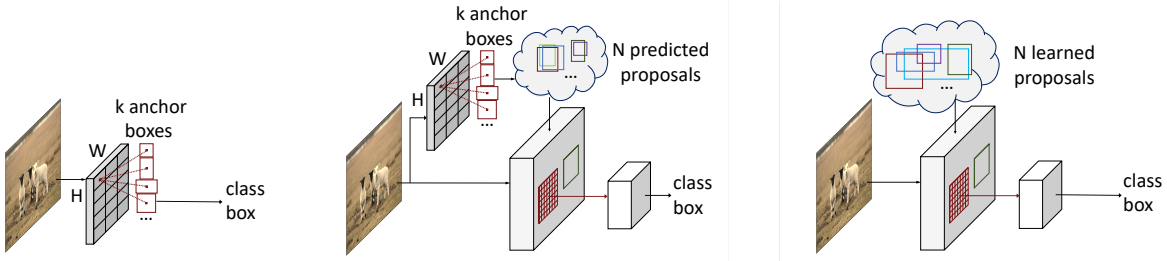
Sparse R-CNN: End-to-End Object Detection with Learnable Proposals

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



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

Figure 1: **Comparisons** of different object detection pipelines. (a) In dense detectors, HWk object candidates enumerate on all image grids, *e.g.* RetinaNet [23]. (b) In dense-to-sparse detectors, they select a small set of N candidates from dense HWk object candidates, and then extract image features within corresponding regions by pooling operation, *e.g.* Faster R-CNN [30]. (c) Our proposed Sparse R-CNN, directly provides a small set of N learned object proposals. Here $N \ll HWk$.

Abstract

*We present Sparse R-CNN, a purely sparse method for object detection in images. Existing works on object detection heavily rely on dense object candidates, such as k anchor boxes pre-defined on all grids of image feature map of size $H \times W$. In our method, however, a fixed sparse set of learned object proposals, total length of N, are provided to object recognition head to perform classification and location. By eliminating HWk (up to hundreds of thousands) hand-designed object candidates to N (*e.g.* 100) learnable proposals, Sparse R-CNN completely avoids all efforts related to object candidates design and many-to-one label assignment. More importantly, final predictions are directly output without non-maximum suppression post-procedure. Sparse R-CNN demonstrates accuracy, run-time and training convergence performance on par with the well-established detector baselines on the challenging COCO dataset,* e.g., *achieving 44.5 AP in standard $3\times$ training schedule and running at 22 fps using ResNet-50 FPN model. We hope our work could inspire re-thinking the convention of dense prior in object detectors. The code is available at:* https://github.com/PeizeSun/SparseR-CNN.

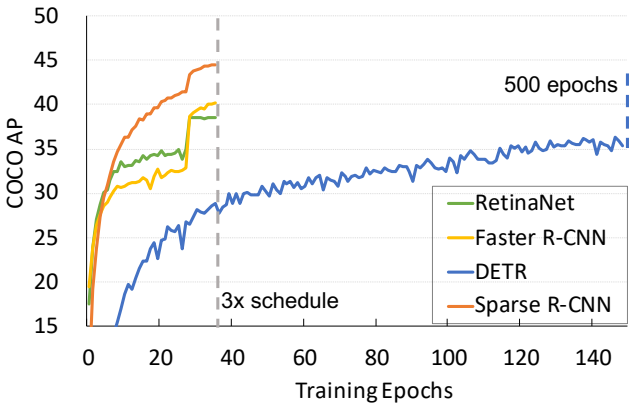


Figure 2: Convergence curves of RetinaNet, Faster R-CNN, DETR and Sparse R-CNN on COCO `val2017` [24]. Sparse R-CNN achieves competitive performance in terms of training efficiency and detection quality.

1. Introduction

Object detection aims at localizing a set of objects and recognizing their categories in an image. Dense prior has always been cornerstone to success in detectors. In classic computer vision, the sliding-window paradigm, in which a classifier is applied on a dense image grid, is leading de-

* Equal contribution.

tection method for decades [6, 9, 38]. Modern mainstream one-stage detectors pre-define marks on a dense feature map grid, such as anchors boxes [23, 29], shown in Figure 1a, or reference points [35, 44], and predict the relative scaling and offsets to bounding boxes of objects, as well as the corresponding categories. Although two-stage pipelines work on a sparse set of proposal boxes, their proposal generation algorithms are still built on dense candidates [11, 30], shown in Figure 1b.

These well-established methods are conceptually intuitive and offer robust performance [8, 24], together with fast training and inference time [40]. Besides their great success, it is important to note that dense-prior detectors suffer some limitations: 1) Such pipelines usually produce redundant and near-duplicate results, thus making non-maximum suppression (NMS) [1, 39] post-processing a necessary component. 2) The many-to-one label assignment problem [2, 42, 43] in training makes the network sensitive to heuristic assign rules. 3) The final performance is largely affected by sizes, aspect ratios and number of anchor boxes [23, 29], density of reference points [19, 35, 44] and proposal generation algorithm [11, 30].

Despite the dense convention is widely recognized among object detectors, a natural question to ask is: *Is it possible to design a sparse detector?* Recently, DETR proposes to reformulate object detection as a direct and sparse set prediction problem [3], whose input is merely 100 learned object queries [37]. The final set of predictions are output directly without any hand-designed post-processing. In spite of its simple and fantastic framework, DETR requires each object query to interact with global image context. This dense property not only slows down its training convergence [45], but also blocks it establishing a thoroughly sparse pipeline for object detection.

We believe the sparse property should be in two aspects: *sparse boxes* and *sparse features*. Sparse boxes mean that a small number of starting boxes (*e.g*. 100) is enough to predict all objects in an image. While sparse features indicate the feature of each box does not need to interactively interact with all other features over the full image. From this perspective, DETR is not a pure spare method since each object query must interact with dense features over full images.

In this paper, we propose Sparse R-CNN, a purely sparse method, without object positional candidates enumerating on *all(dense) image grids* nor object queries interacting with *global(dense) image feature*. As shown in Figure 1c, object candidates are given with a fixed small set of learnable bounding boxes represented by 4-d coordinate. For example of COCO dataset [24], 100 boxes and 400 parameters are needed in total, rather than the predicted ones from hundreds of thousands of candidates in Region Proposal Network (RPN) [30]. These sparse candidates are used as proposal boxes to extract the feature of Region of Interest (RoI) by RoIPool [10] or RoIAlign [13].

The learnable proposal boxes are the statistics of potential object location in the image. Whereas, the 4-d coordinate is merely a rough representation of object and lacks a lot of informative details such as pose and shape. Here we introduce another concept termed *proposal feature*, which is a high-dimension (*e.g*., 256) latent vector. Compared with rough bounding box, it is expected to encode the rich instance characteristics. Specially, proposal feature generates a series of customized parameters for its exclusive object recognition head. We call this operation Dynamic Instance Interactive Head, since it shares similarities with recent dynamic scheme [18, 34]. Compared to the shared 2-fc layers in [30], our head is more flexible and holds a significant lead in accuracy. We show in our experiment that the formulation of head conditioned on unique proposal feature instead of the fixed parameters is actually the key to Sparse R-CNN's success. Both *proposal boxes* and *proposal features* are randomly initialized and optimized together with other parameters in the whole network.

The most remarkable property in our Sparse R-CNN is its sparse-in sparse-out paradigm in the whole time. The initial input is a sparse set of proposal boxes and proposal features, together with the one-to-one dynamic instance interaction. Neither dense candidates [23, 30] nor interacting with global(dense) feature [3] exists in the pipeline. This pure sparsity makes Sparse R-CNN a brand new member in R-CNN family.

Sparse R-CNN demonstrates its accuracy, run-time and training convergence performance on par with the well-established detectors [2, 30, 35] on the challenging COCO dataset [24], *e.g*., achieving 44.5 AP in standard $3\times$ training schedule and running at 22 fps using ResNet-50 FPN model. To our best knowledge, the proposed Sparse R-CNN is the first work that demonstrates a considerably sparse design is qualified yet. We hope our work could inspire rethinking the necessary of dense prior in object detection and exploring next generation of object detector.

2. Related Work

Dense method. Sliding-window paradigm has been popular for many years in object detection. Limited by classical feature extraction techniques [6, 38], the performance has plateaued for decades and the application scenarios are limited. Development of deep convolution neural networks (CNNs) [14, 17, 20] cultivates general object detection achieving significant improvement in performance [8, 24]. One of mainstream pipelines is one-stage detector, which directly predicts the category and location of anchor boxes densely covering spatial positions, scales, and aspect ratios in a single-shot way, such as OverFeat [32], YOLO [29],

SSD [25] and RetinaNet [23]. Recently, anchor-free algorithms [16, 21, 35, 44] are proposed to make this pipeline much simpler by replacing hand-crafted anchor boxes with reference points. All of above methods are built on dense candidates and each candidate is directly classified and regressed. These candidates are assigned to ground-truth object boxes in training time based on a pre-defined principle, *e.g.*, whether the anchor has a higher intersection-over-union (IoU) threshold with its corresponding ground truth, or whether the reference point falls in one of object boxes. Moreover, NMS post-processing [1, 39] is needed to remove redundant predictions during inference time.

Dense-to-sparse method. Two-stage detector is another mainstream pipeline and has dominated modern object detection for years [2, 4, 10, 11, 13, 30]. This paradigm can be viewed as an extension of dense detector. It firstly obtains a sparse set of foreground proposal boxes from dense region candidates, and then refines location of each proposal and predicts its specific category. The region proposal algorithm plays an important role in the first stage in these two-stage methods, such as Selective Search [36] in R-CNN and Region Proposal Networks (RPN) [30] in Faster R-CNN. Similar to dense pipeline, it also needs NMS post-processing and hand-crafted label assignment. There are only a few of foreground proposals from hundreds of thousands of candidates, thus these detectors can be concluded as dense-to-sparse methods.

Recently, DETR [3] is proposed to directly output the predictions without any hand-crafted components, achieving very competitive performance. DETR utilizes a sparse set of object queries, to interact with global(dense) image feature, in this view, it can be seen as another dense-to-sparse formulation.

Sparse method. Sparse object detection has the potential to eliminate efforts to design dense candidates, but has trailed the accuracy of above detectors. G-CNN [27] can be viewed as a precursor to this group of algorithms. It starts with a multi-scale regular grid over the image and iteratively updates the boxes to cover and classify objects. This hand-designed regular prior is obviously sub-optimal and fails to achieve top performance. Instead, our Sparse R-CNN applies learnable proposals and achieves better performance. Concurrently, Deformable-DETR [45] is introduced to restrict each object query to attend to a small set of key sampling points around the reference points, instead of all points in feature map. We hope sparse methods could serve as solid baseline and help ease future research in object detection community.

3. Sparse R-CNN

The central idea of Sparse R-CNN framework is to replace hundreds of thousands of candidates from Region Proposal Network (RPN) with a small set of proposal boxes (*e.g.*, 100). In this section, we first briefly introduce the overall architecture of the proposed method. Then we describe each components in details.

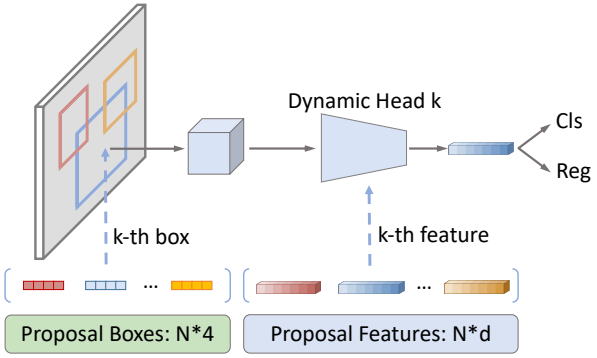


Figure 3: An overview of Sparse R-CNN pipeline. The input includes an image, a set of proposal boxes and proposal features, where the latter two are learnable parameters. The backbone extracts feature map, each proposal box and proposal feature are input into its exclusive dynamic head to generate object feature, and finally outputs classification and location.

3.1. Pipeline

Sparse R-CNN is a simple, unified network composed of a backbone network, a dynamic instance interactive head and two task-specific prediction layers (Figure 3). There are three inputs in total, an image, a set of proposal boxes and proposal features. The latter two are learnable and can be optimized together with other parameters in network.

3.2. Module

Backbone. Feature Pyramid Network (FPN) based on ResNet architecture [14, 22] is adopted as the backbone network to produce multi-scale feature maps from input image. Following [22], we construct the pyramid with levels P_2 through P_5, where l indicates pyramid level and P_l has resolution 2^l lower than the input. All pyramid levels have $C = 256$ channels. Please refer to [22] for more details. Actually, Spare R-CNN has the potential to benefit from more complex designs to further improve its performance, such as stacked encoder layers [3] and deformable convolution network [5], on which a recent work Deformable-DETR [45] is built. However, we align the setting with Faster R-CNN [30] to show the simplicity and effectiveness of our method.

Learnable proposal box. A fixed small set of learnable proposal boxes ($N \times 4$) are used as region proposals, instead of the predictions from Region Proposal Network (RPN).

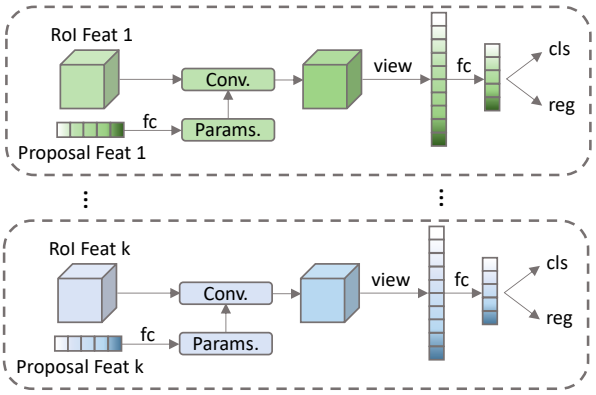


Figure 4: An overview of our dynamic instance interactive module. The filters vary with different instances, *i.e.*, the k-th proposal feature generates dynamic parameters for the corresponding k-th RoI.

These proposal boxes are represented by 4-d parameters ranging from 0 to 1, denoting normalized center coordinates, height and width. The parameters of proposals boxes will be updated with the back-propagation algorithm during training. Thanks to the learnable property, we find in our experiment that the effect of initialization is minimal, thus making the framework much more flexible.

Conceptually, these learned proposal boxes are the statistics of potential object location in the training set and can be seen as an initial guess of the regions that are most likely to encompass the objects in the image, regardless of the input. Whereas, the proposals from RPN are strongly correlated to the current image and provide coarse object locations. We rethink that the first-stage locating is luxurious in the presence of later stages to refine the location of boxes. Instead, a reasonable statistic can already be qualified candidates. In this view, Sparse R-CNN can be categorized as the extension of object detector paradigm from thoroughly dense [23, 25, 28, 35] to dense-to-sparse [2, 4, 11, 30] to thoroughly sparse, shown in Figure 1.

Learnable proposal feature. Though the 4-d proposal box is a brief and explicit expression to describe objects, it provides a coarse localization of objects and a lot of informative details are lost, such as object pose and shape. Here we introduce another concept termed *proposal feature* ($N \times d$), it is a high-dimension (*e.g.*, 256) latent vector and is expected to encode the rich instance characteristics. The number of proposal features is same as boxes, and we will discuss how to use it next.

Dynamic instance interactive head. Given N proposal boxes, Sparse R-CNN first utilizes the RoIAlign operation to extract features for each box. Then each box feature will be used to generate the final predictions using our prediction head.

Figure 4 illustrates the prediction head, termed as Dynamic Instance Interactive Module, motivated by dynamic algorithms [18, 34]. Each RoI feature is fed into its own exclusive head for object location and classification, where each head is conditioned on specific proposal feature. In our design, proposal feature and proposal box are in one-to-one correspondence. For N proposal boxs, N proposal features are employed. Each RoI feature $f_i(S \times S \times C)$ will interact with the corresponding proposal feature $p_i(C)$ to filter out ineffective bins and outputs the final object feature (C). The final regression prediction is computed by a 3-layer perception with ReLU activation function and hidden dimension C, and classification prediction is by a linear projection layer.

For light design, we carry out consecutive 1×1 convolution with ReLU activation function, to implement the interaction process. Each proposal feature p_i will be convolved with the RoI feature to get a more discriminate feature. For more details, please refer to our code. We note that implementation detail of interactive head is not crucial as long as parallel operation is supported for efficiency.

Our proposal feature can be seen as an implementation of attention mechanism, for attending to which bins in a RoI of size $S \times S$. The proposal feature generates kernel parameters of convolution, then RoI feature is processed by the generated convolution to obtain the final feature. In this way, those bins with most foreground information make effect on final object location and classification.

We also adopt the iteration structure to further improve the performance. The newly generated object boxes and object features will serve as the proposal boxes and proposal features of the next stage in iterative process. Thanks to the sparse property and light dynamic head, it introduces only a marginal computation overhead. Self-attention module [37] is embedded into dynamic head to reason about the relations between objects. We note that Relation Network [15] also utilizes attention module. However, it demands geometry attributes and complex rank feature in addition to object feature. Our module is much more simple and only takes object feature as input.

Object query proposed in DETR [3] shares a similar design as proposal feature. However, object query is learned positional encoding. Feature map is required to add spatial positional encoding when interacting with object query, otherwise leads to a significant drop. Our proposal feature is irrelevant to position and we demonstrate that our framework can work well without positional encoding. We provide further comparisons in the experimental section.

Set prediction loss. Sparse R-CNN applies set prediction loss [3, 33, 41] on the fixed-size set of predictions of classification and box coordinates. Set-based loss produces an

Method	Feature	Epochs	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	FPS
RetinaNet-R50 [40]	FPN	36	38.7	58.0	41.5	23.3	42.3	50.3	24
RetinaNet-R101 [40]	FPN	36	40.4	60.2	43.2	24.0	44.3	52.2	18
Faster R-CNN-R50 [40]	FPN	36	40.2	61.0	43.8	24.2	43.5	52.0	26
Faster R-CNN-R101 [40]	FPN	36	42.0	62.5	45.9	25.2	45.6	54.6	20
Cascade R-CNN-R50 [40]	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 [45]	DeformEncoder	50	43.8	62.6	47.7	26.4	47.1	58.0	19
Sparse R-CNN-R50	FPN	36	42.3	61.2	45.7	26.7	44.6	57.6	23
Sparse R-CNN-R101	FPN	36	43.5	62.1	47.2	26.1	46.3	59.7	19
Sparse R-CNN*-R50	FPN	36	44.5	63.4	48.2	26.9	47.2	59.5	22
Sparse R-CNN*-R101	FPN	36	**45.6**	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 [40] or original papers [3, 45]. Here "$*$" indicates that the model is with 300 learnable proposal boxes and random crop training augmentation, similar to Deformable DETR [45]. Run time is evaluated on NVIDIA Tesla V100 GPU.

optimal bipartite matching between predictions and ground truth objects. The matching cost is defined as follows:

$$\mathcal{L} = \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{L1} \cdot \mathcal{L}_{L1} + \lambda_{giou} \cdot \mathcal{L}_{giou} \qquad (1)$$

Here $\mathcal{L}_{cls}$ is focal loss [23] of predicted classifications and ground truth category labels, $\mathcal{L}_{L1}$ and $\mathcal{L}_{giou}$ are L1 loss and generalized IoU loss [31] between normalized center coordinates and height and width of predicted boxes and ground truth box, respectively. λ_{cls}, λ_{L1} and λ_{giou} are coefficients of each component. The training loss is the same as the matching cost except that only performed on matched pairs. The final loss is the sum of all pairs normalized by the number of objects inside the training batch.

R-CNN families [2, 43] have always been puzzled by label assignment problem since many-to-one matching remains. Here we provide new possibilities that directly bypassing many-to-one matching and introducing one-to-one matching with set-based loss. This is an attempt towards exploring end-to-end object detection.

4. Experiments

Dataset. Our experiments are conducted on the challenging MS COCO benchmark [24] using the standard metrics for object detection. All models are trained on the COCO `train2017` split (∼118k images) and evaluated with `val2017` (5k images).

Training details. ResNet-50 [14] is used as the backbone network unless otherwise specified. The optimizer is AdamW [26] with weight decay 0.0001. The mini-batch is 16 images and all models are trained with 8 GPUs. Default training schedule is 36 epochs and the initial learning rate is set to 2.5×10^{-5}, divided by 10 at epoch 27 and 33, respectively. The backbone is initialized with the pre-trained weights on ImageNet [7] and other newly added layers are initialized with Xavier [12]. Data augmentation includes random horizontal, scale jitter of resizing the input images such that the shortest side is at least 480 and at most 800 pixels while the longest at most 1333. Following [3, 45], $\lambda_{cls} = 2$, $\lambda_{L1} = 5$, $\lambda_{giou} = 2$. The default number of proposal boxes, proposal features and iteration is 100, 100 and 6, respectively.

Inference details. The inference process is quite simple in Sparse R-CNN. Given an input image, Sparse R-CNN directly predicts 100 bounding boxes associated with their scores. The scores indicate the probability of boxes containing an object. For evaluation, we directly use these 100 boxes without any post-processing.

4.1. Main Result

We provide two versions of Sparse R-CNN for fair comparison with different detectors. The first one adopts 100 learnable proposal boxes *without* random crop data augmentation, and is used to make comparison with mainstream object detectors, *e.g.* Faster R-CNN and RetinaNet [40]. The second one leverages 300 learnable proposal boxes with random crop data augmentations, and is used to make comparison with DETR-series models [3, 45].

As shown in Table 1, Sparse R-CNN outperforms well-established mainstream detectors, such as RetinaNet and

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 2: 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.

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 3: 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.

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 4: 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.

Faster R-CNN, by a large margin. Surprisingly, Sparse R-CNN based on ResNet-50 achieves 42.3 AP, which has already competed with Faster R-CNN on ResNet-101 in accuracy.

We note that DETR and Deformable DETR usually employ stronger feature extracting method, such as stacked encoder layers and deformable convolution. The stronger implementation of Sparse R-CNN is used to give a more fair comparison with these detectors. Sparse R-CNN exhibits higher accuracy even using the simple FPN as feature extracting method. Moreover, Sparse R-CNN gets much better detection performance on small objects compared with DETR(26.9 AP vs. 22.5 AP).

The training convergence speed of Sparse R-CNN is $10\times$ faster over DETR, as shown in Figure 2. Since proposed, DETR has been suffering from slow convergence, which motivates the proposal of Deformable DETR. Compared with Deformable DETR, Sparse R-CNN exhibits better performance in accuracy (44.5 AP vs. 43.8 AP) and shorter running-time (22 FPS vs. 19 FPS), with shorter training schedule (36 epochs vs. 50 epochs).

The inference time of Sparse R-CNN is on par with other detectors. We notice that the model with 100 proposals is running at 23 FPS, while 300 proposals only decreases to 22 FPS, thanks to the light design of the dynamic instance interactive head.

4.2. Module Analysis

In this section, we analyze each component in Sparse R-CNN. All models are based on ResNet50-FPN backbone, 100 proposals, 3x training schedule, unless otherwise noted.

Learnable proposal box. Starting with Faster R-CNN, we naively replace RPN with a sparse set of learnable proposal boxes. The performance drops from 40.2 AP (Table 1 line 3) to 18.5 (Table 2). We find that there is no noticeable improvement even more fully-connected layers are stacked.

Iterative architecture. Iteratively updating the boxes is an intuitive idea to improve its performance. However, we find that a simple cascade architecture does not make a big difference, as shown in Table 3. We analyze the reason is that compared with the refined proposal boxes in [2] which mainly locating around the objects, the candidates in our case are much more coarse, making it hard to be optimized. We observe that the target object for one proposal box is usually consistent in the whole iterative process. Therefore, the object feature in previous stage can be reused to play a strong cue for the next stage. The object feature encodes rich information such as object pose and location. This minor change of feature reuse results in a huge gain of 11.7 AP on basis of original cascade architecture. Finally, the iterative architecture brings 13.7 AP improvement, as shown in second row of Table 2.

Dynamic head. The dynamic head uses object feature of previous stage in a different way with iterative architecture discussed above. Instead of simply concatenating, the object feature of previous stage is first processed by self-attention module, and then used as proposal feature to implement instance interaction of current stage. The self-attention module is applied to the set of object features for reasoning about the relation between objects. Table 4 shows the benefit of self-attention and dynamic instance interaction. Finally, Sparse R-CNN achieves accuracy performance of 42.3 AP.

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 5: Effect of initialization of proposal boxes. Detection performance is relatively robust to initialization of proposal boxes.

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 6: Effect of number of proposals. Increasing number of proposals leads to continuous improvement, while more proposals take more training time.

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 7: Effect of number of stages. Gradually increasing the number of stages, the performance is saturated at 6 stages.

Initialization of proposal boxes. The dense detectors always heavily depend on design of object candidates, whereas, object candidates in Sparse R-CNN are learnable and thus, all efforts related to designing hand-crafted anchors are avoided. However, one may concern that the initialization of proposal boxes plays a key role in Sparse R-CNN. Here we study the effect of different methods for initializing 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 [27].
- "Random" denotes the center, height and width of proposal boxes are randomly initialized with Gaussian distribution.

Method	AP	AP_{50}	AP_{75}
Multi-head Attention [37]	35.7	54.9	37.7
Dynamic head	42.3(**+6.6**)	61.2	45.7

Table 8: Dynamic head vs. Multi-head Attention. As object recognition head, dynamic head outperforms multi-head attention.

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 9: Proposal feature vs. Object query. Object query is learned positional encoding, while proposal feature is irrelevant to position.

From Table 5 we show that the final performance of Sparse R-CNN is relatively robust to the initialization of proposal boxes.

Number of proposals. The number of proposals largely effects both dense and sparse detectors. Original Faster R-CNN uses 300 proposals [30]. Later on it increases to 2000 [40] and obtains better performance. We also study the effect of proposal numbers on Sparse R-CNN in Table 6. Increasing proposal number from 100 to 500 leads to continuous improvement, indicating that our framework is easily to be used in various circumstances. Whereas, 500 proposals take much more training time, so we choose 100 and 300 as the main configurations.

Number of stages in iterative architecture. Iterative architecture is a widely-used technique to improve object detection performance [2, 3, 38], especially for Sparse R-CNN. Table 7 shows the effect of stage numbers in iterative architecture. Without iterative architecture, performance is merely 21.7 AP. Considering the input proposals of first stage is a guess of possible object positions, this result is not surprising. Increasing to 2 stage brings in a gain of 14.5 AP, up to competitive 36.2 AP. Gradually increasing the number of stages, the performance is saturated at 6 stages. We choose 6 stages as the default configuration.

Dynamic head vs. Multi-head Attention. As discussed in Section 3, dynamic head uses proposal feature to filter RoI feature and finally outputs object feature. We find that multi-head attention module [37] provides another possible implementation for the instance interaction. We carry out the comparison experiments in Table 8, and its performance falls behind 6.6 AP. Compared with linear multi-head attention, our dynamic head is much more flexible, whose pa-

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

Figure 5: Visualization of predicted boxes of each stage in iterative architecture, including learned proposal boxes. Boxes of classification score above 0.2 are shown. Learned proposal boxes are drawn in white color. The boxes from the same proposal are drawn in the same color. The learned proposal boxes are randomly distributed on the image and together cover the whole image. The iterative heads gradually refine box position and remove duplicate ones.

rameters are conditioned on its specific proposal feature and more non-linear capacity can be easily introduced.

Proposal feature vs. Object query. Here we make a comparison of object query [3] proposed in DETR and our proposal feature. As discussed in [3], object query is learned positional encoding, guiding the decoder interacting with the summation of image feature map and spatial positional encoding. Using only image feature map will lead to a significant drop. However, our proposal feature can be seen as a feature filter, which is irrelevant to position. The comparisons are shown in Table 9, DETR drops 7.8 AP if the spatial positional encoding is removed. On the contrary, positional encoding gives no gain in Sparse R-CNN.

4.3. The Proposal Boxes Behavior

Figure 5 shows the learned proposal boxes of a converged model. These boxes are randomly distributed on the image to cover the whole image area. This guarantees the recall performance on the condition of sparse candidates. Further, each stage of cascading heads gradually refines box position and remove duplicate ones. This results in high precision performance. Figure 5 also shows that Sparse R-CNN presents robust performance in both rare and crowd scenarios. For object in rare scenario, its duplicate boxes are removed within a few of stages. Crowd scenarios consume more stages to refine but finally each object is detected precisely and uniquely.

5. Conclusion

We present Sparse R-CNN, a purely sparse method for object detection in images. A fixed sparse set of learned object proposals are provided to perform classification and location by dynamic heads. Final predictions are directly output without non-maximum suppression post-procedure. Sparse R-CNN demonstrates its accuracy, run-time and training convergence performance on par with the well-established detector. We hope our work could inspire rethinking the convention of dense prior and exploring next generation of object detector.

References

[1] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – improving object detection with one line of code. In *ICCV*, 2017. 2, 3

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018. 2, 3, 4, 5, 6, 7

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End object detection with transformers. In *ECCV*, 2020. 2, 3, 4, 5, 7, 8

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NeurIPS*, 2016. 3, 4

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 3

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[8] Mark Everingham, Luc. Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 2

[9] Pedro Felzenszwalb, Ross Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *T-PAMI*, 32(9):1627–1645, 2010. 2

[10] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 2, 3

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2, 3, 4

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 5

[13] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2, 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 5

[15] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018. 4

[16] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. DenseBox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. 3

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[18] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NIPS*, pages 667–675, 2016. 2, 4

[19] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi. Foveabox: Beyound anchor-based object detection. *IEEE Transactions on Image Processing*, 29:7389–7398, 2020. 2

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 2

[21] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In *ECCV*, 2018. 3

[22] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *ICCV*, 2017. 1, 2, 3, 4, 5

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2, 5

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 3, 4

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 5

[27] Mahyar Najibi, Mohammad Rastegari, and Larry S Davis. G-cnn: an iterative grid based object detector. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2369–2377, 2016. 3, 7

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 4

[29] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *CVPR*, 2017. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2, 3, 4, 7

[31] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 5

[32] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Robert Fergus, and Yann Lecun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 2

[33] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016. 4

[34] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. *arXiv preprint arXiv:2003.05664*, 2020. 2, 4

[35] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019. 2, 3, 4

[36] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 3

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 4, 7

[38] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001. 2, 7

[39] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *NIPS*, 2020. 2, 3

[40] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2, 5, 7

[41] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. In *Advances in Neural Information Processing Systems*, pages 6740–6749, 2019. 4

[42] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen. Dynamic R-CNN: Towards high quality object detection via dynamic training. In *ECCV*, 2020. 2

[43] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 2, 5

[44] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2, 3

[45] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2, 3, 5