A. Formulation of Diffusion Model

We provide a detailed review of the formulation of diffusion models, following the notion of [16, 38, 67]. Starting from a data distribution $\boldsymbol{z}_0 \sim q(\boldsymbol{z}_0)$, we define a forward Markovian noising process q which produces data samples $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, ..., $\boldsymbol{z}_T$ by gradually adding Gaussian noise at each timestep t. In particular, the added noise is scheduled by the variance $\beta_t \in (0, 1)$:

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) := \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \tag{3}$$

$$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) := \mathcal{N}(\boldsymbol{z}_t; \sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}) \tag{4}$$

As noted by Ho *et al.* [38], we can directly sample data $\boldsymbol{z}_t$ at an arbitrary timestep t without the need of applying q repeatedly:

$$q(\boldsymbol{z}_t|\boldsymbol{z}_0) := \mathcal{N}(\boldsymbol{z}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0, (1-\bar{\alpha}_t)\boldsymbol{I}) \tag{5}$$

$$:= \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0 + \epsilon\sqrt{1-\bar{\alpha}_t}, \epsilon \in \mathcal{N}(0, \boldsymbol{I}) \tag{6}$$

where $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$ and $\alpha_t := 1 - \beta_t$. Then, we could use $\bar{\alpha}_t$ instead of β_t to define the noise schedule.

Based on Bayes' theorem, it is found that the posterior $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0)$ is a Gaussian distribution as well:

$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_{t-1}; \tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0), \tilde{\beta}_t \mathbf{I}) \tag{7}$$

where

$$\tilde{\mu}_t(\boldsymbol{z}_t, \boldsymbol{z}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{z}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{z}_t \tag{8}$$

and

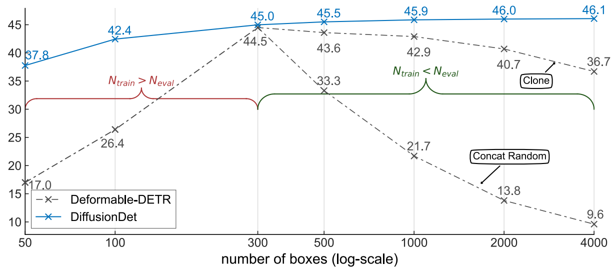
$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{9}$$

are the mean and variance of this Gaussian distribution.

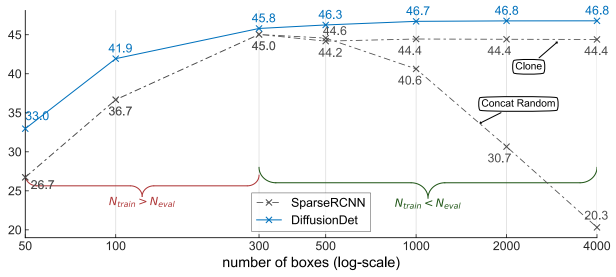
We could get a sample from $q(\boldsymbol{z}_0)$ by first sampling from $q(\boldsymbol{z}_T)$ and running the reversing steps $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ until $\boldsymbol{z}_0$. Besides, the distribution of $q(\boldsymbol{z}_T)$ is nearly an isotropic Gaussian distribution with a sufficiently large T and reasonable schedule of β_t ($\beta_t \to 0$), which making it trivial to sample $\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I})$. Moreover, since calculating $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ exactly should depend on the entire data distribution, we could approximate $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ using a neural network, which is optimized to predict a mean μ_θ and a diagonal covariance matrix Σ_θ:

$$p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) := \mathcal{N}(\boldsymbol{z}_{t-1}; \mu_\theta(\boldsymbol{z}_t, t), \Sigma_\theta(\boldsymbol{z}_t, t)) \tag{10}$$

Instead of directly parameterizing $\mu_\theta(\boldsymbol{z}_t, t)$, Ho *et al.* [38] found learning a network $f_\theta(\boldsymbol{z}_t, t)$ to predict the ϵ or $\boldsymbol{z}_0$ from Equation (6) worked best. We choose to predict $\boldsymbol{z}_0$ in this work.



(a) **Deformable DETR [115] vs. DiffusionDet**



(b) **Sparse R-CNN [91] vs. DiffusionDet**

Figure 5. **Dynamic number of boxes.** All models are trained with 300 candidates (*i.e.*, learnable queries or random boxes). When $N_{train} > N_{eval}$, we directly choose N_{eval} from N_{train} candidates; when $N_{train} < N_{eval}$, we design two strategies, *i.e.*, `clone` and `concat random`.

B. Additional Experiments

We provide some additional experiments in this section for more detailed analysis.

B.1. Dynamic Number of Boxes

We further compare the dynamic box property of DiffusionDet with Deformable DETR [115] and Sparse R-CNN [91] in Figure 5. We directly use the provided models in their official code repositories.[1] [2]

We make Deformable DETR work under $N_{train} \neq N_{eval}$ setting using the same `clone` and `concat random` strategies as DETR as introduced in Section 4.2. For Sparse R-CNN, the strategy `concat random` is slightly different since Sparse R-CNN has both learnable queries and learnable boxes. Therefore, we concatenate $N_{eval} - N_{train}$ boxes to existing N_{train} boxes which are initialized to have the same size as the whole image. Besides, we also concatenate $N_{eval} - N_{train}$ randomly initialized queries to existing N_{train} queries in the same way as DETR and Deformable DETR.

Similar to DETR [10], neither Deformable DETR nor Sparse R-CNN has the dynamic box property. Specifically, the performance of Deformable DETR decreases to 9.6 AP when $N_{eval} = 4000$, far lower than the peak value of 44.5

[1] https://github.com/fundamentalvision/Deformable-DETR
[2] https://github.com/PeizeSun/SparseR-CNN

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)	42.24 (-2.78)
DiffusionDet		45.80	45.74 (-0.06)	45.46 (-0.34)
	✓		46.62 (+0.82)	46.79 (+0.99)

Table 8. **Iterative evaluation.** [E] denotes ensembling predictions from multiple steps. NMS is adopted when using an ensemble strategy. We show the performance differences of each method with respect to their own performance on step 1 by (-) or (+).

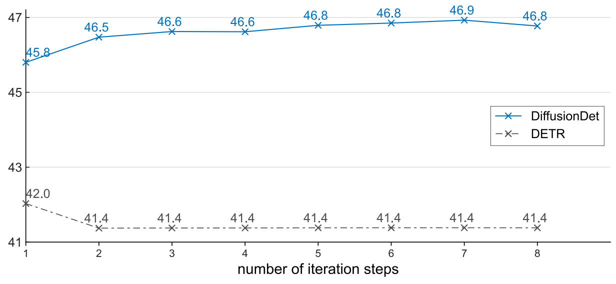
AP. Although Sparse R-CNN has a slower decrease compared with Deformable DETR, its performance is unsatisfactory when the N_{eval} is inconsistent with N_{train}. These findings suggest the distinctive dynamic number of boxes property of DiffusionDet.

B.2. Iterative Evaluation

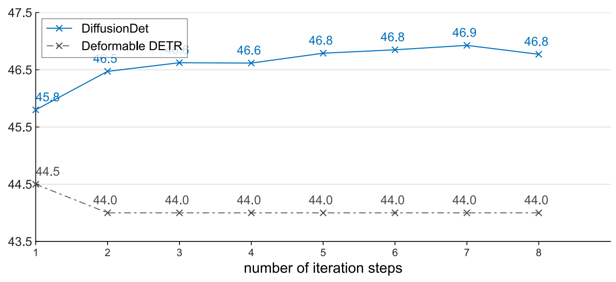
In Table 8 we compare the progressive refinement property of DiffusionDet with some previous approaches like DETR [10], Deformable DETR [115] and Sparse R-CNN [91]. All of these four models have 6 cascading stages as the detection decoder. The refinement refers to the output of the previous 6 stages and is taken as the input of the next 6 stages. All model checkpoints are from Model Zoo in their official code repositories.

We experiment with two settings: (1) only use the output of the last reference step as the final prediction; (2) use the ensemble of the output of multiple steps as the final prediction. For the latter setting, we adopt NMS to remove duplicate predictions among different steps.

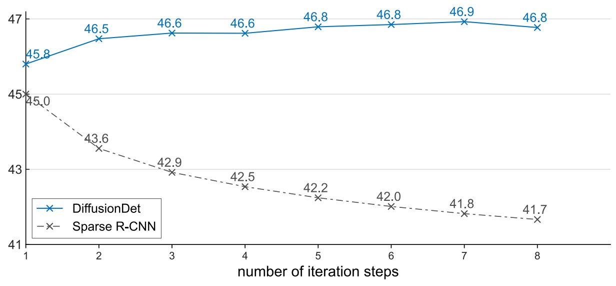
We find that all models have performance drop when evaluated with more than one step without ensemble strategy. However, the performance drop of DiffusionDet and DETR is negligible. Adopting an ensemble would mitigate the performance degradation except for DETR. Nevertheless, previous query-based based methods still have performance drops with more steps. In contrast, DiffusionDet turns performance down to up. Specifically, DiffusionDet has performance gains with more refinement steps. For example, DiffusionDet with five steps has 0.99 AP higher than with a single step. Therefore, we use the ensemble strategy as default. To better compare DiffusionDet with DETR, Deformable DETR, and Sparse R-CNN, we also draw the comparison curves of Table 8 in Figure 6.



(a) **DETR [10] vs. DiffusionDet**



(b) **Deformable DETR [115] vs. DiffusionDet**



(c) **Sparse R-CNN [91] vs. DiffusionDet**

Figure 6. **Iterative evaluation.** All models are the provided models in their official code repositories.

C. Experimental Settings

In this section, we give the detailed experimental settings in Section 4.2 and Section 4.3.

C.1. DETR with 300 Queries

Since the official GitHub repository[3] of only provides DETR [10] with 100 object queries, we reproduce it with 300 object queries using the official code for fair comparison in Section 4.2. Specifically, we train this model with `Detectron2` wrapper based on configuration `https://github.com/facebookresearch/detr/blob/main/d2/configs/detr_256_6_6_torchvision.yaml`, as summarized in Table 9. We note that the configuration only trains the model for about 300 epochs. We only change the `NUM_OBJECT_QUERIES` from 100 to 300 and leave everything else the same as the original one.

[3] https://github.com/facebookresearch/detr

config	value
# object queries	300
optimizer	AdamW [61]
base learning rate	1e-4
weight decay	1e-4
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	64
learning rate schedule	step lr
lr decay steps	(369600,)
warmup iter	10
warmup factor	1.0
training iters	554400
clip gradient type	full model
clip gradient value	0.01
clip gradient norm	2.0
data augmentation	RandomFlip, RandomResizedCrop, RandomCrop

Table 9. **DETR reproduction setting.**

config	value
optimizer	AdamW [61]
base learning rate	2.5e-5
weight decay	1e-4
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	16
learning rate schedule	step lr
lr decay steps	(350000, 420000)
warmup iter	1000
warmup factor	0.01
training iters	450000
clip gradient type	full model
clip gradient value	1.0
clip gradient norm	2.0
data augmentation	RandomFlip, RandomResizedCrop, RandomCrop

Table 10. **COCO setting.**

config	value
optimizer	AdamW [61]
base learning rate	2.5e-5
weight decay	1e-4
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	16
learning rate schedule	step lr
lr decay steps	(210000, 250000)
warmup iter	1000
warmup factor	0.01
training iters	270000
clip gradient type	full model
clip gradient value	1.0
clip gradient norm	2.0
data augmentation	RandomFlip, RandomResizedCrop, RandomCrop
data sampler	RepeatFactorTrainingSampler
repeat thres.	0.001

Table 11. **LVIS setting.**

C.2. Benchmark on COCO and LVIS

In Section 4.3, we benchmark DiffusionDet on COCO dataset and LVIS dataset. The training configuration is in Table 10 and Table 11, respectively.

D. Training Loss

We adopt set prediction loss [10, 91, 115] on the set of N_{train} predictions for DiffusionDet. Set prediction loss requires pairwise matching cost between predictions and ground truth objects, taking into account both the category and box predictions. The matching cost is formulated as :

$$\mathcal{C} = \lambda_{cls} \cdot \mathcal{C}_{cls} + \lambda_{L1} \cdot \mathcal{C}_{L1} + \lambda_{giou} \cdot \mathcal{C}_{giou}, \tag{11}$$

where $\mathcal{C}_{cls}$ the focal loss [56] between prediction and ground truth class labels. Besides, our boxes loss contains $\mathcal{C}_{L1}$ and $\mathcal{C}_{giou}$, which are most commonly-used ℓ_1 loss and generalized IoU (GIoU) loss [75]. λ_{cls}, λ_{L1} and $\lambda_{giou} \in \mathbb{R}$ are weights of each component to balance to overall multiple losses. Following [10, 91, 115], we adopt $\lambda_{cls} = 2.0$, $\lambda_{L1} = 5.0$ and $\lambda_{giou} = 2.0$.

We assign multiple predictions to each ground truth with the optimal transport approach [25, 26]. Specifically, for each ground truth, we select the top-k predictions with the least matching cost as its positive samples, others as negatives. Then, DiffusionDet is optimized with a multi-task loss function:

$$\mathcal{L} = \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{L1} \cdot \mathcal{L}_{L1} + \lambda_{giou} \cdot \mathcal{L}_{giou}, \tag{12}$$

The component of training loss is the same as the matching cost, except that the loss is only performed on the matched pairs.