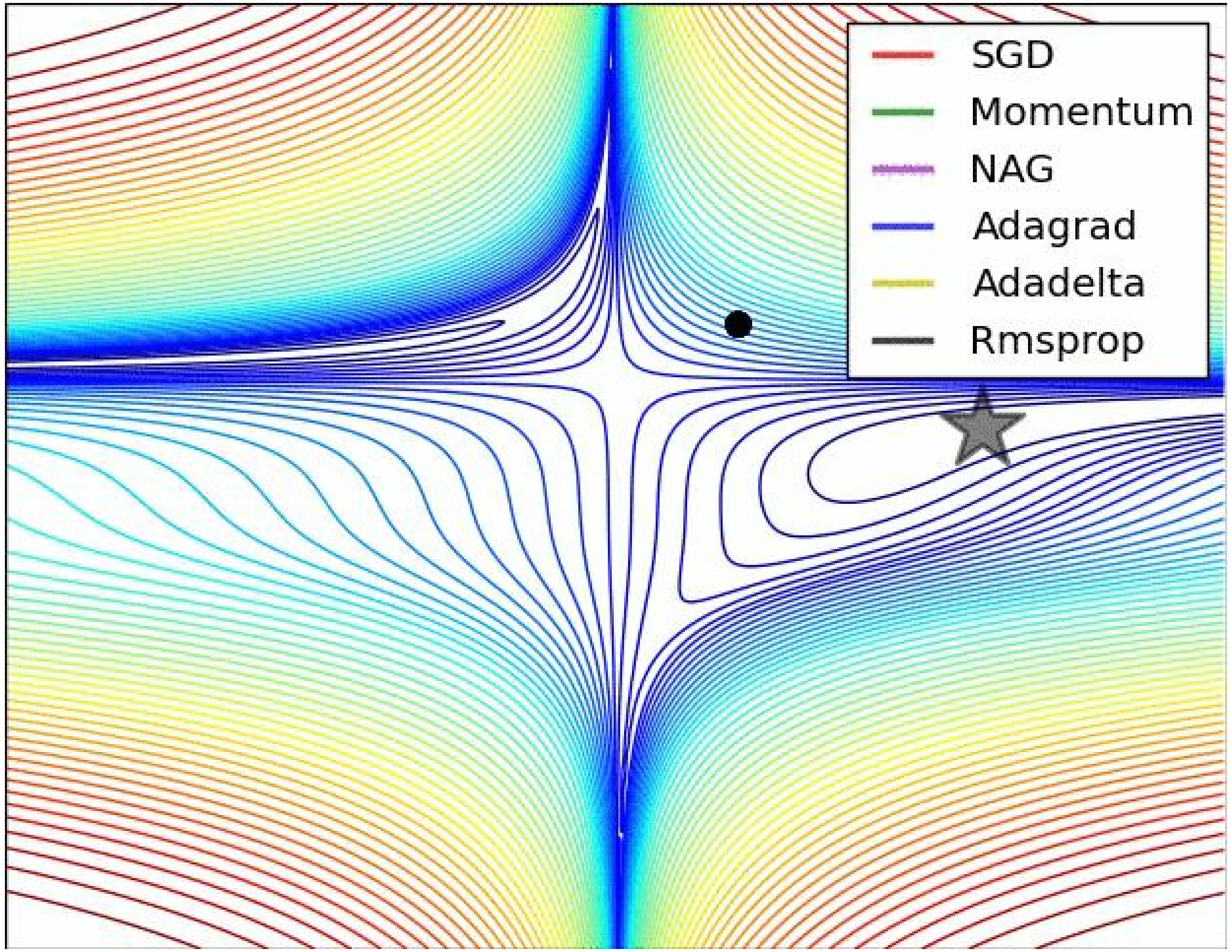
```
computed       | targets
---------------------------------------
0.3   0.3   0.4   | 0   0   1 (democrat)
0.3   0.4   0.3   | 0   1   0 (republican)
0.1   0.2   0.7   | 1   0   0 (other)
```

```
-( (ln(0.3)*0) + (ln(0.3)*0) + (ln(0.4)*1) ) = -ln(0.4)
```

Step Learing Rate

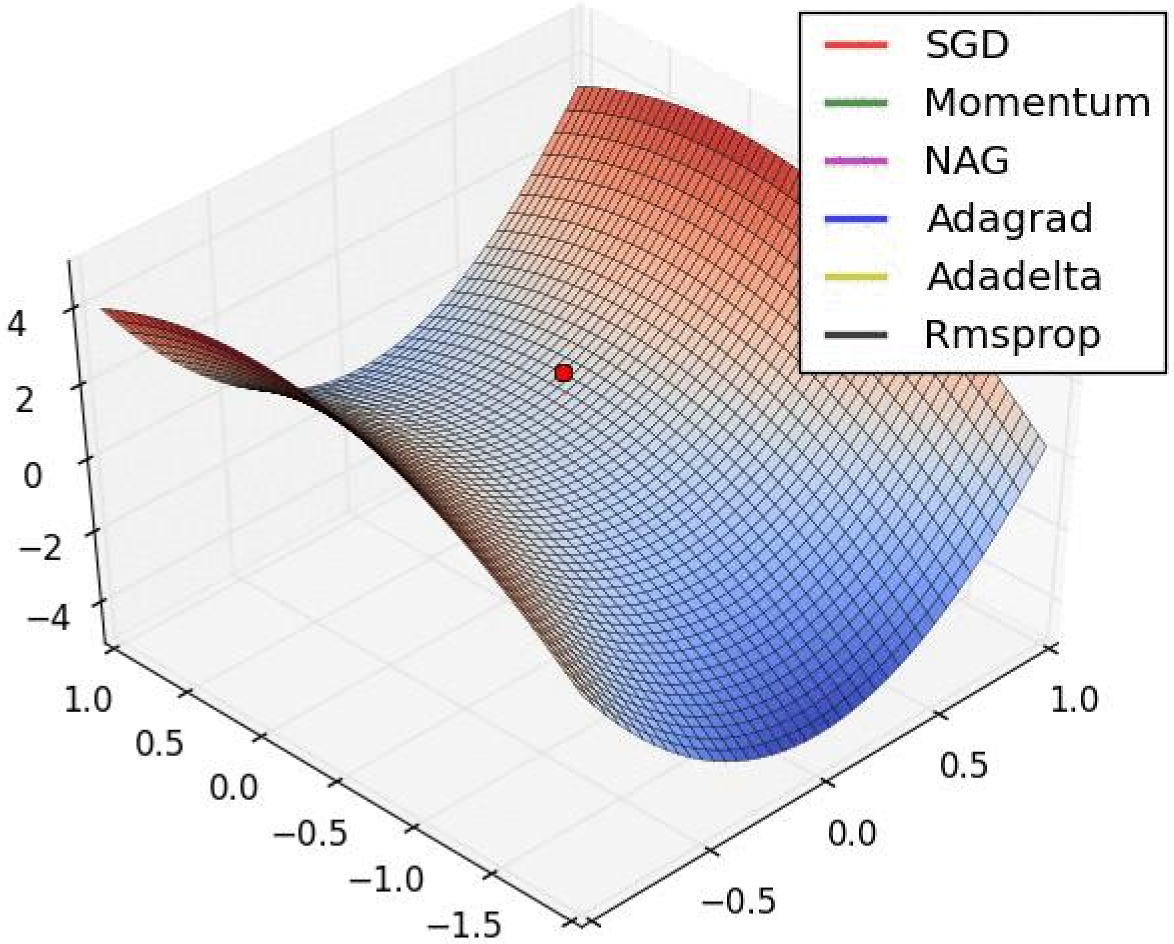


SGD optimization on loss surface contours

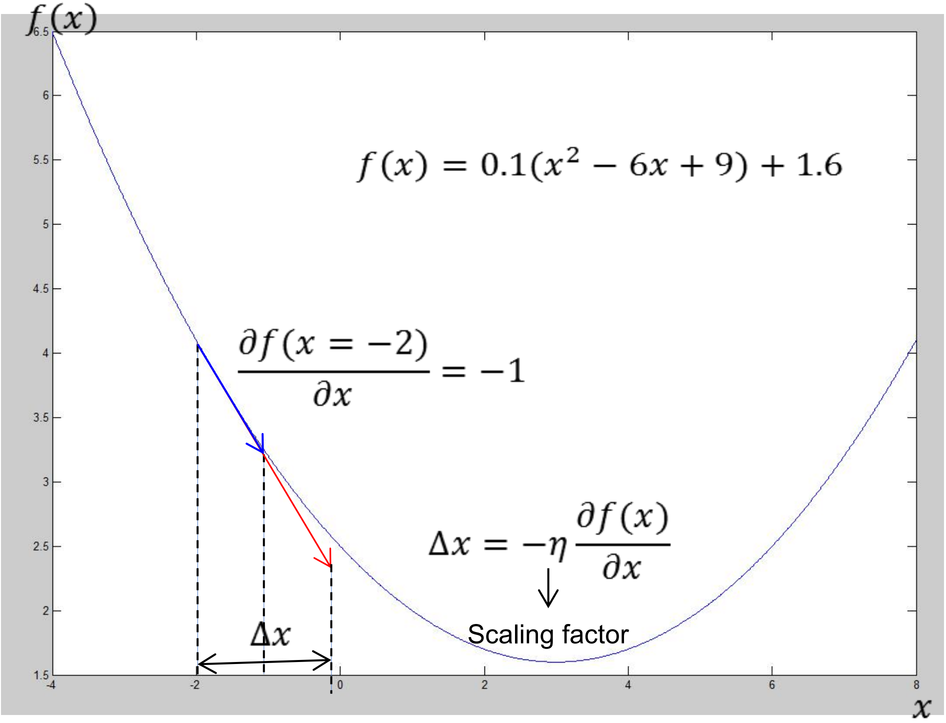


Optimizer

▶ SGD optimization on saddle point



Preliminary - Steepest Descent Method



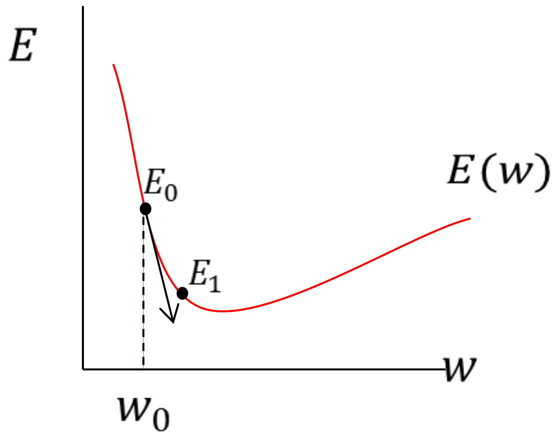
- **Remarks on $\frac{\partial f(x)}{\partial x}$:**
  - Represent the direction of slope.
  - $\ominus$for the left side.
  - $\oplus$ for the right side.
  - Minus sign is added to drive the function to its minimum value.

Mathematical Explanation (1/2)

- We want to update w as:
  - $w_1 = w_0 + \eta\vec{v}$

    Scaling factor

    We don't know yet its exact value



- So: $\Delta E = E_1 - E_0$
  - We want to drive E to its minimum.
    - ΔE should be $\ominus$.
  - And we have:

$$\Delta E = E(\overbrace{w_0 + \eta\vec{v}}^{w_1}) - E(w_0)$$

Mathematical Explanation (2/2)

- Using Taylor series:
  - Centered at w_0.

$$\Delta E = E(w_0 + \eta\vec{v}) - E(w_0)$$

$$a = w_0$$

$$= \cancel{E(w_0)} + \nabla E(w_0)(\cancel{w_0} + \eta\vec{v} - \cancel{w_0}) - \{\cancel{E(w_0)} + \cancel{\nabla E(w_0)(w_0 - w_0)}\}$$

Taylor series:

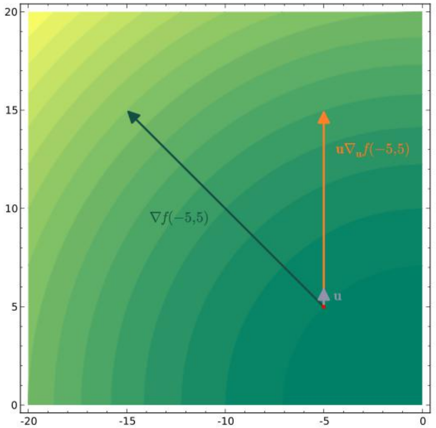
$$f(x) = f(a) + f'(a)(x-a) + HOT$$

gradient descent ignores the high order term

$$\Delta E = \eta \nabla E(w_0)\vec{v}$$

This part is also known as directional derivative

https://en.wikipedia.org/wiki/Directional_derivative



In which case this value will be ⊖?

When their direction are opposite (dot product rule).

So, $\vec{v}$ should be $\vec{v} = -\dfrac{\nabla E(w_0)}{\|\nabla E(w_0)\|}$

Normalize its value since $\vec{v}$ is a normal vector.

Thus: $w_1 = w_0 - \eta \nabla E(w_0)$

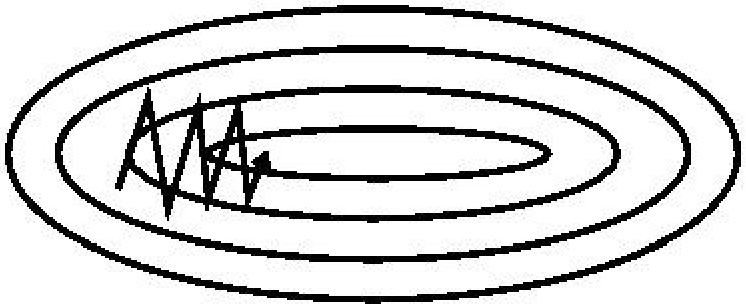
Gradient Descent-based Algorithms

Gradient descent

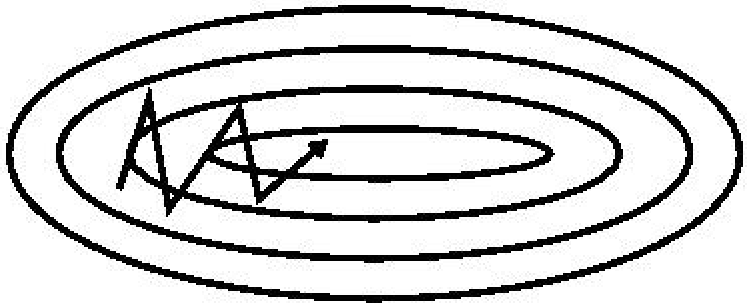
$$w = w - \eta \nabla J(w)$$

Gradient descent with momentum

$$\Delta w = \gamma \Delta w_{t-1} - \eta \nabla J(w) \longrightarrow w = w + \Delta w$$



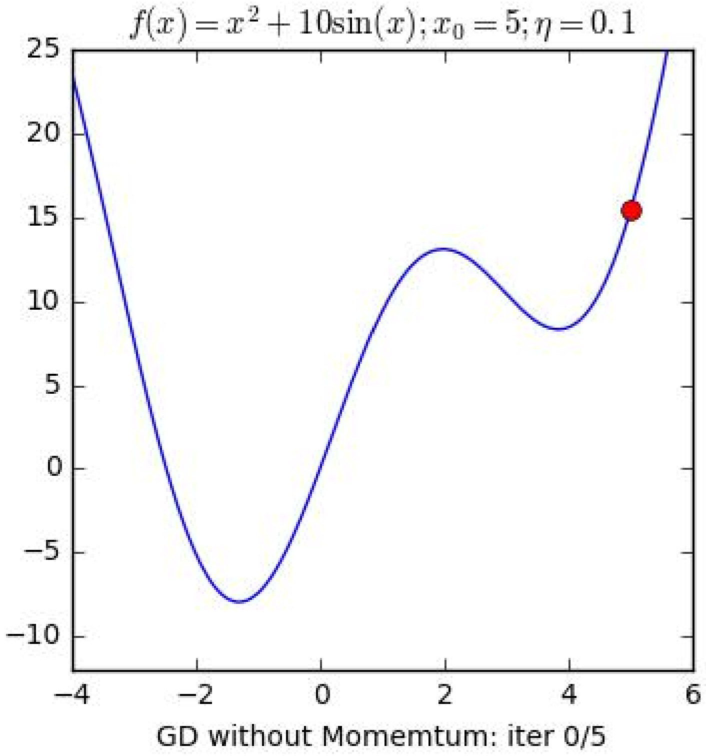
SGD without momentum

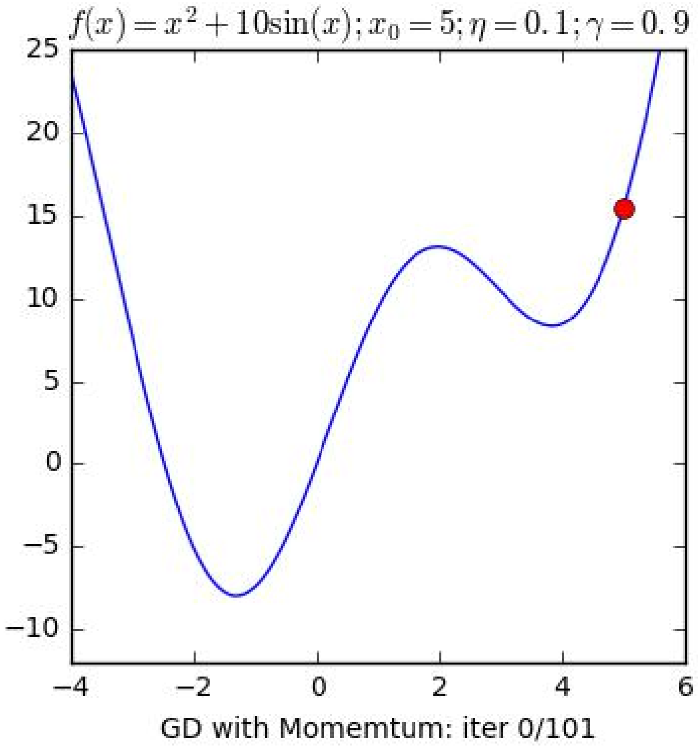


SGD with momentum

http://ruder.io/optimizing-gradient-descent/

- **Function with 2 minimums:** $f(x) = x^2 + 10\sin(x)$





https://www.d2l.ai/chapter_optimization/momentum.html

Gradient Descent-based Algorithms

- **Gradient descent**

$$w = w - \eta \nabla J(w)$$

- **RMS prop (gradient direction and moving average)**
  - Adapts the learning rate to the parameters

$$w = w - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla J(w) \qquad E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

http://ruder.io/optimizing-gradient-descent/

Gradient Descent-based Algorithms

Gradient descent

$$w = w - \eta \nabla J(w)$$

Gradient descent with momentum

$$\Delta w = \beta \Delta w_{t-1} - \eta \nabla J(w) \longrightarrow w = w + \Delta w$$

$$\beta = 0.9$$

RMS prop (gradient direction and moving average)

$$w = w - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla J(w)$$

$$g = \nabla J(w) \qquad \beta_2 = 0.9$$

Adam

- RMS prop + momentum

$$w = w - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} E[g]_t$$

$$E[g^2]_t = \beta_2 E[g^2]_{t-1} + (1 - \beta_2) g_t^2$$

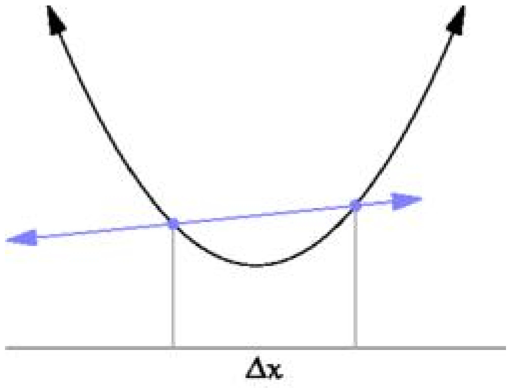
$$E[g]_t = \beta_1 E[g]_{t-1} + (1 - \beta_1) g_t$$

$$g = \nabla J(w) \qquad \beta_1 = 0.9, \beta_2 = 0.999$$

http://ruder.io/optimizing-gradient-descent/

https://johnchenresearch.github.io/demon/

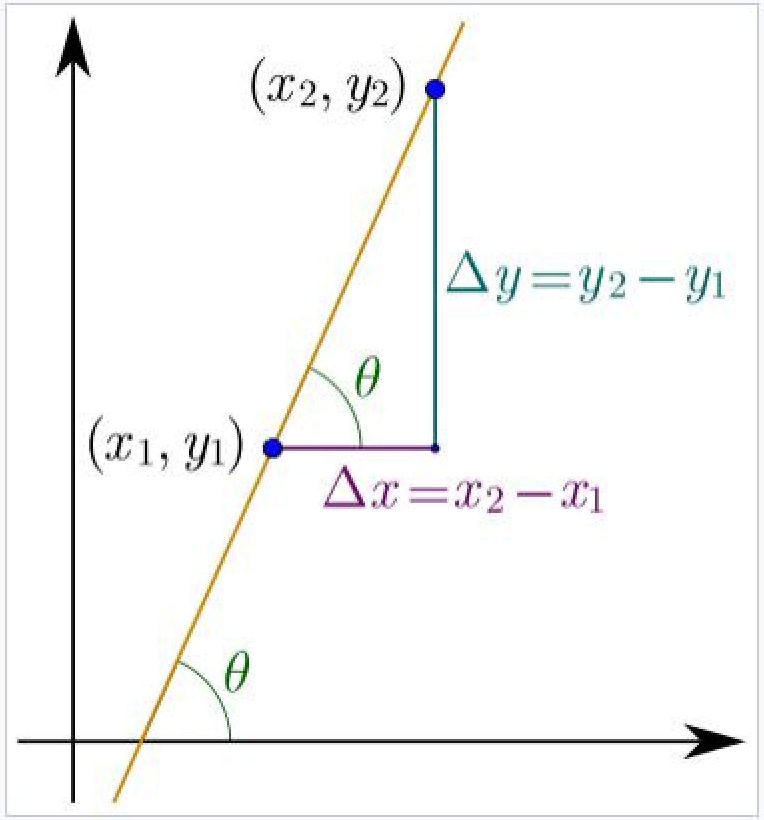
- Derivative is a slope of the tangent line



$$m = \frac{\Delta f(a)}{\Delta a} = \frac{f(a+h) - f(a)}{(a+h) - (a)} = \frac{f(a+h) - f(a)}{h}.$$

- The slope is when $\Delta x \rightarrow 0$

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$



Slope of a linear function:

$$m = \frac{\Delta y}{\Delta x}$$

https://en.wikipedia.org/wiki/Derivative

$$F(x) = f(g(x))$$

Derivative at a

$$F'(a) = \lim_{h \to 0} \frac{f\big(g(a+h)\big) - f\big(g(a)\big)}{h}$$

Let $a + h = x$

$$F'(a) = \lim_{x-a \to 0} \frac{f\big(g(x)\big) - f\big(g(a)\big)}{x - a}$$

Multiply by $\frac{g(x)-g(a)}{g(x)-g(a)}$ and re-arrange

$$F'(a) = \lim_{x \to a} \frac{f\big(g(x)\big) - f\big(g(a)\big)}{g(x) - g(a)} \frac{g(x) - g(a)}{x - a}$$

recall

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Chain Rule (2/3)

$$F(x) = f(g(x))$$

Derivative at a

$$F'(a) = \lim_{h \to 0} \frac{f(g(a+h)) - f(g(a))}{h}$$

Let $a + h = x$

$$F'(a) = \lim_{h \to 0} \frac{f(g(a+h)) - f(g(a))}{h}$$

Multiply by $\frac{g(x)-g(a)}{g(x)-g(a)}$
and re-arrange

$$F'(a) = \lim_{x \to a} \frac{f(g(a+h)) - f(g(a))}{g(a+h) - g(a)} \frac{g(a+h) - g(a)}{h}$$

recall

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

$$F'(a) = \lim_{x \to a} \frac{f(g(a+h)) - f(g(a))}{g(a+h) - g(a)} g'(a)$$

Chain Rule (3/3)

$$F(a) = f(g(a))$$

Derivative at a

$$F'(a) = \lim_{h\to 0} \frac{f(g(a+h)) - f(g(a))}{h}$$

Let $a + h = x$

$$F'(a) = \lim_{h\to 0} \frac{f(g(a+h)) - f(g(a))}{h}$$

Multiply by $\frac{g(x)-g(a)}{g(x)-g(a)}$
and re-arrange

$$F'(a) = \lim_{h\to 0} \frac{f(g(a+h)) - f(g(a))}{g(a+h)-g(a)} \frac{g(a+h)-g(a)}{h}$$

recall

$$f'(a) = \lim_{h\to 0} \frac{f(a+h) - f(a)}{h}$$

Other form   Let $a + h = x$

$$f'(a) = \lim_{x\to a} \frac{f(x) - f(a)}{x-a}$$

$$F'(a) = \left[\lim_{h\to 0} \frac{f(g(a+h)) - f(g(a))}{g(a+h)-g(a)}\right] g'(a)$$

Change was
respect to a

$$F'(a) = \left[\lim_{h\to 0} \frac{f(g(a+h)) - f(g(a))}{g(a+h)-g(a)}\right] \frac{dg(a)}{da}$$

When $h \to 0$,
$g(a+h) - g(a) \to 0$

$$F'(a) = \left[\lim_{g(a+h)\to g(a)} \frac{f(g(a+h)) - f(g(a))}{g(a+h)-g(a)}\right] \frac{dg(a)}{da}$$

$$F'(a) = \frac{df(g(a))}{dg(a)} \frac{dg(a)}{da}$$

Same form

Chain Rule …details

recall $f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$

$$F'(a) = \left[\lim_{g(a+h) \to g(a)} \frac{f\big(g(a+h)\big) - f\big(g(a)\big)}{g(a+h) - g(a)}\right] \frac{dg(a)}{da}$$

simplify

$$F'(a) = \left[\lim_{g(x) \to g(a)} \frac{f\big(g(x)\big) - f\big(g(a)\big)}{g(x) - g(a)}\right] \frac{dg(a)}{da}$$

Same form, now the point is
represented as function output

simplify

$$F'(a) = \frac{df(g(a))}{dg(a)} \frac{dg(a)}{da} \longrightarrow F'(a) = \frac{df}{dg} \frac{dg}{da}$$

Chain Rule- Graphical Illustration

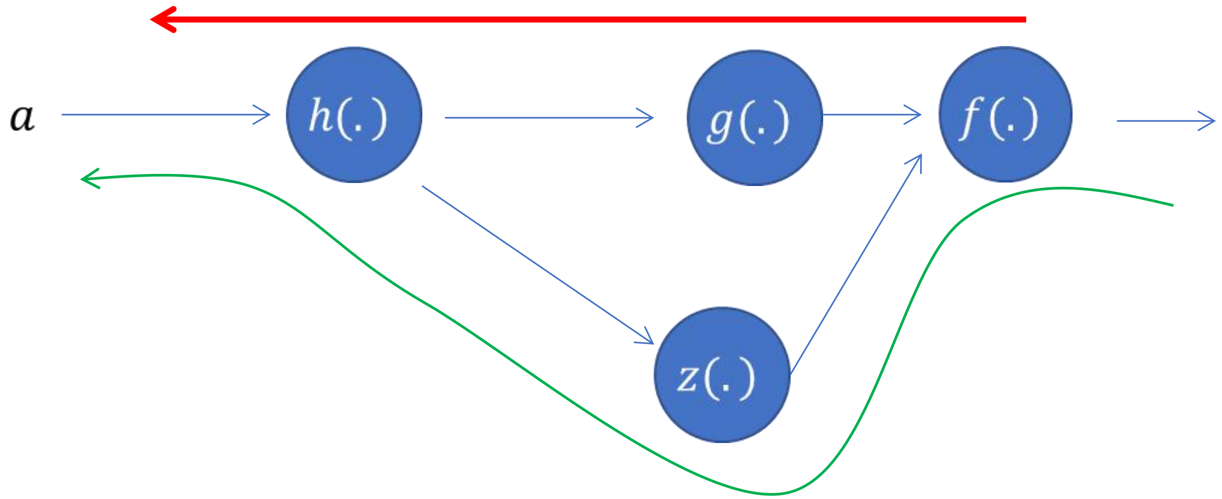
$$a \longrightarrow g(.) \longrightarrow f(.) \longrightarrow f(g(a))$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial a}$$

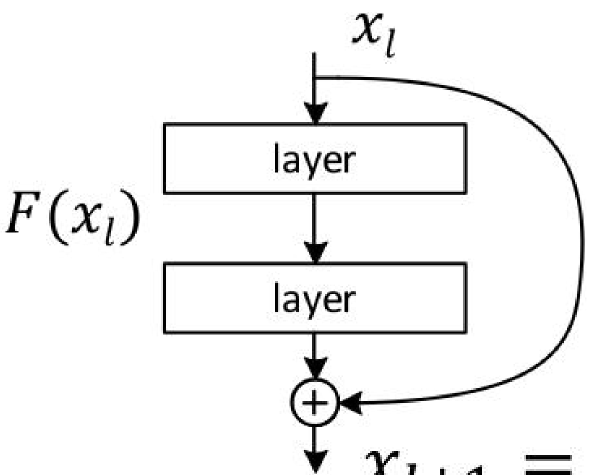
$$a \longrightarrow h(.) \longrightarrow g(.) \longrightarrow f(.) \longrightarrow$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial a}$$

Chain Rule- Graphical Illustration



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial a} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial h}\frac{\partial h}{\partial a}$$



$$x_{l+1} = x_l + F(x_l)$$

- shortcut mapping: h = identity
- ~~after-add mapping: f = ReLU~~
- What if f = identity?

Very smooth forward propagation

$$x_{l+1} = x_l + F(x_l)$$

$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

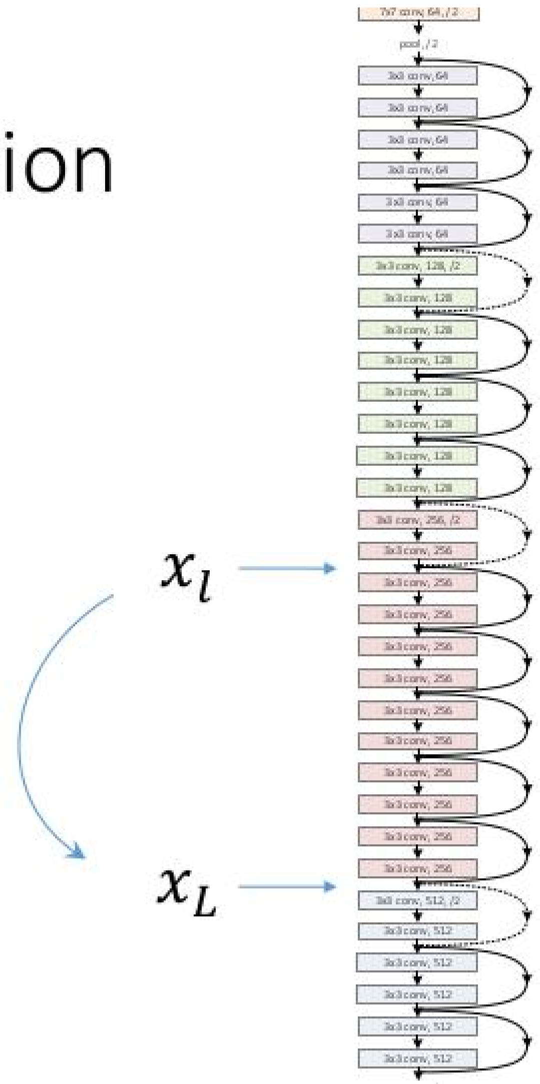
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

ResNet (3/)

Very smooth forward propagation

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

- Any x_l is directly forward-prop to any x_L, plus residual.
- Any x_L is an additive outcome.
  - in contrast to multiplicative: $x_L = \prod_{i=l}^{L-1} W_i x_l$



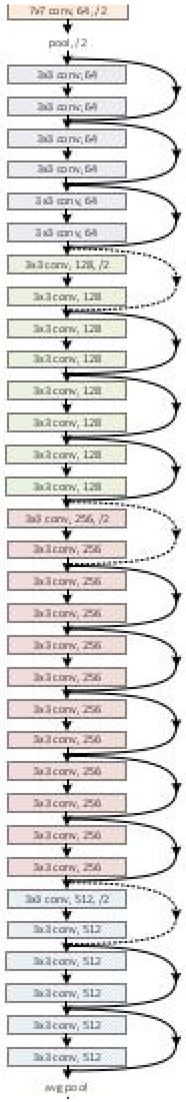
Very smooth backward propagation

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$$

$$\frac{\partial E}{\partial x_l}$$

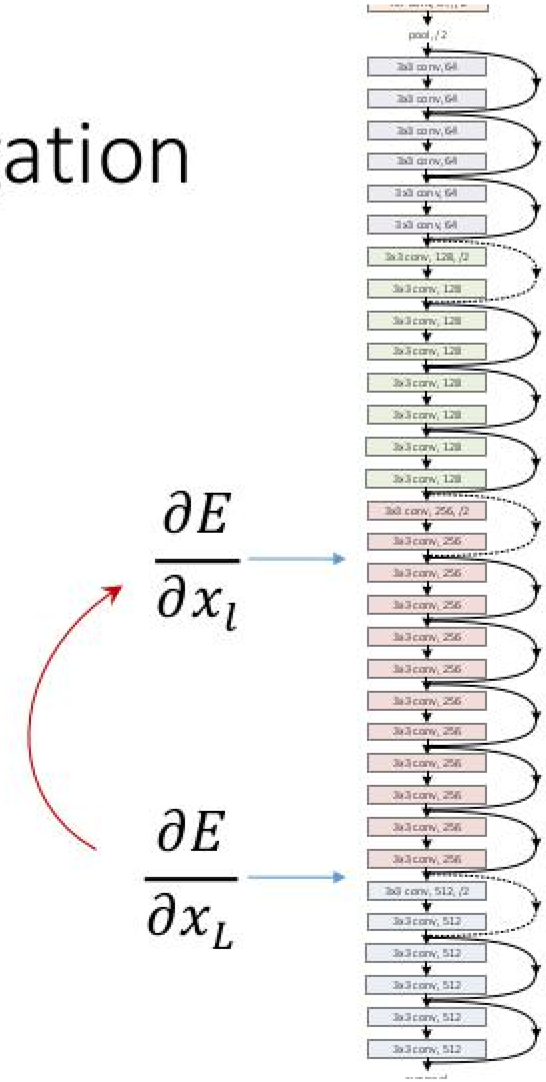
$$\frac{\partial E}{\partial x_L}$$



Very smooth backward propagation

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$$

- Any $\frac{\partial E}{\partial x_L}$ is directly back-prop to any $\frac{\partial E}{\partial x_l}$, plus residual.
- Any $\frac{\partial E}{\partial x_l}$ is additive; unlikely to vanish
  - in contrast to multiplicative: $\frac{\partial E}{\partial x_l} = \prod_{i=l}^{L-1} W_i \frac{\partial E}{\partial x_L}$



Residual for every layer

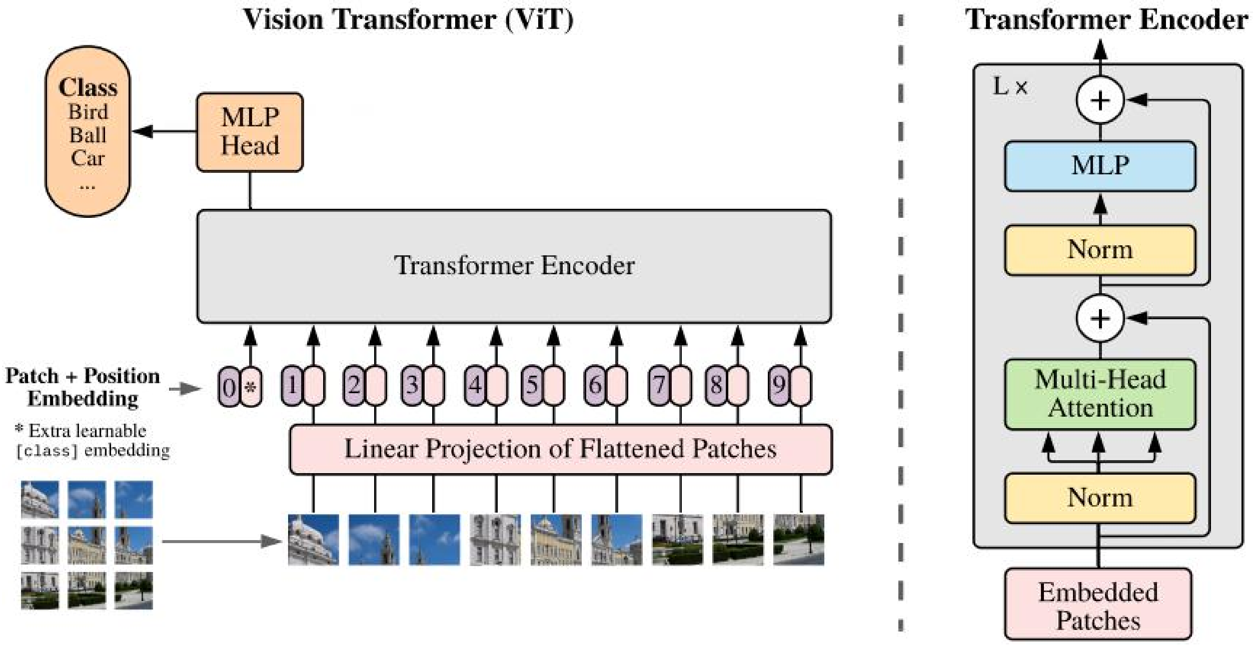
forward: $x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$

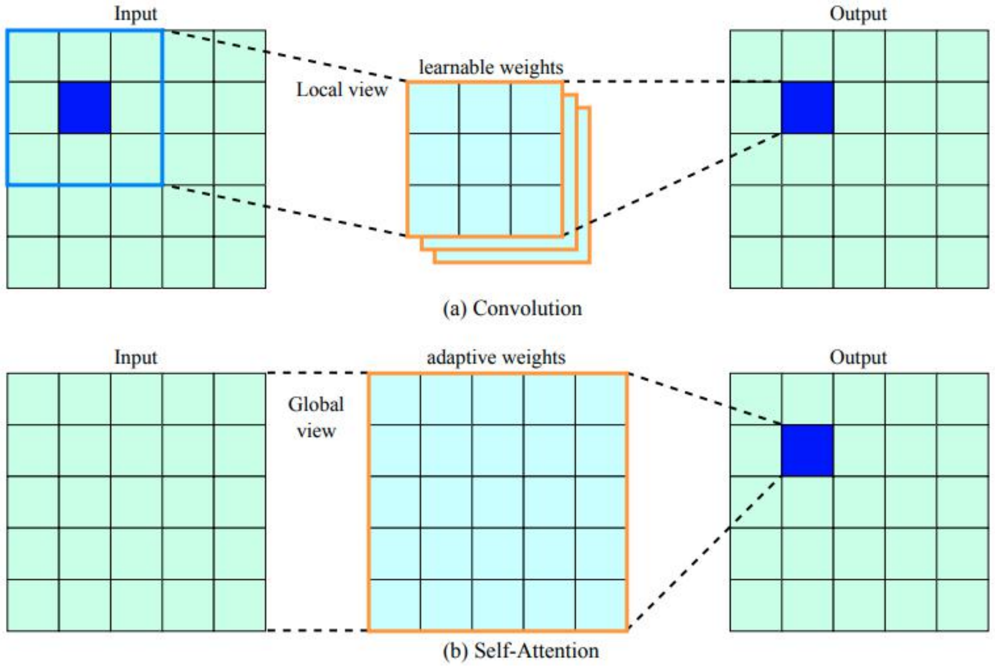
Enabled by:

- shortcut mapping: h = identity
- after-add mapping: f = identity

backward: $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$

Vision Transformer





Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." ICLR 2021

Token Embedding

▶ A word ~ a token

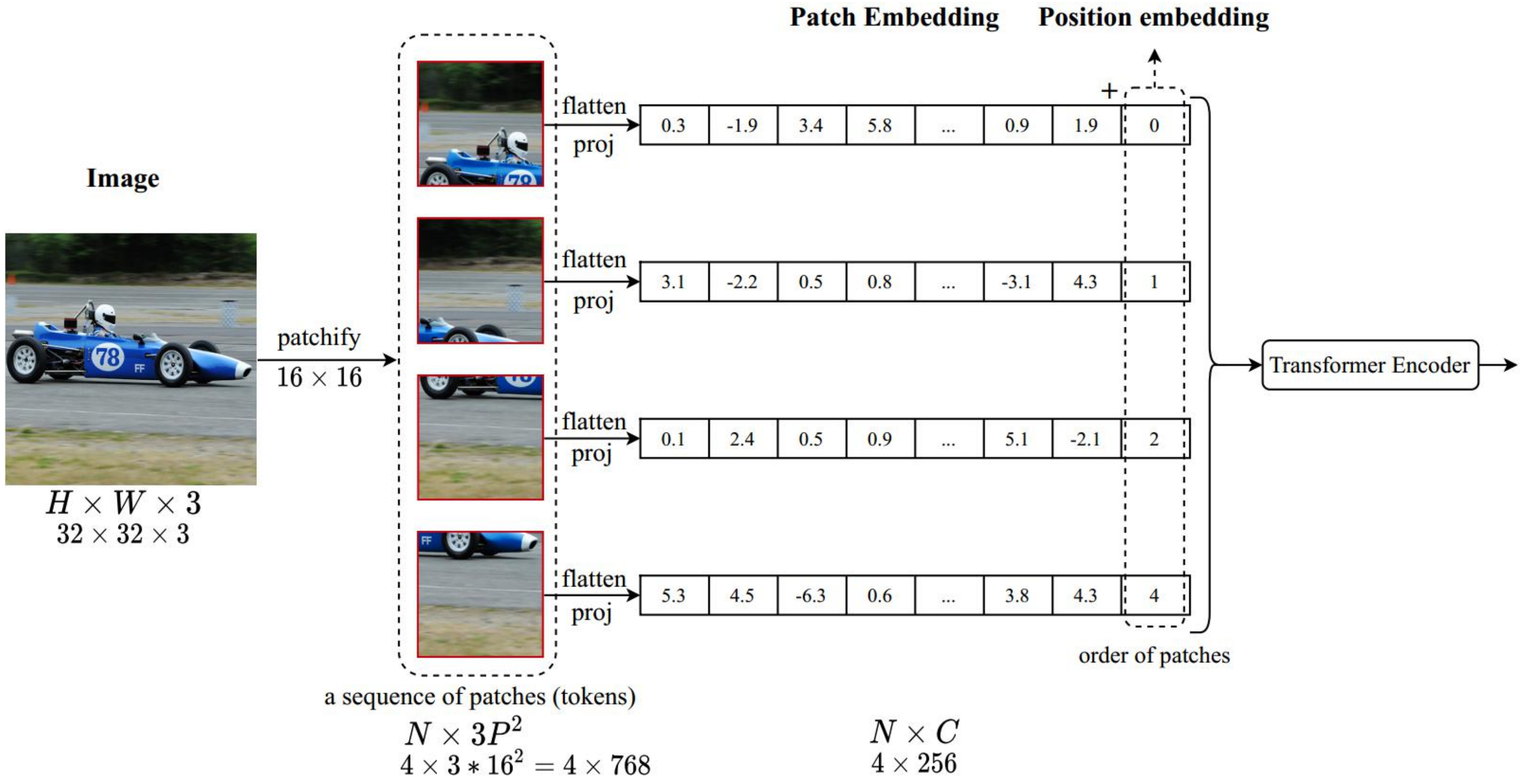
Natural Language Processing

Token	Token Embedding			
This	-0.5	...	2.1	0
is	1.7	...	3.4	1
a	-5.3	...	-1.4	2
car	5.2	...	4.3	3

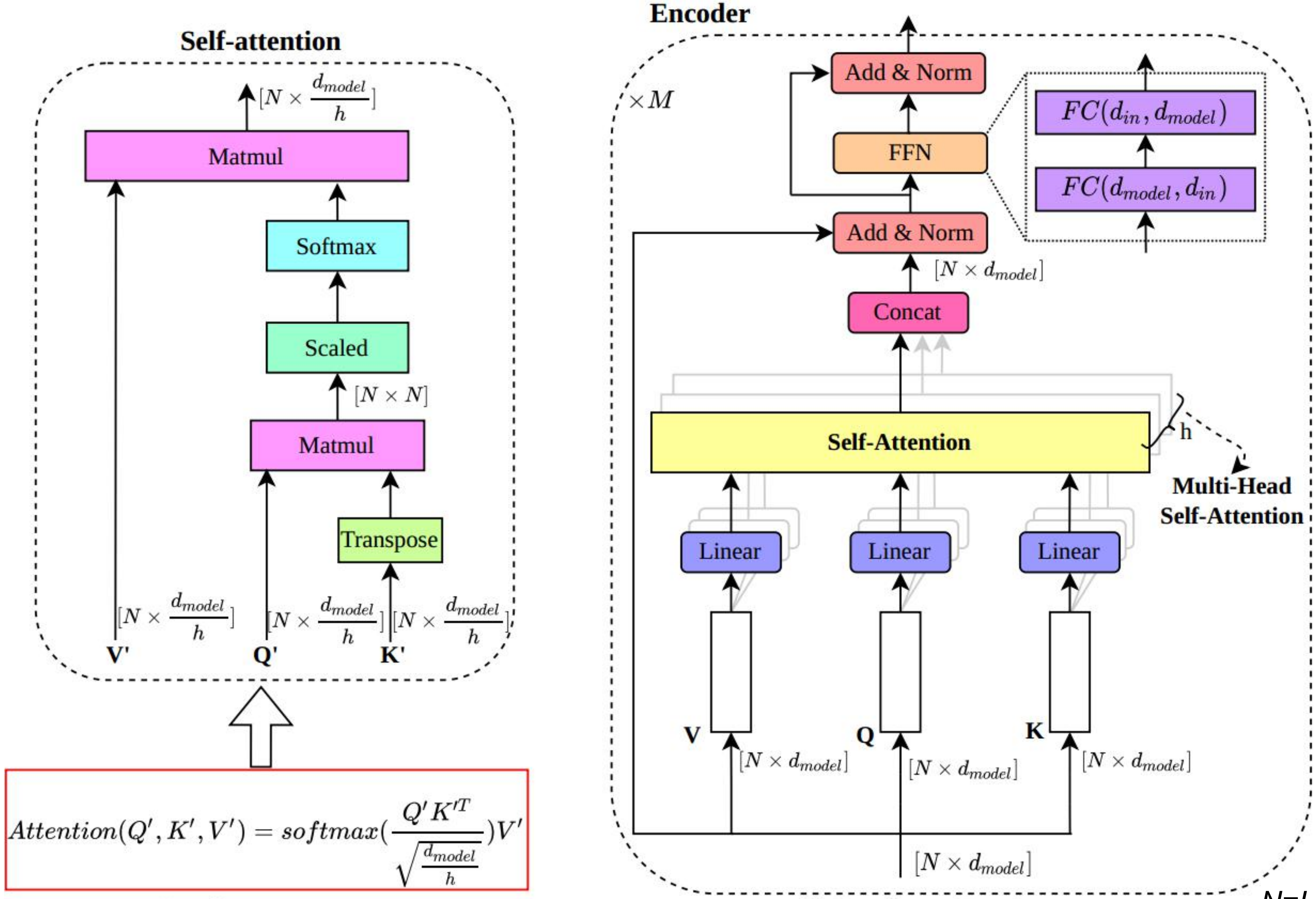
→ Transformer Encoder →

▶ A patch ~ a token

Computer Vision



Transformer Encoder



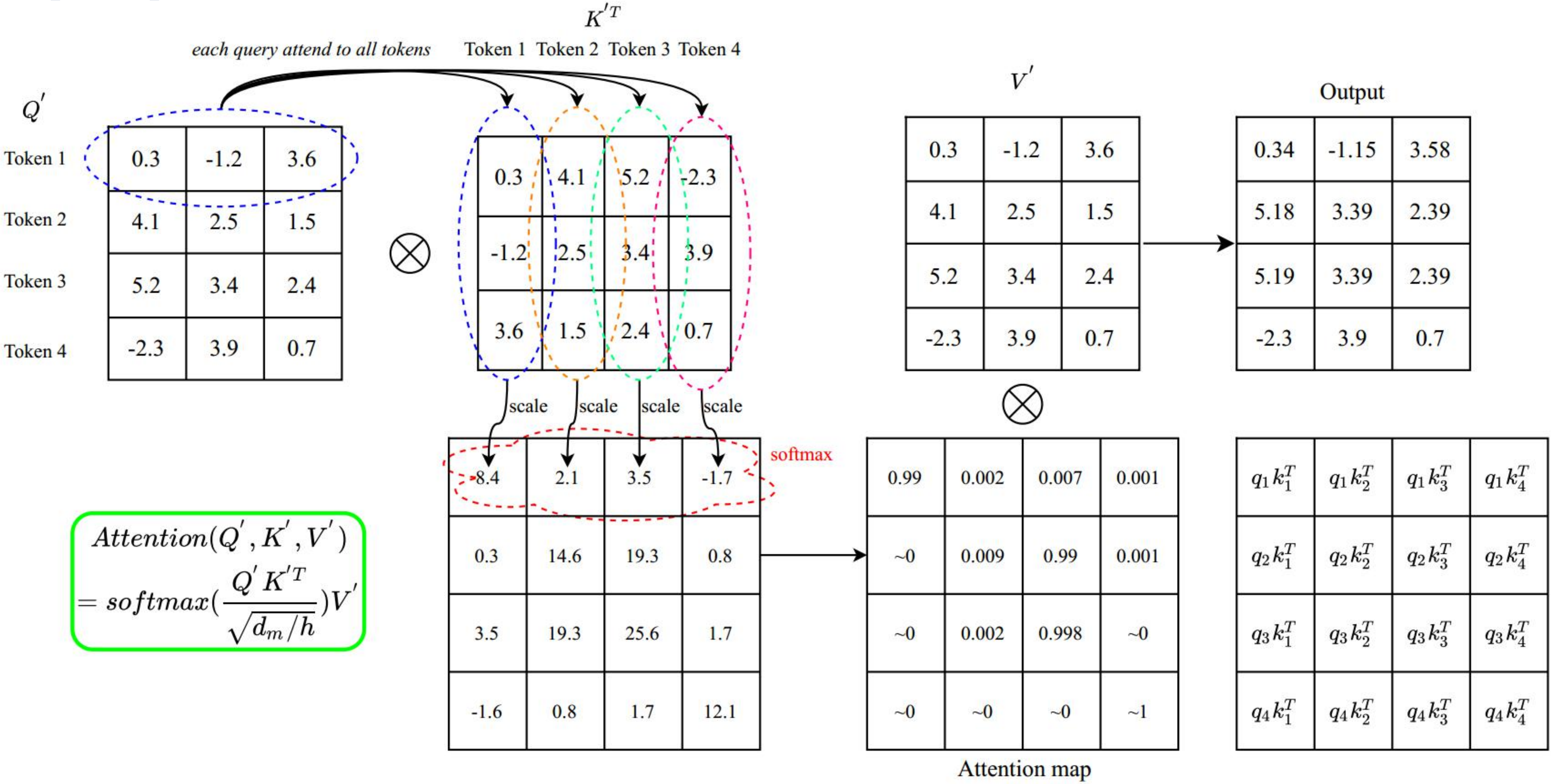
(a) Self-attention operation

(b) Transformer Encoder

*N=H*W*

Self-attention Example

- **Self-Attention: capture global dependencies from tokens**
  - Example: input has 4 tokens and each token has 3 channels



$$Attention(Q', K', V') = softmax\left(\frac{Q'K'^T}{\sqrt{d_m/h}}\right)V'$$

Why do we need to scale attention matrix?

Self-attention:

$$Attention(Q', K', V') = softmax\left(\frac{Q'K'^T}{\boxed{\sqrt{d_m/h}}}\right)V'$$

scaling factor

$$Q' \in \mathbb{R}^{N \times d_h}, \quad \text{a vector } q_i' \in \mathbb{R}^{d_h}, d_h \in d_m/h$$

$$K' \in \mathbb{R}^{N \times d_h}, \quad \text{a vector } k_j' \in \mathbb{R}^{d_h}$$

$$q_i' k_j'^T = \sum_{n=1}^{d_h} q_{i,n}' k_{j,n}' = q_{i,1}' k_{j,1}' + q_{i,2}' k_{j,2}' + \ldots$$

Assume that q_i', k_j' are independent random variables with mean 0 and variance 1

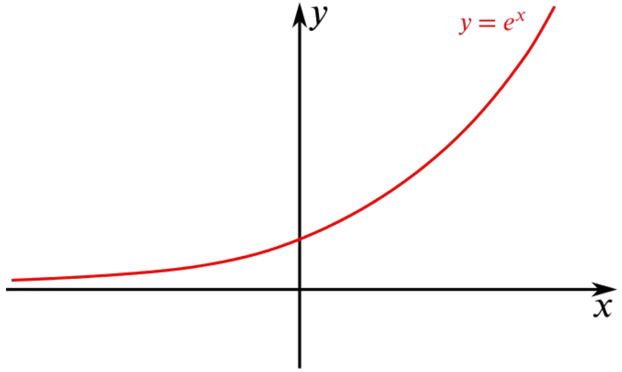
$$\mathbb{E}(q_i' k_j'^T) = \mathbb{E}(q_i')\mathbb{E}(k_j'^T) = 0$$

$$Var(q_i' k_j'^T) = Var(q_{i,1}' k_{j,1}' + q_{i,2}' k_{i,2}' + \ldots) = 1 + 1 + \ldots = d_h = d_m/h$$

$$Std(q_i' k_j'^T) = \sqrt{Var(q_i' k_j'^T)} = \sqrt{d_m/h}$$

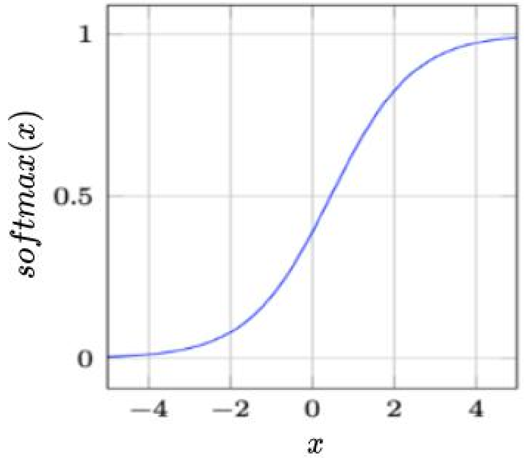
Max versus Softmax

$$max(x = [8.4, 2.1, 3.5, -1.7]) = 8.4$$



$$softmax(x) = \frac{\boxed{e^{x_i}}}{\sum_{j=1}^{N} e^{x_j}} = [0.99, 0.002, 0.007, 0.001]$$

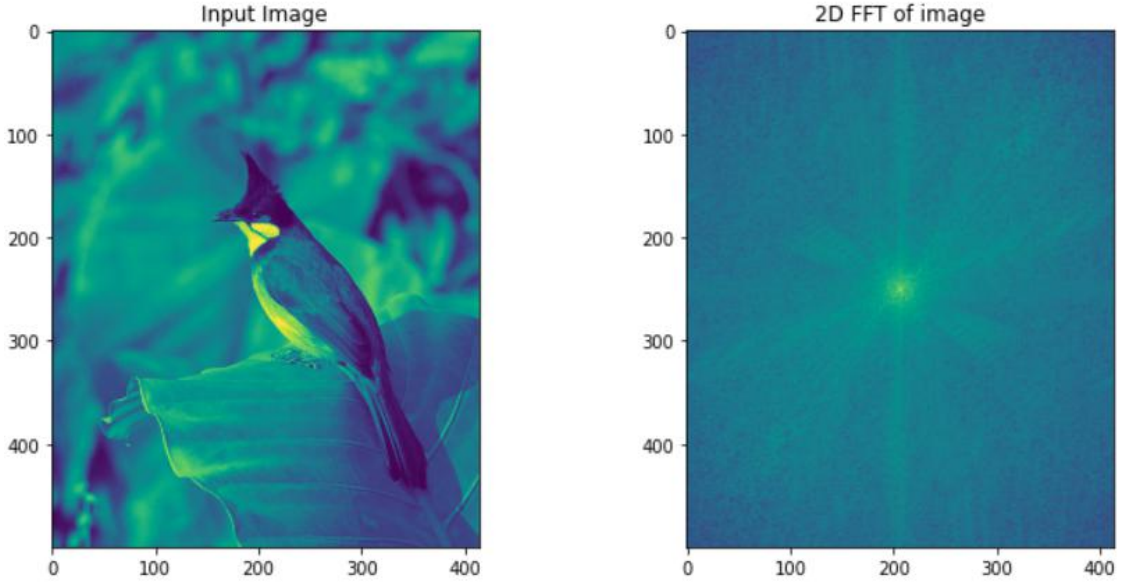
- ✓ turn values into probability distribution
- ✓ dependencies between elements



2D Dicrete Fourier Transform

- One-to-one mapping: spatial domain → complex frequency domain.
- Preserve all the information of the input.
- The output features has a wide range of the frequencies
- Goal: extract helpful frequencies from Fourier features --> increase representation ability.

$$\mathcal{X}[:, u, v] = \mathcal{F}(X) = \sum_{m}^{H_P-1} \sum_{n}^{W_P-1} X[:, m, n] e^{-j2\pi\left(\frac{um}{H_P} + \frac{vn}{W_P}\right)}$$



- There has a conjugate symmetry of the complex tensor.
  - A half of complex tensor needs to be computed and restored.

Example - Amplitude Spectrum

-0.5	1.2	2.5	2.2	1.5
3.4	-1.9	2.3	4.1	5.3
-0.8	2.6	1.1	2.5	-2.1
1.3	4.4	3.2	1.4	0.9
3.3	-2.4	0.9	4.2	3.4

$X : [5 \times 5]$

2D DFT

$$\mathcal{X}[u,v] = \sum_{m=0}^{H-1}\sum_{n=0}^{W-1} X[m,n]e^{-j2\pi(\frac{um}{H}+\frac{vn}{W})}$$

frequencies: $\omega_m = \dfrac{2\pi m}{H}$, $\omega_n = \dfrac{2\pi n}{W}$

$$\mathcal{X}[0,0] = \sum_{m=0}^{H-1}\sum_{n=0}^{W-1} X[m,n]e^{0} = 44.0$$

$$\mathcal{X}[2,1] = \sum_{m=0}^{H-1}\sum_{n=0}^{W-1} X[m,n]e^{-j2\pi(\frac{2m}{4}+\frac{n}{4})} = -4.2 - j17.3$$

$$\mathcal{X}[u,v] = \mathcal{X}^*[H-u, W-v]$$

44.0	-9.0+j7.4	3.8-j1.1	3.8+j1.1	-9.0-j7.4
2.1+j	0.5+j13.1	2.1+j5.4	2.2-j4.4	-1.2-j9.5
-6.9-j9.7	-4.2-j17.3	-2.2-j0.9	-12.0+j5.4	-3.3+j12.2
-6.9+j9.7	-3.3-j12.2	-12.0-j5.4	-2.2+j0.9	-4.2+j17.3
2.1-j	-1.2+j9.5	2.2+j4.4	2.1-j5.4	0.5-j13.1

$\mathcal{X} : [5 \times 5]$

shifting

-2.2+j0.9	-4.2+j17.3	-6.9+j9.7	-3.3-j12.2	-12.0-j5.4
2.1-j5.4	0.5-j13.1	2.1-j	-1.2+j9.5	2.2+j4.4
3.8+j1.1	-9.0-j7.4	44.0	-9.0+j7.4	3.8-j1.1
2.2-j4.4	-1.2-j9.5	2.1+j	0.5+j13.1	2.1+j5.4
-12.0+j5.4	-3.3+j12.2	-6.9-j9.7	-4.2-j17.3	-2.2-j0.9

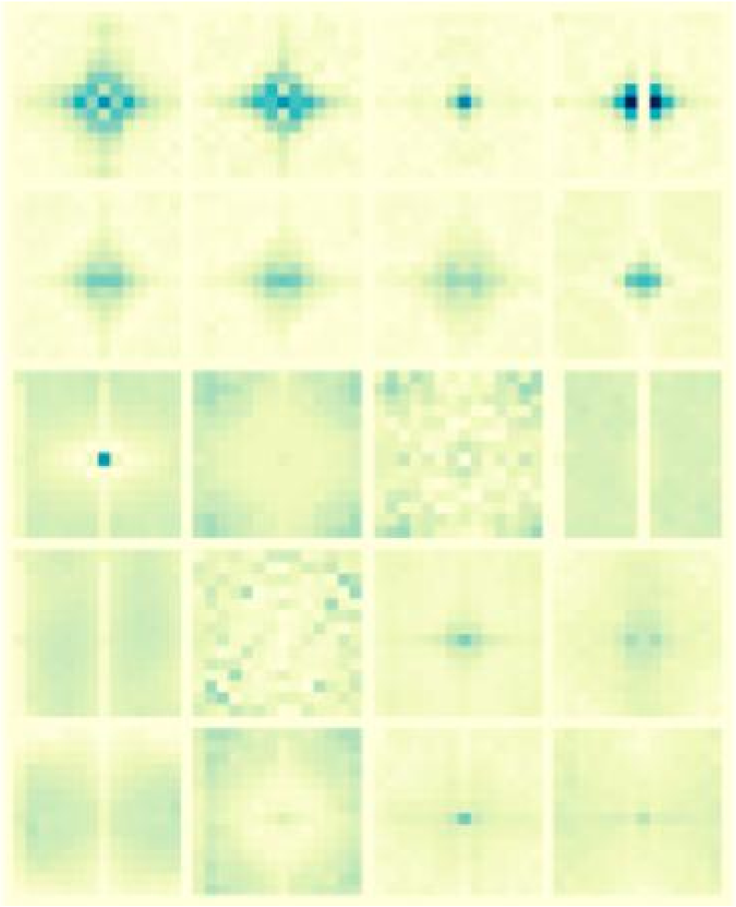
amplitude

$$r = \sqrt{a^2+b^2}$$

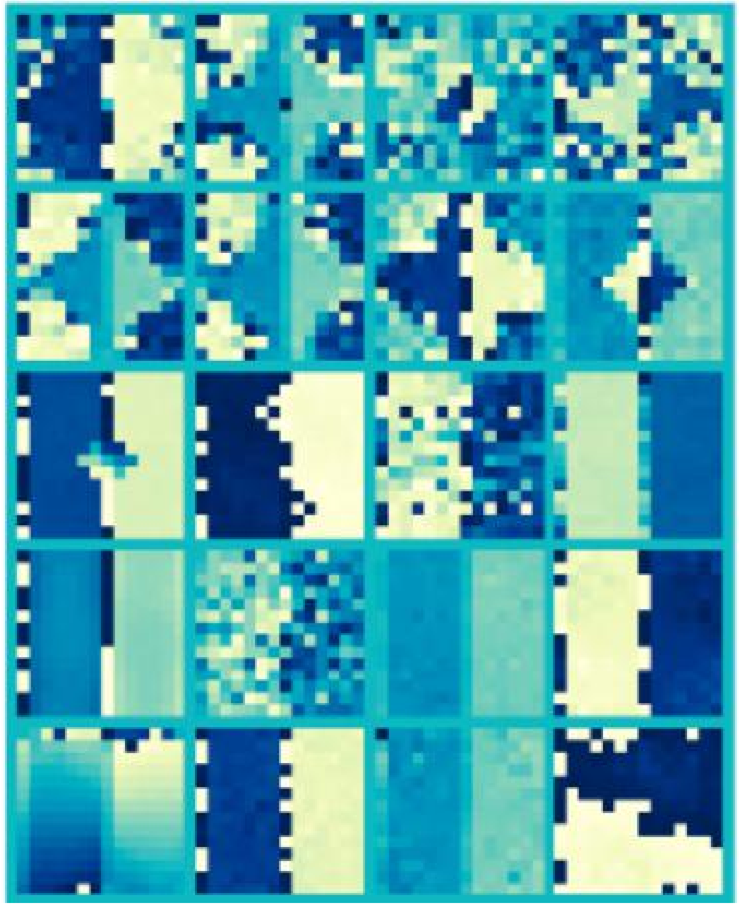
2.4	17.8	11.9	12.7	13.2
5.8	13.1	2.3	9.6	4.9
3.9	11.7	44.0	11.7	3.9
4.9	9.6	2.3	13.1	5.8
13.2	12.7	11.9	17.8	2.4

Amplitude spectrum

Amplitude and Phase Spectrum

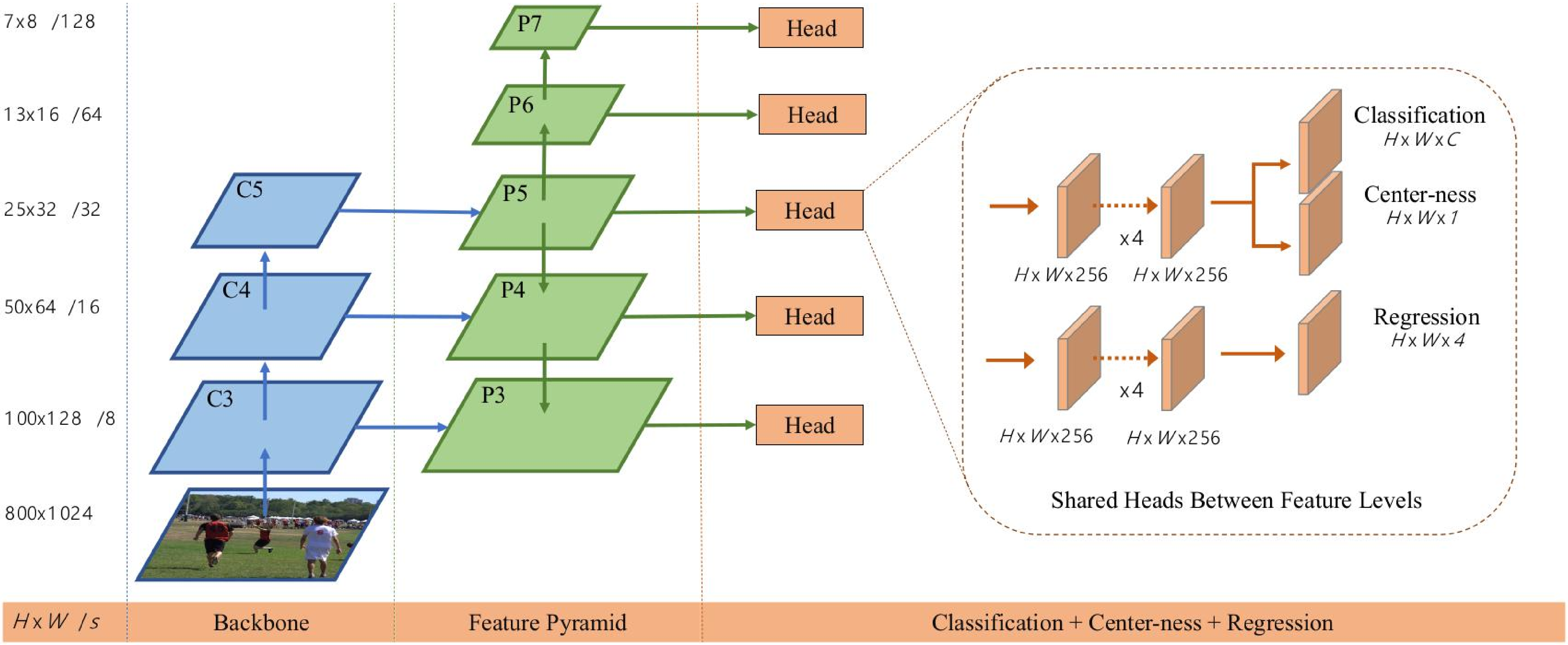


Amplitude spectrum



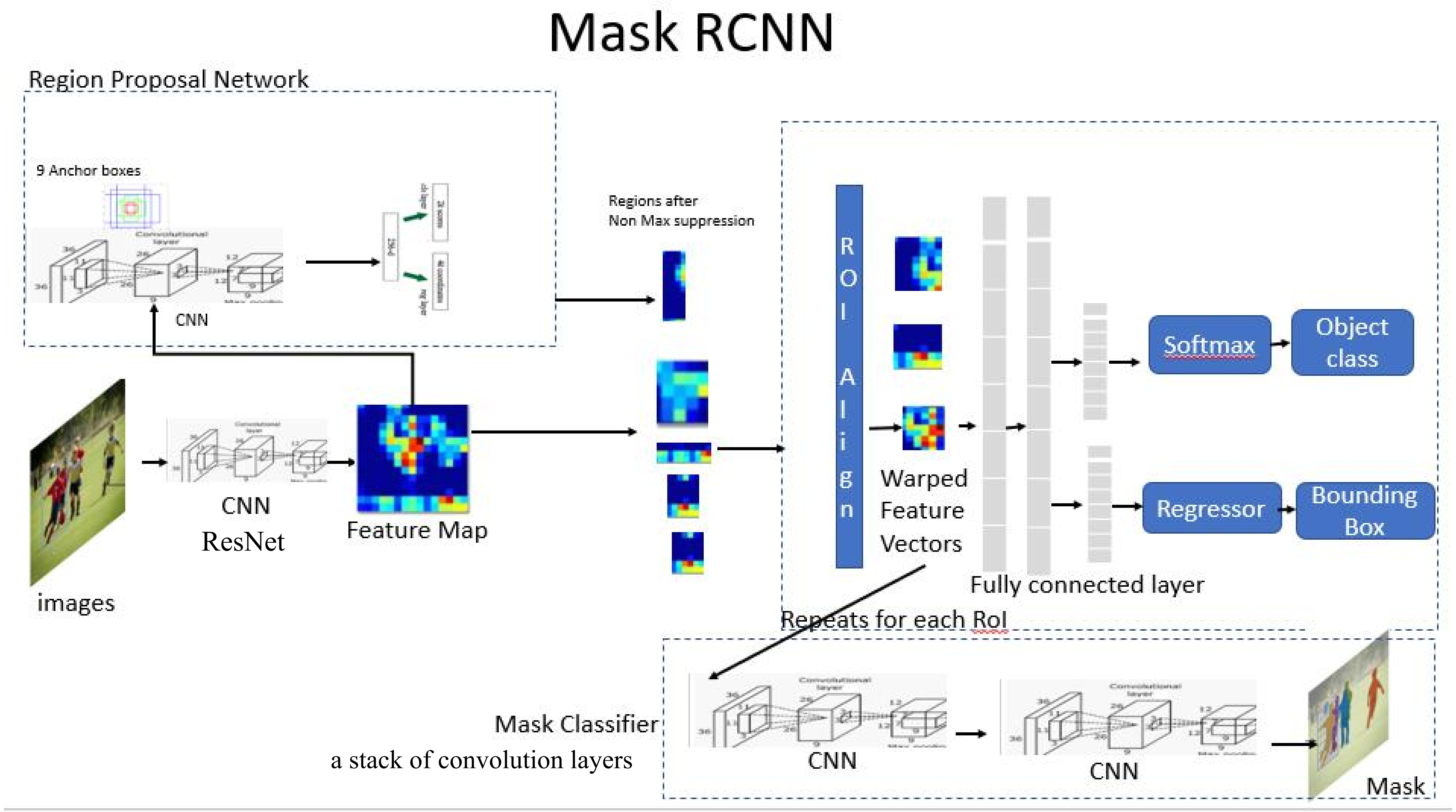
Phase spectrum

RetinaNet



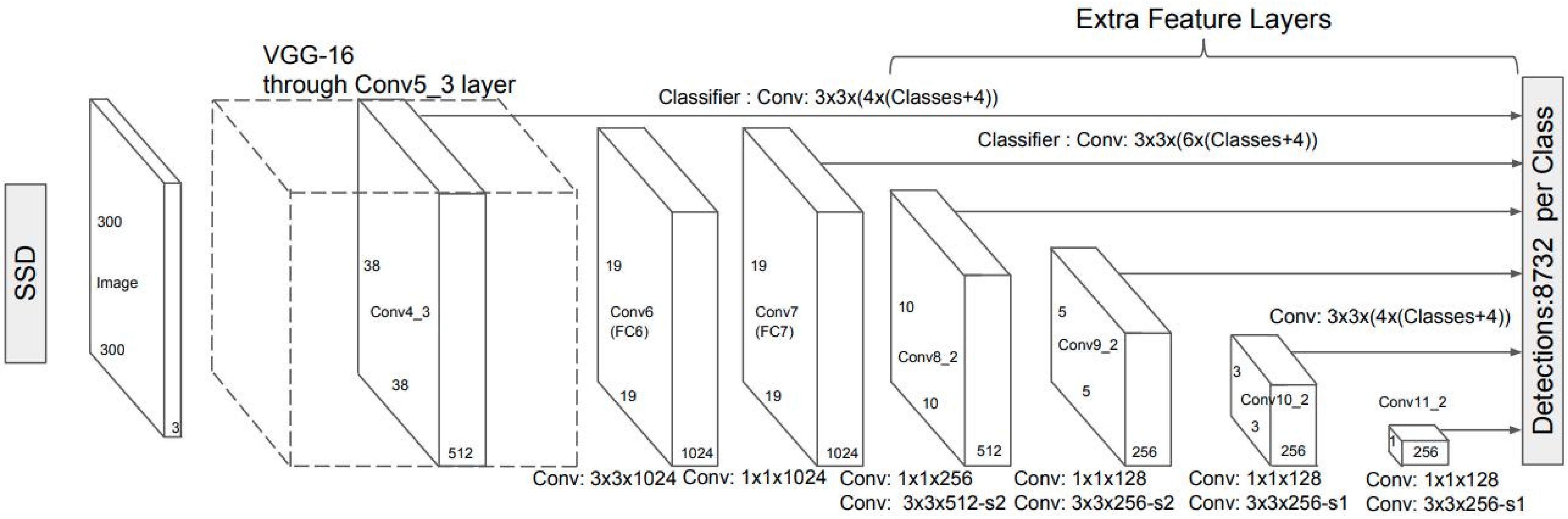
Focal Loss for Dense Object Detection, ICCV`2017

Mask R-CNN



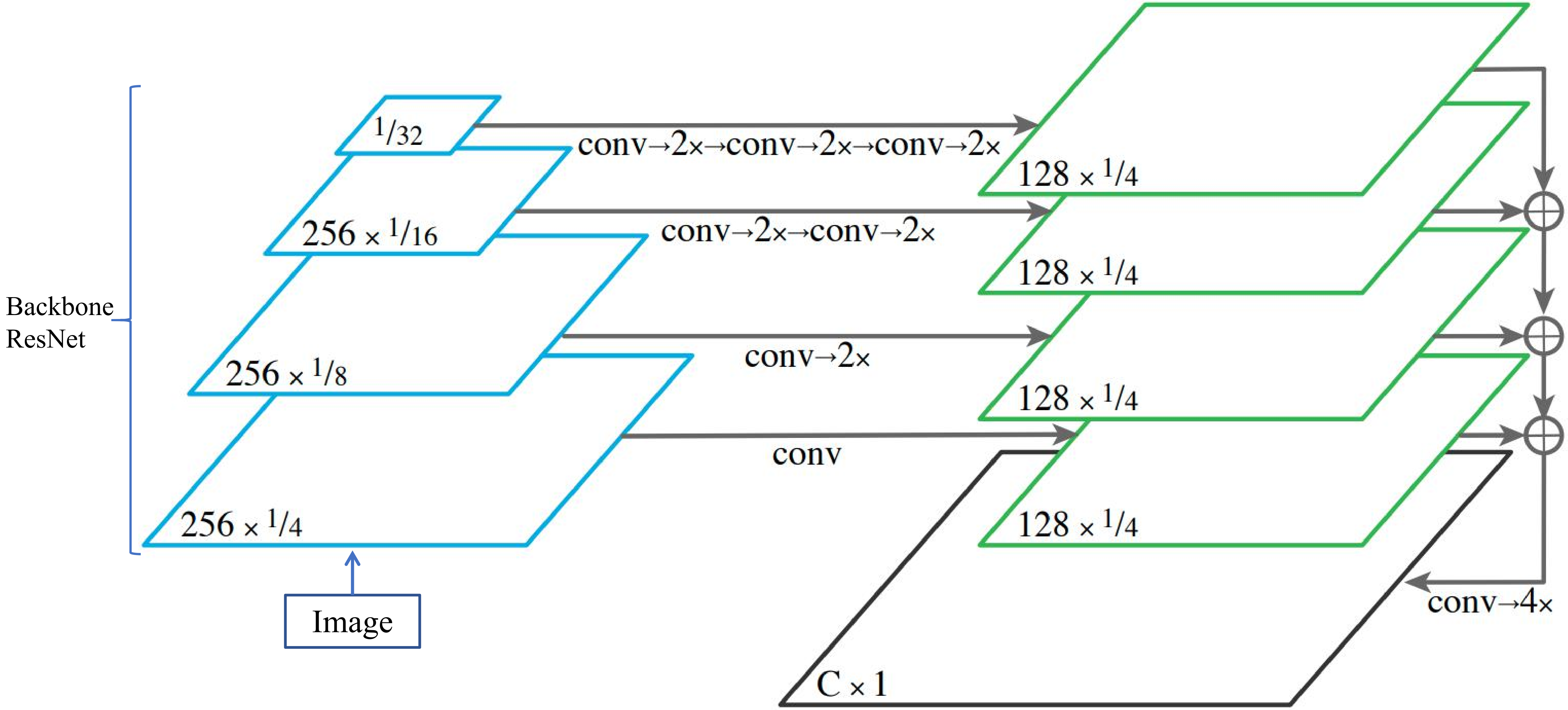
https://towardsdatascience.com/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1

SSD



SSD: Single Shot MultiBox Detector, ECCV`2016

Semantic FPN



Panoptic Feature Pyramid Networks, CVPR`2019

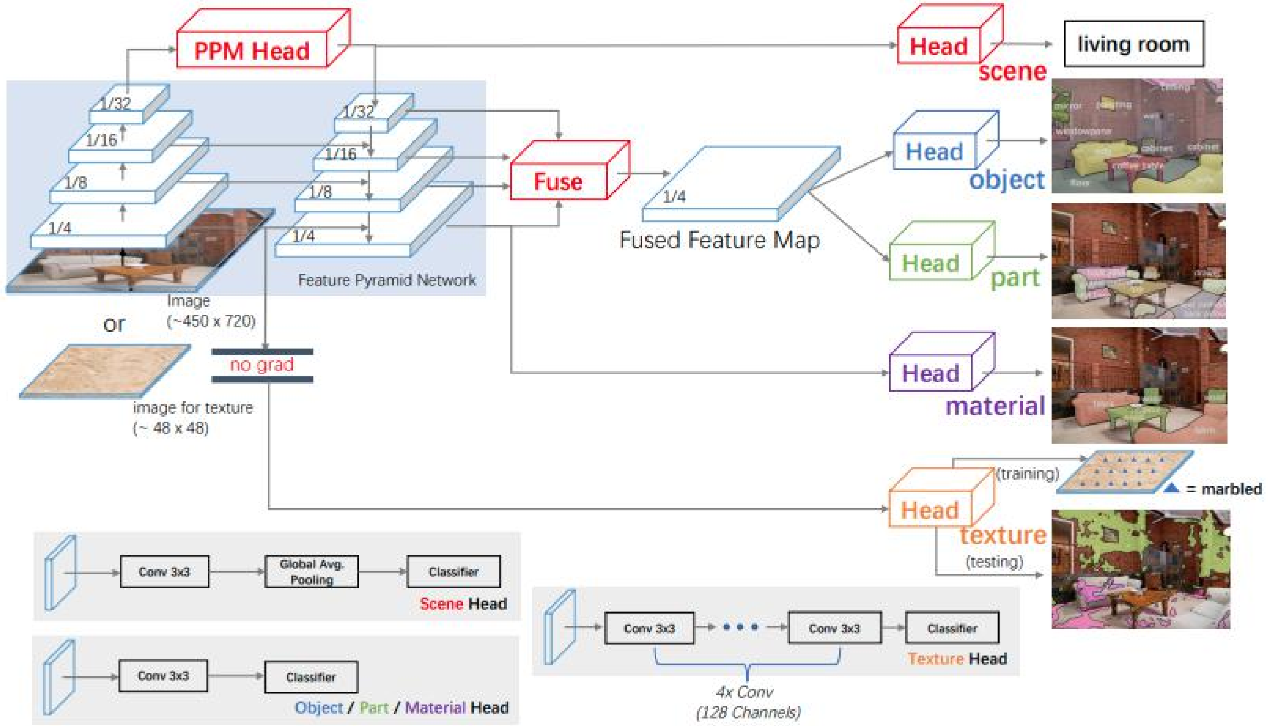
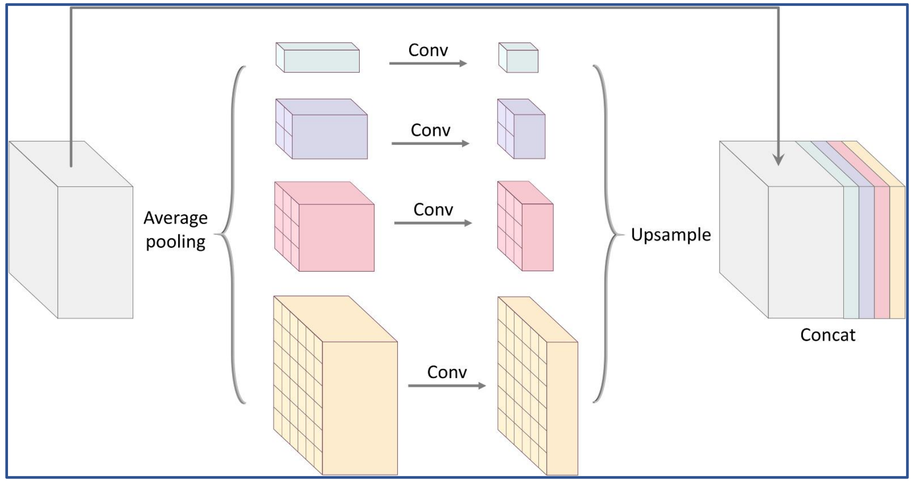


Fig. 4. UPerNet framework for Unified Perceptual Parsing. Top-left: The Feature Pyramid Network (FPN) [31] with a Pyramid Pooling Module (PPM) [16] appended on the last layer of the back-bone network before feeding it into the top-down branch in FPN. Top-right: We use features at various semantic levels. Scene head is attached on the feature map directly after the PPM since image-level information is more suitable for scene classification. Object and part heads are attached on the feature map fused by all the layers put out by FPN. Material head is attached on the feature map in FPN with the highest resolution. Texture head is attached on the Res-2 block in ResNet [1], and fine-tuned after the whole network finishes training on other tasks. Bottom: The illustrations of different heads. Details can be found in Section 3.

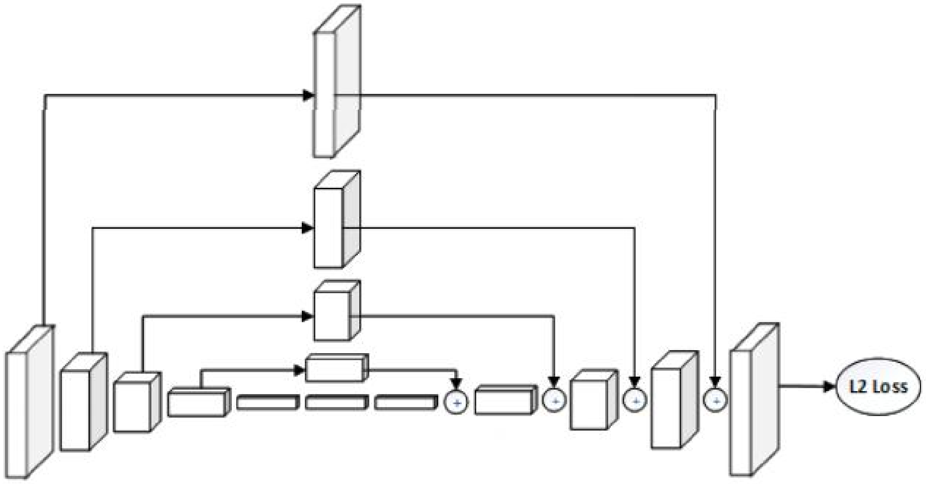


Pyramid Pooling Module (PPM)

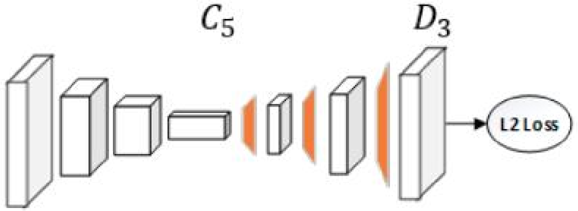
Unified Perceptual Parsing for Scene Understanding, ECCV`2018

UPerNet

Parts
Materials
- sofa is composed of seat cushion, arm, back pillow and seat base.
- coffee table is composed of leg, apron and top.
back pillow
door
drawer
arm
seat cushion
top
seat base
apron
leg
seat cushion
back pillow
brick
fabric
wood
wood
wood
carpet
fabric
- cabinet is made of wood.
- coffee table is made of wooc
- wall is made of brick.
- sofa is made of fabric.
- floor is made of carpet.
Scene
Objects
Textures
Living room
ceiling
mirror
painting
wall
windowpane
cabinet
sofa
cabinet
coffee table
floor
sofa
stratified
stratified
knitted
waffled
porous
- coffee table is waffled.
- sofa is knitted.
- wall is stratified.
- living room is composed of wall, floor, ceiling, coffee table, cabinet and painting.
scene
object
part
material
texture
zen garden
windowpane
wall
flower
bottle
vase
plate
chair
table
window
window
shop window
ceramic
painted
wood
fabric
grooved
runway
sky
building
bridge
tree
wall
plant
wall
river
arm
bed
cabinet
seat cushion
back pillow
arm
seat cushion
metal
wood
knitted

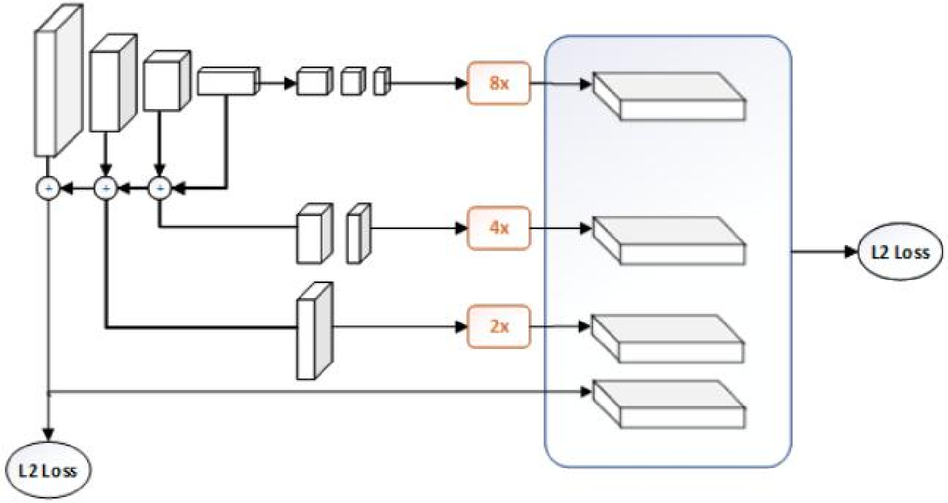


(a) Hourglass

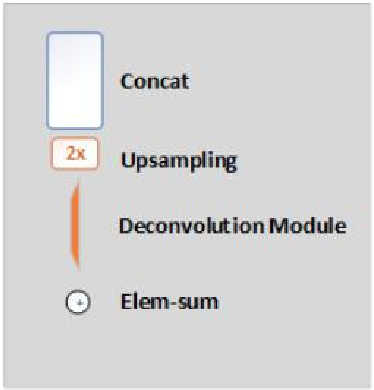


ResNet + transposed convolution

(c) Our Network



(b) CPN

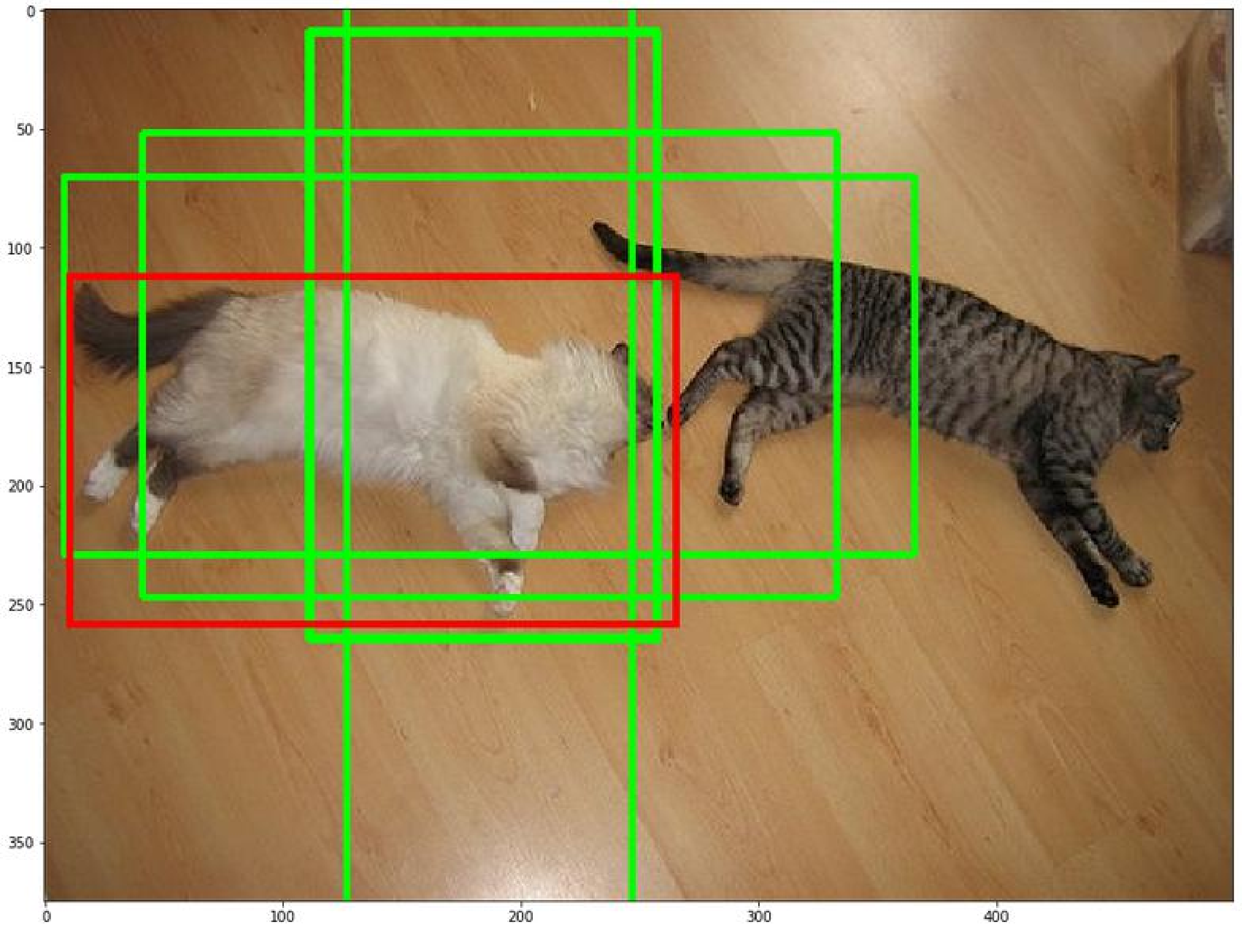


Simple Baselines for Human Pose Estimation and Tracking, ECCV`2018

Keypoint Heatmaps

Stacked Hourglass Networks for Human Pose Estimation, ECCV`2016

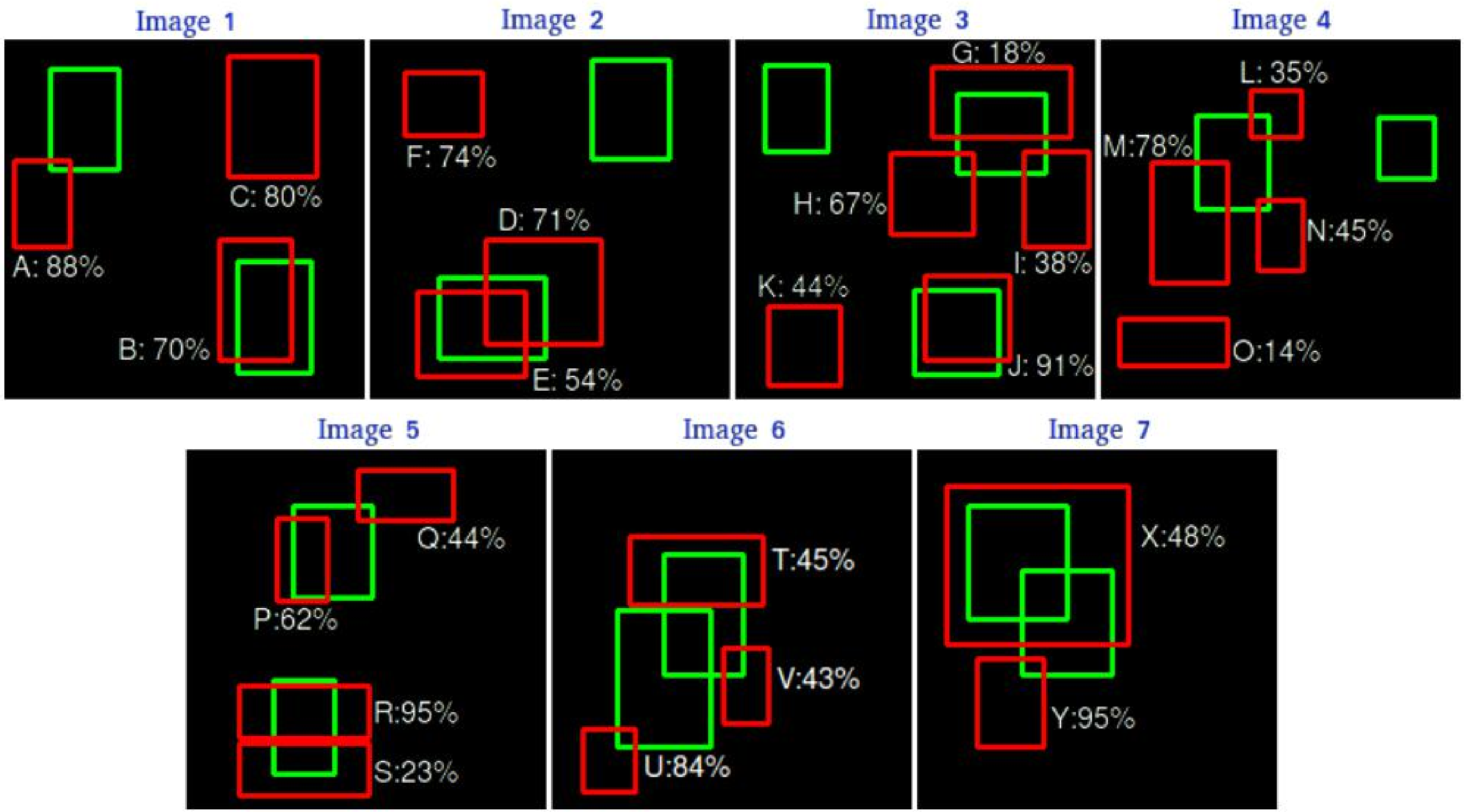
Prediction Formatting



dx	dy	dw	dh	L

Object Detection AP (1/6)

- https://github.com/rafaelpadilla/Object-Detection-Metrics



15 ground truth boxes and 26 predicted boxes

Object Detection AP (2/6)

Images	Detections	Confidences	TP or FP
Image 1	A	88%	FP
Image 1	B	70%	TP
Image 1	C	80%	FP
Image 2	D	71%	FP
Image 2	E	54%	TP
Image 2	F	74%	FP
Image 3	G	18%	TP
Image 3	H	67%	FP
Image 3	I	38%	FP
Image 3	J	91%	TP
Image 3	K	44%	FP
Image 4	L	35%	FP
Image 4	M	78%	FP
Image 4	N	45%	FP
Image 4	O	14%	FP
Image 5	P	62%	TP
Image 5	Q	44%	FP
Image 5	R	95%	TP
Image 5	S	23%	FP
Image 6	T	45%	FP
Image 6	U	84%	FP
Image 6	V	43%	FP
Image 7	X	48%	TP
Image 7	Y	95%	FP

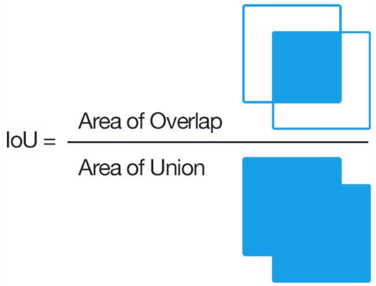
Object Detection AP (3/6)

Images	Detections	Confidences	TP	FP	Acc TP	Acc FP	Precision	Recall
Image 5	R	95%	1	0	1	0	1	0.0666
Image 7	Y	95%	0	1	1	1	0.5	0.0666
Image 3	J	91%	1	0	2	1	0.6666	0.1333
Image 1	A	88%	0	1	2	2	0.5	0.1333
Image 6	U	84%	0	1	2	3	0.4	0.1333
Image 1	C	80%	0	1	2	4	0.3333	0.1333
Image 4	M	78%	0	1	2	5	0.2857	0.1333
Image 2	F	74%	0	1	2	6	0.25	0.1333
Image 2	D	71%	0	1	2	7	0.2222	0.1333
Image 1	B	70%	1	0	3	7	0.3	0.2
Image 3	H	67%	0	1	3	8	0.2727	0.2
Image 5	P	62%	1	0	4	8	0.3333	0.2666
Image 2	E	54%	1	0	5	8	0.3846	0.3333
Image 7	X	48%	1	0	6	8	0.4285	0.4
Image 4	N	45%	0	1	6	9	0.4	0.4
Image 6	T	45%	0	1	6	10	0.375	0.4
Image 3	K	44%	0	1	6	11	0.3529	0.4

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}},$$

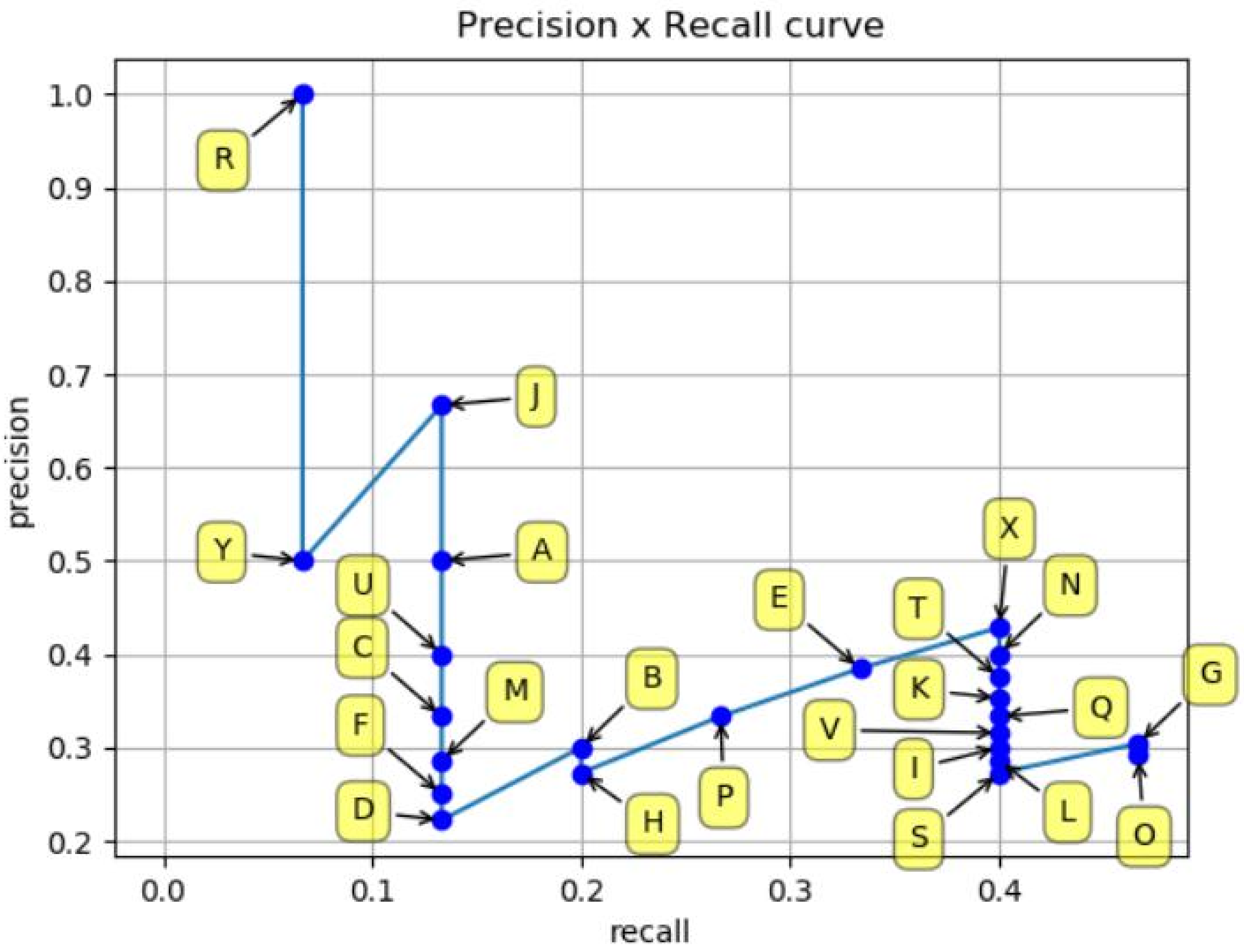
$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

TP if **IoU**(pred, gt) >= threshold
else FPs

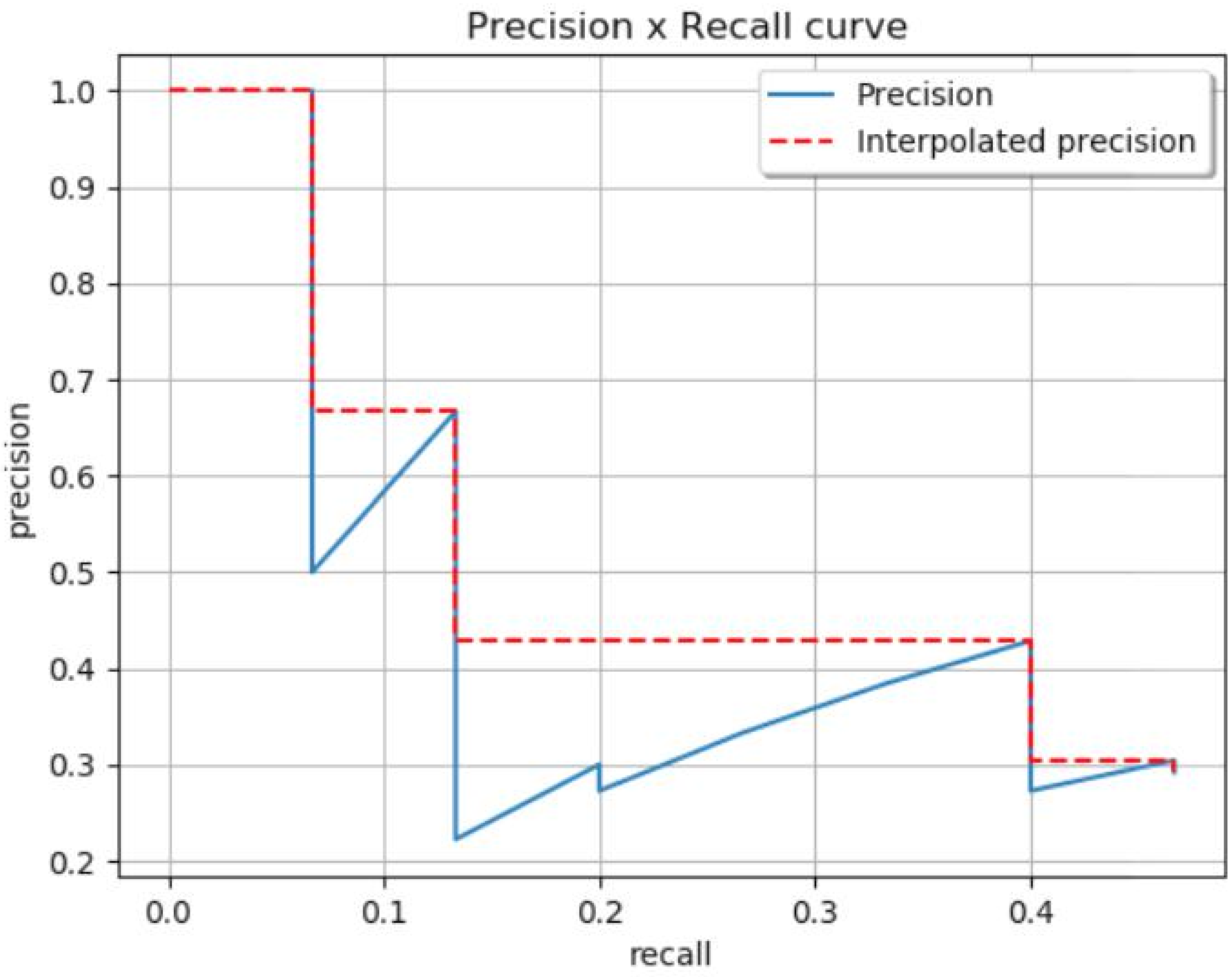


- ✓ mask: IoU based on binary masks and logical operations
- ✓ keypoint: IoU based on L2 distance term

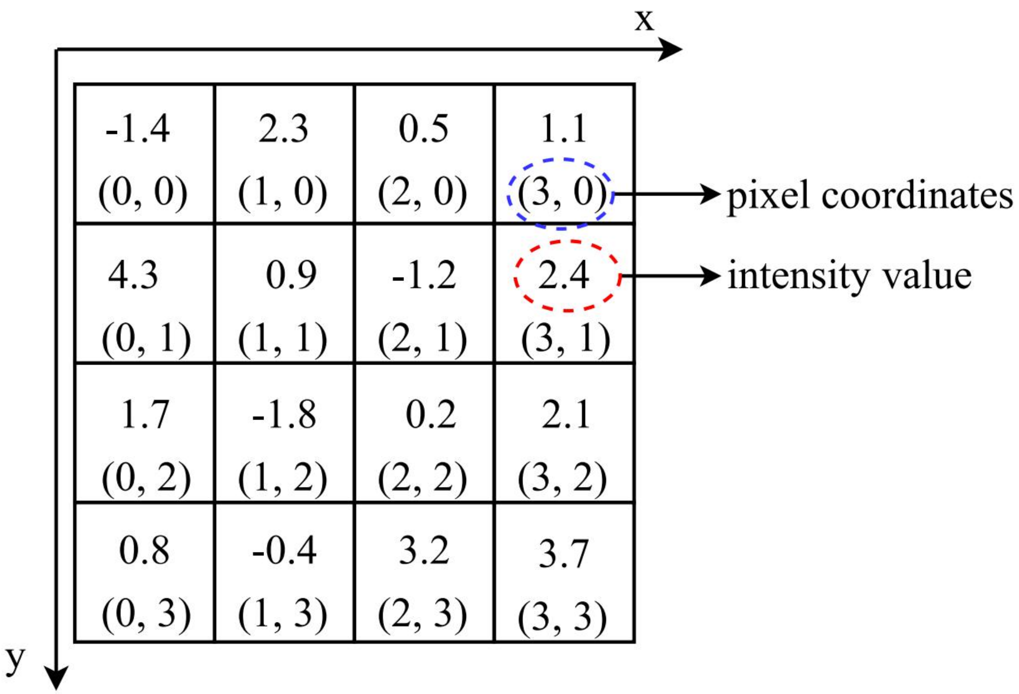
Object Detection AP (4/6)

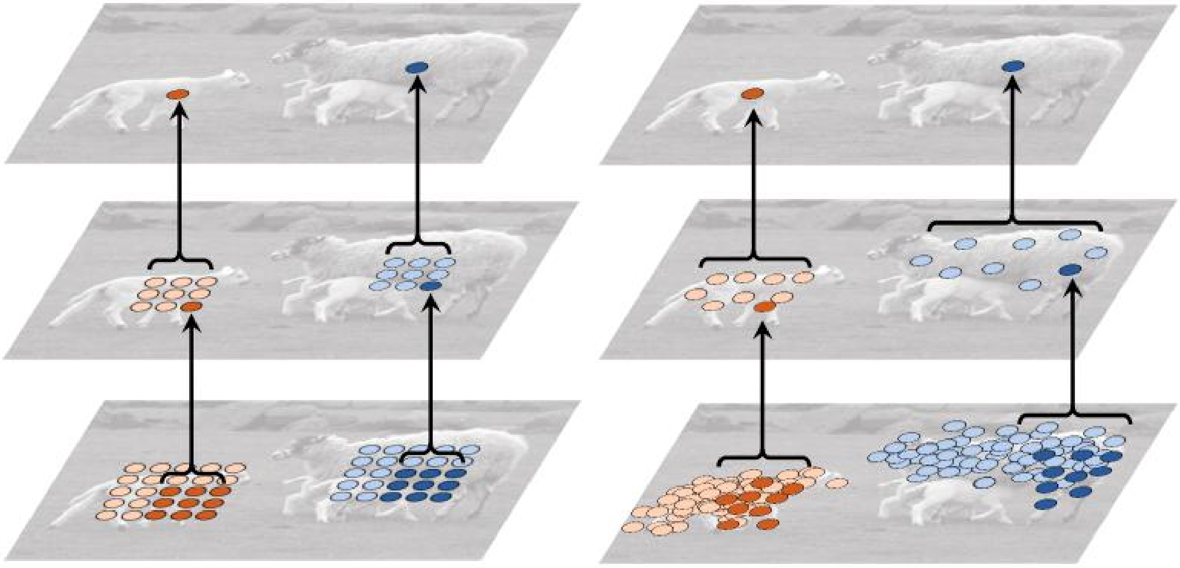


Object Detection AP (5/6)



Reference Points and Image Content





common patch embedding
- uniform way

bilinear patch embedding
- adaptive way

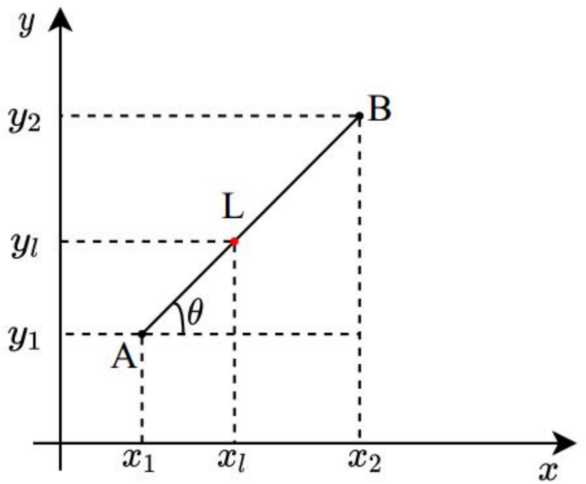
Bilinear Interpolation

Linear Interpolation

+ Given A, B

+ Target position x_l . Find y_l ?

Assume that the function behaves linearly between two known points



$$tan(\theta) = \frac{y_l - y_1}{x_l - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

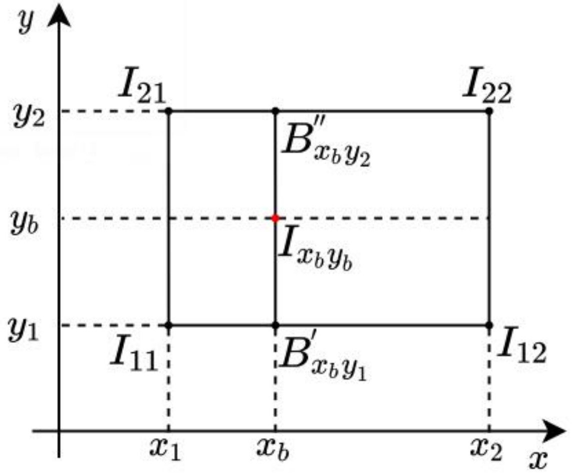
$$\Rightarrow y_l = \frac{x_2 - x_l}{x_2 - x_1} * y_1 + \frac{x_l - x_1}{x_2 - x_1} * y_2$$

Bilinear Interpolation

Given intensity values at four pixel locations

$$I_{11}, I_{12}, I_{21}, I_{22}$$

Find intensity value $I_{x_b y_b}$ at given point (x_b, y_b)



Compute linear interpolation for $B'_{x_b y_1}, B''_{x_b y_2}$

$$B'_{x_b y_1} = \frac{x_2 - x_b}{x_2 - x_1} * I_{11} + \frac{x_b - x_1}{x_2 - x_1} * I_{12}$$

$$B''_{x_b y_2} = \frac{x_2 - x_b}{x_2 - x_1} * I_{21} + \frac{x_b - x_1}{x_2 - x_1} * I_{22}$$

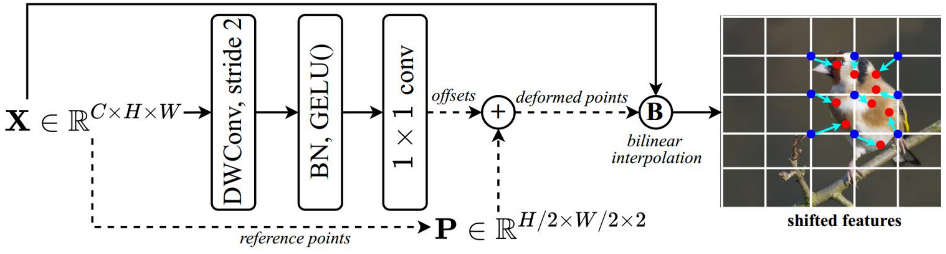
Compute linear interpolation for $I_{x_b y_b}$

$$I_{x_b y_b} = \frac{y_2 - y_b}{y_2 - y_1} * B'_{x_b y_1} + \frac{y_b - y_1}{y_2 - y_1} * B''_{x_b y_2}$$

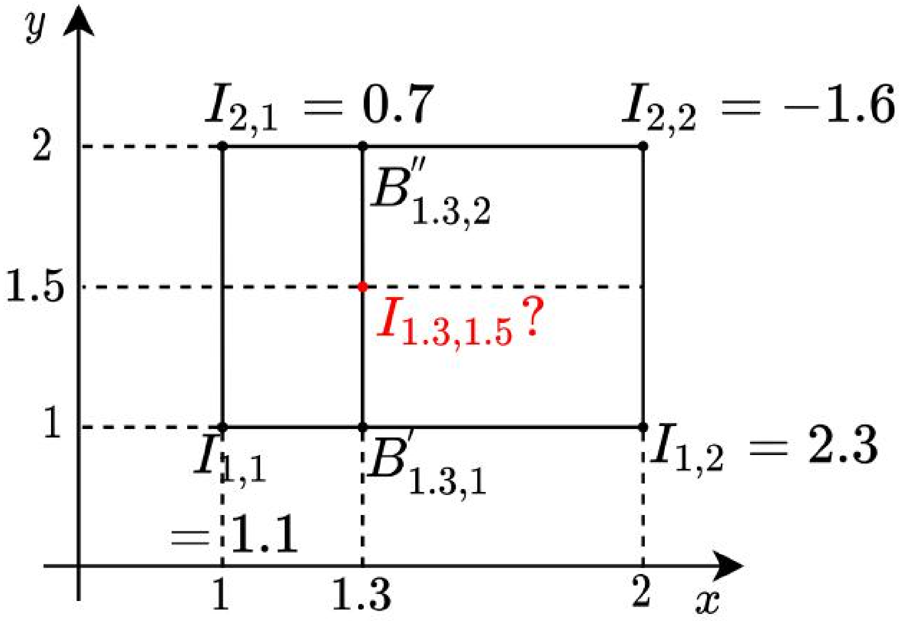
In our Bilinear PE, (x_b, y_b) is learned by small network where parameters are conditioned on the image content.

$$(x_b, y_b) = (x_i, y_j) + N(I)$$

$N(I) = (\Delta_x, \Delta_y)$: offsets



Bilinear Interpolation



$$B'_{1.3,1} = \frac{2-1.3}{2-1} * 1.1 + \frac{1.3-1}{2-1} * 2.3 = 1.46$$

$$B''_{1.3,2} = \frac{2-1.3}{2-1} * 0.7 + \frac{1.3-1}{2-1} * (-1.6) = 0.01$$

$$I_{1.3,1.5} = \frac{2-1.5}{2-1} * 1.46 + \frac{1.5-1}{2-1} * 0.01 = 0.735$$

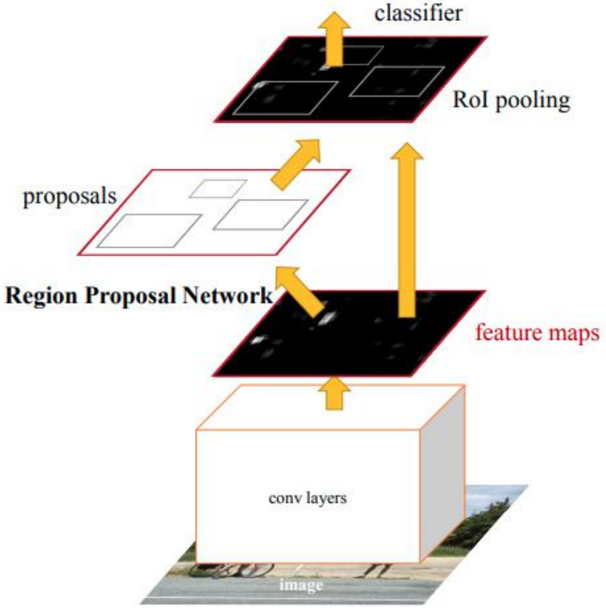
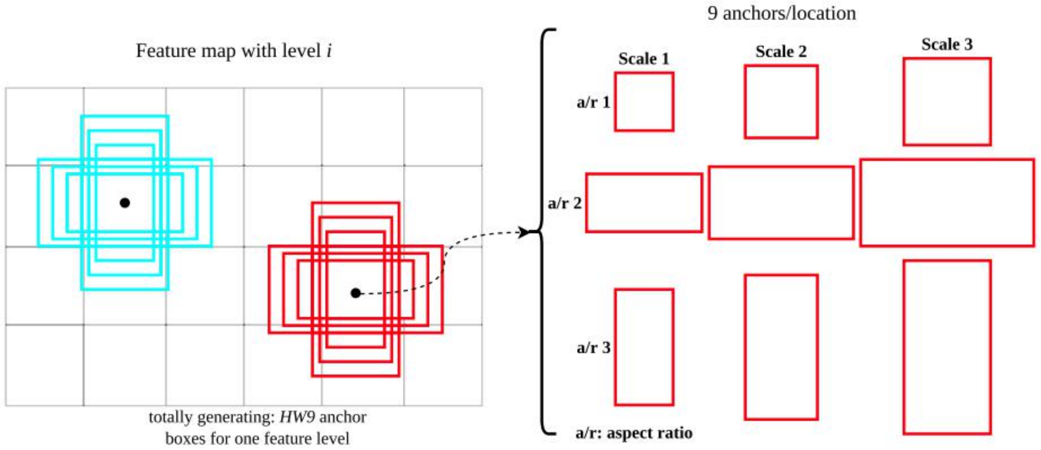
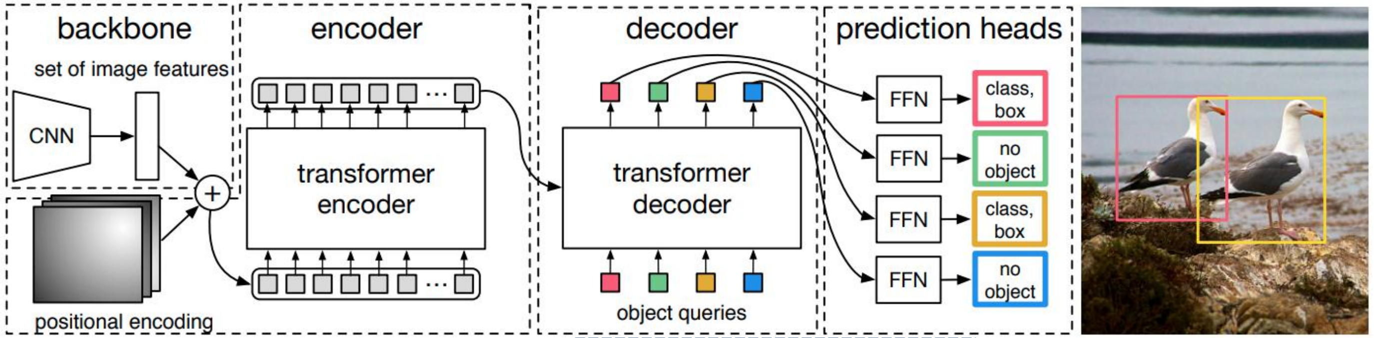


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the ‘attention’ of this unified network.



Detection Transformer

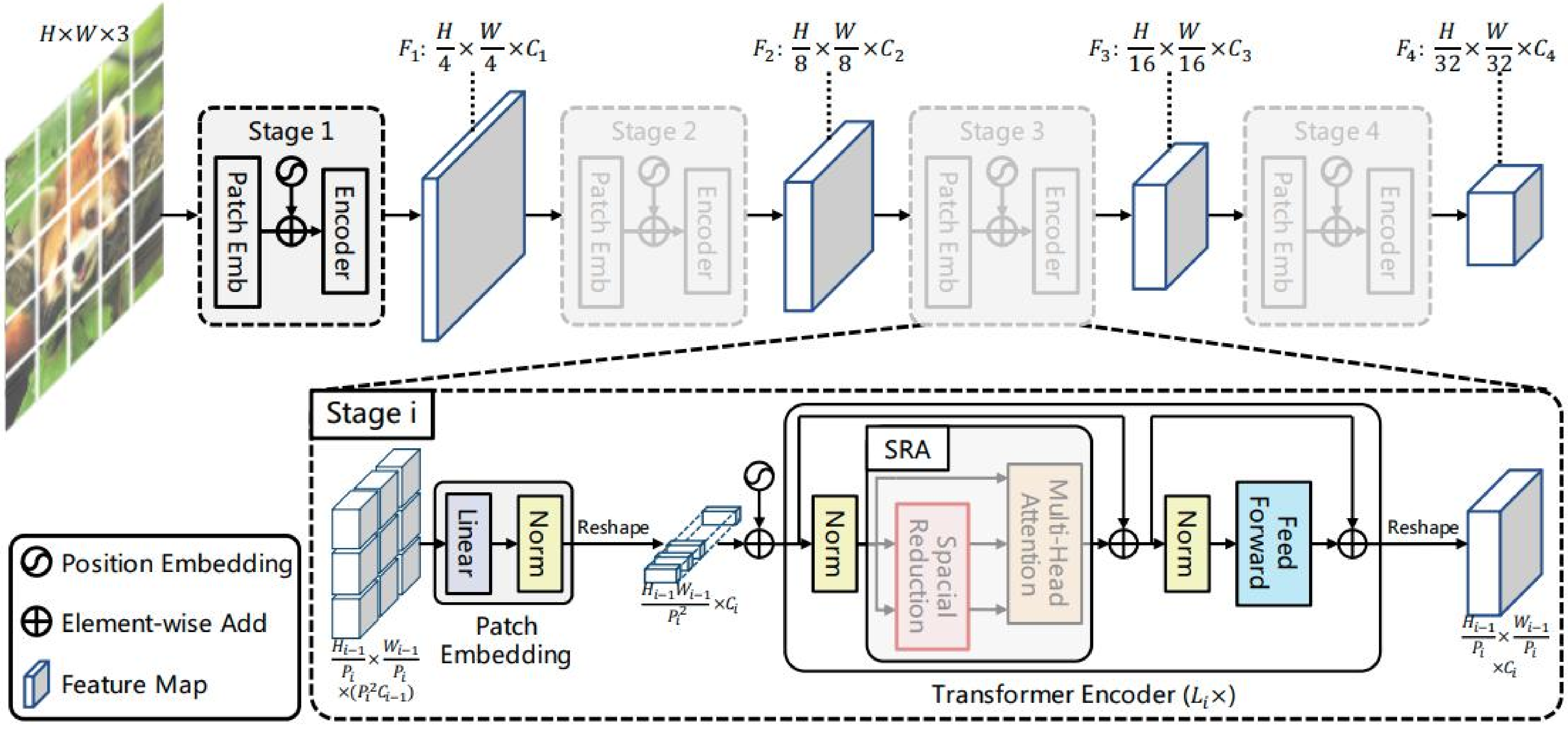


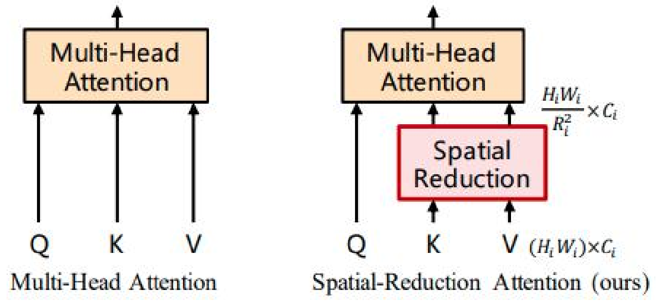
N×4 (4: x, y, w, h)

object queries are *learnable* and *interacted with image features* to reason about box prediction

PVT-Pyramid Vision Transformer

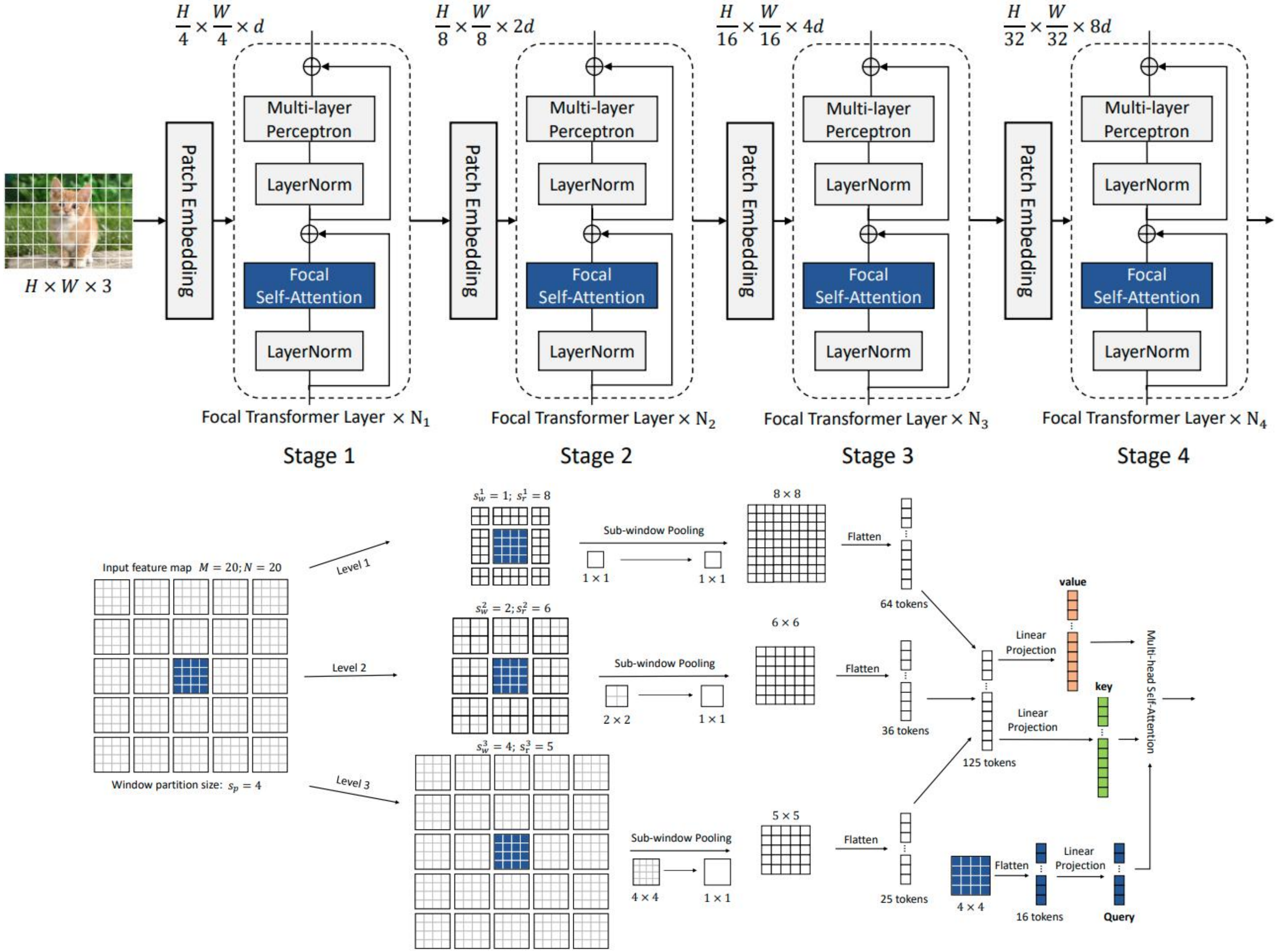
▶ Key idea: spatial reduction attention





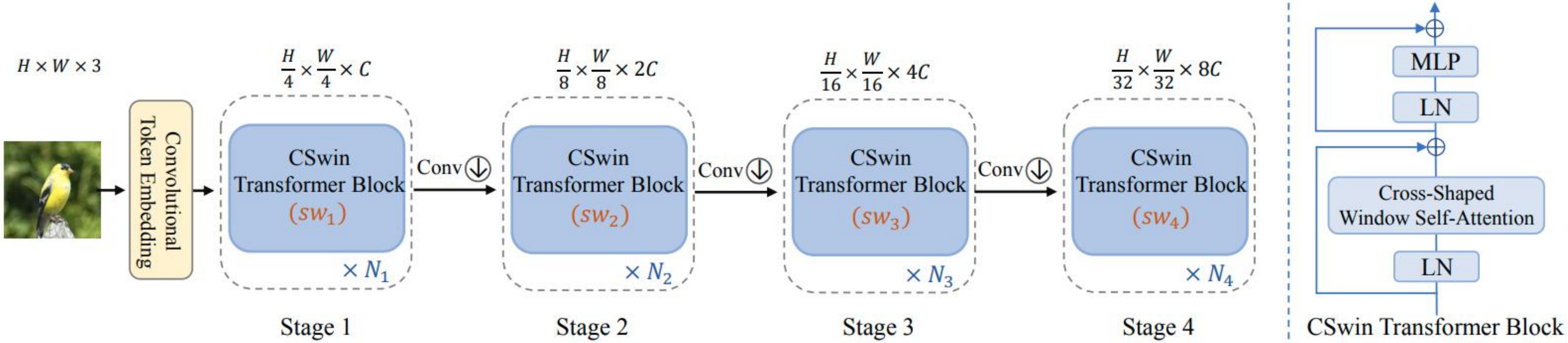
Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction Without Convolutions, ICCV`2021

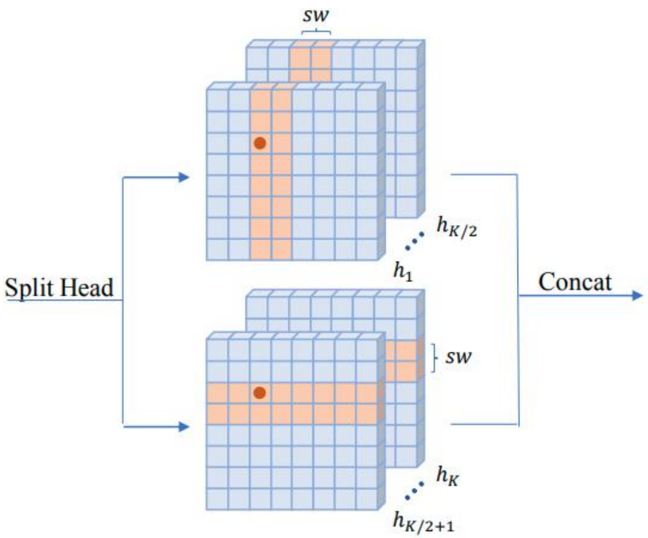
Focal ViT: *multi-scale self-attentions*



FocalViT` NeurIPS`2021

CSWin Transformer

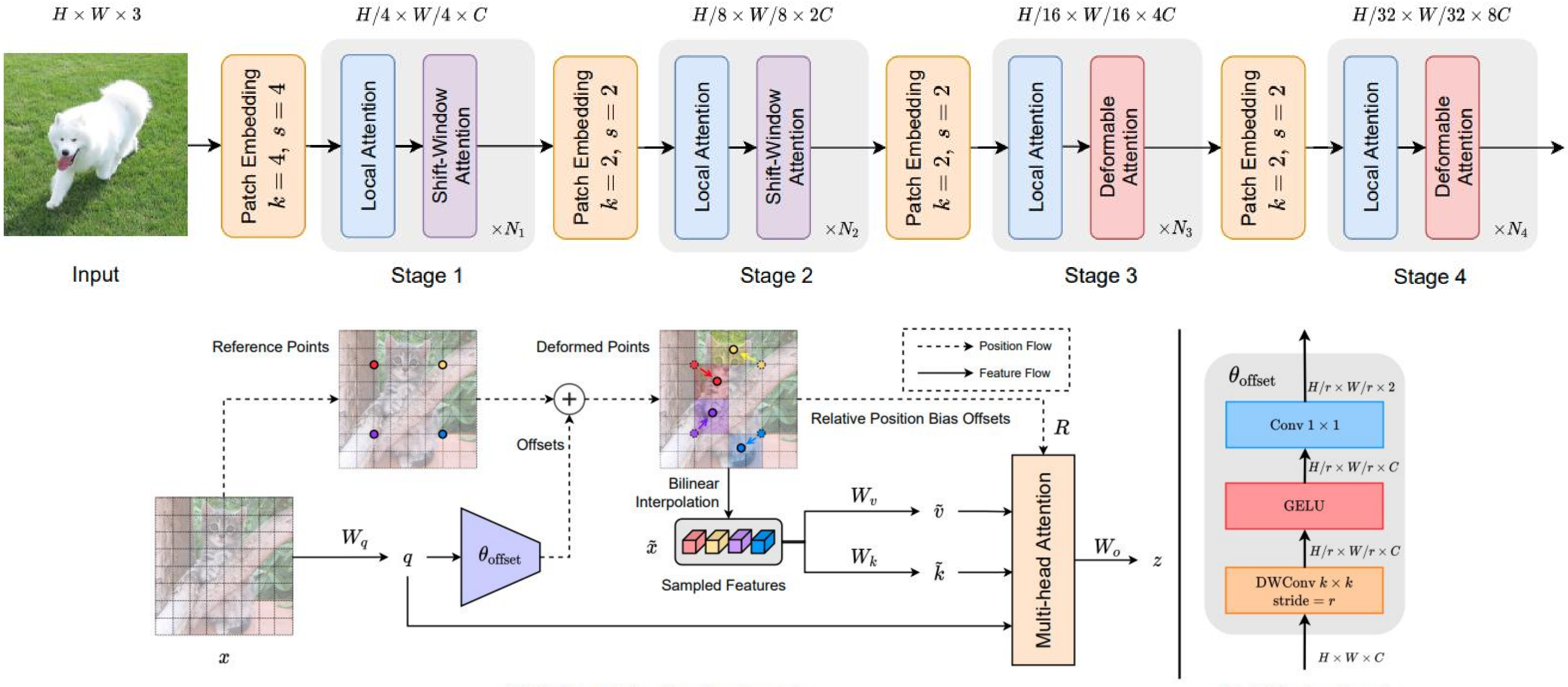




Dynaic Stripe Window + Parallel Grouing Heads = CSWin

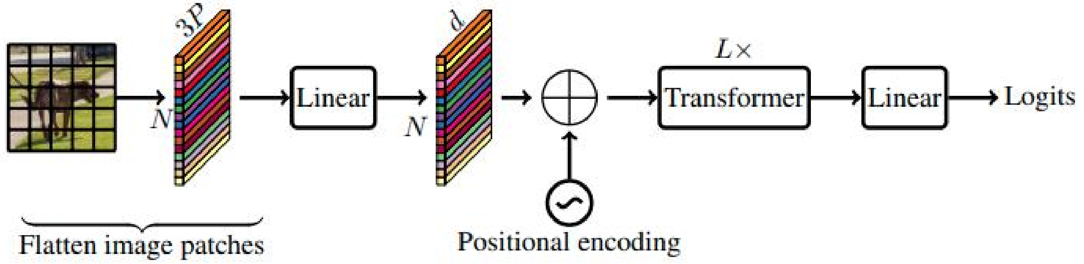
CSWin Transformer: A General Vision Transformer Backbone With Cross-Shaped Windows, CVPR`2022

DAT - Deformable Attention

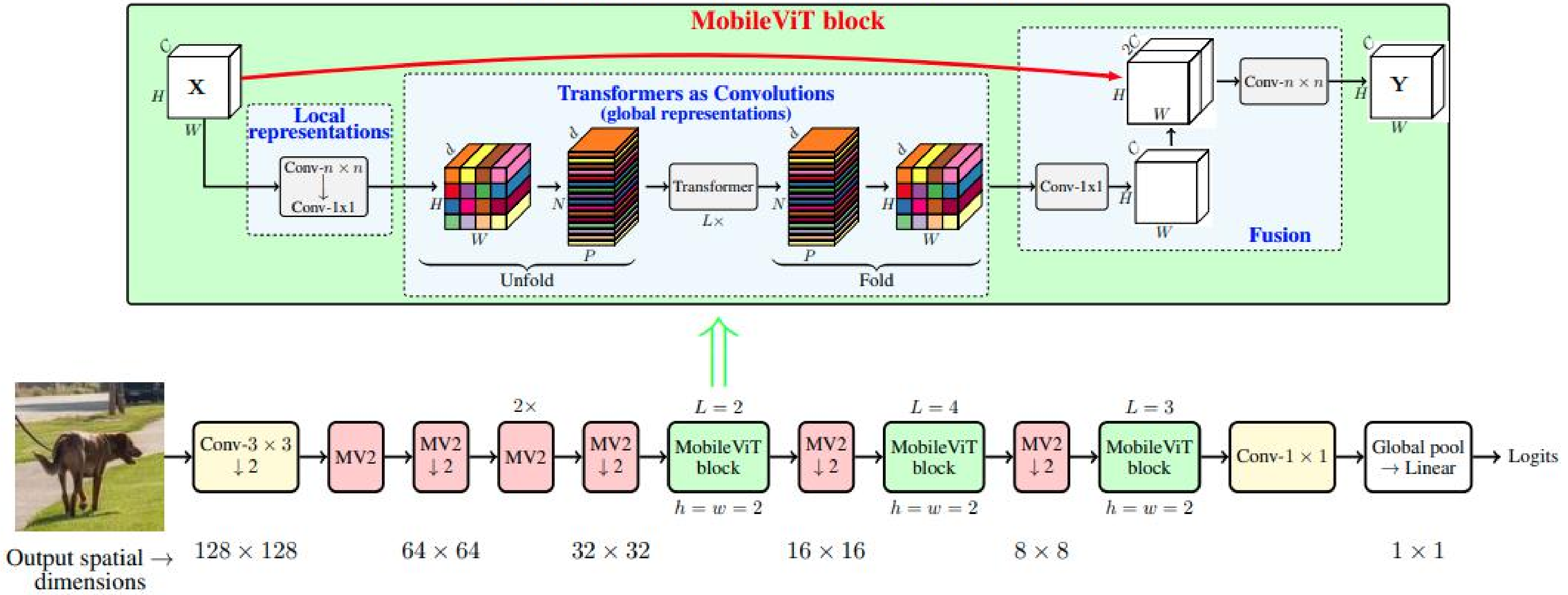


(a) Deformable attention module **(b) Offset network**

Vision Transformer With Deformable Attention, CVPR`2022



(a) **Standard visual transformer (ViT)**



(b) **MobileViT**. Here, Conv-$n \times n$ in the MobileViT block represents a standard $n \times n$ convolution and MV2 refers to MobileNetv2 block. Blocks that perform down-sampling are marked with ↓ 2.

Figure 1: **Visual transformers vs. MobileViT**

Layer	Output size	Output stride	Repeat	Output channels		
				XXS	XS	S
Image	256×256	1				
Conv-3×3, $\downarrow 2$ MV2	128×128	2	1 1	16 16	16 32	16 32
MV2, $\downarrow 2$ MV2	64×64	4	1 2	24 24	48 48	64 64
MV2, $\downarrow 2$ MobileViT block ($L = 2$)	32×32	8	1 1	48 48 ($d = 64$)	64 64 ($d = 96$)	96 96 ($d = 144$)
MV2, $\downarrow 2$ MobileViT block ($L = 4$)	16×16	16	1 1	64 64 ($d = 80$)	80 80 ($d = 120$)	128 128 ($d = 192$)
MV2, $\downarrow 2$ MobileViT block ($L = 3$) Conv-1×1	8×8	32	1 1 1	80 80 ($d = 96$) 320	96 96 ($d = 144$) 384	160 160 ($d = 240$) 640
Global pool Linear	1×1	256	1	1000	1000	1000
Network Parameters				1.3 M	2.3 M	5.6 M

Table 4: **MobileViT architecture.** Here, d represents dimensionality of the input to the transformer layer in MobileViT block (Figure 1b). By default, in MobileViT block, we set kernel size n as three and spatial dimensions of patch (height h and width w) in MobileViT block as two.

MobileViTv2 Block

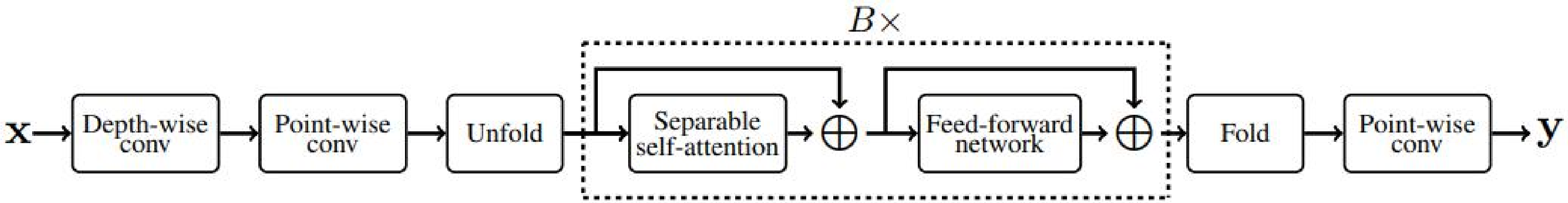
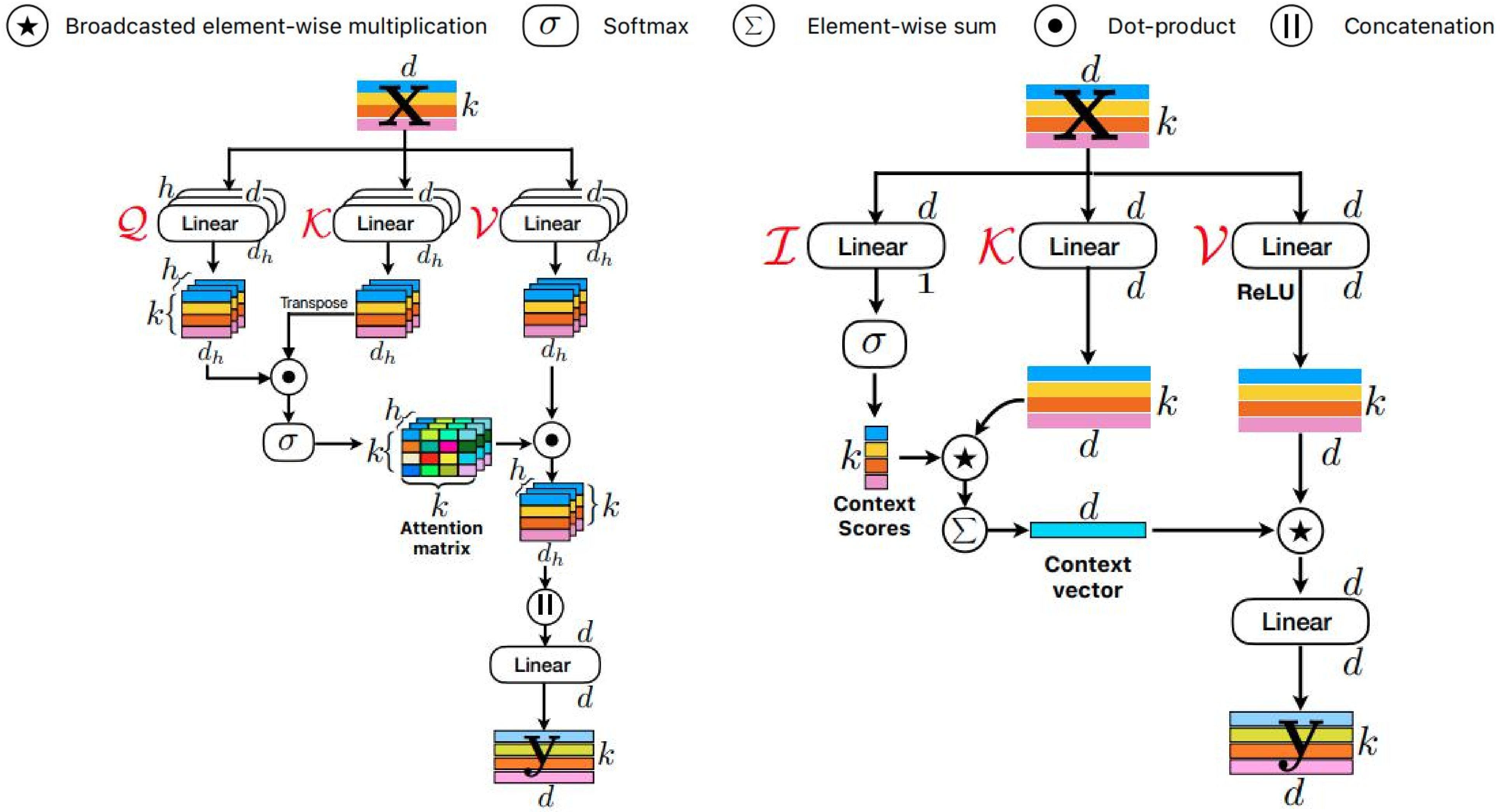


Figure 6: `MobileViTv2` **block**. Here, depth-wise convolution uses a kernel size of 3×3 to encode local representations. Similar to [4], unfolding and folding operations uses a patch height and width of two respectively. The separable self-attention and feed-forward layers are repeated $B\times$ before applying the folding operation.



(a) Transformer [1]

(b) Linear Transformer in MobileViTv2 [2]

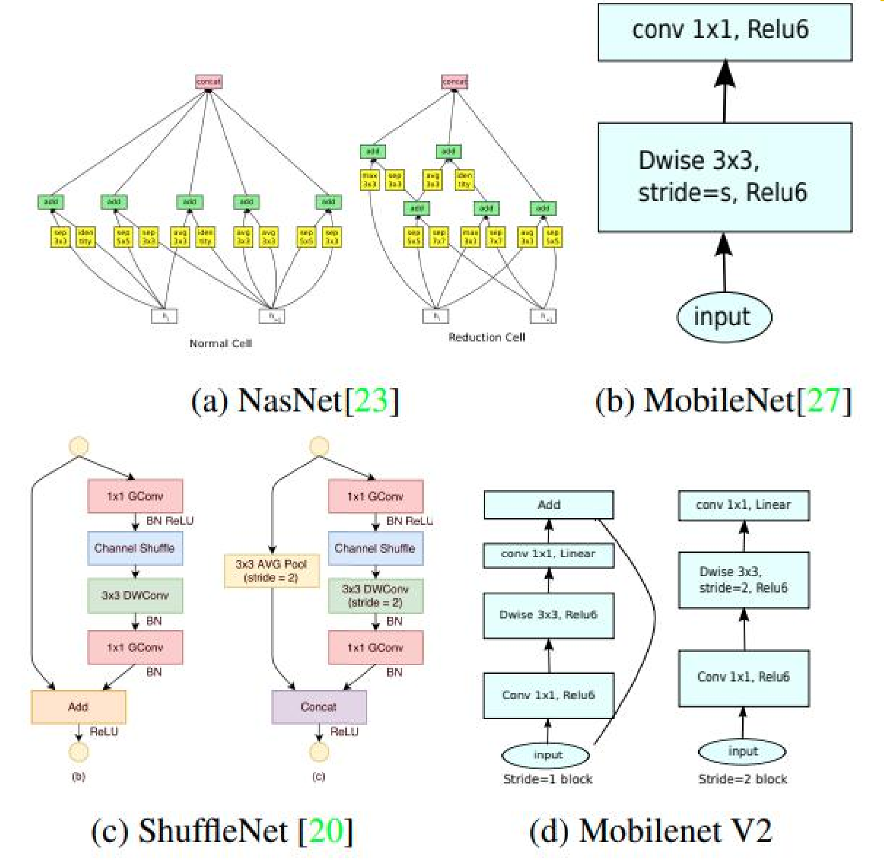
[1] Vaswani, Ashish, et al. "Attention is all you need." NeurIPS`2017
[2] Mehta, Sachin, and Mohammad Rastegari. "Separable Self-attention for Mobile Vision Transformers." arXiv 2022

MobileViTv2

Table 5: `MobileViTv2` **architecture.** Here, d represents dimensionality of the input to the separable self-attention layer, B denotes the repetition of transformer block with separable self-attention inside the `MobileViTv2` block (Fig. 6), and MV2 indicates MobileNetv2 block. Similar to MobileViTv1 block, we set kernel size as three and spatial dimensions of patch (height h and width w) as two in the `MobileViTv2` block.

Layer	**Output size**	**Output stride**	**Repeat**	**Output channels**
Image	256×256	1		
Conv-3×3, $\downarrow 2$ MV2	128×128	2	1 1	32α 64α
MV2, $\downarrow 2$ MV2	64×64	4	1 2	128α 128α
MV2, $\downarrow 2$ `MobileViTv2` block (Fig. 6; $B = 2$)	32×32	8	1 1	256α $256 * \alpha$ ($d = 128\alpha$)
MV2, $\downarrow 2$ `MobileViTv2` block (Fig. 6; $B = 4$)	16×16	16	1 1	384α 384α ($d = 192\alpha$)
MV2, $\downarrow 2$ `MobileViTv2` block (Fig. 6; $B = 3$)	8×8	32	1 1	512α 512α ($d = 256\alpha$)
Global pool Linear	1×1	256	1	512α 1000

MobileNetV2



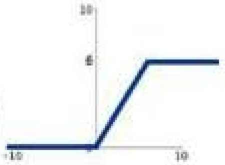
(a) NasNet[23] (b) MobileNet[27]

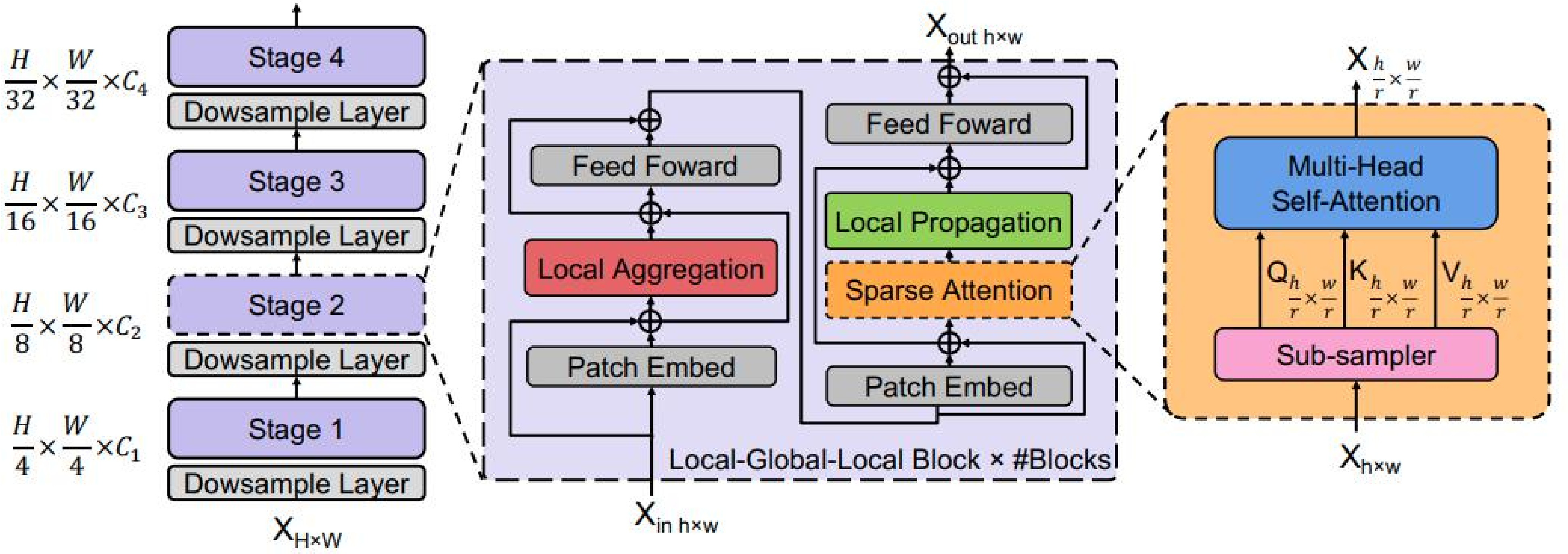
(c) ShuffleNet [20] (d) Mobilenet V2

Figure 4: Comparison of convolutional blocks for different architectures. ShuffleNet uses Group Convolutions [20] and shuffling, it also uses conventional residual approach where inner blocks are narrower than output. ShuffleNet and NasNet illustrations are from respective papers.

ReLU6

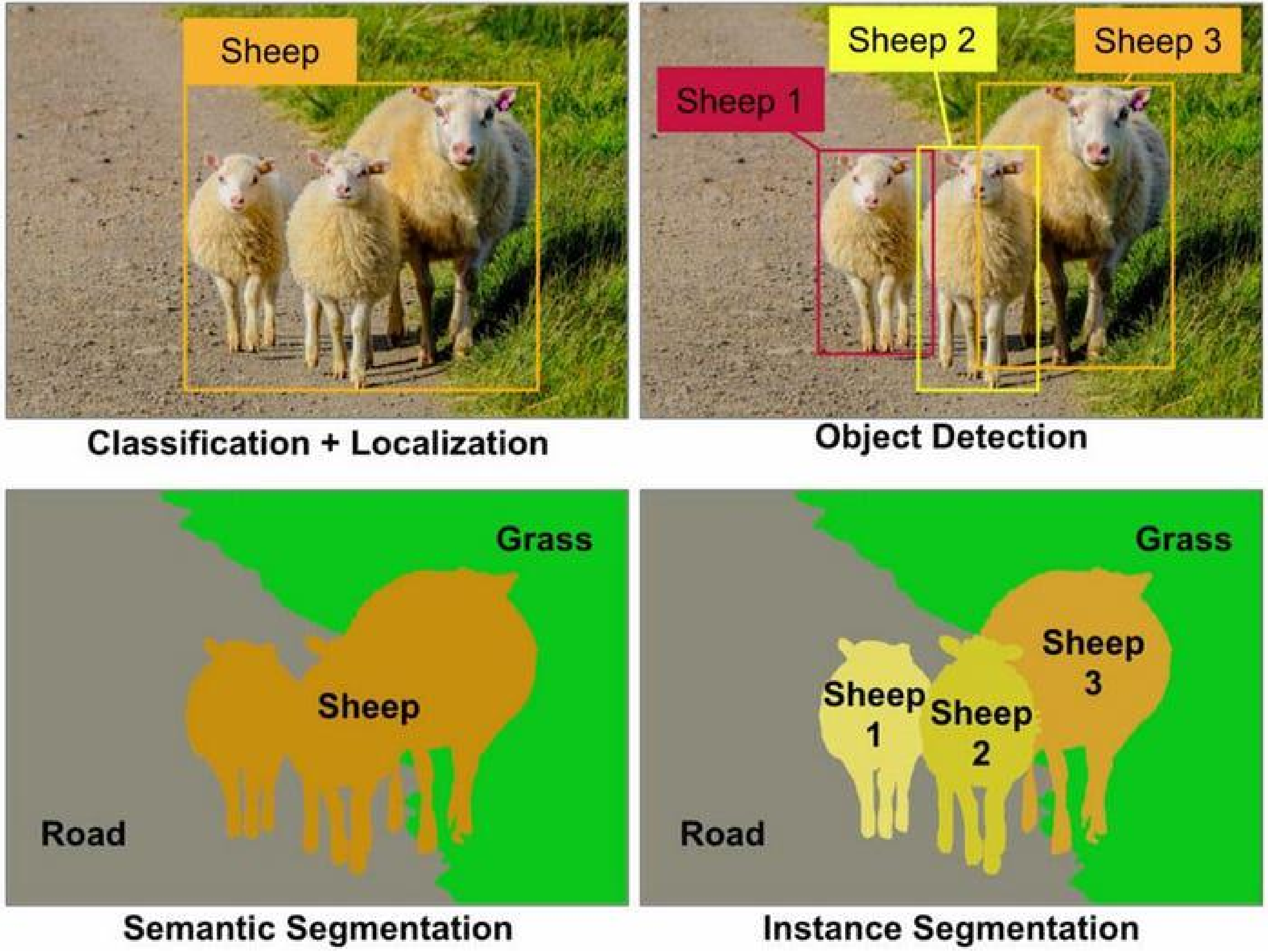
$\min(\max(0, x), 6)$





(a) Architecture b) Local-Global-Local Bottleneck (c) Global Sparse Attention

Instance and Semantic Segmentation



https://nirmalamurali.medium.com/image-classification-vs-semantic-segmentation-vs-instance-segmentation-625c33a08d50