Yejin Song
BOAZ Data Analysis Dept. 19th
yjyjsssong@gmail.com

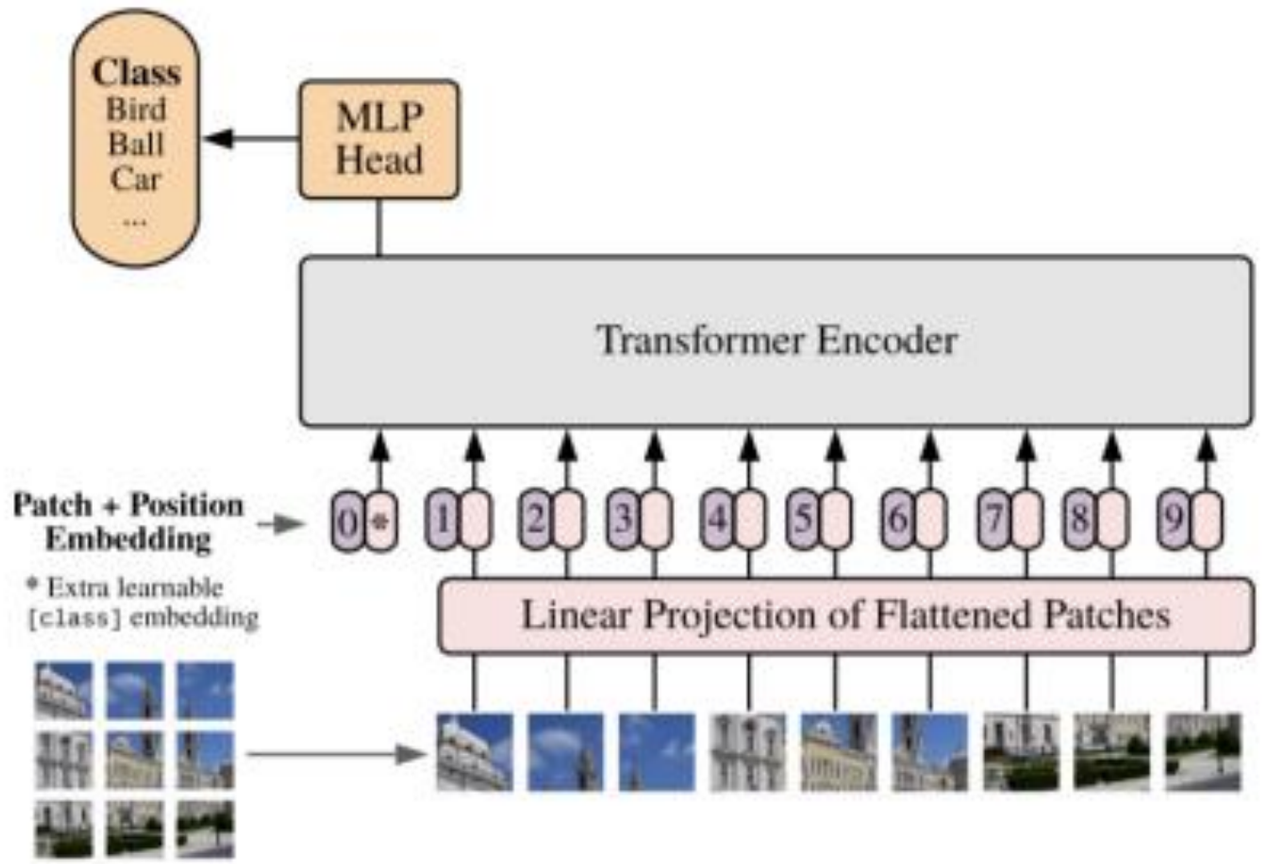
Advanced Object Detection

2023.02.07

Table of Content

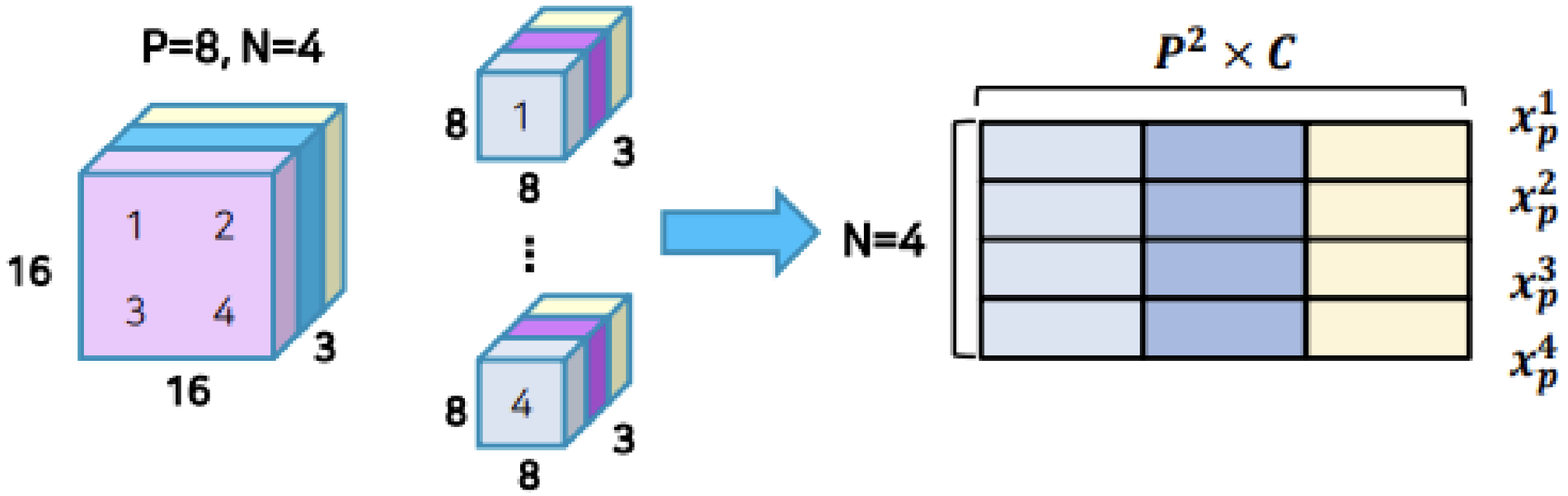
Vision Transformer (ViT)

- 1) Flatten 3D to 2D (Patch 단위로 나누기)
- 2) Learnable한 embedding 처리
- 3) Add class embedding, position embedding
- 4) Transformer
- 5) Predict



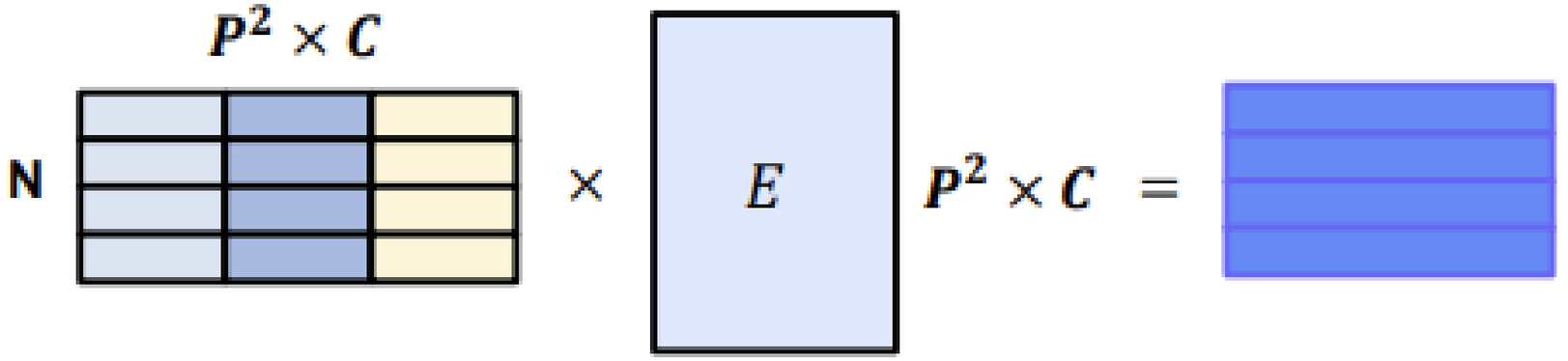
Vision Transformer (ViT)

❑ 1) Flatten 3D to 2D (Patch 단위로 나누기)



Vision Transformer (ViT)

- 2) Learnable한 embedding 처리
  - E 라는 matrix를 이용해서 학습을 가능하게 만들어줌



Vision Transformer (ViT)

- 3) Add class embedding, position embedding
  - 앞서 만들어진 embedding 값에 class embedding 추가 ([CLS]Token) •
  - 이미지의 위치 따라 학습하기 위해 position embedding 추가

Vision Transformer (ViT)

- ViT의 문제점
  - ViT의 실험부분을 보면 굉장히 많은량의 Data를 학습하여야 성능이 나옴
  - Transformer 특성상 computational cost 큼
  - 일반적인 backbone으로 사용하기 어려움

Vision Transformer (ViT)

- ViT의 문제점
  - ViT의 실험부분을 보면 굉장히 많은량의 Data를 학습하여야 성능이 나옴
  - Transformer 특성상 computational cost 큼
  - 일반적인 backbone으로 사용하기 어려움

- 해결법
  - CNN과 유사한 구조로 설계
  - Window라는 개념을 활용하여 cost를 줄임

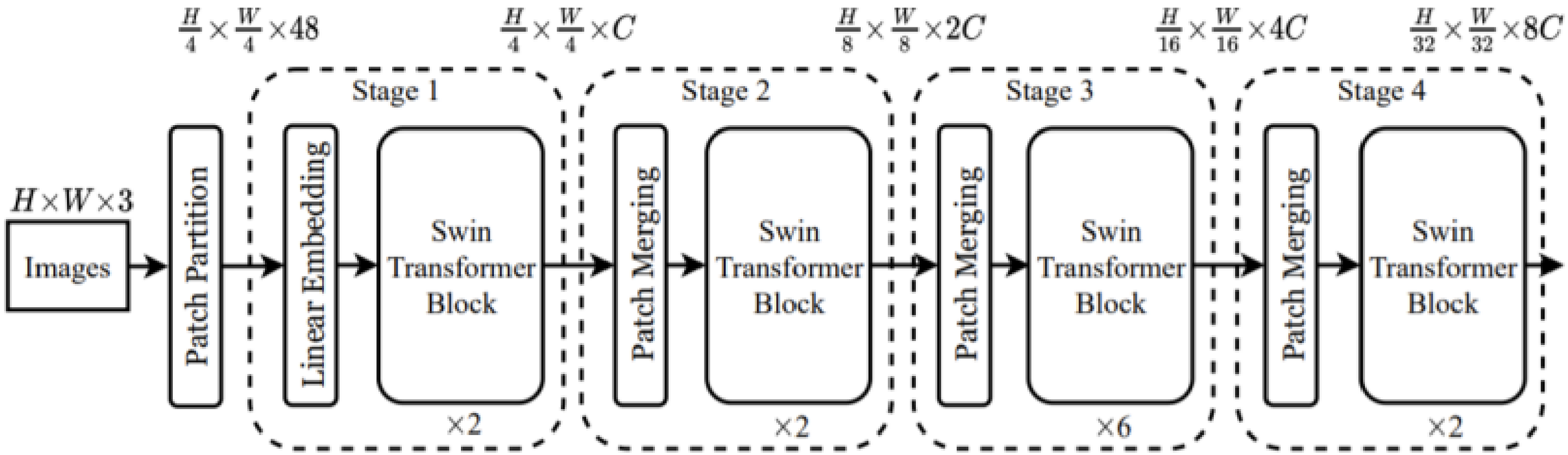
Vision Transformer (ViT)

- ViT의 문제점
  - ViT의 실험부분을 보면 굉장히 많은량의 Data를 학습하여야 성능이 나옴
  - Transformer 특성상 computational cost 큼
  - 일반적인 backbone으로 사용하기 어려움

- 해결법
  - CNN과 유사한 구조로 설계
  - Window라는 개념을 활용하여 cost를 줄임

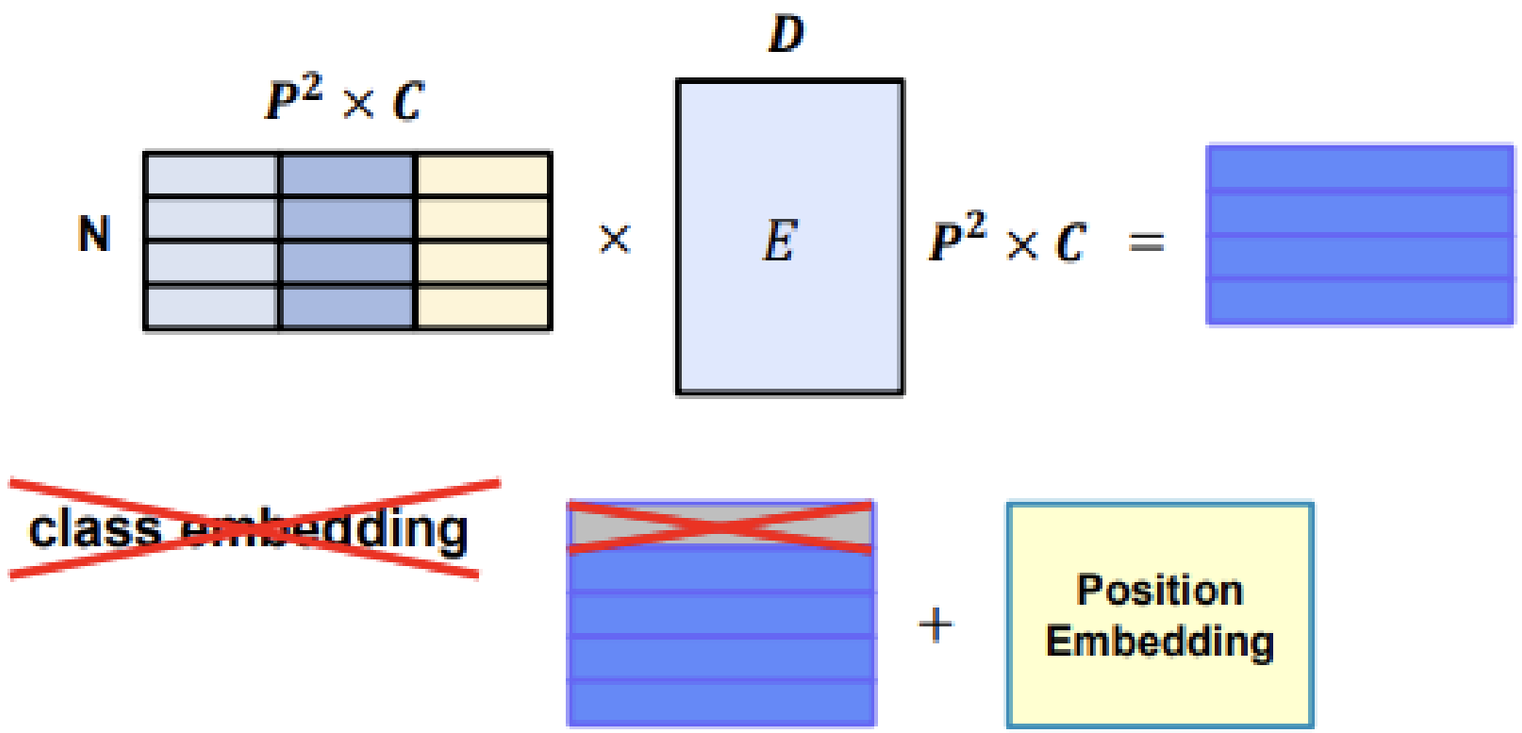
Vision Transformer (ViT)

- Swin Transformer
  - Patch Partitioning
  - Linear Embedding
  - Swin Transformer Block
  - Window Multi-head Attention
  - Patch Merging



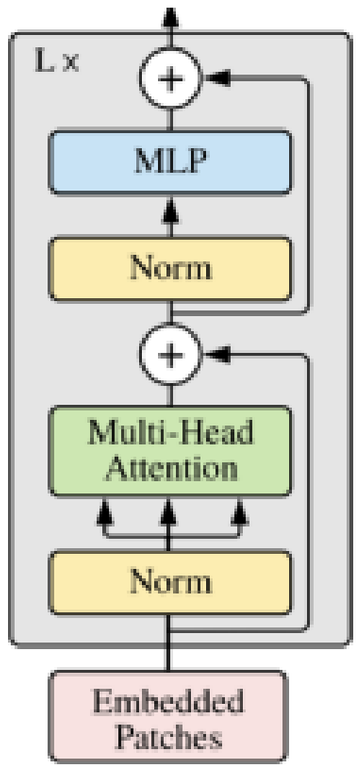
Vision Transformer (ViT)

- Linear Embedding
  - ViT와 embedding 방식은 동일, 그러나 class embedding 제거

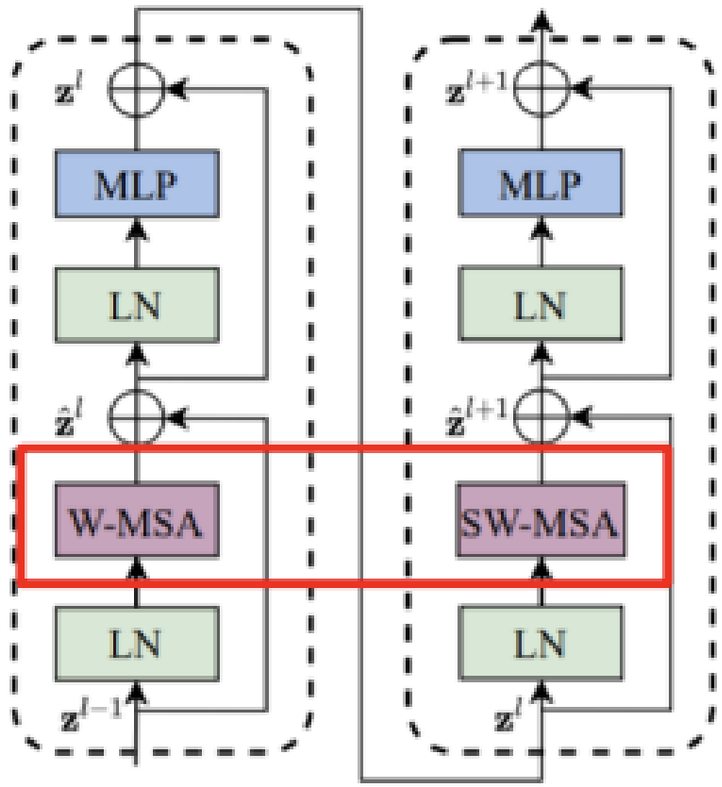


Vision Transformer (ViT)

- Swin Transformer Block



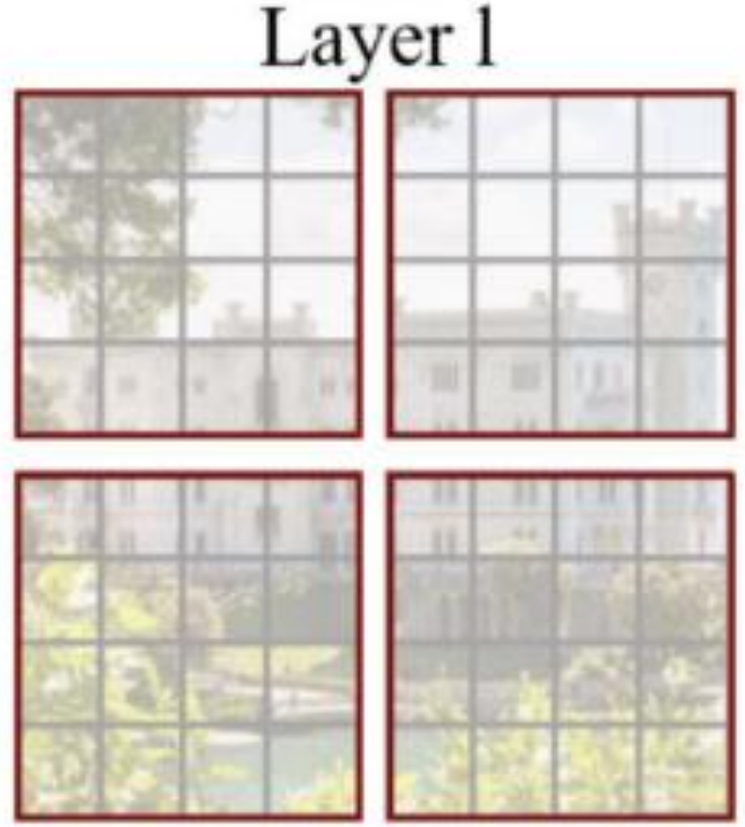
Vision Transformer



Swin Transformer

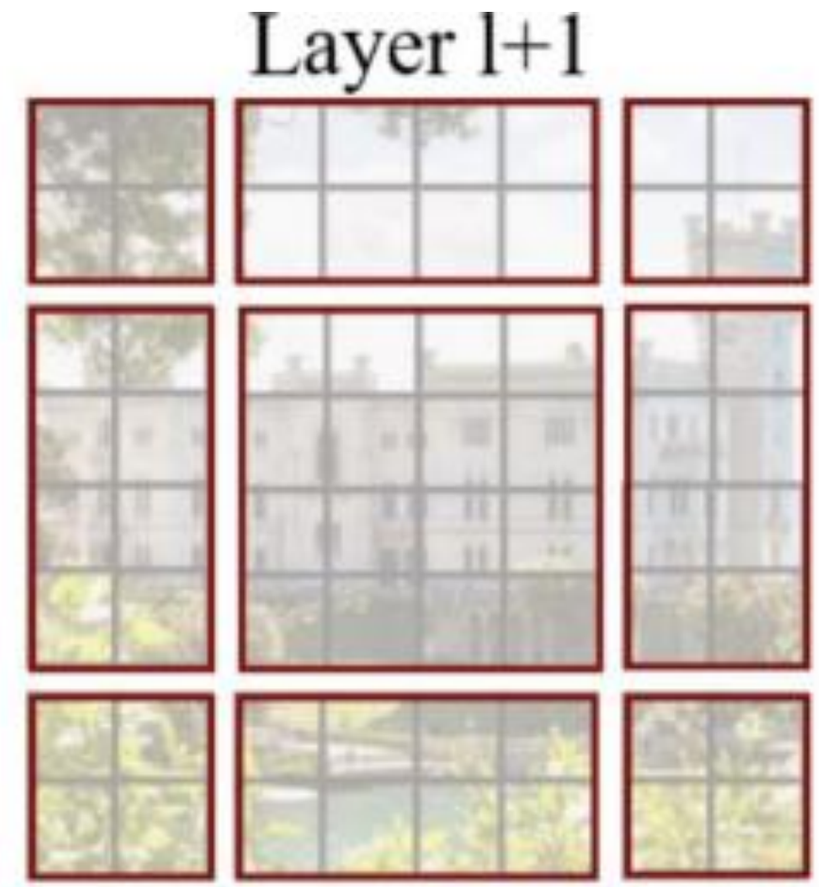
Vision Transformer (ViT)

- Swin Transformer Block
  - Window Multi-Head Attention
    - Window 단위로 embedding을 나눔.
    - 기존 ViT같은 경우 모든 embedding을 Transformer에 입력
    - Swin-Transformer는 Window 안에서만 Transformer 연산 수행
    - 따라서 이미지 크기에 따라 증가되던 computational cost가 Window 크기에 따라 조절 가능
    - Window 안에서만 수행하여 receptive field를 제한하는 단점 존재



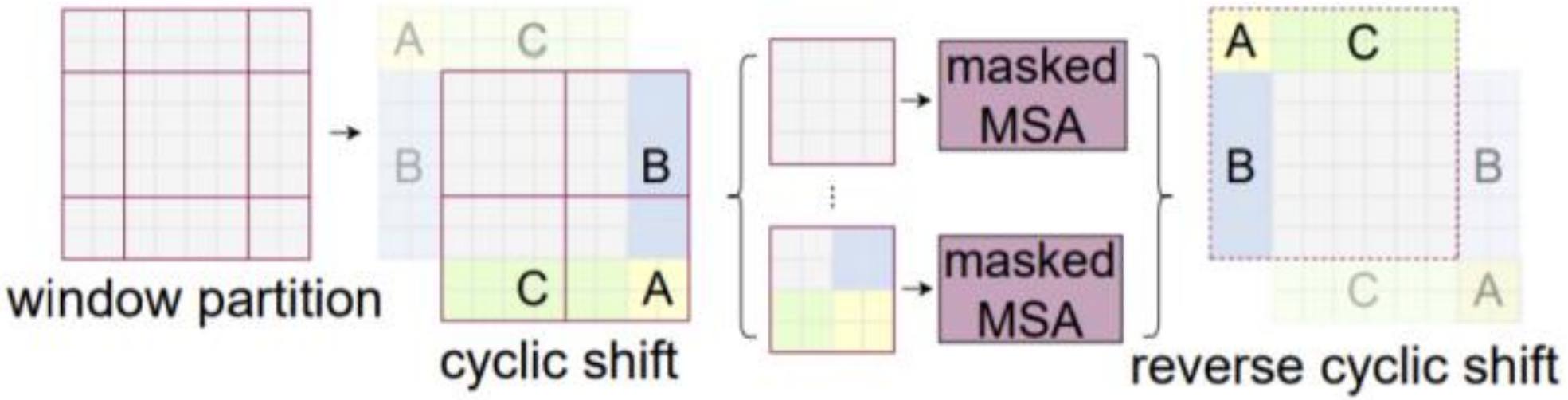
Vision Transformer (ViT)

- Swin Transformer Block
  - Shifted Window Multi-Head Attention
    - W-MSA의 경우 Window 안에서만 수행하여 receptive field를 제한하는 단점 존재
    - 이를 해결하기 위해 Shifted Window Multi-Head Attention을 Transformer Block 2번째 layer에서 수행
    - Window size와 다르게 나뉜 부분들 해결 필요



Vision Transformer (ViT)

- Swin Transformer Block
  - Shifted Window Multi-Head Attention
    - 남는 부분들 (A, B, C)를 그림과 같이 옮겨줌
    - 이 때 남는 부분들을 masking 처리하여 self-attention 연산이 되지 않도록 함



Vision Transformer (ViT)

- Experiment

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

(c) System-level Comparison

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DANet [22]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [10]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [23]	ResNet-101	45.9	38.5	-		
DNL [68]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [70]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [66]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [70]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [10]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [10]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [78]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Vision Transformer (ViT)

- Swin Transformer 정리
  - 적은 Data에도 학습이 잘 이루어짐
  - Window 단위를 이용하여 computation cost를 대폭 줄임
  - CNN과 비슷한 구조로 Object Detection, Segmentation 등의 backbone으로 general하게 활용

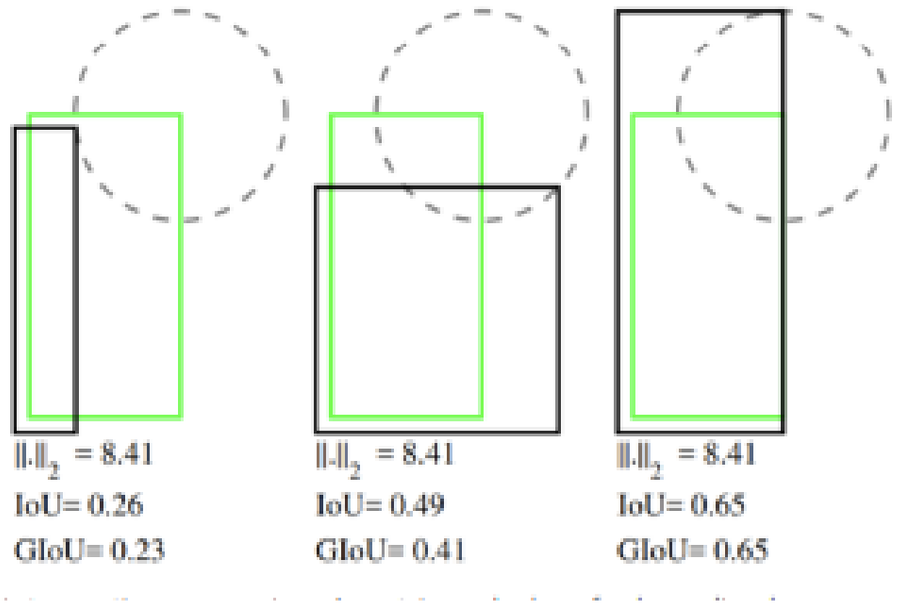
YOLO v4

- Background
  - Object Detection에서 사용하는 최신 방법들을 소개 및 실험
    - Object Detection model을 디자인하거나 향상 할 수 있는 아이디어 다양
  - 최신 Detection에서 정확도는 크게 향상시켰지만, 많은 양의 GPU를 필요
  - 실시간 요구하는 task에 부적합
    - Ex) 자율주행
  - 다른 detector들 보다 빠르면서 정확도가 높음

- Contribution
  - 하나의 GPU에서 훈련할 수 있는 빠르고 정확한 Object detector
  - BOF, BOS 방법들을 실험을 통해서 증명하고 조합을 찾음
    - BOF (Bag of Freebies) : inference 비용을 늘리지 않고 정확도 향상시키는 방법
    - BOS (Bag of Specials) : inference 비용을 조금 높이지만 정확도가 크게 향상하는 방법
  - GPU 학습에 더 효율적이고 적합하도록 방법들을 변형

Bag of Freebies

- Data Augmentation
  - 입력 이미지의 변화시켜 과적합(overfitting)을 막고, 다양한 환경에서도 강력해지는 방법 (ex. CutMix)
- Semantic Distribution Bias
  - 데이터셋에 특정 라벨(배경)이 많은 경우 불균형을 해결하기 위한 방법
  - Label smoothing
    - 라벨에 0 또는 1로 설정하는 것이 아니라 smooth하게 부여
      - Ex) 원래 0이었던 라벨을 0.1로 부여, 1이었던 라벨을 0.9로 부여
    - 모델의 overfitting 막아주고 regularization의 효과
- Bounding Box Regression
  - Bounding box 좌표값들을 예측하는 방법(MSE)은 거리가 일정하더라도 IoU가 다를 수 있음
  - IoU 기반 loss 제안 (IoU는 1에 가까울수록 잘 예측한 것이므로 loss처럼 사용 가능)
  - GIoU
    - IoU 기반 loss
    - IoU가 0인 경우에 대해서 차별화하여 loss 부여



Bag of Specials

- Enhance receptive field
  - Feature map의 receptive field를 키워서 검출 성능을 높이는 방법 (ex. SPP)
- Attention Module
  - SE,CBAM
- Feature Integration
  - Feature map 통합하기 위한 방법
- Activation Function
  - 좋은 activation 함수는 gradient가 더 효율적으로 전파
  - ReLU
    - Gradient vanishing 문제 해결하기 위한 활성 함수로 등장
    - 음수 값이 나오면 훈련이 되지 않는 현상 발생
  - Swish / Mish
    - 약간의 음수 허용하기 때문에 ReLU의 zero bound보다 gradient 흐름에 좋은 영향
    - 모든 구간에서 미분 가능Enhance receptive field
- Post-processing Method
  - 불필요한 Bbox 제거하는 방법

BoF and BoS for YOLOv4 backbone

- Activations : ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish
- Bounding box regression loss : MSE, IoU, GIoU, CIoU, DIoU
- Data augmentation : CutOut, MixUp, CutMix
- Regularization method : DropOut, DropPath, Spatial DropOut, DropBlock
- Normalization : Batch Normalization (BN), Cross-GPU Batch Normalization (CGBN or SyncBN), Filter Response Normalization (FRN), Cross-Iteration Batch Normalization (CBN)
- Skip-connections : Residual connections, Weighted residual connections, Multi-input weighted residual connections, Cross stage partial connections (CSP)
- Others : label smoothing

Cross Stage Partial Network (CSPNet)

- ❑ 정확도 유지하면서 경량화
- ❑ 메모리 cost 감소
- ❑ 다양한 backbone에서 사용가능
- ❑ 연산 bottleneck 제거

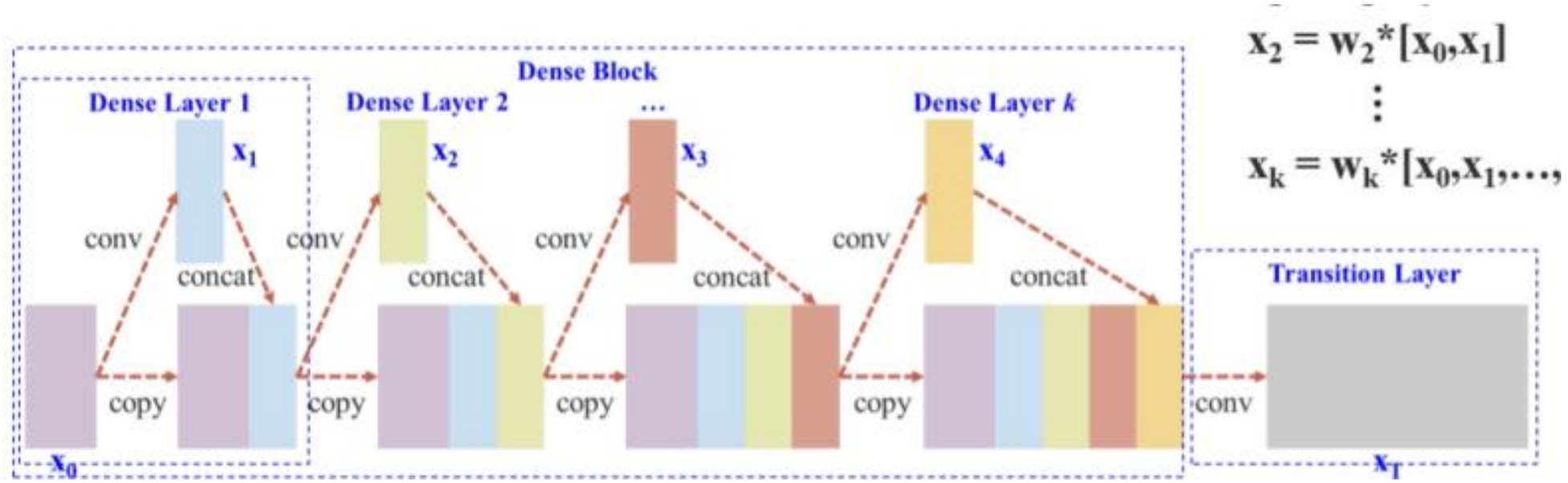
- ❑ 기존 DenseNet의 문제점
  - ■ 가중치 업데이트 할 때 gradient 정보가 재사용

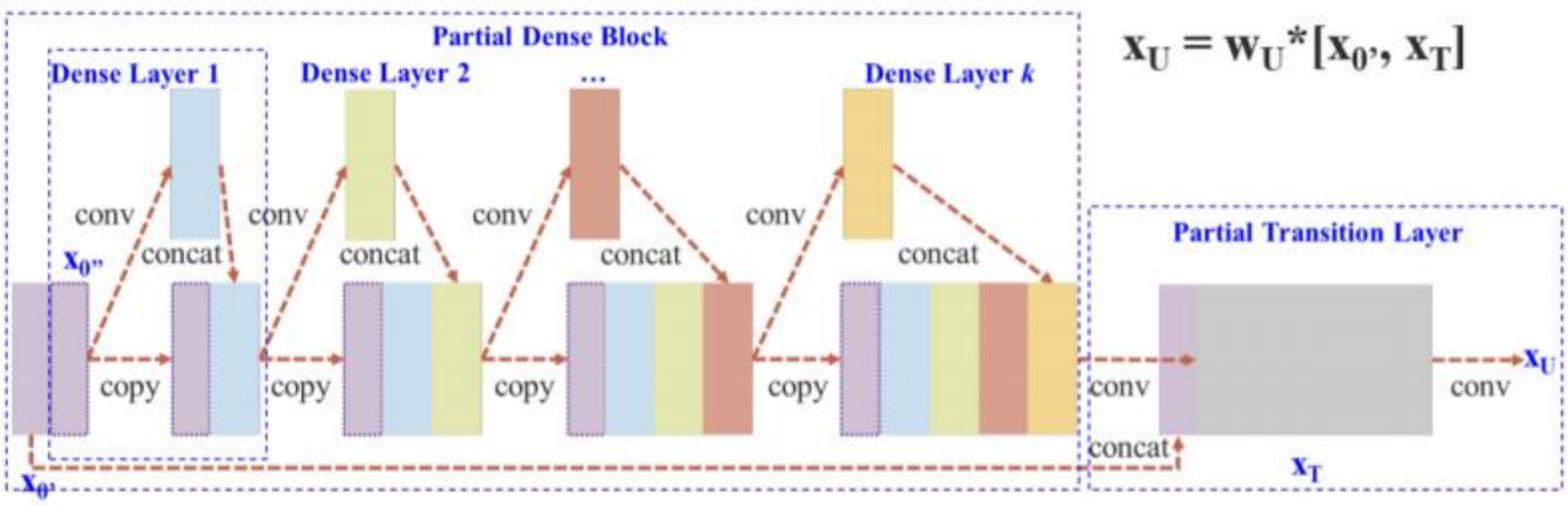
$$x_1 = w_1 * x_0$$
$$x_2 = w_2 * [x_0, x_1]$$
$$\vdots$$
$$x_k = w_k * [x_0, x_1, \ldots, x_{k-1}]$$

$$w_1' = f(w_1, g_0)$$
$$w_2' = f(w_2, g_0, g_1)$$
$$w_3' = f(w_3, g_0, g_1, g_2)$$
$$\vdots$$
$$w_k' = f(w_k, g_0, g_1, g_2, \ldots, g_{k-1})$$

CSPDenseNet

- CSPDenseNet
  - gradient information 많아지는 것 방지





BoF and BoS for Backbone Classification

Table 2: Influence of BoF and Mish on the CSPResNeXt-50 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	**94.0%**
	✓						**78.0%**	**94.3%**
		✓					**78.1%**	**94.5%**
			✓				77.5%	93.8%
				✓			**78.1%**	**94.4%**
					✓		64.5%	86.0%
						✓	**78.9%**	**94.5%**
	✓	✓		✓			**78.5%**	**94.8%**
	✓	✓		✓		✓	**79.8%**	**95.2%**

Table 3: Influence of BoF and Mish on the CSPDarknet-53 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.2%	93.6%
	✓	✓		✓			**77.8%**	**94.4%**
	✓	✓		✓		✓	**78.7%**	**94.8%**

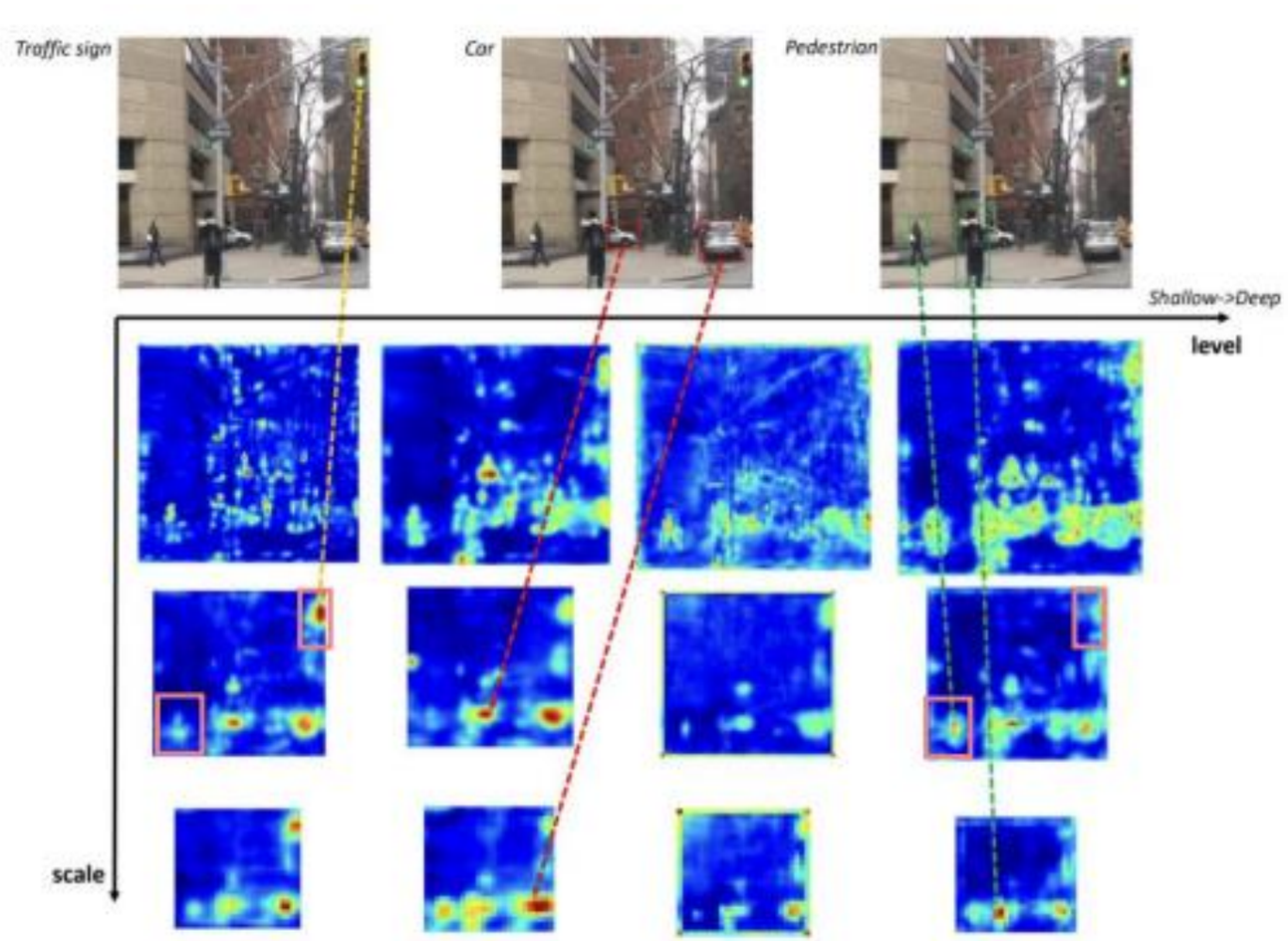
BoF and BoS for Object Detection

Table 4: Ablation Studies of Bag-of-Freebies. (CSPResNeXt50-PANet-SPP, 512x512).

S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP_{50}	AP_{75}
									MSE	38.0%	60.0%	40.8%
✓									MSE	37.7%	59.9%	40.5%
	✓								MSE	**39.1%**	**61.8%**	**42.0%**
		✓							MSE	36.9%	59.7%	39.4%
			✓						MSE	**38.9%**	**61.7%**	**41.9%**
				✓					MSE	33.0%	55.4%	35.4%
					✓				MSE	**38.4%**	**60.7%**	**41.3%**
						✓			MSE	**38.7%**	**60.7%**	**41.9%**
							✓		MSE	35.3%	57.2%	38.0%
✓									GIoU	**39.4%**	59.4%	**42.5%**
✓									DIoU	**39.1%**	58.8%	**42.1%**
✓									CIoU	**39.6%**	59.2%	**42.6%**
✓	✓	✓	✓						CIoU	**41.5%**	**64.0%**	**44.8%**
	✓		✓					✓	CIoU	36.1%	56.5%	38.4%
✓	✓	✓	✓					✓	MSE	**40.3%**	**64.0%**	**43.1%**
✓	✓	✓	✓					✓	GIoU	**42.4%**	**64.4%**	**45.9%**
✓	✓	✓	✓					✓	CIoU	**42.4%**	**64.4%**	**45.9%**

M2Det

- Feature pyramid 한계점
  - Backbone으로부터 feature pyramid 구성
    - Classification task를 위해 설계된 backbone은 object detection task를 수행하기에 충분하지 않음
    - Backbone network는 single-level layer로, single-level 정보만 나타냄
    - 일반적으로, low-level feature는 간단한 외형을, high-level feature는 복잡한 외형을 나타내는데 적합합니다

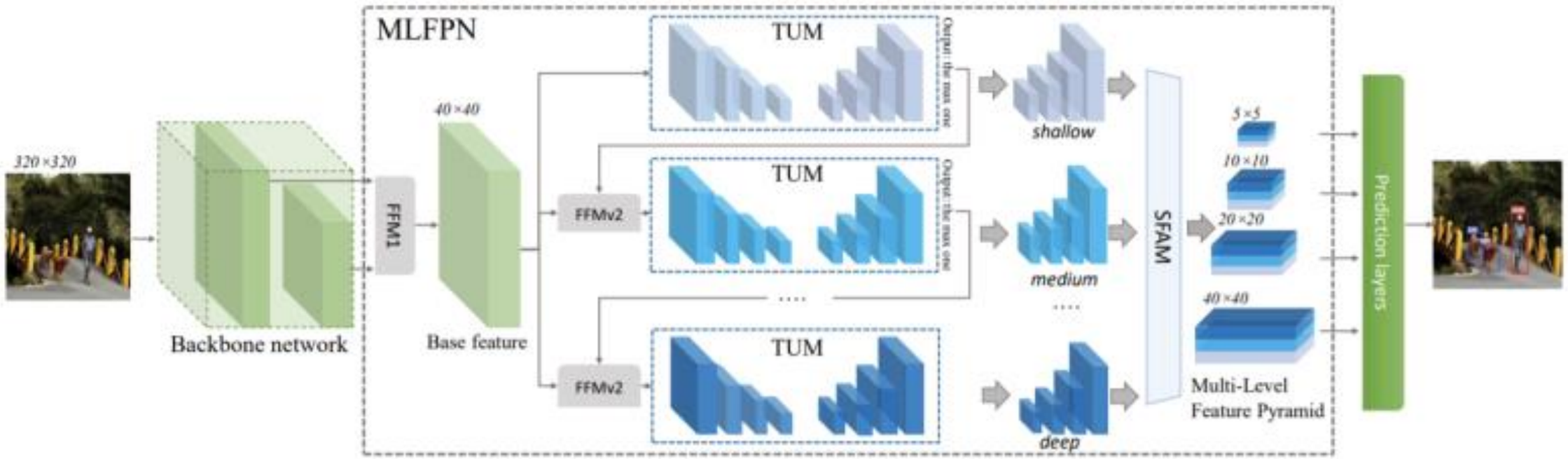


M2Det

- ❑ Multi-level, multi-scale feature pyramid 제안(MLFPN)
- ❑ • SSD에 합쳐서 M2Det라는 one stage detector 제안

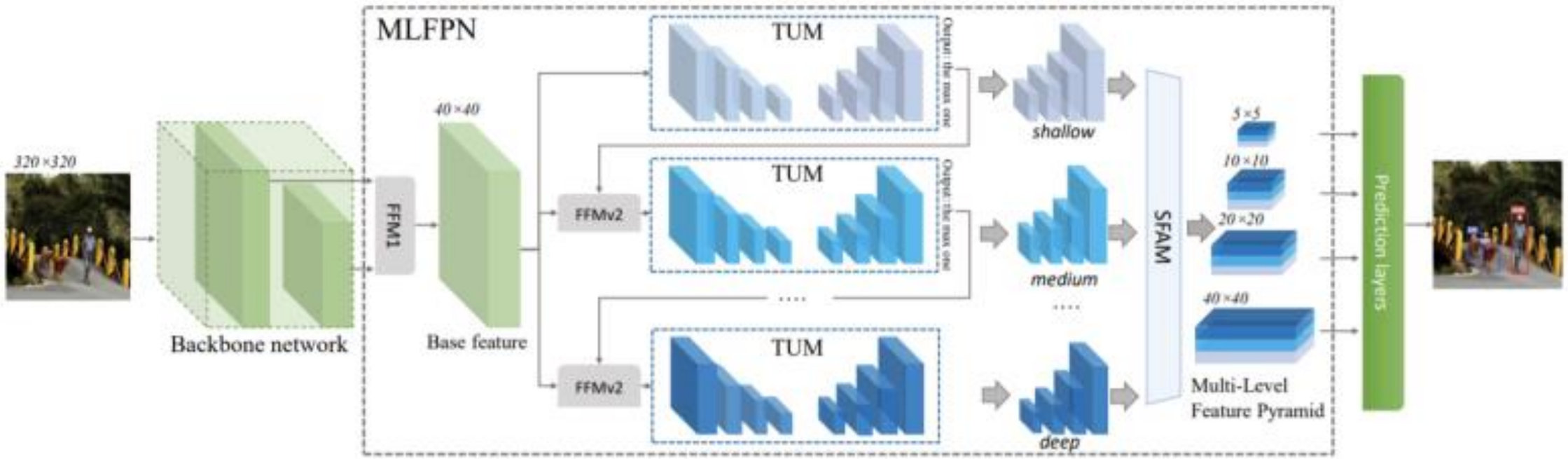
MLFPN

- FFM : Feature Fusion Module
  - FFMv1 : base feature 생성
  - Base feature : 서로 다른 scale의 2 feature map을 합쳐 semantic 정보가 풍부함



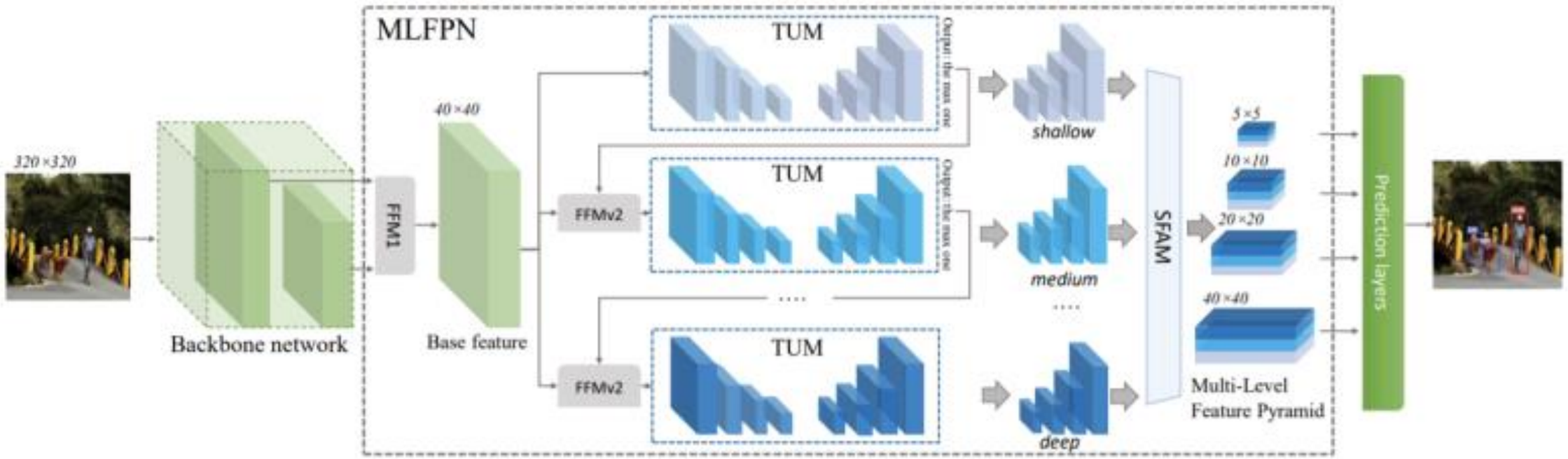
MLFPN

- TUM : Thinned U-shape Module
  - Encode-decoder 구조
  - Decoder의 출력 : 현재 level에서의 multi-scale features



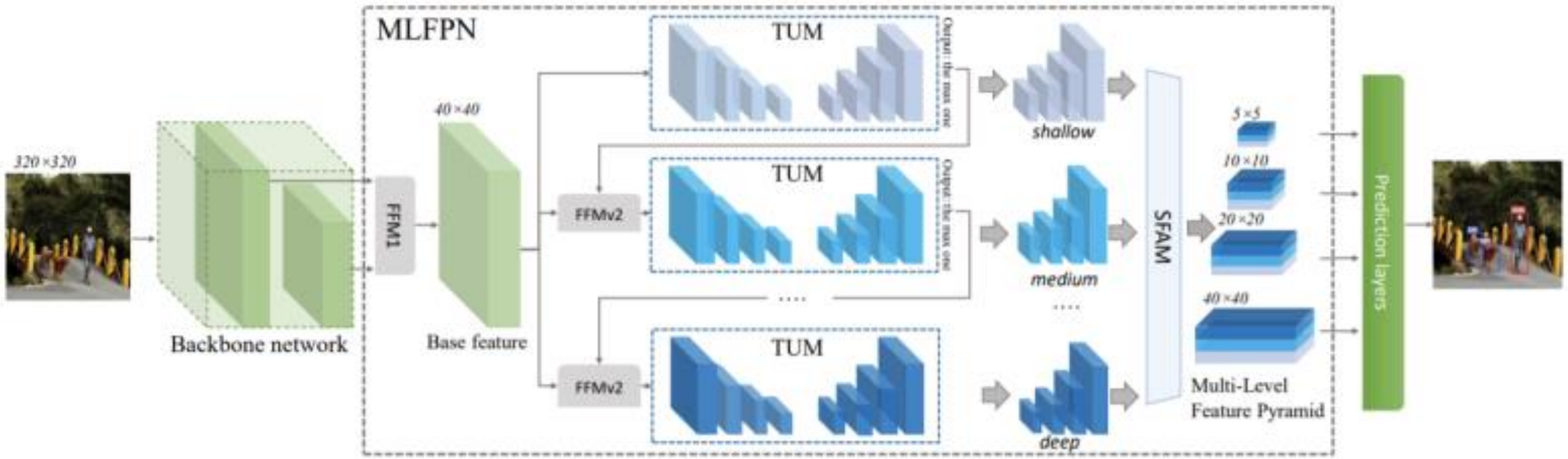
MLFPN

- FFM : Feature Fusion Module
  - FFMv2 : base feature와 이전 TUM 출력 중에서 가장 큰 feature concat
  - 다음 TUM의 입력으로 들어감



MLFPN

- SFAM : Scale-wise Feature Aggregation Module
  - TUMs에서 생성된 multi-level multi-scale을 합치는 과정
  - 동일한 크기를 가진 feature들끼리 연결 (scale-wise concatenation)
  - 각각의 scale의 feature들은 multi-level 정보를 포함
  - Channel-wise attention 도입 (SE block)
    - 채널별 가중치를 계산하여 각각의 feature를 강화시키거나 약화시킴



M2Det

- Results

Method	Backbone	Input size	MultiScale	FPS	Avg. Precision, IoU:			Avg. Precision, Area:		
					0.5:0.95	0.5	0.75	S	M	L
two-stage:										
Faster R-CNN (Ren et al. 2015)	VGG-16	~1000×600	False	7.0	21.9	42.7	-	-	-	-
OHEM++ (Shrivastava et al. 2016)	VGG-16	~1000×600	False	7.0	25.5	45.9	26.1	7.4	27.7	40.3
R-FCN (Dai et al. 2016)	ResNet-101	~1000×600	False	9	29.9	51.9	-	10.8	32.8	45.0
CoupleNet (Zhu et al. 2017)	ResNet-101	~1000×600	False	8.2	34.4	54.8	37.2	13.4	38.1	50.8
Faster R-CNN w FPN (Lin et al. 2017a)	Res101-FPN	~1000×600	False	6	36.2	59.1	39.0	18.2	39.0	48.2
Deformable R-FCN (Dai et al. 2017)	Inc-Res-v2	~1000×600	False	-	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN (He et al. 2017)	ResNeXt-101	~1280×800	False	3.3	39.8	62.3	43.4	22.1	43.2	51.2
Fitness-NMS (Tychsen-Smith and Petersson 2018)	ResNet-101	~1024×1024	True	5.0	41.8	60.9	44.9	21.5	45.0	57.5
Cascade R-CNN (Cai and Vasconcelos 2018)	Res101-FPN	~1280×800	False	7.1	42.8	62.1	46.3	23.7	45.5	55.2
SNIP (Singh and Davis 2018)	DPN-98	-	True	-	45.7	67.3	51.1	29.3	48.8	57.1
one-stage:										
SSD300* (Liu et al. 2016)	VGG-16	300×300	False	43	25.1	43.1	25.8	6.6	25.9	41.4
RON384++ (Kong et al. 2017)	VGG-16	384×384	False	15	27.4	49.5	27.1	-	-	-
DSSD321 (Fu et al. 2017)	ResNet-101	321×321	False	9.5	28.0	46.1	29.2	7.4	28.1	47.6
RetinaNet400 (Lin et al. 2017b)	ResNet-101	~640×400	False	12.3	31.9	49.5	34.1	11.6	35.8	48.5
RefineDet320 (Zhang et al. 2018)	VGG-16	320×320	False	38.7	29.4	49.2	31.3	10.0	32.0	44.4
RefineDet320 (Zhang et al. 2018)	ResNet-101	320×320	True	-	38.6	59.9	41.7	21.1	41.7	52.3
M2Det (Ours)	VGG-16	320×320	False	33.4	33.5	52.4	35.6	14.4	37.6	47.6
M2Det (Ours)	VGG-16	320×320	True	-	38.9	59.1	42.4	24.4	41.5	47.6
M2Det (Ours)	ResNet-101	320×320	False	21.7	34.3	53.5	36.5	14.8	38.8	47.9
M2Det (Ours)	ResNet-101	320×320	True	-	**39.7**	**60.0**	**43.3**	**25.3**	**42.5**	**48.3**
YOLOv3 (Redmon and Farhadi 2018)	DarkNet-53	608×608	False	19.8	33.0	57.9	34.4	18.3	35.4	41.9
SSD512* (Liu et al. 2016)	VGG-16	512×512	False	22	28.8	48.5	30.3	10.9	31.8	43.5
DSSD513 (Fu et al. 2017)	ResNet-101	513×513	False	5.5	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet500 (Lin et al. 2017b)	ResNet-101	~832×500	False	11.1	34.4	53.1	36.8	14.7	38.5	49.1
RefineDet512 (Zhang et al. 2018)	VGG-16	512×512	False	22.3	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet512 (Zhang et al. 2018)	ResNet-101	512×512	True	-	41.8	62.9	45.7	25.6	45.1	54.1
CornerNet (Law and Deng 2018)	Hourglass	512×512	False	4.4	40.5	57.8	45.3	20.8	44.8	56.7
CornerNet (Law and Deng 2018)	Hourglass	512×512	True	-	42.1	57.8	45.3	20.8	44.8	56.7
M2Det (Ours)	VGG-16	512×512	False	18.0	37.6	56.6	40.5	18.4	43.4	51.2
M2Det (Ours)	VGG-16	512×512	True	-	42.9	62.5	47.2	28.0	47.4	52.8
M2Det (Ours)	ResNet-101	512×512	False	15.8	38.8	59.4	41.7	20.5	43.9	53.4
M2Det (Ours)	ResNet-101	512×512	True	-	**43.9**	**64.4**	**48.0**	**29.6**	**49.6**	**54.3**
RetinaNet800 (Lin et al. 2017b)	Res101-FPN	~1280×800	False	5.0	39.1	59.1	42.3	21.8	42.7	50.2
M2Det (Ours)	VGG-16	800×800	False	11.8	41.0	59.7	45.0	22.1	46.5	53.8
M2Det (Ours)	VGG-16	800×800	True	-	**44.2**	**64.6**	**49.3**	**29.2**	**47.9**	**55.1**

Thank you