

You Only Look One-Level Feature

Qiang Chen et al. (CVPR 2021 accepted)

Presenter: Dongmin Choi

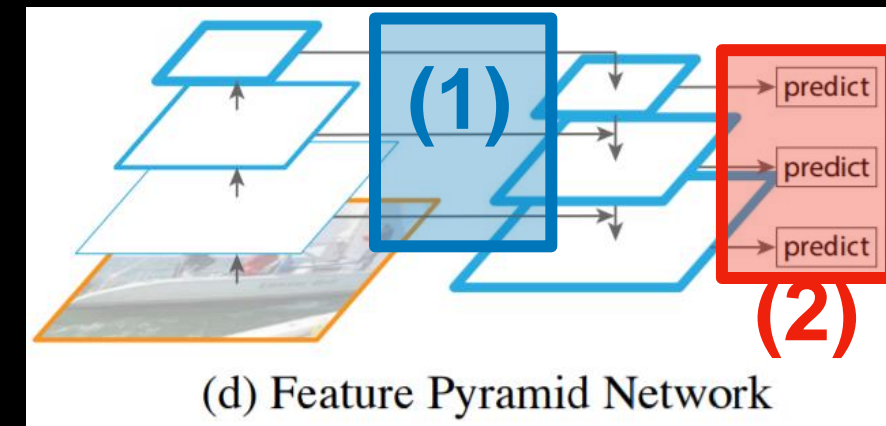
Abstract

Revisiting Feature Pyramid Network (FPN)

- Various FPN-based One-stage Detectors
- What is the main component for these networks to achieve high performance?
- Not multi-scale feature fusion but divide-and-conquer
- Instead of adopting the complex feature pyramid, let's utilize only one-level feature
- You Only Look One-level Feature (YOLOF)
 - Dilated Encoder
 - Uniform Matching
 - 2.5x faster than RetinaNet and 7x less training than DETR

Introduction

Feature Pyramid Network (FPN)

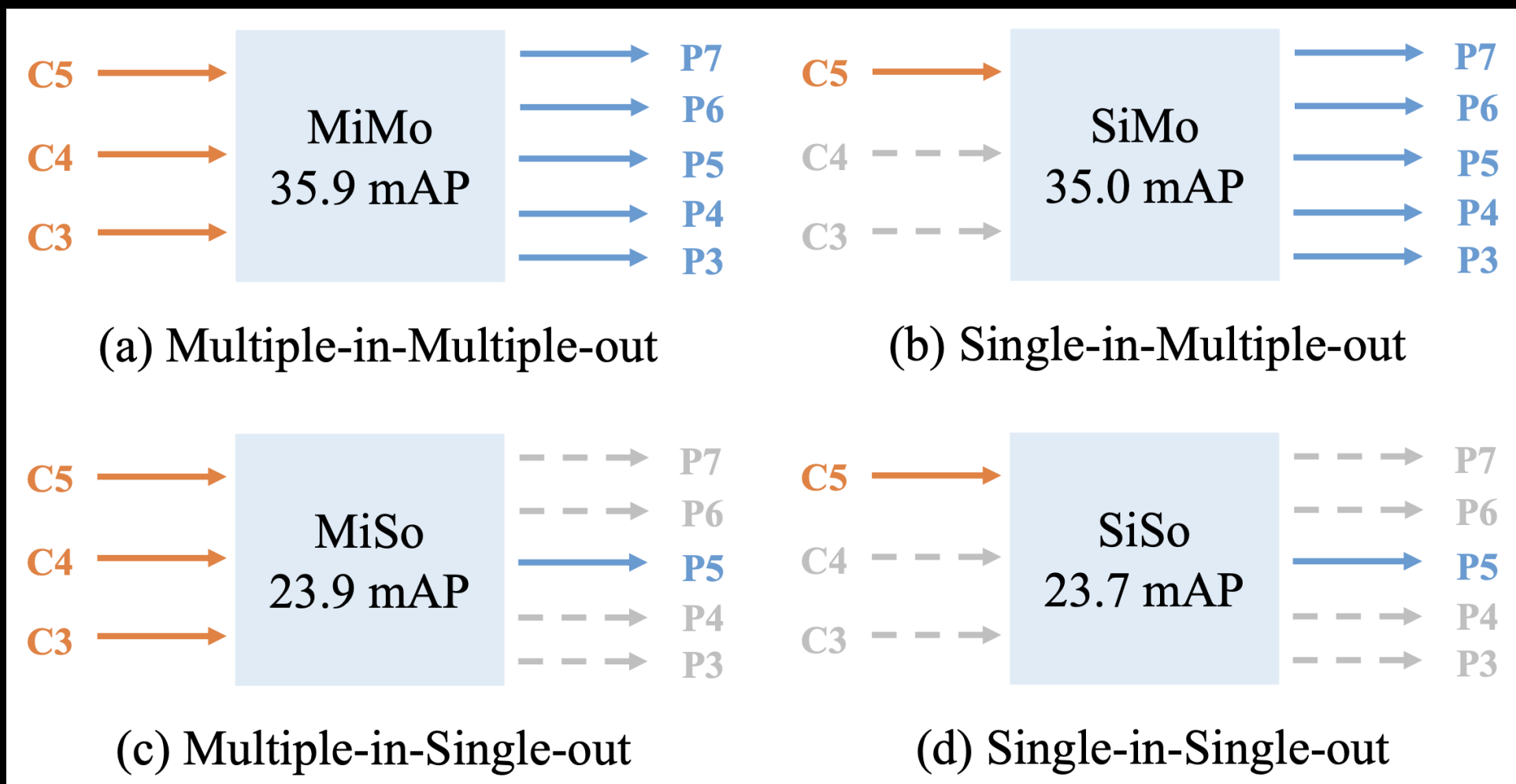


- Essential component for both two-stage and one-stage detectors
- Two main benefits:
 - (1) multi-scale feature fusion: fusing multiple low- and high-res feature inputs
 - (2) divide-and-conquer: detecting objects on different levels
- Common belief: multi-scale feature fusion is important
- Designing complex fusion methods manually or via NAS have been proposed actively
- However, few studies on the function of divide-and-conquer

Introduction

Multi-scale Feature Fusion vs. Divide-and-Conquer

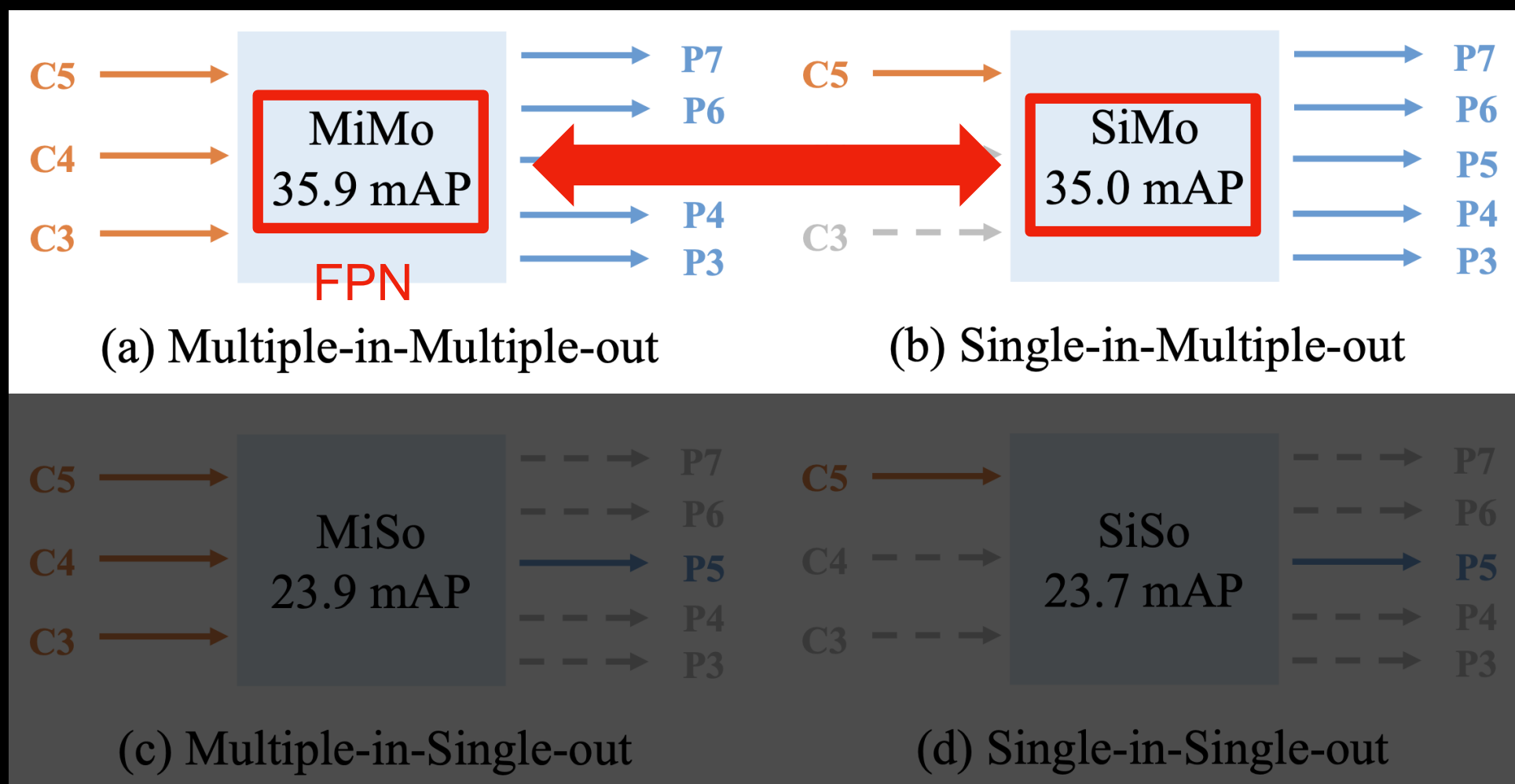
- Let's decouple the multi-scale feature fusion and divide-and-conquer functionalities with RetinaNet



Introduction

Multi-scale Feature Fusion vs. Divide-and-Conquer

- Let's decouple the multi-scale feature fusion and divide-and-conquer functionalities with RetinaNet

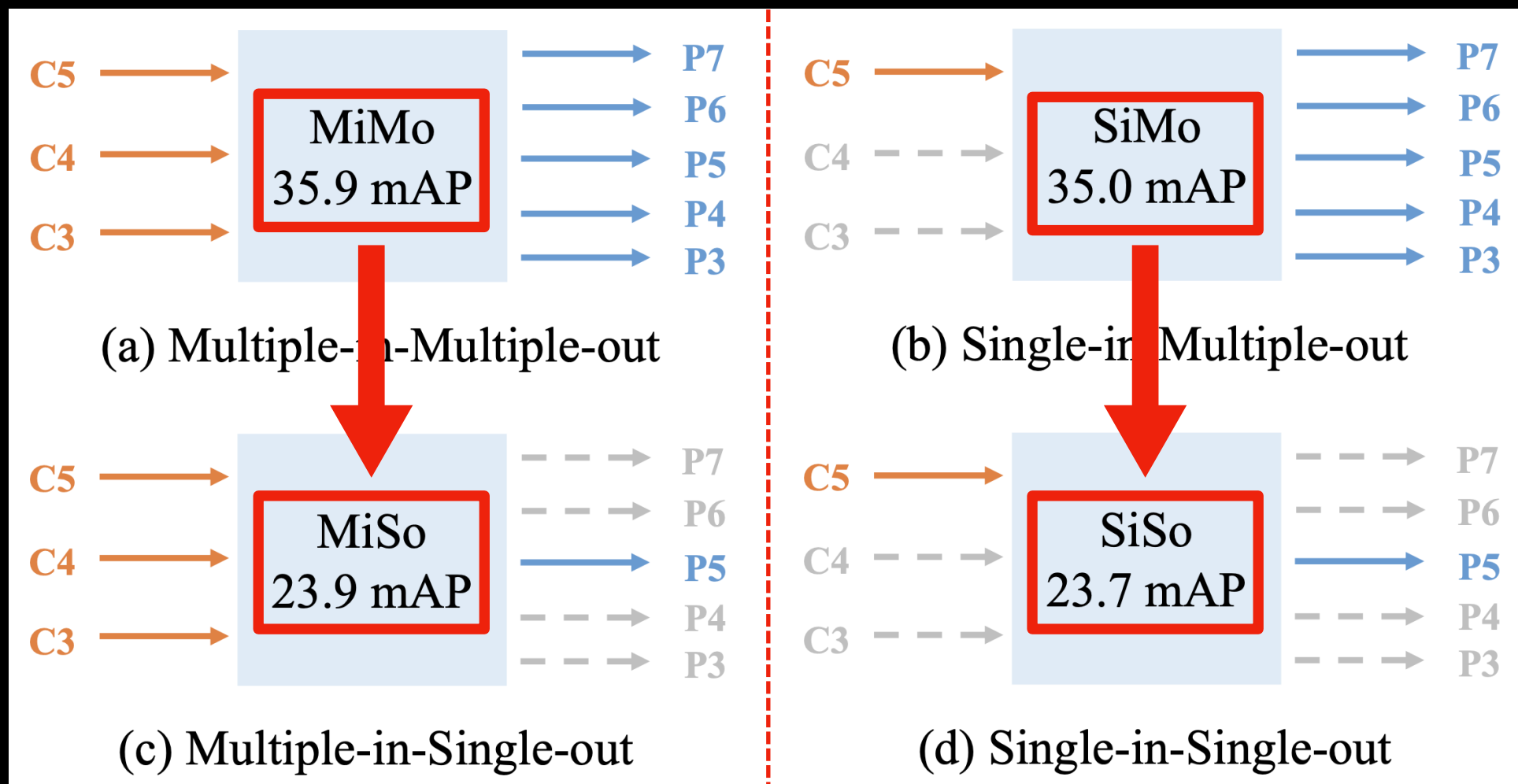


SiMo can achieve comparable performance with MiMo

Introduction

Multi-scale Feature Fusion vs. Divide-and-Conquer

- Let's decouple the multi-scale feature fusion and divide-and-conquer functionalities with RetinaNet



Performance drops dramatically in MiSo and SiSo

Introduction

Multi-scale Feature Fusion vs. Divide-and-Conquer

- Let's decouple the multi-scale feature fusion and divide-and-conquer functionalities with RetinaNet
- SiMO can achieve comparable performance with MiMo
- Performance drops dramatically in MiSo and SiSo
- Two facts:
 - (1) C5 feature carries sufficient context for detecting objects on various scales
 - (2) divide-and-conquer >> multi-scale feature fusion

Introduction

Divide-and-Conquer

- divides the complex detection problem into several sub-problems by object scales; optimization problem
- But memory burden, slow and complex
- Can we solve this problem with SiSo encoder?
 - (1) extract the multi-scale contexts for objects on various scales only with a single-level feature
 - (2) solve the imbalance problem of positive anchors raised by the sparse anchors in the single feature

Introduction

Divide-and-Conquer

- divides the complex detection problem into several sub-problems by object scales; optimization problem
- But memory burden, slow and complex
- Can we solve this problem with SiSo encoder?
 - (1) extract the multi-scale contexts for objects on various scales only with a single-level feature → Dilated Encoder
 - (2) solve the imbalance problem of positive anchors raised by the sparse anchors in the single feature → Uniform Matching

YOLOF

Introduction

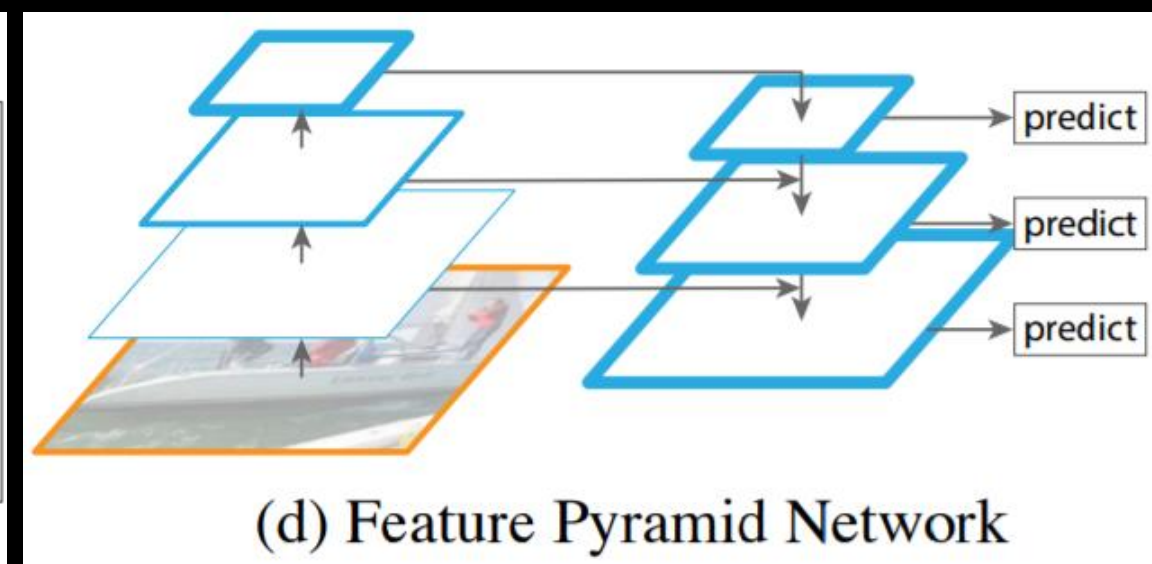
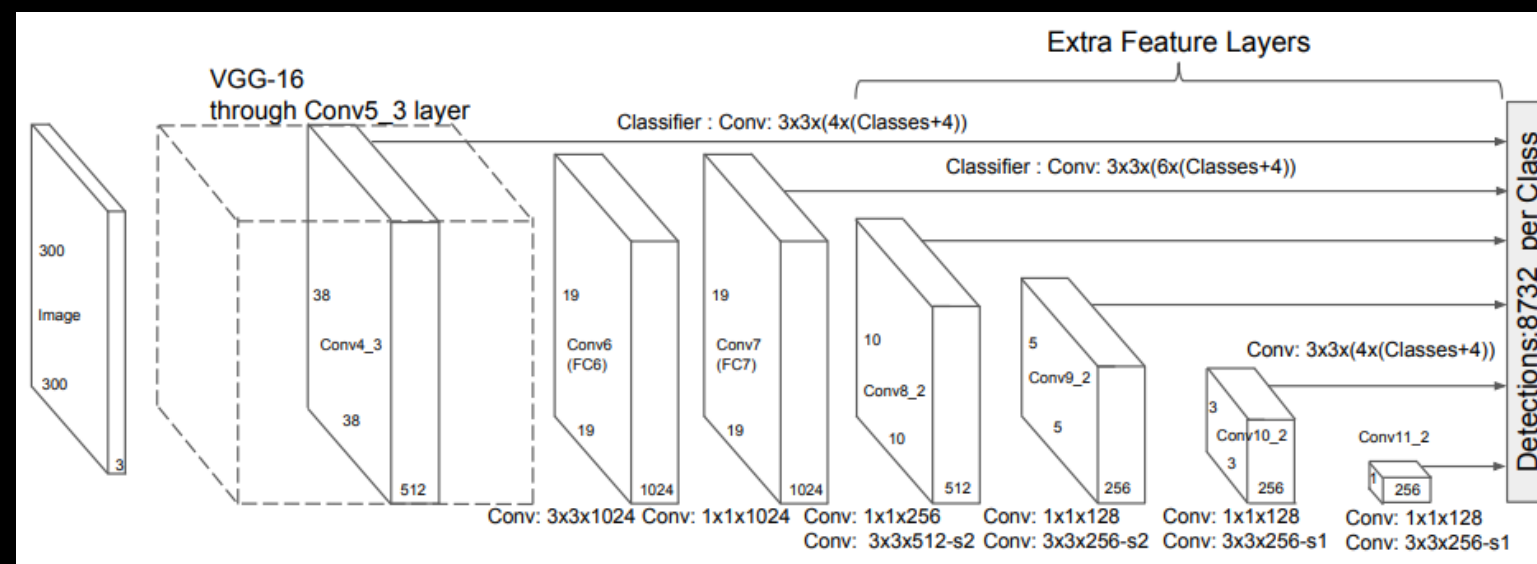
Contributions

- show that the most significant benefits of FPS is its **divide-and-conquer** solution rather than **multi-scale** feature fusion
- YOLOF, a simple and efficient baseline without using FPN
- On COCO benchmark, YOLOF can achieve comparable results with a faster speed on GPUs with RetinaNet, DETR and YOLOv4

Related Works

Multi-level Feature Detectors

- A conventional technique to employ multiple features
- SSD¹: first utilizes multi-scale features and performs detection on each scale for different scales objects
- FPN²: semantic-riched feature pyramids by combining shallow features and deep features (dominant)



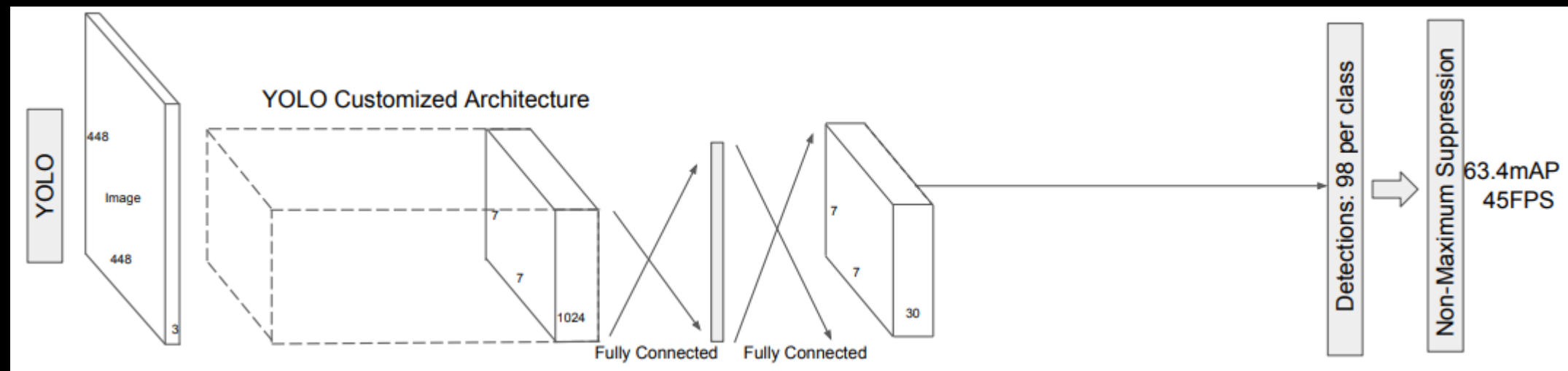
¹W Liu et al., SSD: Single Shot MultiBox Detector, ECCV 2016

²T.Y Lin et al., Feature Pyramid Networks for Object Detection, CVPR 2017

Related Works

Single-level Feature Detectors

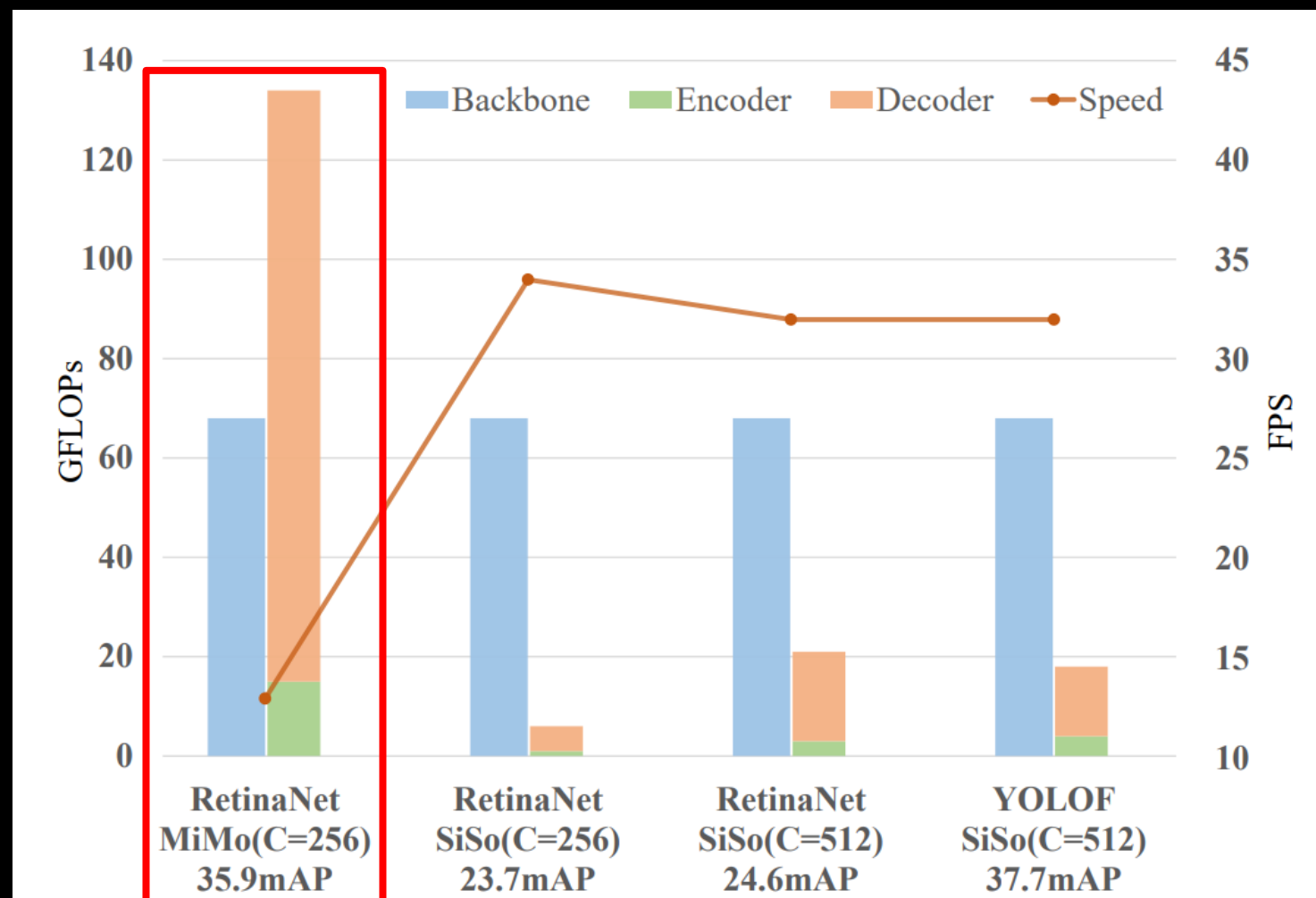
- R-CNN series
 - YOLO and YOLOv2
 - CornerNet and CenterNet: high-resolution feature map brings enormous memory cost
 - DETR: needs a long training schedule due to the totally anchor-free mechanism and transformer learning phase
- Super fast but low performance**



Cost Analysis of MiMo Encoders

Quantitative Study on the Cost of MiMo Encoders

- Based on RetinaNet w/ ResNet-50
- Pipeline: Backbone + Encoder + Decoder
- The MiMo encoder and decoder bring enormous memory burdens



Method

Replacing MiMo encoder with SiSo encoder

- Two problems of SiSo encoders which cause performance drop:

(1) Limited Scale Range

- the range of scales matching to the C5 feature's receptive field is limited
- forbid the detection performance for objects across various scales

(2) Imbalance Problem on Positive Anchors

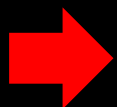
- the imbalance problem on positive anchors
- raised by sparse anchors in the single-level feature

Method

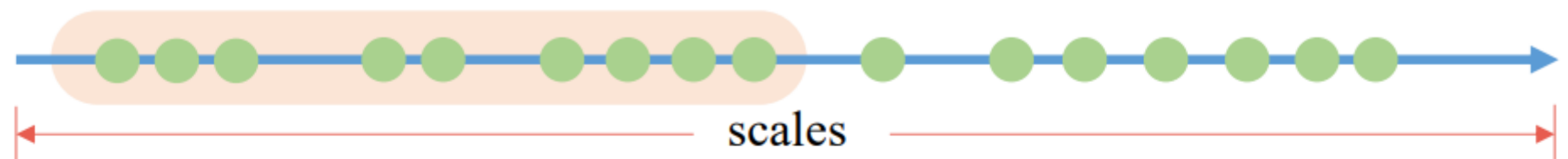
1. Limited Scale Range

- In SiSo encoders, the single-level feature setting has a receptive field which is a constant.

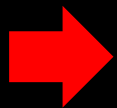
C5



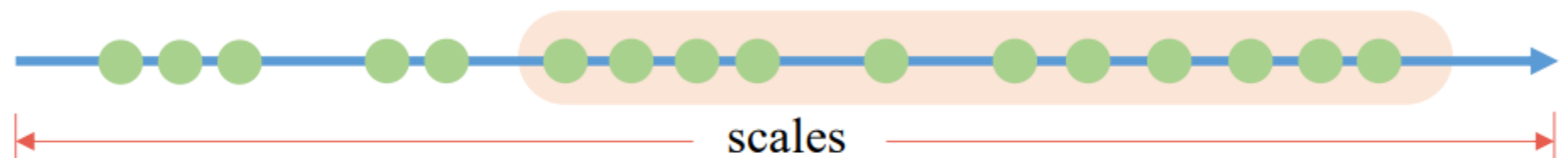
(a)



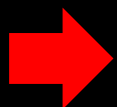
A



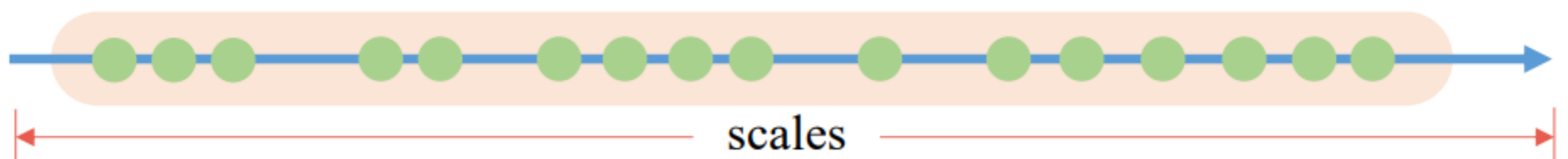
(b)



B



(c)



● scales of objects

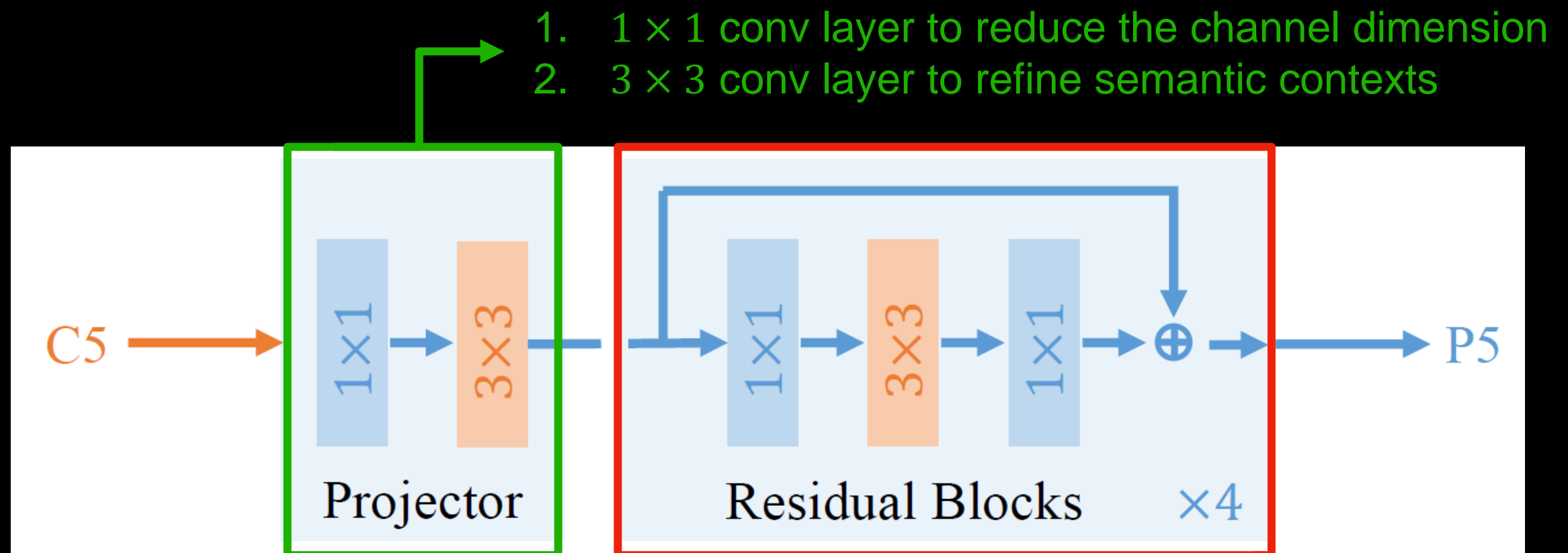
○ scale range covered by feature

- **A**: enlarging the receptive field by stacking standard and dilated conv.
- **B**: combine the original scale range and **A**

Method

1. Limited Scale Range

- Dilated Encoder (SiSo)
: the **Projector** and the **Residual Blocks**

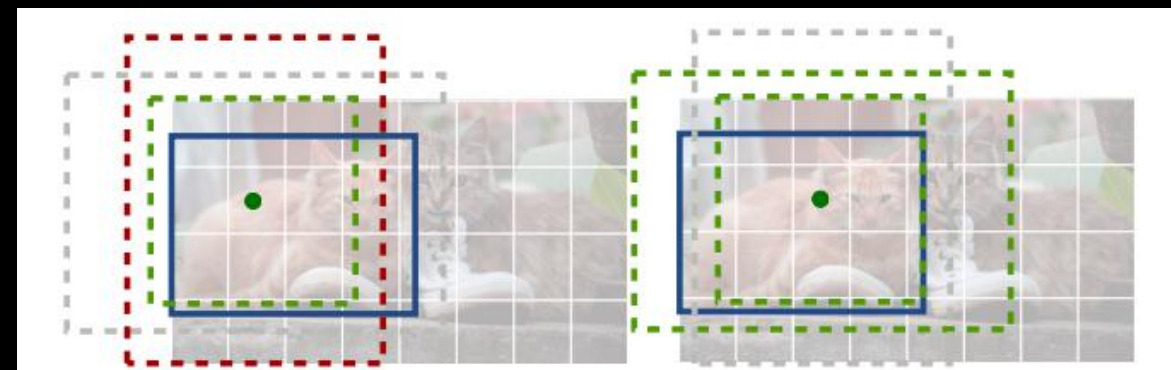


Stacking four dilated residual blocks with different dilations rates (2, 4, 6, 8) to generate output features with multiple receptive fields

Method

2. Imbalance Problem on Positive Anchors

- **Max-IoU matching**
: strategy to define **positive anchors** by measuring the IoUs between anchors and ground-truth boxes
- In **MiMo** encoders, the anchors are pre-defined on multiple levels. So, **Max-IoU matching** generates a sufficient number of positive anchors
- However, in **SiSo** encoder, the number of anchors diminish extensively from about $100k$ to $5k$. **Sparse anchors** raise a matching problem.



(a) Standard anchor based detection. Anchors count as **positive** with an overlap $IoU > 0.7$ to any **object**, **negative** with an overlap $IoU < 0.3$, or are **ignored** otherwise.

Method

2. Imbalance Problem on Positive Anchors

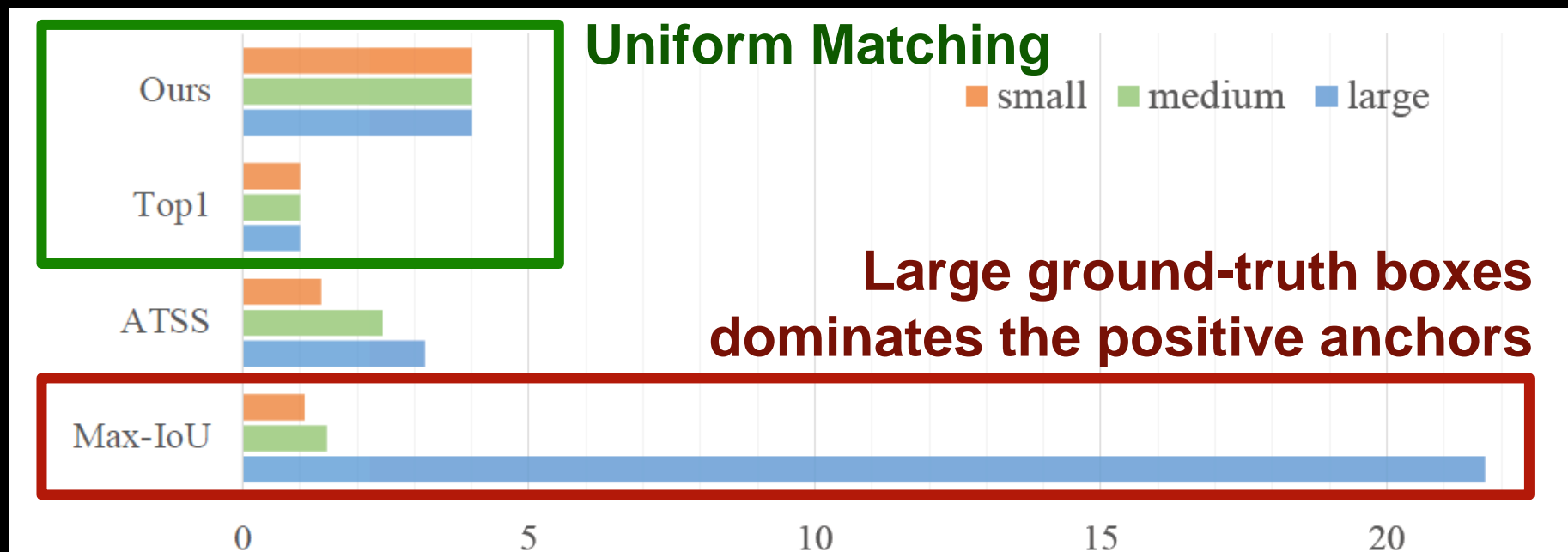


Figure 6. Distribution of the generated positive anchors in various matching methods with single feature. This figure aims to show the balancedness of the generated positive anchors. The positive anchors in the Max-IoU are dominated by large ground-truth boxes, causing huge imbalance across object scales. ATSS alleviates the imbalance problem by adaptively sampling positive anchors when training. The Top1 and Ours adopt a uniform matching, generating positive anchors in a balanced manner regardless of small, medium, and large objects.

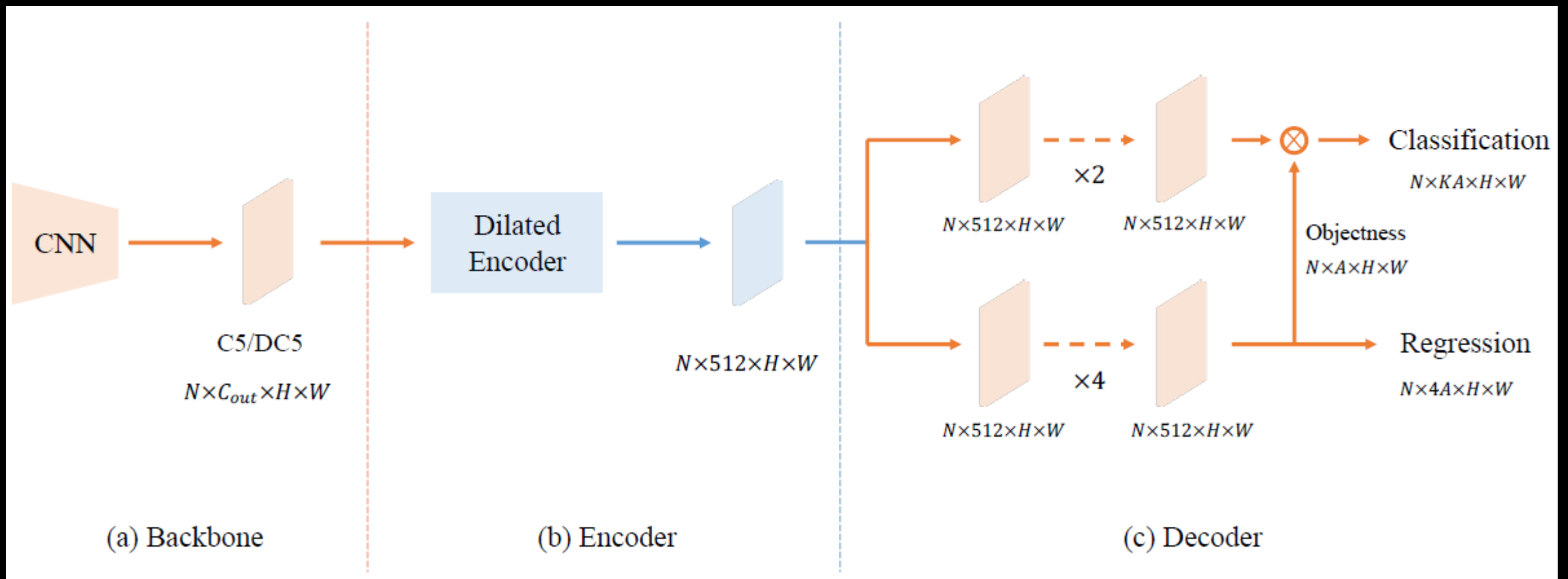
Method

2. Imbalance Problem on Positive Anchors

- Uniform Matching
: adopting the k nearest anchor as positive anchors for each ground-truth box (top- k matching)
- IoU thresholds
: ignore large IoU (>0.7) negative anchors and small IoU (<0.15) positive anchors

Method

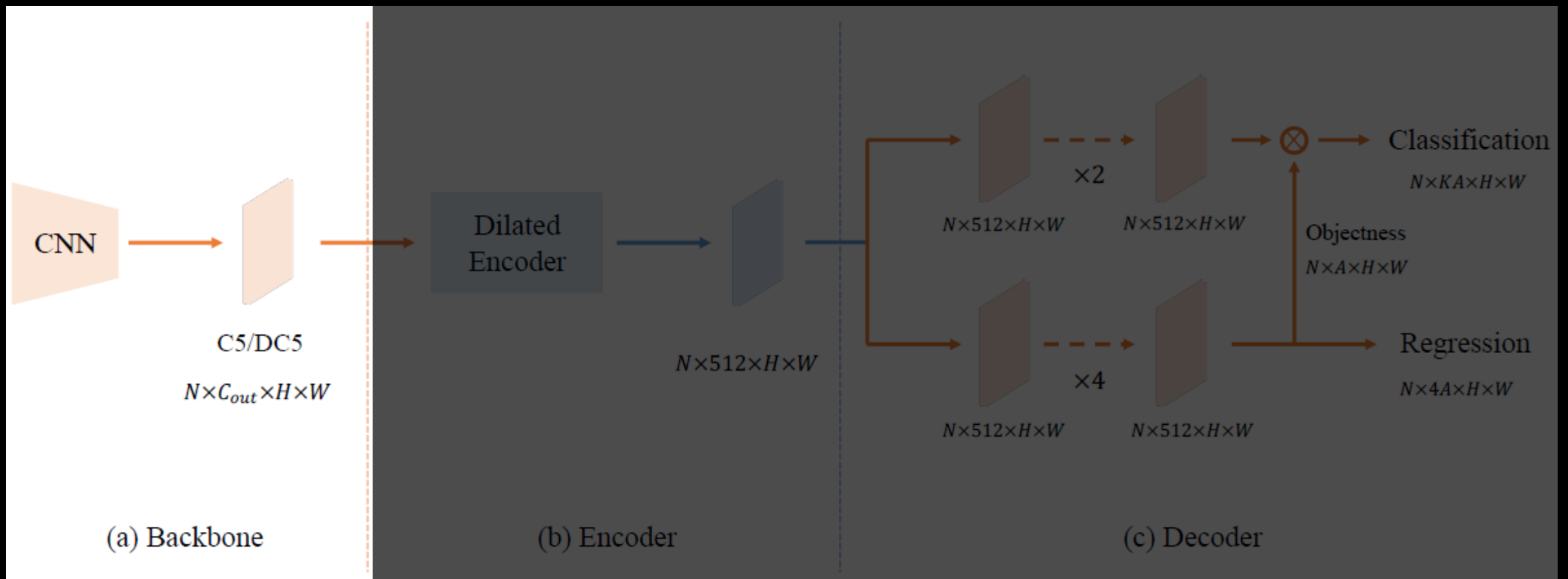
3. YOLOF



- A fast and straightforward framework with single-level feature
- Three parts: the backbone, the encoder, and the decoder

Method

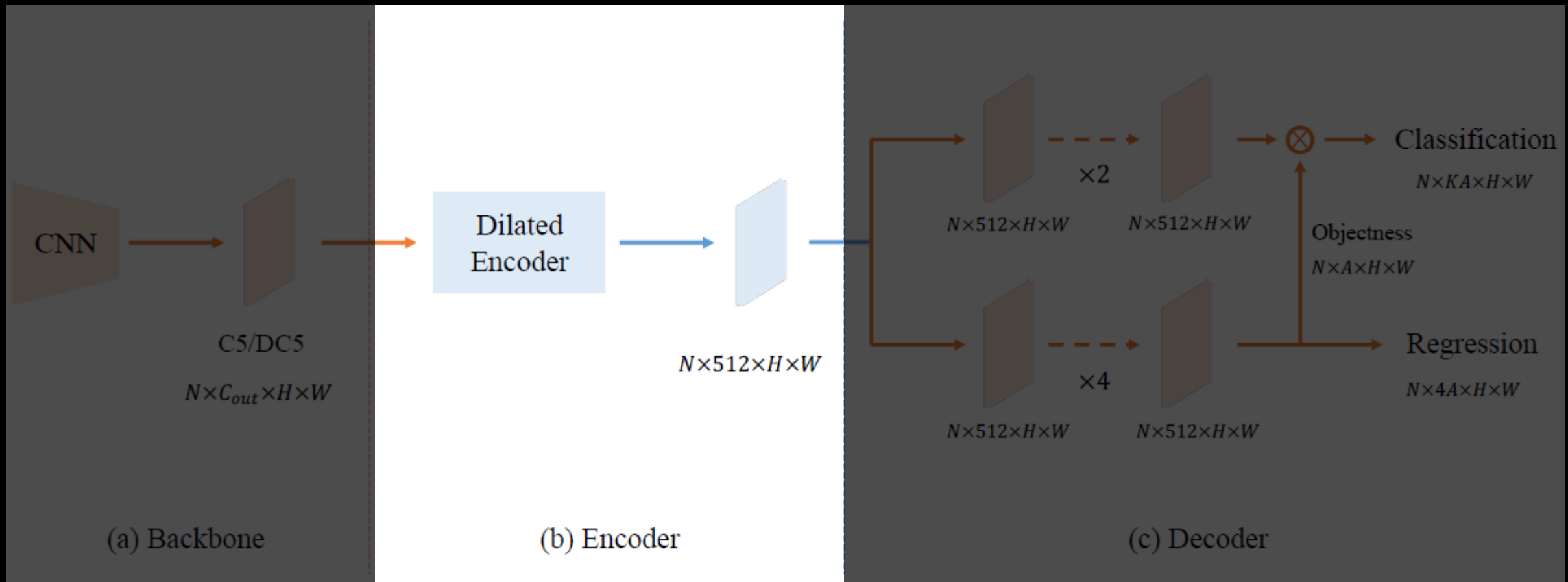
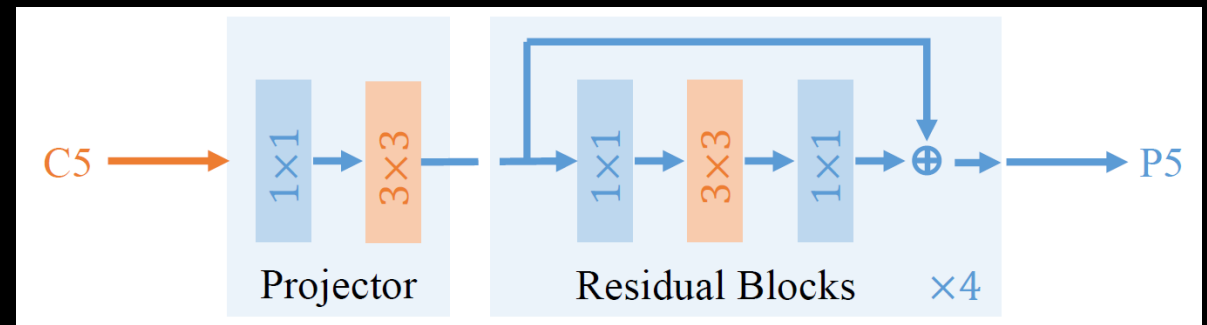
3. YOLOF: Backbone



- ResNet and ResNeXt series as the backbone
- Pre-trained on ImageNet
- Output: C5 feature map (2048 channels, output stride 32)

Method

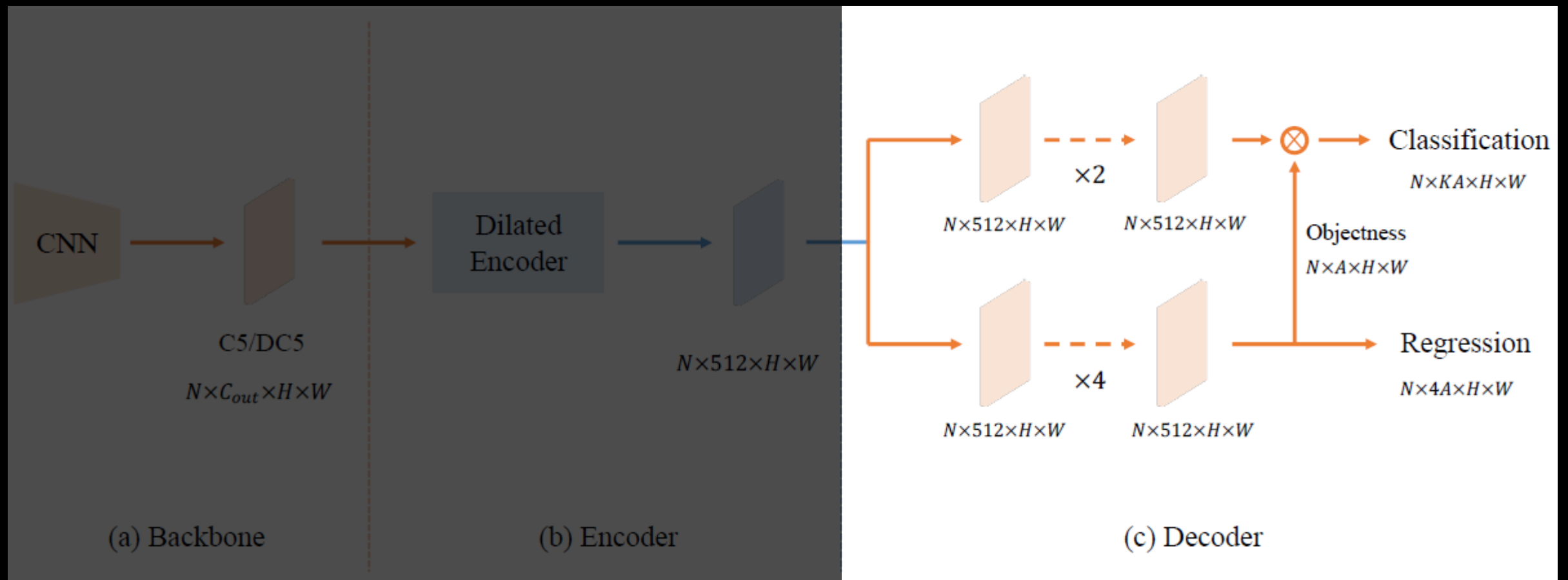
3. YOLOF: Encoder



- Two projection layer (1×1 - 3×3) to reduce channels ($2048 \rightarrow 512$)
- Stacked residual blocks (1×1 - 3×3 w/ dilation - 1×1)

Method

3. YOLOF: Decoder

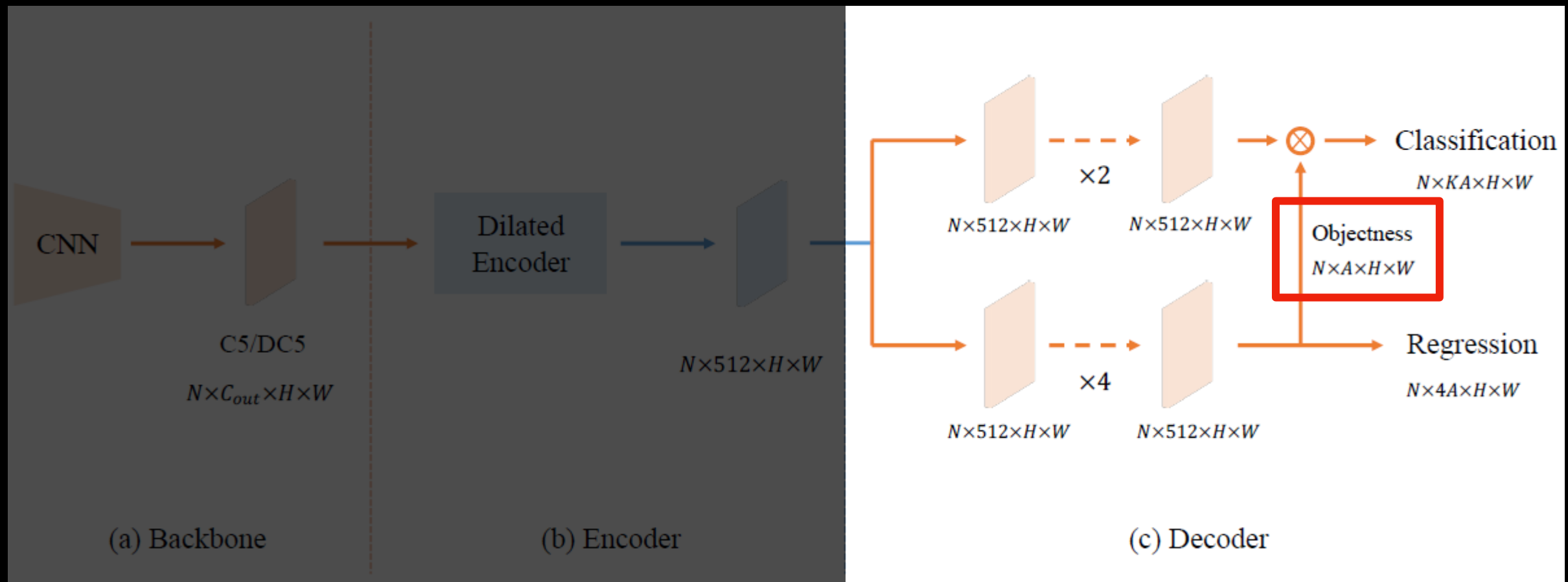


- Two parallel task-specific heads: the clf head and the reg head
- Two minor modification
 - (1) follow the design of FFN in DETR & different # of conv.
 - (2) follow AutoAssign and implicit objectness prediction

Method

3. YOLOF: Decoder

Objectness (FG or BG) without direct supervision



- Two parallel task-specific heads: the clf head and the reg head
- Two minor modification
 - (1) follow the design of FFN in DETR & different # of conv.
 - (2) follow AutoAssign and implicit objectness prediction

Method

3. YOLOF: Decoder

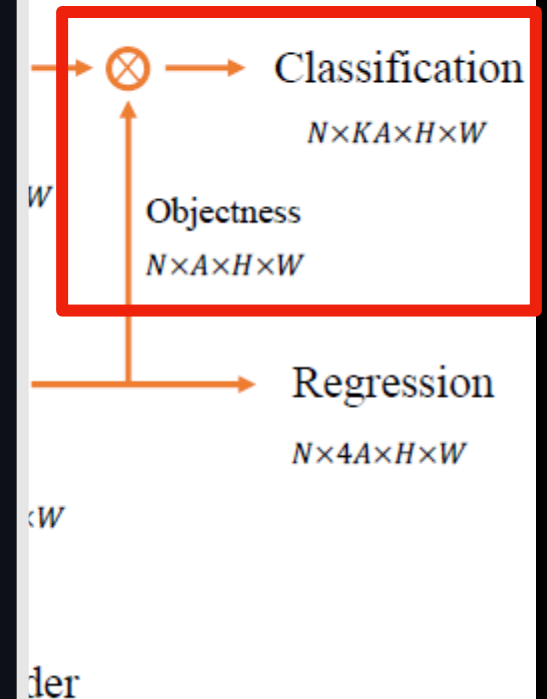
Objectness (FG or BG) without direct supervision

```
def forward(self,
            feature: torch.Tensor) -> Tuple[torch.Tensor, torch.Tensor]:
    cls_score = self.cls_score(self.cls_subnet(feature))
    N, _, H, W = cls_score.shape
    cls_score = cls_score.view(N, -1, self.num_classes, H, W)

    reg_feat = self.bbox_subnet(feature)
    bbox_reg = self.bbox_pred(reg_feat)

    objectness = self.object_pred(reg_feat) # 3x3 Conv

    # implicit objectness
    objectness = objectness.view(N, -1, 1, H, W)
    normalized_cls_score = cls_score + objectness - torch.log(
        1. + torch.clamp(cls_score.exp(), max=self.INF) + torch.clamp(
            objectness.exp(), max=self.INF))
    normalized_cls_score = normalized_cls_score.view(N, -1, H, W)
    return normalized_cls_score, bbox_reg
```



reg head

(2) follow AutoAssign and implicit objectness prediction

Method

3. YOLOF: Other Details

Random Shift Operation

- Sparse pre-defined anchors in YOLOF
 - decrease the match quality between anchors and ground-truth boxes
- Random shift operation (max 32 pixels) to inject noises into the position
- Increase the prob. of GT boxes matching with high-quality anchors
- Restriction on the anchors' center's shift is also helpful (< 32 pixels)

Experiments

Implementation Details

- COCO benchmark
- Synchronized SGD over 8 GPUs with 64 batch-size
- Initial learning rate: 0.12
- Following DETR, a smaller learning rate for the backbone (1/3)
- Warmup iteration: 1500
- '1x' schedule: a total of 22.5k iterations (lr decay at 15k and 20k)
- Inference: NMS (threshold = 0.6)
- Other hyperparameters follow the settings of RetinaNet

Experiments

1. Comparison with Previous Works: RetinaNet

Model	schedule	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#params	GFLOPs	FPS
RetinaNet [23]	1x	35.9	55.7	38.5	19.4	39.5	48.2	38M	201	13
RetinaNet-R101 [23]	1x	38.3	58.5	41.3	21.7	42.5	51.2	57M	266	11
RetinaNet+	1x	37.7	58.1	40.2	22.2	41.7	49.9	38M	201	13
RetinaNet-R101+	1x	40.0	60.4	42.7	23.2	44.1	53.3	57M	266	10
YOLOF	1x	37.7	56.9	40.6	19.1	42.5	53.2	44M	86	32
YOLOF-R101	1x	39.8	59.4	42.9	20.5	44.5	54.9	63M	151	21
YOLOF-X101	1x	42.2	62.1	45.7	23.2	47.0	57.7	102M	289	10
A YOLOF-X101 [†]	3x	44.7	64.1	48.6	25.1	49.2	60.9	102M	289	10
B YOLOF-X101 ^{†‡}	3x	47.1	66.4	51.2	31.8	50.9	60.6	102M	-	-

- RetinaNet+: implicit objectness and GN layer is adopted to RetinaNet
- '1x' models: single scale training (short side as 800 px, longer side as at most 1333)
- **A**: multi-scale training
- **B**: multi-scale training + multi-scale testing

Experiments

1. Comparison with Previous Works: DETR

Model	Epochs	#params	GFLOPS/FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR [4]	500	41M	86/24*	42.0	62.4	44.2	20.5	45.8	61.1
DETR-R101 [4]	500	60M	152/17*	43.5	63.8	46.4	21.9	48.0	61.8
YOLOF	72	44M	86/32	41.6	60.5	45.0	22.4	46.2	57.6
YOLOF-R101	72	63M	151/21	43.7	62.7	47.4	24.3	48.3	58.9

- DETR achieved amazing performance only adopting a single C5 features. That means transformer layers can capture global dependencies.
- YOLOF shows that a conventional local convolution layers can also achieve this goal!

Experiments

1. Comparison with Previous Works: YOLOv4

Model	Epochs	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4 [1]	273	53*	43.5	65.7	47.3	26.7	47.6	53.3
YOLOF-DC5	184	60 [†]	44.3	62.9	47.5	24.0	48.5	60.4

- YOLOF-DC5: add **dilations** on the last stage of the backbone
- YOLOF-DC5 can run 13% faster than YOLOv4 with a 0.8 mAP improvement

Experiments

2. Ablation Experiments: Dilated Encoder and Uniform Matching

<i>Dilated Encoder</i>	<i>Uniform Matching</i>	AP	Δ	AP_S	AP_M	AP_L
		21.1	-16.6	8.6	31.1	34.5
✓		29.1	-8.6	9.5	32.2	50.6
	✓	33.8	-3.9	17.7	40.9	43.8
✓	✓	37.7	-	19.1	42.5	53.2

Experiments

2. Ablation Experiments: Dilated Encoder and Uniform Matching

<i>Dilated Encoder</i>	<i>Uniform Matching</i>	AP	Δ	AP_S	AP_M	AP_L
		21.1	-16.6	8.6	31.1	34.5
✓		29.1	-8.6	9.5	32.2	50.6
	✓	33.8	-3.9	17.7	40.9	43.8
✓	✓	37.7	-	19.1	42.5	53.2

- Dilated Encoder has a significant impact on large objects

Experiments

2. Ablation Experiments: Dilated Encoder and Uniform Matching

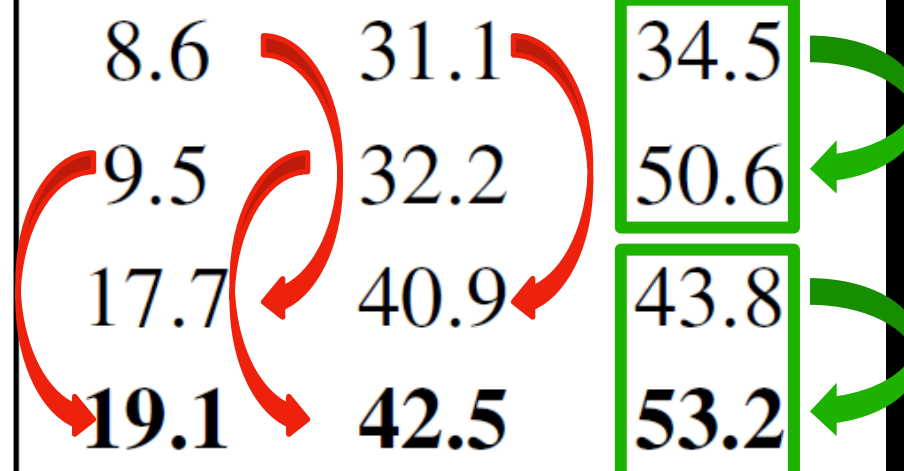
<i>Dilated Encoder</i>	<i>Uniform Matching</i>	AP	Δ	AP_S	AP_M	AP_L
		21.1	-16.6	8.6	31.1	34.5
✓		29.1	-8.6	9.5	32.2	50.6
	✓	33.8	-3.9	17.7	40.9	43.8
✓	✓	37.7	-	19.1	42.5	53.2

- Dilated Encoder has a significant impact on large objects
- Uniform Matching has a significant impact on small and medium objects

Experiments

2. Ablation Experiments: Dilated Encoder and Uniform Matching

<i>Dilated Encoder</i>	<i>Uniform Matching</i>	AP	Δ	AP_S	AP_M	AP_L
		21.1	-16.6	8.6	31.1	34.5
✓		29.1	-8.6	9.5	32.2	50.6
	✓	33.8	-3.9	17.7	40.9	43.8
✓	✓	37.7	-	19.1	42.5	53.2



- Dilated Encoder has a significant impact on large objects
- Uniform Matching has a significant impact on small and medium objects

Experiments

2. Ablation Experiments: ResBlocks and Dilations

N	AP	AP_s	AP_m	AP_l
0	33.8	17.7	40.9	43.8
2	34.9	17.8	41.3	46.8
4	35.5	17.6	41.4	48.4
6	36.0	17.7	41.9	49.5
8	36.6	18.5	42.0	50.7
10	36.9	18.3	42.4	50.4

(a) Number of ResBlocks (ResNet-50):

More residual blocks bring more gains. N represent the number of ResBlocks. To keep YOLOF simple and neat, we add 4 blocks in the encoder by default.

$Dilations$	AP	AP_s	AP_m	AP_l
1,1,1,1	35.5	17.6	41.4	48.4
2,2,2,2	36.4	18.1	41.8	50.2
3,3,3,3	36.9	18.4	42.1	51.0
1,2,3,4	37.4	18.6	42.6	51.8
2,4,6,8	37.7	19.1	42.5	53.2
3,6,9,12	37.3	18.7	42.1	52.6

(b) Different dilations (ResNet-50-N4):

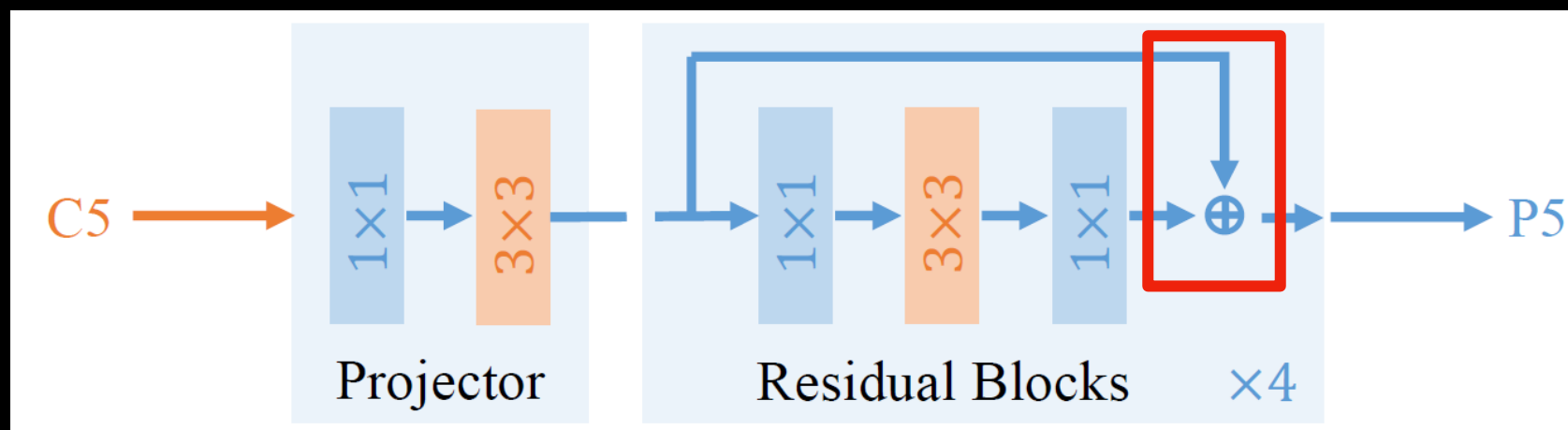
'N4' means we add 4 ResBlocks in the encoder. Dilation in the residual block gives large gains on large objects and slightly improve the performance of small and medium objects.

Experiments

2. Ablation Experiments: Add Short Cut of Not

<i>Dilations & Shortcut</i>	AP	AP _s	AP _m	AP _l
2,4,6,8 ✓	37.7	19.1	42.5	53.2
2,4,6,8 -	34.1	16.2	38.4	47.5
1,1,1,1 ✓	35.5	17.6	41.4	48.4
1,1,1,1 -	32.6	15.0	38.4	44.2

(c) **Add shortcut or not (ResNet-50):** YOLOF results with shortcuts or not on various dilation settings. Shortcut brings considerable gains on all object scales and becomes more important when the dilations are adopted (+3.6 AP with dilations 2,4,6,8 vs. +2.9 AP when dilations are all ones).



Experiments

2. Ablation Experiments: Number of Positives

$topk$	AP	AP ₅₀	AP ₇₅	AP _s	AP _m	AP _l
top1	35.9	55.6	38.4	17.5	40.3	50.2
top2	37.2	56.7	39.9	18.9	41.6	52.0
top3	37.5	57.1	40.2	18.6	41.9	52.5
top4	37.7	56.9	40.6	19.1	42.5	53.2
top5	37.5	56.7	40.3	18.1	42	53.2

(d) **Number of positives (ResNet-50-N4):** Number of positive anchors in *Uniform Matching*. Increase the positive anchor for each ground-truth box can improve the performance while it saturates when too many positive anchors. We choose the top4 anchors in YOLOF which achieves best results.

Experiments

2. Ablation Experiments: Uniform Matching vs. Others

<i>Matching Methods</i>	AP	AP ₅₀	AP ₇₅	AP _s	AP _m	AP _l
Max-IoU Matching [23]	29.1	45.9	29.6	9.5	32.2	50.6
ATSS(topk=9) [48]	34.6	54.3	37.1	17.7	40.6	46.9
ATSS(topk=15) [48]*	36.5	55.9	38.6	18.1	41.4	50.8
Hungarian Matching [4]	35.8	55.5	38.3	18.2	39.9	50.2
Uniform Matching	37.7	56.9	40.6	19.1	42.5	53.2

(e) **Uniform matching vs. other matchings (ResNet-50-N4):** Comparison with other matching methods. Uniform Matching achieve balance in positive anchors and get the best results among other matching methods, which is consistent with the comparison in Figure 6. Note that '*' represents that we get the best result for ATSS [48] when setting topk as 15. More details can be found in the Appendix.

Conclusion

- The success of FPN is due to its divide-and-conquer solution to the optimization problem in dense object detection
- YOLOF: a simple but highly efficient method without FPN
- Comparable performance with RetinaNet and DETR only with a single-level feature

Thank you