DiffusionDet: Diffusion Model for Object Detection

Shoufa Chen et al.

The Univ. of Hong Kong, Tencent AI Lab

Arxiv

Presented by Minho Park

Contribution

- Formulate object detection as a generative denoising process.
- Dynamic boxes and progressive refinement.
- DiffusionDet achieves favorable performance against previous well-established detectors.

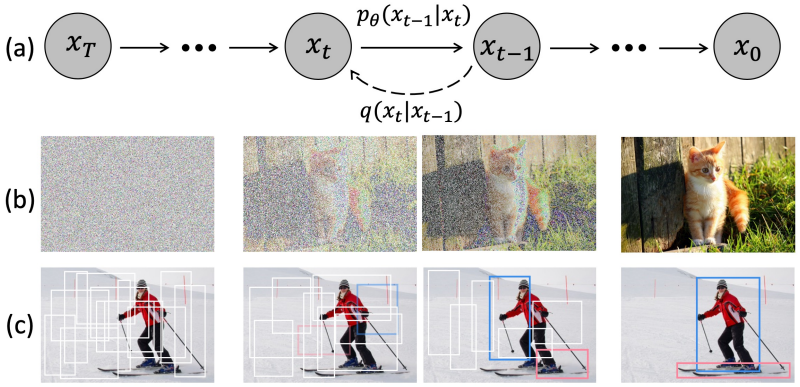


Figure 1. **Diffusion model for object detection**. (a) A diffusion model where q is the diffusion process and p_θ is the reverse process. (b) Diffusion model for image generation task. (c) We propose to formulate object detection as a denoising diffusion process from noisy boxes to object boxes.

Motivation

- Recently, Sparse R-CNN, DETR proposes learnable object queries to eliminate the hand-designed components.
- However, they still have a dependency on a fixed set of learnable queries.

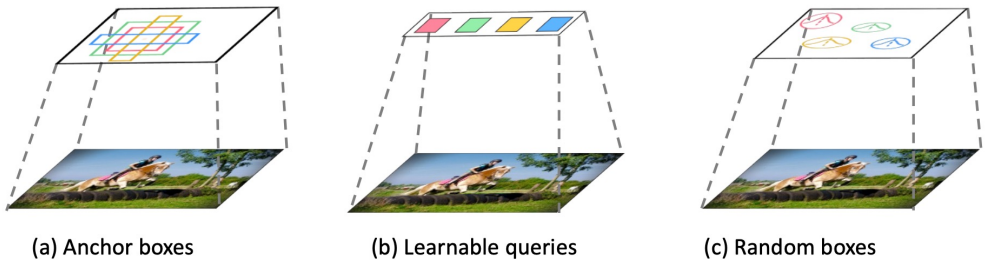


Figure 2. **Comparisons of different object detection paradigms.** (a) Detection from empirical object priors [64, 66]; (b) Detection from learnable queries [10, 81, 102]; (c) Detection from random boxes **(Ours)**.

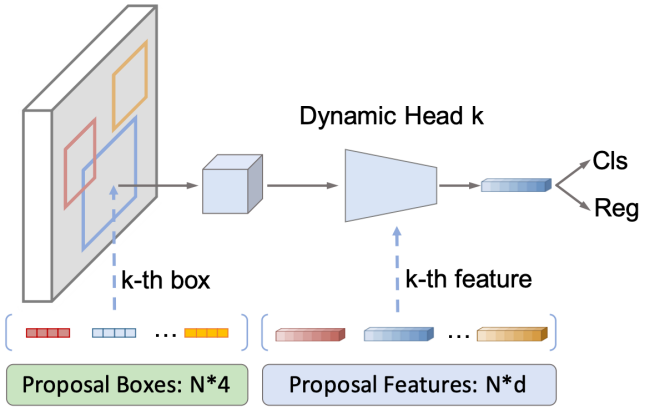


Figure 3 – An overview of Sparse R-CNN pipeline. The input includes an image, a set of proposal boxes and proposal features, where the latter two are learnable parameters. The backbone extracts feature map, each proposal box and proposal feature are fed into its exclusive dynamic head to generate object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

Sun, Peize, et al. "Sparse r-cnn: End-to-end object detection with learnable proposals." *CVPR*. 2021.

DDPM

- Reverse process

$$p_\theta(x_{0:T}) \coloneqq p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) \coloneqq \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$$

- Forward process or diffusion process

$$q(x_{1:T}|x_0) \coloneqq \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_{t-1}|x_t) \coloneqq \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$$

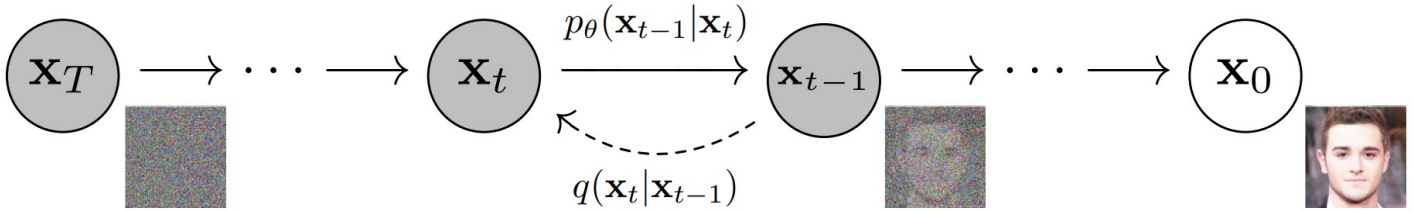


Figure 2: The directed graphical model considered in this work.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *NeurIPS,* 2020.
Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." *ICML*, 2015.

Overview

- Image synthesis decoder has changed to “Detection Decoder”.
  - Sparse R-CNN
- For computational efficiency, apply the detection decoder on the extracted deep feature representation from the image encoder.
  - ResNet, Swin backbone.

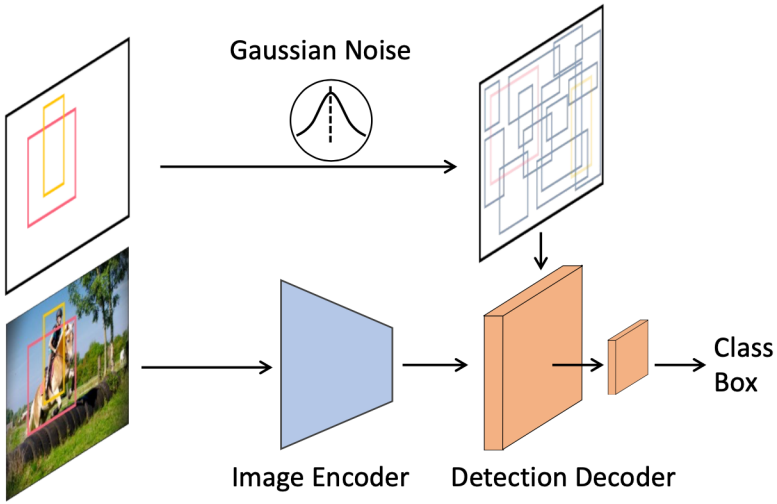


Figure 3. DiffusionDet Framework

DiffusionDet Training Algorithm

- Image encoder
- Ground truth boxes padding (N_{train})
  - Clone GT boxes
  - Concatenating random boxes
- Box corruption
  - Add Gaussian noise to (c_x, c_y, w, h).
- Training losses
  - Set prediction loss (DETR, Sparse R-CNN)

Algorithm 1 DiffusionDet Training

```
def train_loss(images, gt_boxes):
  """
  images: [B, H, W, 3]
  gt_boxes: [B, *, 4]
  # B: batch
  # N: number of proposal boxes
  """

  # Encode image features
  feats = image_encoder(images)

  # Pad gt_boxes to N
  pb = pad_boxes(gt_boxes) # padded boxes: [B, N, 4]

  # Signal scaling
  pb = (pb * 2 - 1) * scale

  # Corrupt gt_boxes
  t = randint(0, T)            # time step
  eps = normal(mean=0, std=1) # noise: [B, N, 4]
  pb_crpt = sqrt(    alpha_cumprod(t)) * pb +
            sqrt(1 - alpha_cumprod(t)) * eps

  # Predict
  pb_pred = detection_decoder(pb_crpt, feats, t)

  # Set prediction loss
  loss = set_prediction_loss(pb_pred, gt_boxes)

  return loss
```

`alpha_cumprod(t)`: cumulative product of α_i, *i.e.*, $\prod_{i=1}^{t} \alpha_i$

DiffusionDet Training Algorithm

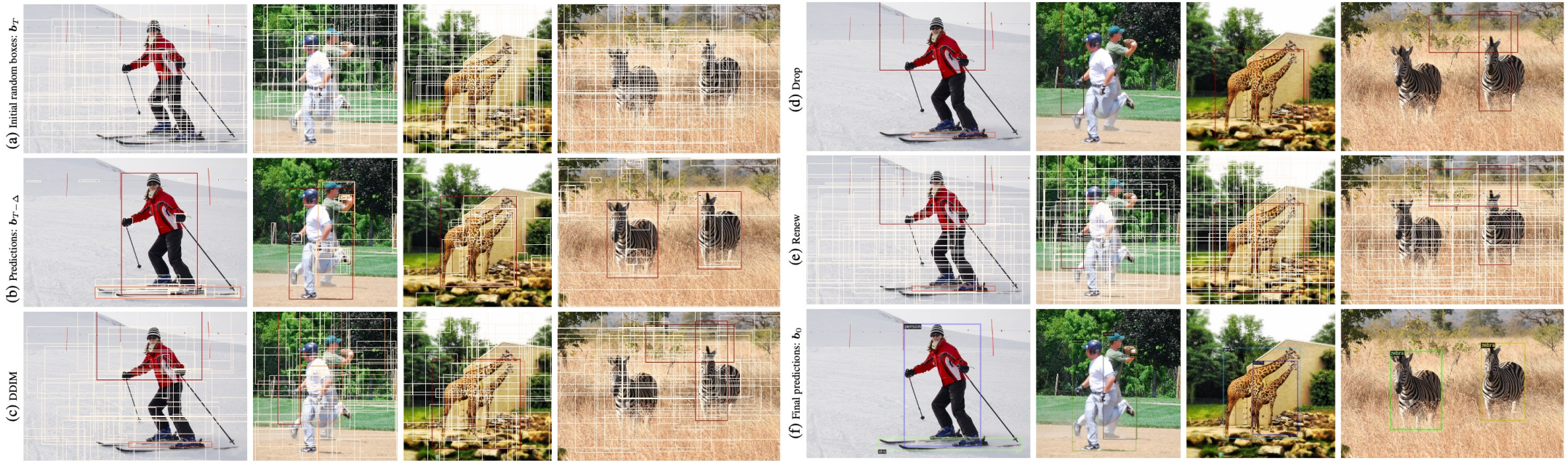
- Image encoder.
- Start with Gaussian noise.
- Denoising with the trained Detection Decoder.
- DDIM sampling.
  - 1000 steps -> 1 / 4 / 8 steps
- Box renewal.
  - Filter out undesired boxes with scores lower than a particular threshold.
  - Concatenate the remaining boxes with new random boxes sampled from Gaussian distribution.

Algorithm 2 DiffusionDet Sampling

```
def infer(images, steps, T):
  """
  images: [B, H, W, 3]
  # steps: number of sample steps
  # T: number of time steps
  """

  # Encode image features
  feats = image_encoder(images)

  # noisy boxes: [B, N, 4]
  pb_t = normal(mean=0, std=1)

  # uniform sample step size
  times = reversed(linespace(-1, T, steps))

  # [(T-1, T-2), (T-2, T-3), ..., (1, 0), (0, -1)]
  time_pairs = list(zip(times[:-1], times[1:])

  for t_now, t_next in zip(time_pairs):
    # Predict pb_0 from pb_t
    pb_pred = detection_decoder(pb_t, feats, t_now)

    # Estimate pb_t at t_next
    pb_t = ddim_step(pb_t, pb_pred, t_now, t_next)

    # Box renewal
    pb_t = box_renewal(pb_t)

  return pb_pred
```

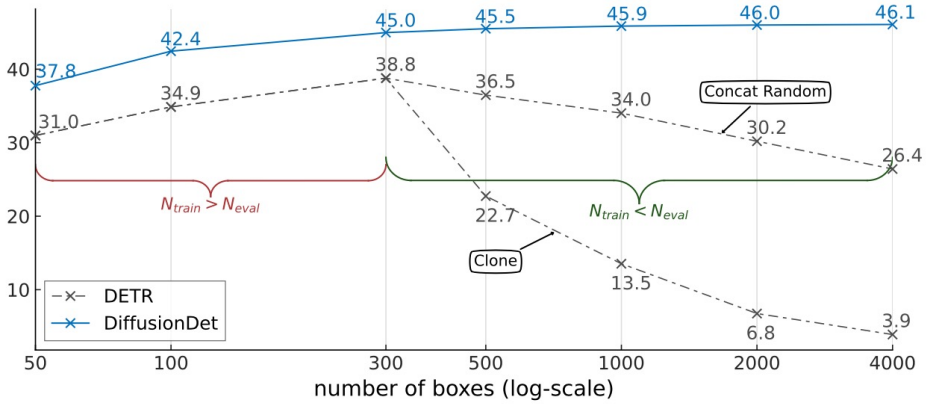
`linespace`: generate evenly spaced values

Visualization of Sampling Step in Inference



Once-for-all Properties of DiffusionDet

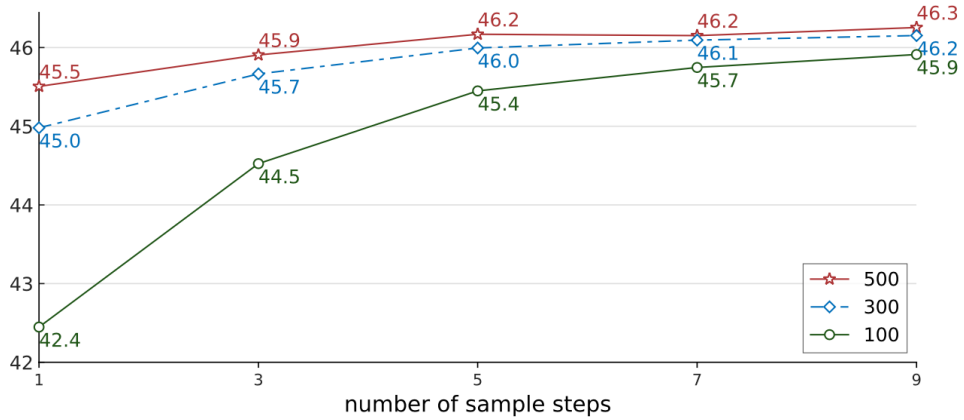
- Dynamic boxes
  - Trained with 300 object queries.
  - Evaluate with {50, 100, 300, 500, 1000, 2000, 4000} queries or boxes.
  - $N_{train} > N_{eval}$: directly choose
  - $N_{train} < N_{eval}$: Concat random



(a) **Dynamic boxes.** Both DETR and DiffusionDet are trained with 300 object queries or proposal boxes. More proposal boxes in inference bring accuracy improvement on DiffusionDet, while degenerate DETR.

Once-for-all Properties of DiffusionDet

- Progressive refinement
  - Increasing the iterative steps from 1 to 9.
  - Trade-off accuracy against inference speed.



(b) **Progressive refinement.** DiffusionDet is trained with 300 proposal boxes and evaluated with different numbers of proposal boxes. For all cases, the accuracy increases with refinement times.

method	[E]	step 1	step 3	step 5
DETR		42.03	42.00 (-0.03)	41.88 (-0.15)
	✓		41.35 (-0.68)	41.36 (-0.67)
Deformable DETR		44.46	43.45 (-1.01)	43.40 (-1.06)
	✓		44.03 (-0.43)	44.04 (-0.42)
Sparse R-CNN		45.02	1.32 (-43.70)	0.32 (-44.70)
	✓		42.90 (-2.12)	42.25 (-2.77)
DiffusionDet		44.98	44.93 (-0.05)	44.93 (-0.05)
	✓		45.67 (+0.69)	46.00 (+1.02)

Table 4. **Progressive refinement.** [E] denotes ensembling predictions from multiple steps. NMS is adopted when using ensemble strategy. We show the performance differences of each method with respect to their own performance on step 1 by (-) or (+).

Quantitative Results

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
			ResNet-50 [34]			
RetinaNet [93]	38.7	58.0	41.5	23.3	42.3	50.3
Faster R-CNN [93]	40.2	61.0	43.8	24.2	43.5	52.0
Cascade R-CNN [93]	44.3	62.2	48.0	26.6	47.7	57.7
DETR [10]	42.0	62.4	44.2	20.5	45.8	61.1
Deformable DETR [102]	43.8	62.6	47.7	26.4	47.1	58.0
Sparse R-CNN [81]	45.0	63.4	48.2	26.9	47.2	59.5
DiffusionDet (1 step)	45.5	65.1	48.7	27.5	48.1	61.2
DiffusionDet (4 step)	46.1	66.0	49.2	28.6	48.5	61.3
DiffusionDet (8 step)	**46.2**	**66.4**	**49.5**	**28.7**	**48.5**	**61.5**
			ResNet-101 [34]			
RetinaNet [93]	40.4	60.2	43.2	24.0	44.3	52.2
Faster R-CNN [93]	42.0	62.5	45.9	25.2	45.6	54.6
Cascade R-CNN [11]	45.5	63.7	49.9	27.6	49.2	59.1
DETR [10]	43.5	63.8	46.4	21.9	48.0	61.8
Sparse R-CNN [81]	46.4	64.6	49.5	28.3	48.3	61.6
DiffusionDet (1 step)	46.6	66.3	50.0	30.0	49.3	62.8
DiffusionDet (4 step)	46.9	66.8	50.4	**30.6**	49.5	62.6
DiffusionDet (8 step)	**47.1**	**67.1**	**50.6**	30.2	**49.8**	**62.7**
			Swin-Base [54]			
Cascade R-CNN [54]	51.9	70.9	56.5	35.4	55.2	67.4
Sparse R-CNN	52.0	72.2	57.0	35.8	55.1	68.2
DiffusionDet (1 step)	52.3	72.7	56.3	34.8	56.0	68.5
DiffusionDet (4 step)	52.7	73.5	56.8	36.1	56.0	**68.9**
DiffusionDet (8 step)	**52.8**	**73.6**	**56.8**	**36.1**	**56.2**	68.8

Table 1. **Comparisons with different object detectors on COCO 2017 `val` set**. The reference after each method indicates the source of its results. The method without reference is our implementation.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP_r	AP_c	AP_f
				ResNet-50 [34]					
Faster R-CNN†	22.5	37.1	23.6	16.5	29.6	34.9	9.9	21.1	29.7
Cascade R-CNN†	26.3	37.8	27.8	18.4	34.4	41.9	12.3	24.9	34.1
Faster R-CNN	25.2	40.6	26.9	18.5	32.2	37.7	16.4	23.4	31.1
Cascade R-CNN	29.4	41.4	30.9	20.6	37.5	44.3	20.0	27.7	35.4
Sparse R-CNN	29.2	41.0	30.7	20.7	36.9	44.2	20.6	27.7	34.6
DiffusionDet (1 step)	30.4	42.8	31.8	20.6	38.6	47.6	23.5	28.1	36.0
DiffusionDet-(4 step)	31.8	45.0	33.2	22.5	39.9	48.3	24.8	29.3	37.6
DiffusionDet-(8 step)	**31.9**	**45.3**	**33.1**	**22.8**	**40.2**	**48.1**	**24.0**	**29.5**	**38.1**
				ResNet-101 [34]					
Faster R-CNN†	24.8	39.8	26.1	17.9	32.2	36.9	13.7	23.1	31.5
Cascade R-CNN†	28.6	40.1	30.1	19.8	37.1	43.8	15.3	27.3	35.9
Faster R-CNN	27.2	42.9	29.1	20.3	35.0	40.4	18.8	25.4	33.0
Cascade R-CNN	31.6	43.8	33.4	22.3	39.7	47.3	23.9	29.8	37.0
Sparse R-CNN	30.1	42.0	31.9	21.3	38.5	45.6	23.5	27.5	35.9
DiffusionDet (1 step)	31.9	44.6	33.1	21.6	40.3	49.0	23.4	30.5	37.1
DiffusionDet-(4 step)	32.9	46.5	34.3	23.3	41.1	49.9	24.2	31.3	38.6
DiffusionDet-(8 step)	**33.5**	**47.3**	**34.7**	**23.6**	**41.9**	**49.8**	**24.8**	**32.0**	**39.0**
				Swin-Base [54]					
DiffusionDet-(1 step)	40.6	54.8	42.7	28.3	50.0	61.6	33.6	39.8	44.6
DiffusionDet-(4 step)	41.9	57.1	44.0	30.3	50.6	62.3	34.9	40.7	46.3
DiffusionDet-(8 step)	**42.1**	**57.8**	**44.3**	**31.0**	**51.3**	**62.5**	**34.3**	**41.0**	**46.7**

Table 2. **Comparisons with different object detectors on LVIS v1.0 `val` set**. We re-implement all detectors using federated loss [100] except for the rows in light gray (with †).

Ablation Studies

scale	AP	AP_{50}	AP_{75}
0.1	38.5	54.2	41.4
1.0	44.3	63.2	47.6
2.0	45.0	64.3	48.1
3.0	44.8	63.9	48.2

(a) **Signal scale**. A large scaling factor can improve detection performance.

case	AP	AP_{50}	AP_{75}
Repeat	43.7	62.6	47.0
Cat Gaussian	45.0	64.3	48.1
Cat Uniform	44.7	63.7	48.3
Cat Full	44.8	63.9	47.9

(b) **GT boxes padding.** Concatenating Gaussian boxes works best.

DDIM	box renewal	step 1	step 4	step 8
		45.0	43.4	43.4
✓		45.0	45.5	45.4
	✓	45.0	45.6	45.6
✓	✓	45.0	45.8	46.2

(c) **Sampling strategy.** Using both DDIM and box renewal works best.

score thresh.	AP	AP_{50}	AP_{75}
0.0	45.4	65.2	48.8
0.3	45.9	65.7	49.2
0.5	46.2	66.1	49.4
0.7	46.0	66.2	49.0

(d) **Box renewal** at evaluation of step 8. The threshold of 0.5 works best.

eval \ train	100	300	500
100	<u>42.5</u>	42.4	41.1
300	43.8	<u>45.0</u>	44.8
500	44.2	45.5	<u>45.6</u>
1000	44.5	45.9	<u>46.1</u>

(e) **Matching between** N_{train} and N_{eval}. The best for each row is underlined.

# boxes	step	AP	AP_{50}	AP_{75}	FPS
[81]	1	45.0	63.4	48.2	31.4
100	1	42.5	60.3	45.9	31.6
300	1	45.0	64.3	48.1	31.3
300	4	45.8	65.7	49.2	12.4

(f) **Accuracy vs. speed.** Using more boxes bring performance gain at the cost of latency.

Table 3. **DiffusionDet ablation experiments** on MS-COCO. We report AP, AP_{50}, and AP_{75}. If not specified, the default setting is: the backbone is ResNet-50 [34] with FPN [49], the signal scale is 2.0, ground-truth boxes padding method is concatenating Gaussian random boxes, DDIM and box renewal are used in sampling step, where the score threshold in box renewal is 0.5, both training and evaluation use 300 boxes. Default settings are marked in gray.