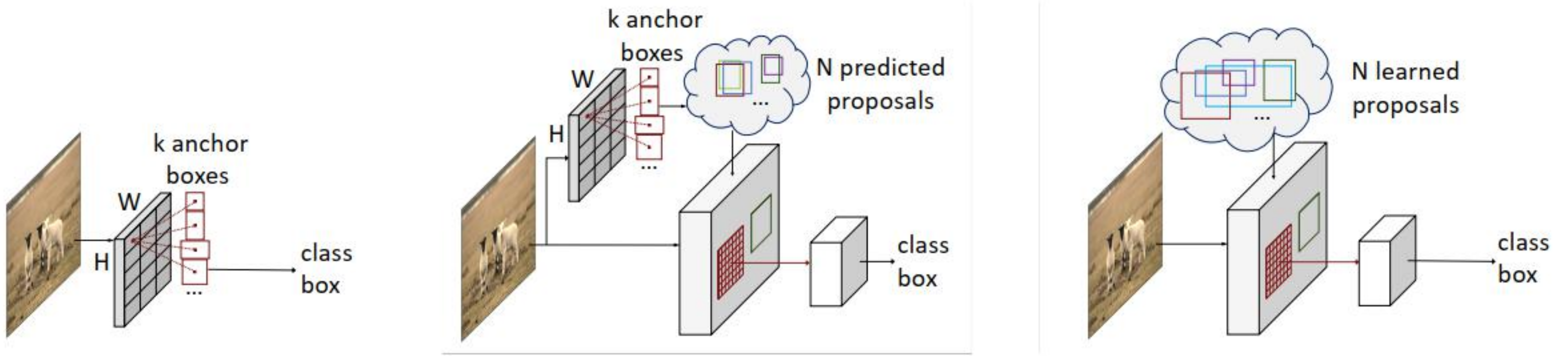
Sparse R-CNN: End-to-End Object Detection with Learnable Proposals

Peize Sun[1*], Rufeng Zhang[2*], Yi Jiang[3*], Tao Kong[3], Chenfeng Xu[4], Wei Zhan[4], Masayoshi Tomizuka[4], Lei Li[3], Zehuan Yuan[3], Changhu Wang[3], Ping Luo[1]

[1]The University of Hong Kong [2]Tongji University
[3]ByteDance AI Lab [4]University of California, Berkeley

Comparisons of different object detection pipelines

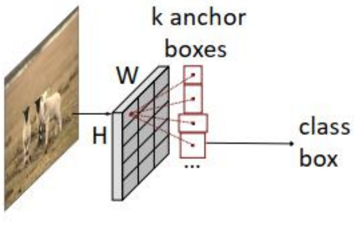


(a) **Dense**: RetinaNet

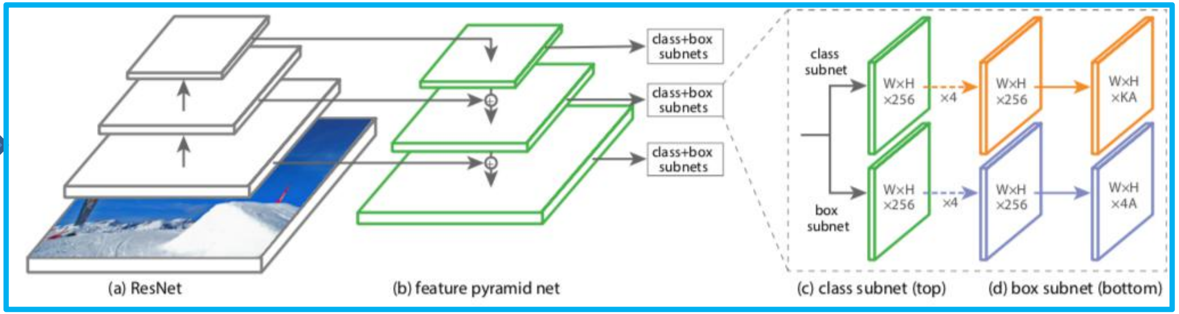
(b) **Dense-to-Sparse**: Faster R-CNN

(c) **Sparse**: Sparse R-CNN

Comparisons of different object detection pipelines



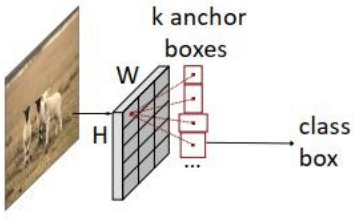
(a) **Dense**: RetinaNet



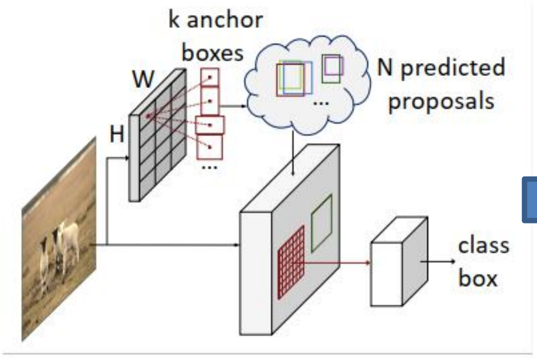
HW k object candidates

RetinaNet： Focal Loss for Dense Object Detection

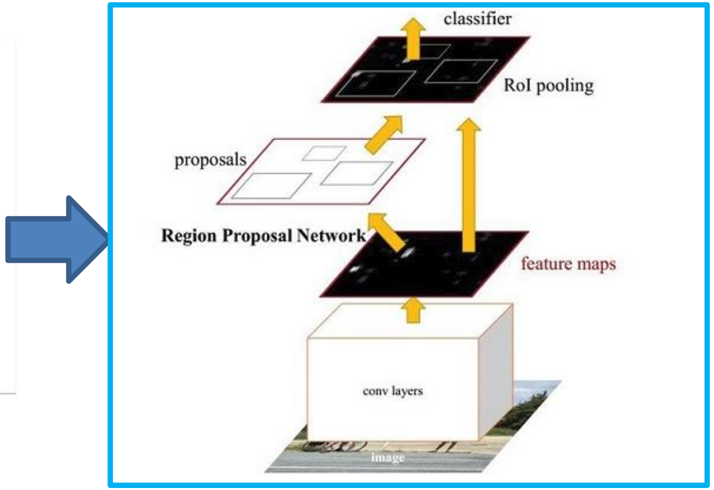
Comparisons of different object detection pipelines



(a) **Dense**: RetinaNet



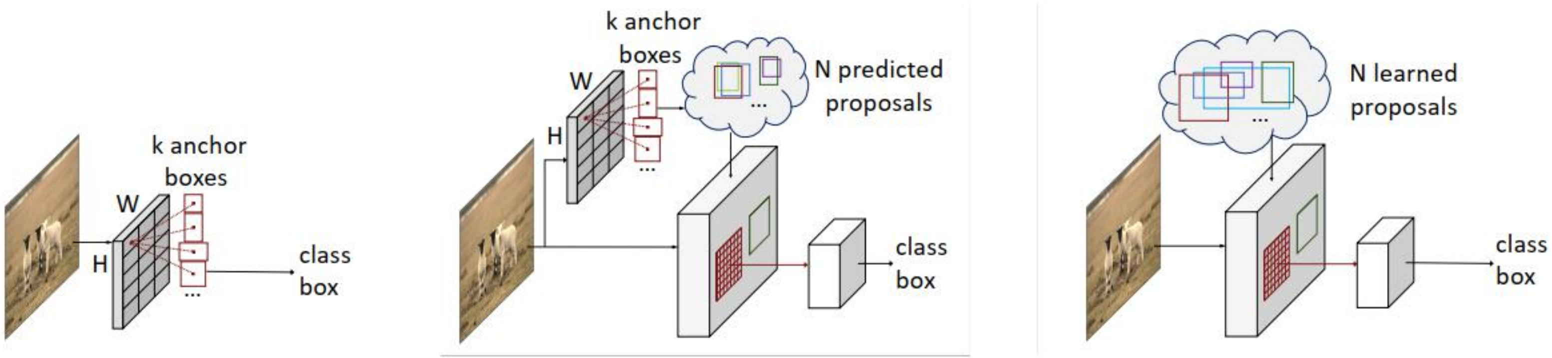
(b) **Dense-to-Sparse**: Faster R-CNN



select a small set of *N* candidates from
dense *HW k* object candidates(RPN)

extract image features within corresponding
regions by pooling operation

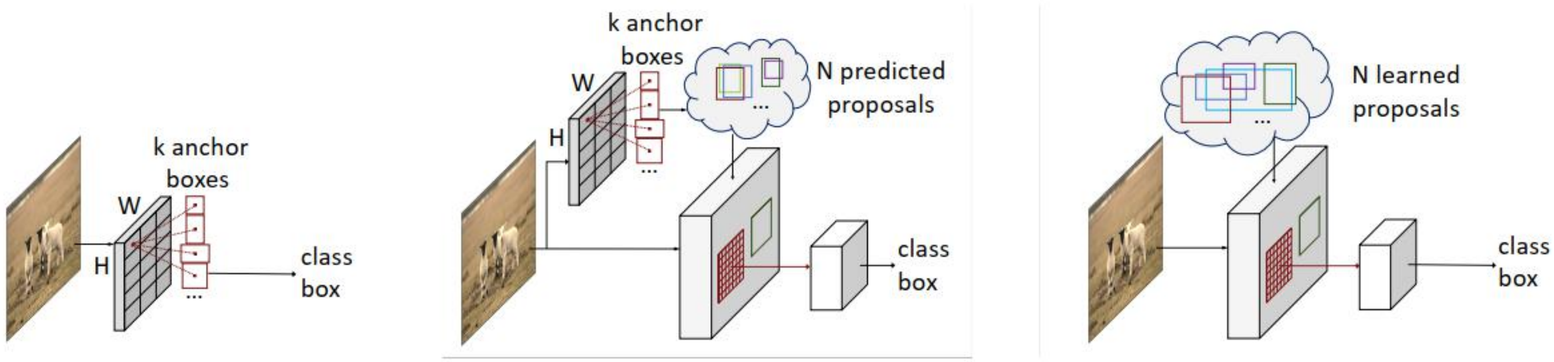
Comparisons of different object detection pipelines



(a) **Dense**: RetinaNet (b) **Dense-to-Sparse**: Faster R-CNN (c) **Sparse**: Sparse R-CNN

directly provides a small set of *N* learned object proposals.

Comparisons of different object detection pipelines



(a) **Dense**: RetinaNet (b) **Dense-to-Sparse**: Faster R-CNN (c) **Sparse**: Sparse R-CNN

directly provides a small set of *N* learned object proposals.

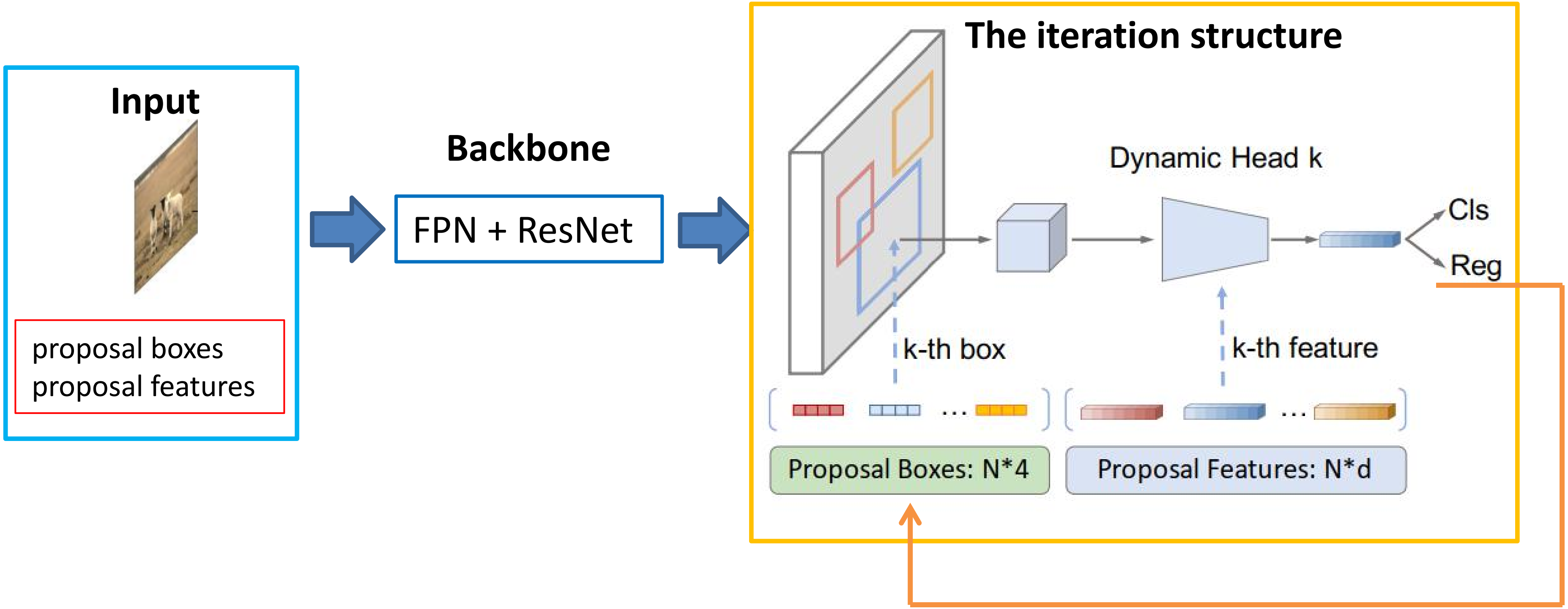
Here *N* << *HW k*

HW k (up to hundreds of thousands)

N (e.g.*100)*

without non-maximum suppression post procedure

Overview of Sparse R-CNN



Overview of Sparse R-CNN

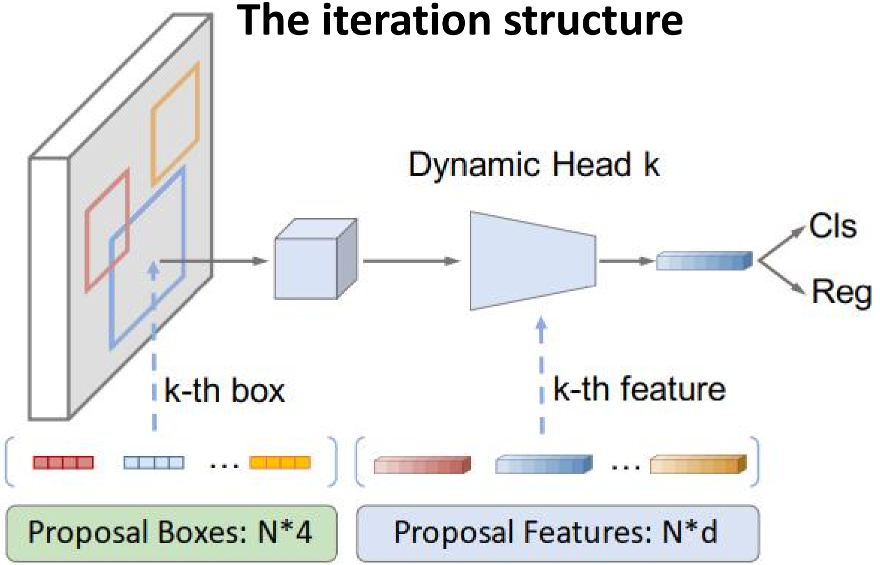
Input

proposal boxes
proposal features

Backbone

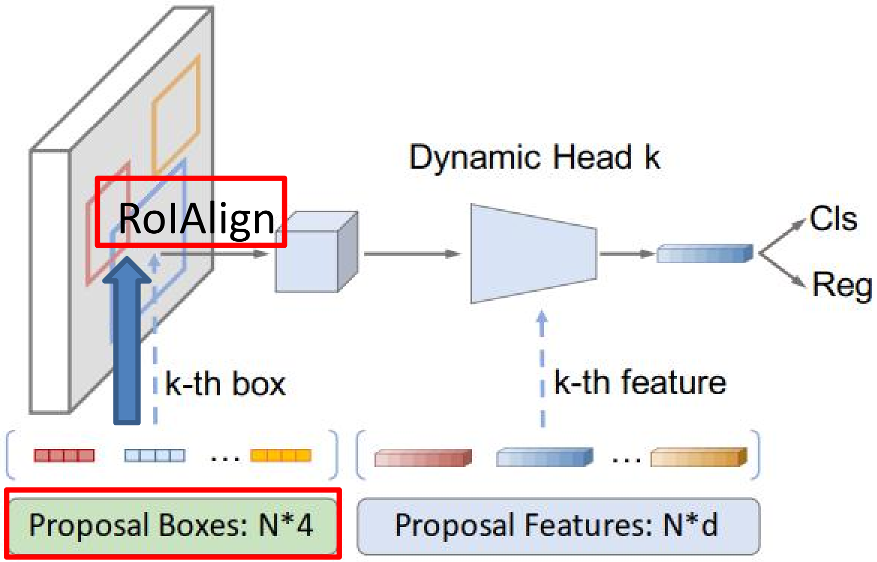
FPN + ResNet

The iteration structure



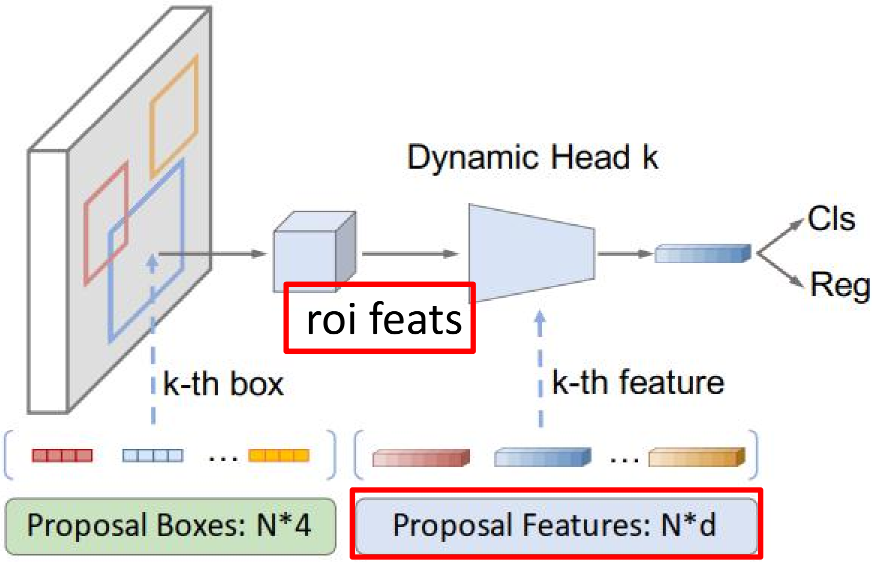
- **1. Input**: an image, a set of proposal boxes and proposal features,
  the latter two are learnable parameters.
  2. Backbone: FPN + ResNet
  3. Regression prediction :3-layer perception
  4.Classification prediction :a linear projection.

Details of Sparse R-CNN



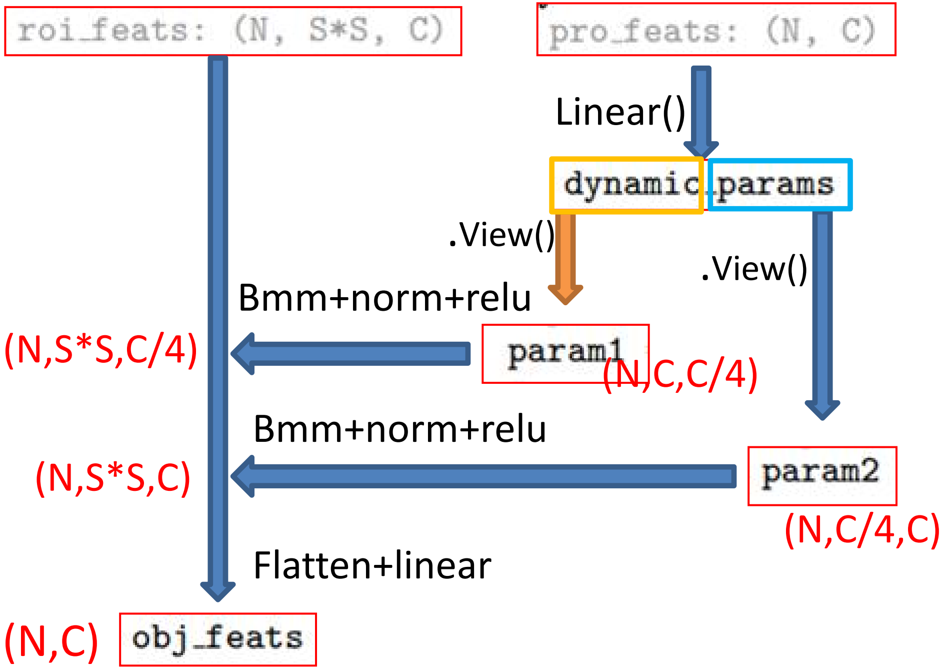
coarse localization of objects

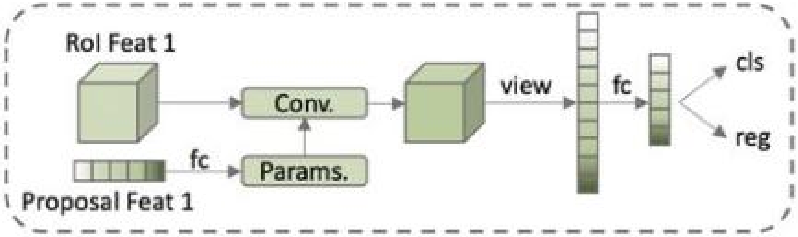
Details of Sparse R-CNN



coarse localization of objects

Dynamic Instance Interaction





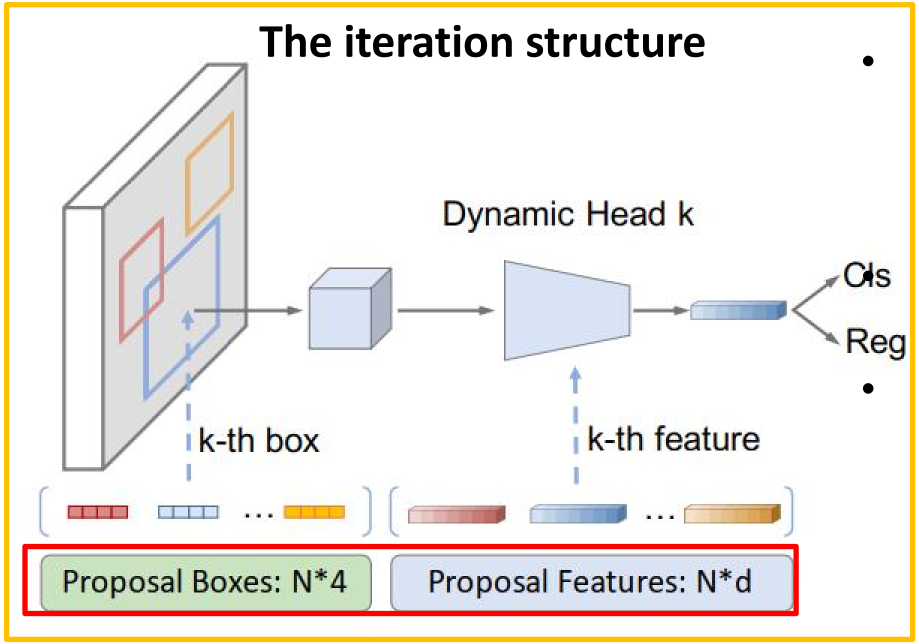

```python
def dynamic_instance_interaction(pro_feats, roi_feats):
    # pro_feats: (N, C)
    # roi_feats: (N, S*S, C)

    # parameters of two 1x1 convs: (N, 2 * C * C/4)
    dynamic_params = linear1(pro_features)
    # parameters of first conv: (N, C, C/4)
    param1 = dynamic_params[:, :C*C/4].view(N, C, C/4)
    # parameters of second conv: (N, C/4, C)
    param2 = dynamic_params[:, C*C/4:].view(N, C/4, C)

    # instance interaction for roi_features: (N, S*S, C)
    roi_feats = relu(norm(bmm(roi_feats, param1)))
    roi_feats = relu(norm(bmm(roi_feats, param2)))

    # roi_feats are flattened: (N, S*S*C)
    roi_feats = roi_feats.flatten(1)
    # obj_feats: (N, C)
    obj_feats = linear2(roi_feats)
    return obj_feats
```

Details of Sparse R-CNN



coarse localization of objects

- **the iteration structure**: the newly generated object boxes and object features will serve as the proposal boxes and proposal features of the next stage in iterative process .

  (Cascade R-CNN)

- **self-attention module** :Before dynamic instance interaction, on object features.(Attention is all you need）

prediction loss

$$\boxed{\mathcal{L} = \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{L1} \cdot \mathcal{L}_{L1} + \lambda_{giou} \cdot \mathcal{L}_{giou}}$$

focal loss L1 loss generalized IoU loss

$$\lambda_{cls} = 2,\ \lambda_{L1} = 5,\ \lambda_{giou} = 2.$$

- The final loss only performed on matched pairs
- optimal bipartite matching between predictions and ground truth objects

Training details

- 1. ResNet-50 as backbones. The backbone is initialized with the pre-trained weights on ImageNet
- 2. The mini-batch is 16 images and all models are trained with 8 GPUs (NVIDIA Tesla V100 GPU)
- 3. 36 epochs
- 4. proposal boxes, proposal features: 100
- 5. iteration is 6

Experiments

Method	Feature	Epochs	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	FPS
RetinaNet-R50 [53]	FPN	36	38.7	58.0	41.5	23.3	42.3	50.3	24
RetinaNet-R101 [53]	FPN	36	40.4	60.2	43.2	24.0	44.3	52.2	18
Faster R-CNN-R50 [53]	FPN	36	40.2	61.0	43.8	24.2	43.5	52.0	26
Faster R-CNN-R101 [53]	FPN	36	42.0	62.5	45.9	25.2	45.6	54.6	20
Cascade R-CNN-R50 [53]	FPN	36	44.3	62.2	48.0	26.6	47.7	57.7	19
DETR-R50 [3]	Encoder	500	42.0	62.4	44.2	20.5	45.8	61.1	28
DETR-R101 [3]	Encoder	500	43.5	63.8	46.4	21.9	48.0	61.8	20
DETR-DC5-R50 [3]	Encoder	500	43.3	63.1	45.9	22.5	47.3	61.1	12
DETR-DC5-R101 [3]	Encoder	500	44.9	**64.7**	47.7	23.7	**49.5**	**62.3**	10
Deformable DETR-R50 [63]	DeformEncoder	50	43.8	62.6	47.7	26.4	47.1	58.0	19
Sparse R-CNN-R50	FPN	36	42.8	61.2	45.7	26.7	44.6	57.6	23
Sparse R-CNN-R101	FPN	36	44.1	62.1	47.2	26.1	46.3	59.7	19
Sparse R-CNN*-R50	FPN	36	45.0	63.4	48.2	26.9	47.2	59.5	22
Sparse R-CNN*-R101	FPN	36	**46.4**	64.6	**49.5**	**28.3**	48.3	61.6	18

Table 1 – Comparisons with different object detectors on COCO 2017 val set. The top section shows results from Detectron2 [53] or original papers [3, 63]. Here "∗" indicates that the model is with 300 learnable proposal boxes and random crop training augmentation, similar to Deformable DETR [63]. Run time is evaluated on NVIDIA Tesla V100 GPU.

Ablation study

Sparse	Iterative	Dynamic	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
✓			18.5	35.0	17.7	8.3	21.7	26.4
✓	✓		32.2 **(+13.7)**	47.5 (+12.5)	34.4 (+16.7)	18.2 (+9.9)	35.2 (+13.5)	41.7 (+15.3)
✓	✓	✓	42.3 **(+10.1)**	61.2 (+13.7)	45.7 (+11.3)	26.7 (+8.5)	44.6 (+9.4)	57.6 (+15.9)

Table 3 – Ablation studies on each components in Sparse R-CNN. Starting from Faster R-CNN, we gradually add learnable proposal boxes, iterative architecture, and dynamic head in Sparse R-CNN. All models are trained with set prediction loss.

从Faster R-CNN(40.2 AP)出发，直接将RPN替换为a sparse set of learnable proposal boxes，AP降到18.5；引入iterative结构提升AP到32.2；引入dynamic instance interaction最终提升到42.3 AP。

The effect of feature reuse

Cascade	Feature reuse	AP	AP_{50}	AP_{75}
		18.5	35.0	17.7
✓		20.5(**+2.0**)	29.3	20.7
✓	✓	32.2(**+11.7**)	47.5	34.4

Table 4 – The effect of feature reuse in iterative architecture. Original cascading implementation makes no big difference. Concatenating object feature of previous stage to object feature of current stage leads to a huge improvement.

The effect of instance-interaction in dynamic head

Self-att.	Ins. interact	AP	AP_{50}	AP_{75}
		32.2	47.5	34.4
✓		37.2(**+5.0**)	54.8	40.1
✓	✓	42.3(**+5.1**)	61.2	45.7

Table 5 – The effect of instance-interaction in dynamic head. Without instance interaction, dynamic head degenerates to self-attention. The gain comes from both self-attention and instance-interaction.

Initialization of proposal boxes

Init.	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Center	41.5	59.6	45.0	25.6	43.9	56.1
Image	42.3	61.2	45.7	26.7	44.6	57.6
Grid	41.0	59.4	44.2	23.8	43.7	55.6
Random	42.1	60.3	45.3	24.5	44.6	57.9

Table 6 – Effect of initialization of proposal boxes. Detection performance is relatively robust to initialization of proposal boxes.

- "Center" means all proposal boxes are located in the center of image at beginning, height and width is set to 0.1 of image size.
- "Image" means all proposal boxes are initialized as the whole image size.
- "Grid" means proposal boxes are initialized as regular grid in image, which is exactly the initial boxes in G-CNN [34].
- "Random" denotes the center, height and width of proposal boxes are randomly initialized with Gaussian

number of proposals

Proposals	AP	AP_{50}	AP_{75}	FPS	Training time
100	42.3	61.2	45.7	23	19h
300	43.9	62.3	47.4	22	24h
500	44.6	63.2	48.5	20	60h

Table 7 – Effect of number of proposals. Increasing number of proposals leads to continuous improvement, while more proposals take more training time.

number of stages

Stages	AP	AP_{50}	AP_{75}	FPS	Training time
1	21.7	36.7	22.3	35	12h
2	36.2	52.8	38.8	33	13h
3	39.9	56.8	43.2	29	15h
6	42.3	61.2	45.7	23	19h
12	41.6	60.2	45.0	17	30h

Table 8 – Effect of number of stages. Gradually increasing the number of stages, the performance is saturated at 6 stages.

Method	Pos. encoding	AP	AP_{50}	AP_{75}
DETR [3]	✓	40.6	61.6	-
DETR [3]		32.8 (**-7.8**)	55.2	-
Sparse R-CNN	✓	41.9	60.9	45.0
Sparse R-CNN		42.3(**+0.4**)	61.2	45.7

Table 10 – Proposal feature vs. Object query. Object query is learned positional encoding, while proposal feature is irrelevant to position.

(a) Learned proposal boxes (b) Stage1 boxes (c) Stage3 boxes (d) Stage6 boxes