DiffusionDet: Diffusion Model for Object Detection

Shoufa Chen[1], Peize Sun[1], Yibing Song[2], Ping Luo[1]
[1]The University of Hong Kong [2]Tencent AI Lab
{sfchen, pzsun, pluo}@cs.hku.hk yibingsong.cv@gmail.com

Abstract

We propose DiffusionDet, a new framework that formulates object detection as a denoising diffusion process from noisy boxes to object boxes. During training stage, object boxes diffuse from ground-truth boxes to random distribution, and the model learns to reverse this noising process. In inference, the model refines a set of randomly generated boxes to the output results in a progressive way. The extensive evaluations on the standard benchmarks, including MS-COCO and LVIS, show that DiffusionDet achieves favorable performance compared to previous well-established detectors. Our work brings two important findings in object detection. First, random boxes, although drastically different from pre-defined anchors or learned queries, are also effective object candidates. Second, object detection, one of the representative perception tasks, can be solved by a generative way. Our code is available at https://github.com/ShoufaChen/DiffusionDet.

1. Introduction

Object detection aims to predict a set of bounding boxes and associated category labels for targeted objects in one image. As a fundamental visual recognition task, it has become the cornerstone of many related recognition scenarios, such as instance segmentation [33, 48], pose estimation [9, 20], action recognition [29, 73], object tracking [41, 58], and visual relationship detection [40, 56].

Modern object detection approaches have been evolving with the development of object candidates, *i.e.*, from empirical object priors [24, 53, 64, 66] to learnable object queries [10, 81, 102]). Specifically, the majority of detectors solve detection tasks by defining surrogate regression and classification on empirically designed object candidates, such as sliding windows [25, 71], region proposals [24, 66], anchor boxes [50, 64] and reference points [17, 97, 101]. Recently, DETR [10] proposes learnable object queries to eliminate the hand-designed components and set up an end-to-end detection pipeline, attracting great attention on query-based detection paradigm [21, 46, 81, 102].

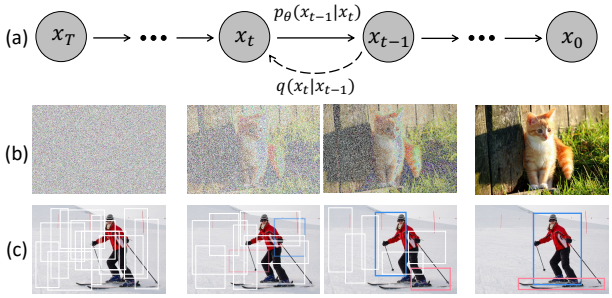


Figure 1. **Diffusion model for object detection**. (a) A diffusion model where q is the diffusion process and p_θ is the reverse process. (b) Diffusion model for image generation task. (c) We propose to formulate object detection as a denoising diffusion process from noisy boxes to object boxes.

While these works achieve a simple and effective design, they still have a dependency on a fixed set of learnable queries. A natural question is: *is there a simpler approach that does not even need the surrogate of learnable queries*?

We answer this question by designing a novel framework that directly detects objects from a set of random boxes. Starting from purely random boxes, which do not contain learnable parameters that need to be optimized in training, we expect to gradually refine the positions and sizes of these boxes until they perfectly cover the targeted objects. This *noise-to-box* approach does not require heuristic object priors nor learnable queries, further simplifying the object candidates and pushing the development of the detection pipeline forward.

Our motivation is illustrated in Figure 1. We think of the philosophy of noise-to-box paradigm is analogous to *noise-to-image* process in the denoising diffusion models [15, 35, 79], which are a class of likelihood-based models to generate the image by gradually removing noise from an image via the learned denoising model. Diffusion models have achieved great success in many generation tasks [3, 4, 37, 63, 85] and start to be explored in perception tasks like image segmentation [1, 5, 6, 12, 28, 42, 89]. However, to the best of our knowledge, there is no prior arts that successfully adopt it to object detection.

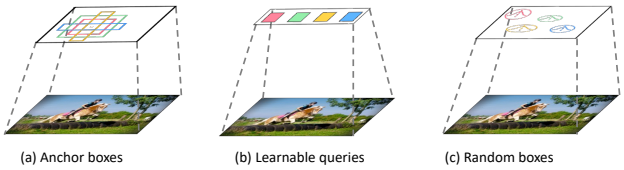


Figure 2. **Comparisons of different object detection paradigms.** (a) Detection from empirical object priors [64, 66]; (b) Detection from learnable queries [10, 81, 102]; (c) Detection from random boxes **(Ours)**.

In this work, we propose DiffusionDet, which tackles the object detection task with a diffusion model by casting detection as a generative task over the space of the positions (center coordinates) and sizes (widths and heights) of bounding boxes in the image. At training stage, Gaussian noise controlled by a variance schedule [35] is added to ground truth boxes to obtain *noisy* boxes. Then these noisy boxes are used to crop [33, 66] features of Region of Interest (RoI) from the output feature map of the backbone encoder, *e.g.*, ResNet [34], Swin Transformer [54]. Finally, these RoI features are sent to the detection decoder, which is trained to predict the ground-truth boxes without noise. With this training objective, DiffusionDet is able to predict the ground truth boxes from random boxes. At inference stage, DiffusionDet generates bounding boxes by reversing the learned diffusion process, which adjusts a noisy prior distribution to the learned distribution over bounding boxes.

The noise-to-box pipeline of DiffusionDet has the appealing advantage of Once-for-All: we can train the network once and use the same network parameters under diverse settings in inference. (1) *Dynamic boxes*: Leveraging random boxes as object candidates, DiffusionDet decouples the training and evaluation. DiffusionDet can be trained with N_{train} random boxes while being evaluated with N_{eval} random boxes, where the N_{eval} is arbitrary and does not need to be equal to N_{train}. (2) *Progressive refinement*: The diffusion model benefits DiffusionDet by iterative refinement. We can adjust the number of denoising sampling steps to improve the detection accuracy or accelerate the inference speed. This flexibility enables DiffusionDet to suit different detection scenarios where accuracy and speed are required differently.

We evaluate DiffusionDet on MS-COCO [51] dataset. With ResNet-50 [34] backbone, DiffusionDet achieves 45.5 AP using a single sampling step, which significantly outperforms Faster R-CNN [66] (40.2 AP), DETR [10] (42.0 AP) and on par with Sparse R-CNN [81] (45.0 AP). Besides, we can further improve DiffusionDet up to 46.2 AP by increasing the number of sampling steps. On the contrary, existing approaches [10, 81, 102] do not have this refinement property and would have a remarkable performance drop when evaluated in an iterative way. Moreover, we further conduct experiments on challenging LVIS [31] dataset, and DiffusionDet also performs well on this long-tailed dataset, achieving 42.1 AP with Swin-Base [54] backbone.

Our **contributions** are summarized as follows:

- We formulate object detection as a generative denoising process, which is the first study to apply the diffusion model to object detection to the best of our knowledge.
- Our noise-to-box detection paradigm has several appealing properties, such as decoupling training and evaluation stage for dynamic boxes and progressive refinement.
- We conduct extensive experiments on MS-COCO and LVIS benchmarks. DiffusionDet achieves favorable performance against previous well-established detectors.

2. Related Work

Object detection. Most modern object detection approaches perform box regression and category classification on empirical object priors, such as proposals [24, 66], anchors [50, 64, 65], points [84, 87, 101]. Recently, Carion *et al*. proposed DETR [10] to detect objects using a fixed set of learnable queries. Since then, the query-based detection paradigm has attracted great attention and inspired a series of following works [46, 52, 57, 80, 81, 102]. In this work, we push forward the development of the object detection pipeline further with DiffusionDet, as compared in Figure 2.

Diffusion model. As a class of deep generative models, diffusion models [35, 77, 79] start from the sample in random distribution and recover the data sample via a gradual denoising process. Diffusion models have recently demonstrated remarkable results in fields including computer vision [4, 19, 30, 32, 36, 60, 63, 68, 69, 74, 96, 99], nature language processing [3, 27, 47], audio processing [38, 43, 45, 62, 82, 92, 95], interdisciplinary applications [2, 37, 39, 70, 85, 91, 94], etc. More applications of diffusion models can be found in recent surveys [8, 96].

Diffusion model for perception tasks. While Diffusion models have achieved great success in image generation [15, 35, 79], their potential for discriminative tasks has yet to be fully explored. Some pioneer works tried to adopt the diffusion model for image segmentation tasks [1, 5, 6, 12, 28, 42, 89], for example, Chen *et al*. [12] adopted Bit Diffusion model [13] for panoptic segmentation [44] of images and videos. However, despite significant interest in this idea, there are no previous solutions that successfully adapt generative diffusion models for object detection, the progress of which remarkably lags behind that of segmentation. We argue that this may be because segmentation tasks are processed in an image-to-image style, which is more conceptually similar to the image generation tasks, while object detection is a set prediction problem [10] which requires assigning object candidates [10, 49, 66] to ground truth objects. To the best of our knowledge, this is the first work that adopts a diffusion model for object detection.

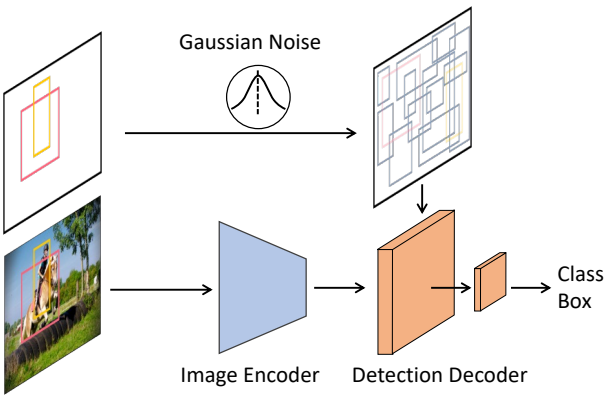


Figure 3. **DiffusionDet framework.** The image encoder extracts feature representation from an input image. The detection decoder takes noisy boxes as input and predicts category classification and box coordinates. During training, the noisy boxes are constructed by adding Gaussian noise to ground-truth boxes. In inference, the noisy boxes are randomly sampled from the Gaussian distribution.

3. Approach

3.1. Preliminaries

Object detection. The learning objective of object detection is input-target pairs $(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{c})$, where $\boldsymbol{x}$ is the input image, $\boldsymbol{b}$ and $\boldsymbol{c}$ are a set of bounding boxes and category labels for objects in the image $\boldsymbol{x}$, respectively. More specifically, we formulate the i-th box in the set as $\boldsymbol{b}^i = (c_x^i, c_y^i, w^i, h^i)$, where (c_x^i, c_y^i) is the center coordinates of the bounding box, (w^i, h^i) are width and height of that bounding box, respectively.

Diffusion model. Diffusion models [35, 75–77] are a classes of likelihood-based models inspired by nonequilibrium thermodynamics [77, 78]. These models define a Markovian chain of diffusion forward process by gradually adding noise to sample data. The forward noise process is defined as

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

which transforms data sample $\boldsymbol{z}_0$ to a latent noisy sample $\boldsymbol{z}_t$ for $t \in \{0, 1, ..., T\}$ by adding noise to $\boldsymbol{z}_0$. $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s = \prod_{s=0}^{t}(1-\beta_s)$ and β_s represents the noise variance schedule [35]. During training, a neural network $f_\theta(\boldsymbol{z}_t, t)$ is trained to predict $\boldsymbol{z}_0$ from $\boldsymbol{z}_t$ by minimizing the training objective with ℓ_2 loss [35]: 神经网络来提供去噪函数.

$$\mathcal{L}_{\text{train}} = \frac{1}{2}||f_\theta(\boldsymbol{z}_t, t) - \boldsymbol{z}_0||^2. \tag{2}$$

At inference stage, data sample $\boldsymbol{z}_0$ is reconstructed from noise $\boldsymbol{z}_T$ with the model f_θ and an updating rule [35, 76] in an iterative way, *i.e.*, $\boldsymbol{z}_T \rightarrow \boldsymbol{z}_{T-\Delta} \rightarrow ... \rightarrow \boldsymbol{z}_0$. More detailed formulation of diffusion models can be found in Appendix A.

In this work, we aim to solve the object detection task via the diffusion model. In our setting, data samples are a set of bounding boxes $\boldsymbol{z}_0 = \boldsymbol{b}$, where $\boldsymbol{b} \in \mathbb{R}^{N\times 4}$ is a set of N boxes. A neural network $f_\theta(\boldsymbol{z}_t, t, \boldsymbol{x})$ is trained to predict $\boldsymbol{z}_0$ from noisy boxes $\boldsymbol{z}_t$, conditioned on the corresponding image $\boldsymbol{x}$. The corresponding category label $\boldsymbol{c}$ is produced accordingly.

3.2. Architecture

Since the diffusion model generates data samples iteratively, it needs to run model f_θ multiple times at the inference stage. However, it would be computationally intractable to directly apply f_θ on the raw image at every iterative step. Therefore, we propose to separate the whole model into two parts, *image encoder* and *detection decoder*, where the former runs only once to extract a deep feature representation from the raw input image $\boldsymbol{x}$, and the latter takes this deep feature as condition, instead of the raw image, to progressively refine the box predictions from noisy boxes $\boldsymbol{z}_t$.

Image encoder. Image encoder takes as input the raw image and extracts its high-level features for the following detection decoder. We implement DiffusionDet with both Convolutional Neural Networks such as ResNet [34] and Transformer-based models like Swin [54]. Feature Pyramid Network [49] is used to generate multi-scale feature maps for both ResNet and Swin backbones following [49, 54, 81].

Detection decoder. Borrowed from Sparse R-CNN [81], the detection decoder takes as input a set of proposal boxes to crop RoI-feature [33, 66] from feature map generated by image encoder, and sends these RoI-features to detection head to obtain box regression and classification results. Following [10, 81, 102], our detection decoder is composed of 6 cascading stages. The differences between our decoder and the one in Sparse R-CNN are that (1) DiffusionDet begins from random boxes while Sparse R-CNN uses a fixed set of learned boxes in inference; (2) Sparse R-CNN takes as input pairs of the proposal boxes and its corresponding proposal feature, while DiffusionDet needs the proposal boxes only; (3) DiffusionDet re-uses the detector head in iterative sampling steps and the parameters are shared across different steps, each of which is specified to the diffusion process by timestep embedding [35, 86], while Sparse R-CNN uses the detection decoder only once in the forward pass.

3.3. Training

During training, we first construct the diffusion process from ground-truth boxes to noisy boxes and then train the model to reverse this process. Algorithm 1 provides the pseudo-code of DiffusionDet training procedure.

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                                  扩充时间轴
  pb = pad_boxes(gt_boxes) # padded boxes: [B, N, 4]

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

  # Corrupt gt_boxes                                   加上噪音的box
  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                                            把加了噪音的box恢复
  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.*, $\prod_{i=1}^{t} \alpha_i$

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]                             box最后变成白噪声了.所以这里面生成pb_t时间的值.
  pb_t = normal(mean=0, std=1)

  # uniform sample step size
  times = reversed(linespace(-1, T, steps))          steps用来调节迭代次数,不需要跑完全部的T.

  # [(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用于创建一个等差数列的一维数组，常用到的一般是三个参数:

起始值,
终止值（默认包含自身）,
数列个数.

`linespace`: generate evenly spaced values

Ground truth boxes padding. For modern object detection benchmarks [18, 31, 51, 72], the number of instances of interest typically varies across images. Therefore, we first *pad* some extra boxes to original ground truth boxes such that all boxes are summed up to a fixed number N_{train}. We explore several padding strategies, for example, repeating existing ground truth boxes, concatenating random boxes or image-size boxes. Comparisons of these strategies are in Section 4.4, and concatenating random boxes works best.

Box corruption. We add Gaussian noises to the padded ground truth boxes. The noise scale is controlled by α_t (in Eq. (1)), which adopts the monotonically decreasing cosine schedule for α_t in different time step t, as proposed in [59]. Notably, the ground truth box coordinates need to be scaled as well since the signal-to-noise ratio has a significant effect on the performance of diffusion model [12]. We observe that object detection favors a relatively higher signal scaling value than image generation stask [13, 15, 35]. More discussions are in Section 4.4.

Training losses. The detection detector takes as input N_{train} corrupted boxes and predicts N_{train} predictions of category classification and box coordinates. We apply set prediction loss [10, 81, 102] on the set of N_{train} predictions. We assign multiple predictions to each ground truth by selecting the top k predictions with the least cost by an optimal transport assignment method [16, 22, 23, 90].

3.4. Inference

The inference procedure of DiffusionDet is a denoising sampling process from noise to object boxes. Starting from boxes sampled in Gaussian distribution, the model progressively refines its predictions, as shown in Algorithm 2.

Sampling step. In each sampling step, the random boxes or the estimated boxes from the last sampling step are sent into the detection decoder to predict the category classification and box coordinates. After obtaining the boxes of the current step, DDIM [76] is adopted to estimate the boxes for the next step. We note that sending the predicted boxes without DDIM to the next step is also an optional progressive refinement strategy. However, it brings significant deterioration, as discussed in Section 4.4.

Box renewal. After each sampling step, the predicted boxes can be coarsely categorized into two types, *desired* and *undesired* predictions. The desired predictions contain boxes that are properly located at corresponding objects, while the undesired ones are distributed arbitrarily. Directly sending these undesired boxes to the next sampling iteration would not bring a benefit since their distribution is not constructed by box corruption in training. To make inference better align with training, we propose the strategy of *box renewal* to revive these undesired boxes by replacing them with random boxes. Specifically, we first filter out undesired boxes with scores lower than a particular threshold. Then, we concatenate the remaining boxes with new random boxes sampled from a Gaussian distribution.

Once-for-all. Thanks to the random boxes design, we can evaluate DiffusionDet with an arbitrary number of random boxes and the number of sampling steps, which do not need to be equal to the training stage. As a comparison, previous approaches [10, 81, 102] rely on the same number of processed boxes during training and evaluation, and their detection decoders are used only once in the forward pass.

4. Experiments

We first show the once-for-all property of DiffusionDet. Then we compare DiffusionDet with previous well-established detectors on MS-COCO [51] and LVIS [31] dataset. Finally, we provide ablation studies on the components of DiffusionDet.

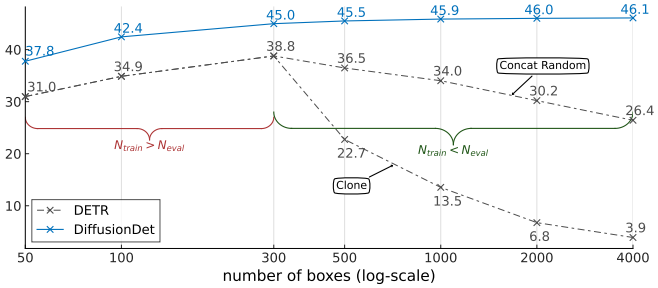
MS-COCO [51] dataset contains about 118K training images in the `train2017` set and 5K validation images in the `val2017` set. There are 80 object categories in total. We report box average precision over multiple IoU thresholds (AP), threshold 0.5 (AP_{50}) and 0.75 (AP_{75}).

LVIS v1.0 [31] dataset is a large-vocabulary object detection and instance segmentation dataset which has 100K training images and 20K validation images. LVIS shares the same source images as MS-COCO, while its annotations capture the long-tailed distribution in 1203 categories. We adopt MS-COCO style box metric AP, AP_{50} and AP_{75} in LVIS evaluation.

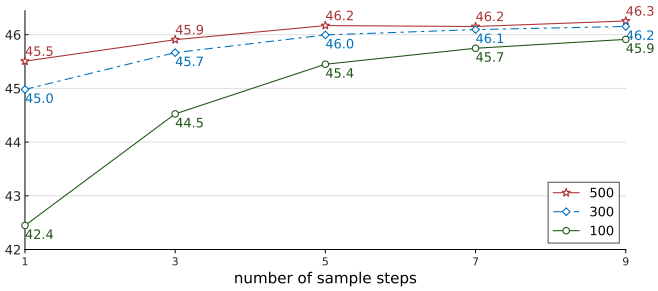
4.1. Implementation Details.

Training schedules. The ResNet and Swin backbone are initialized with pre-trained weights on ImageNet-1K and ImageNet-21K [14], respectively. The newly added detection decoder is initialized with Xavier init [26]. We train DiffusionDet using AdamW [55] optimizer with the initial learning rate as 2.5×10^{-5} and the weight decay as 10^{-4}. All models are trained with a mini-batch size 16 on 8 GPUs. For MS-COCO, the default training schedule is 450K iterations, with the learning rate divided by 10 at 350K and 420K iterations. For LVIS, the training schedule is 210K, 250K, 270K. Data augmentation strategies contain random horizontal flip, scale jitter of resizing the input images such that the shortest side is at least 480 and at most 800 pixels while the longest is at most 1333 [93], and random crop augmentations. We do not use the EMA and some strong data augmentation like MixUp [98] or Mosaic [23].

Inference details. At the inference stage, the detection decoder iteratively refines the predictions from Gaussian random boxes. We select top-100 and top-300 scoring predictions for MS-COCO and LVIS, respectively. The predictions at each sampling step are ensembled together by NMS to get the final predictions.



(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.



(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.

Figure 4. **Once-for-all properties of DiffusionDet**. All experiments are trained on COCO 2017 `train` set and evaluated on COCO 2017 `val` set. DiffusionDet uses the same network parameters for all settings in Figure 4a and 4b. Our proposed DiffusionDet is able to benefit from more proposal boxes and iterative refinements using the same network parameters.

4.2. Main Properties

The main properties of DiffusionDet lie on *once training for all inference cases*. Once the model is trained, it can be used with changing the number of boxes and number of sample steps in inference, as shown in Figure 4. DiffusionDet can achieve better accuracy by using more boxes or/and more refining steps at the cost of higher latency. Therefore, we can deploy a single DiffusionDet to multiple scenarios and obtain a desired speed-accuracy trade-off without retraining the network.

Dynamic boxes. We compare DiffusionDet with DETR [10] to show the advantage of dynamic boxes. Comparisons with other detectors are in Appendix B. We reproduce DETR [10] with 300 object queries using the official code and default settings for 300 epochs training. We train DiffusionDet with 300 random boxes such that the number of candidates is consistent with DETR for a fair comparison. The evaluation is on {50, 100, 300, 500, 1000, 2000, 4000} queries or boxes.

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.

Since the learnable queries are fixed after training in the original setting of DETR, we propose a simple workaround to enable DETR work with different number of queries: when $N_{eval} < N_{train}$, we directly choose N_{eval} queries from N_{train} queries; when $N_{eval} > N_{train}$, we concatenate extra $N_{eval} - N_{train}$ randomly initialized queries (a.k.a. `concat random`). We present results in Figure 4a. The performance of DiffusionDet increases steadily with the number of boxes used for evaluation. On the contrary, DETR has a clear performance drop when the N_{eval} is different with N_{train}, *i.e.*, 300. Besides, this performance drop becomes larger when the difference between N_{eval} and N_{train} increases. For example, when the number of boxes increases to 4000, DETR only has 26.4 AP with `concat random` strategy, which is 12.4 lower than the peak value (*i.e.*, 38.8 AP with 300 queries). As a comparison, DiffusionDet can achieve 1.1 AP gain with 4000 boxes.

We also implement another method for DETR when $N_{eval} > N_{train}$, cloning existing N_{train} queries up to N_{eval} (a.k.a. `clone`). We observe that `concat random` strategy consistently performs better than the `clone`. It is reasonable because the cloned queries will produce similar detection results as the original queries. In contrast, random queries introduce more diversity to the detection results.

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 †).

Progressive refinement. The performance of DiffusionDet can be improved not only by increasing the number of random boxes but by iterating more steps. We evaluate DiffusionDet with 100, 300, and 500 random boxes by increasing their iterative steps from 1 to 9. The results are presented in Figure 4b. We see that DiffusionDet with these three settings all have steady performance gains with more refining steps. Besides, DiffusionDet with fewer random boxes tends to have a larger gain with refinement. For example, the AP of DiffusionDet instance with 100 random boxes increases from 42.4 (1 step) to 45.9 (9 steps), an absolute 3.5 AP improvement. This accuracy performance is comparable with 45.0 (1 step) of 300 random boxes and 45.5 (1 step) of 500, showing that high accuracy in DiffusionDet could be achieved by either increasing the number of proposal boxes or the iterative steps.

In comparison, we find that previous approaches [10, 81, 102] do not have this refinement property. They can use the detection decoder only once. Using two or more iteration steps will drop the performance. More detailed comparison can be found in Appendix C.

4.3. Benchmarking on Detection Datasets

We compare DiffusionDet with previous detectors [7, 10, 50, 66, 81, 102] on MS-COCO and LVIS dataset. We adopt 500 boxes for both training and inference in this subsection. More detailed experimental settings are in Appendix D.

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	42.5	42.4	41.1
300	43.8	45.0	44.8
500	44.2	45.5	45.6
1000	44.5	45.9	46.1

(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.

MS-COCO. In Table 1 we compare the object detection performance of DiffusionDet with previous detectors on MS-COCO. DiffusionDet without refinement (*i.e.*, step 1) achieves 45.5 AP with ResNet-50 backbone, outperforming well-established methods such as Faster R-CNN, RetinaNet, DETR and Sparse R-CNN by a non-trival margin. Besides, DiffusionDet can make its superiority more remarkable when using more iterative refinements. For example, when using ResNet-50 as the backbone, DiffusionDet outperforms Sparse R-CNN by 0.5 AP (45.5 vs. 45.0) with a single step while by 1.2 AP (46.2 vs. 45.0) with 8 steps.

DiffusionDet shows steady improvement when the backbone size scales up. DiffusionDet with ResNet-101 achieves 46.6 AP (1 step) and 47.1 AP (8 steps). When using ImageNet-21k pre-trained Swin-Base [54] as the backbone, DiffusionDet obtains 52.3 AP for a single step and 52.8 AP with 8 steps, outperforming strong baselines such as Cascade R-CNN and Sparse R-CNN.

LVIS v1.0. We compare the results on a more challenging LVIS dataset in Table 2. We reproduce Faster R-CNN and Cascade R-CNN based on `detectron2` [93] while Sparse R-CNN on its original code. We first reproduce Faster R-CNN and Cascade R-CNN using the default settings of `detectron2`, achieving 22.5/24.8 and 26.3/28.8 AP (with † in Table 2) with ResNet-50/101 backbone, respectively. Further, we boost their performance using the federated loss in [100]. Since images in LVIS are annotated in a federated way [31], the negative categories are sparsely annotated, which deteriorates the training gradients, especially for rare classes [83]. Federated loss is proposed to mitigate this issue by sampling a subset S of classes for each training image that includes all positive annotations and a random subset of negative ones. Following [100], we choose $|S| = 50$ in all experiments. Faster R-CNN and Cascade R-CNN earn about 3 AP gains with federated loss. All following comparisons are based on this loss.

We see that DiffusionDet attains remarkable gains using more refinement steps, with both small and large backbones. Moreover, we note that refinement brings more gains on LVIS compared with MS-COCO. For example, its performance increases from 45.5 to 46.2 (+ 0.7 AP) on MS-COCO while from 30.4 to 31.9 (+1.5 AP) on LVIS, which demonstrates that our refining strategy would become more helpful for a more challenging benchmark.

4.4. Ablation Study

We conduct ablation experiments on MS-COCO to study DiffusionDet in detail. All experiments use ResNet-50 with FPN as the backbone and 300 boxes for training and inference without further specification.

Signal scaling. The signal scaling factor controls the signal-to-noise ratio (SNR) of the diffusion process. We study the influence of scaling factors in Table 3a. Results demonstrate that the scaling factor of 2.0 achieves optimal AP performance, outperforming the standard value of 1.0 in image generation task [13, 35] and 0.1 used for panoptic segmentation [12]. We explain that it is because one box only has four representation parameters, *i.e.*, center coordinates (c_x, c_y) and box size (w, h), which is coarsely analogous to an image with only four pixels in image generation. The box representation is more fragile than the dense representation, *e.g.*, 512×512 mask presentation in panoptic segmentation [13]. Therefore, DiffusionDet prefers an easier training objective with an increased signal-to-noise ratio compared to image generation and panoptic segmentation.

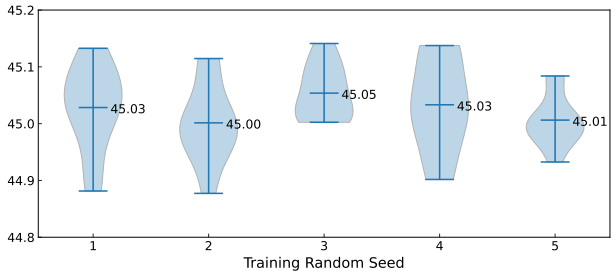


Figure 5. **Statistical results** over 5 independent training instances, each is evaluated 10 times with different random seeds. The number in the figure shows mean values.

GT boxes padding strategy. As introduced in Section 3.3, we need to *pad* additional boxes to the original ground truth boxes such that each image has the same number of boxes. We study different padding strategies in Table 3b, including (1.) repeating original ground truth boxes evenly until the total number reaches pre-defined value N_{train}; (2.) padding random boxes that follow Gaussian distribution; (3.) padding random boxes that follow uniform distribution; (4.) padding boxes that have the same size as the whole image, which is the default initialization of learnable boxes in [81]. Concatenating Gaussian random boxes works best for DiffusionDet. We use this padding strategy as default.

Sampling strategy. We compare different sampling strategies in Table 3c. When evaluating DiffusionDet that does not use DDIM, we directly take the output prediction of the current step as input of the next step. We found that the AP of DiffusionDet degrades with more evaluation steps when neither DDIM nor box renewal is adopted. Besides, only using DDIM or box renewal would bring slight benefits at 4 steps and does not bring further improvements when using more steps. Moreover, our DiffusionDet attains remarkable gains when equipped with both DDIM and renewal. These experiments together verify the necessity of both DDIM and box renewal in the sampling step.

Box renewal threshold. As discussed in Section 3.4, the box renewal strategy is proposed to reactivate the predictions whose scores are lower than a specific threshold. Table 3d shows the effect of the score threshold for box renewal. A threshold of 0.0 is no box renewal used. The results demonstrate the threshold of 0.5 performs slightly better than other ones.

Matching between N_{train} and N_{eval}. As discussed in Sec. 4.2, DiffusionDet has an appealing property of evaluating with an arbitrary number of random boxes. To study how the number of training boxes effects inference performance, we train DiffusionDet with $N_{train} \in \{100, 300, 500\}$ random boxes separately and then evaluate each of these models with $N_{eval} \in \{100, 300, 500, 1000\}$. The results are summarized in Table 3e. First, no matter how many random boxes DiffusionDet uses for training, the accuracy increases steadily with the N_{eval} until the saturated point at around 2000 random boxes. Second, DiffusionDet tends to perform better when the N_{train} and N_{eval} *matches* with each other. For example, DiffusionDet trained with $N_{train} = 100$ boxes behaves better than $N_{train} = 300$ and 500 when $N_{eval} = 100$.

Accuracy vs. speed. We test the inference speed of DiffusionDet in Table 3f. The run time is evaluated on a single NVIDIA A100 GPU with a mini-batch size of 1. We experiment with multiple choices of $N_{train} \in \{100, 300\}$ and keep N_{eval} same as the corresponding N_{train}. We see that increasing N_{train} from 100 to 300 brings 2.5 AP gains while negligible latency cost (31.6 FPS vs. 31.3 FPS) We also test the inference speed of 4 steps when $N_{train} = 300$. We observe that more refinements cost brings more inference times and results in less FPS. Increasing the refining step from 1 to 4 provides 0.8 AP gains but makes the detector slower. For reference, we compare DiffusionDet against Sparse R-CNN with 300 proposals, and DiffusionDet with 300 boxes has a very close FPS to Sparse R-CNN.

Random seed. Since DiffusionDet is given random boxes as input at the start of inference, one may ask whether there is a large performance variance across different random seeds. We evaluate the stability of DiffusionDet by training five models independently with strictly the same configurations except for random seed. Then, we evaluate each model instance with ten different random seeds to measure the distribution of performance, inspired by [61, 88]. As shown in Figure 5, most evaluation results are distributed closely to 45.0 AP. Besides, the mean values are all above 45.0 AP, and the performance differences among different model instances are marginal, which demonstrates that DiffusionDet is robust to the random boxes and produces reliable results.

5. Conclusion and Future Work

In this work, we propose a novel detection paradigm, DiffusionDet, by viewing object detection as a denoising diffusion process from noisy boxes to object boxes. Our noise-to-box pipeline has several appealing properties, including dynamic box and progressive refinement, enabling us to use the same network parameters to obtain the desired speed-accuracy trade-off without re-training the model. Experiments on standard detection benchmarks show that DiffusionDet achieves favorable performance compared to well-established detectors.

To further explore the potential of diffusion model to solve object-level recognition tasks, several future works are beneficial. An attempt is to apply DiffusionDet to video-level tasks, for example, object tracking and action recognition. Another is to extend DiffusionDet from close-world to open-world or open-vocabulary object detection.

References

[1] Tomer Amit, Eliya Nachmani, Tal Shaharbany, and Lior Wolf. Segdiff: Image segmentation with diffusion probabilistic models. *arXiv preprint arXiv:2112.00390*, 2021. 1, 2

[2] Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. *arXiv preprint arXiv:2205.15019*, 2022. 2

[3] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 1, 2

[4] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022. 1, 2

[5] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *International Conference on Learning Representations*, 2022. 1, 2

[6] Emmanuel Asiedu Brempong, Simon Kornblith, Ting Chen, Niki Parmar, Matthias Minderer, and Mohammad Norouzi. Denoising pretraining for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4175–4186, 2022. 1, 2

[7] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: high quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1483–1498, 2019. 6

[8] Hanqun Cao, Cheng Tan, Zhangyang Gao, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion model. *arXiv preprint arXiv:2209.02646*, 2022. 2

[9] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 1, 2, 3, 4, 5, 6, 13, 14, 15

[11] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6

[12] Ting Chen, Lala Li, Saurabh Saxena, Geoffrey Hinton, and David J Fleet. A generalist framework for panoptic segmentation of images and videos. *arXiv preprint arXiv:2210.06366*, 2022. 1, 2, 4, 7

[13] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022. 2, 4, 7

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[15] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 1, 2, 4, 13

[16] Yuming Du, Wen Guo, Yang Xiao, and Vincent Lepetit. 1st place solution for the uvo challenge on image-based open-world segmentation 2021. *arXiv preprint arXiv:2110.10239*, 2021. 4

[17] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019. 1

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 4

[19] Wanshu Fan, Yen-Chun Chen, Dongdong Chen, Yu Cheng, Lu Yuan, and Yu-Chiang Frank Wang. Frido: Feature pyramid diffusion for complex scene image synthesis. *ArXiv*, abs/2208.13753, 2022. 2

[20] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 2334–2343, 2017. 1

[21] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3621–3630, October 2021. 1

[22] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 303–312, 2021. 4, 15

[23] Z Ge, S Liu, F Wang, Z Li, and J Sun. Yolox: Exceeding yolo series in 2021. arxiv. *arXiv preprint arXiv:2107.08430*, 2021. 4, 5, 15

[24] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 2

[25] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[26] 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. JMLR Workshop and Conference Proceedings, 2010. 5

[27] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022. 2

[28] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *arXiv preprint arXiv:2206.09012*, 2022. 1, 2

[29] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018. 1

[30] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706, 2022. 2

[31] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019. 2, 4, 5, 7

[32] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *arXiv preprint arXiv:2205.11495*, 2022. 2

[33] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 2, 3

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 3, 6, 7

[35] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 2, 3, 4, 7, 13

[36] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. 2

[37] Emiel Hoogeboom, Victor Garcia Satorras, Clement Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. *arXiv e-prints*, pages arXiv–2203, 2022. 1, 2

[38] Rongjie Huang, Zhou Zhao, Huadai Liu, Jinglin Liu, Chenye Cui, and Yi Ren. Prodiff: Progressive fast diffusion model for high-quality text-to-speech. *arXiv preprint arXiv:2207.06389*, 2022. 2

[39] Bowen Jing, Gabriele Corso, Regina Barzilay, and Tommi S Jaakkola. Torsional diffusion for molecular conformer generation. In *ICLR2022 Machine Learning for Drug Discovery*, 2022. 2

[40] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015. 1

[41] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2011. 1

[42] Boah Kim, Yujin Oh, and Jong Chul Ye. Diffusion adversarial representation learning for self-supervised vessel segmentation. *arXiv preprint arXiv:2209.14566*, 2022. 1, 2

[43] Sungwon Kim, Heeseung Kim, and Sungroh Yoon. Guided-tts 2: A diffusion model for high-quality adaptive text-to-speech with untranscribed data. *arXiv preprint arXiv:2205.15370*, 2022. 2

[44] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 2

[45] Alon Levkovitch, Eliya Nachmani, and Lior Wolf. Zero-shot voice conditioning for denoising diffusion tts models. *arXiv preprint arXiv:2206.02246*, 2022. 2

[46] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022. 1, 2

[47] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *arXiv preprint arXiv:2205.14217*, 2022. 2

[48] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2359–2367, 2017. 1

[49] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2, 3, 7

[50] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2, 6, 15

[51] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 4, 5

[52] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *International Conference on Learning Representations*, 2022. 2

[53] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[54] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 3, 6, 7

[55] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 5, 15

[56] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European conference on computer vision*, pages 852–869. Springer, 2016. 1

[57] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3651–3660, 2021. 2

[58] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 1

[59] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 4, 13

[60] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16784–16804. PMLR, 17–23 Jul 2022. 2

[61] David Picard. Torch. manual_seed (3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision. *arXiv preprint arXiv:2109.08203*, 2021. 8

[62] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *ICML*, 2021. 2

[63] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022. 1, 2

[64] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[65] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2

[66] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2017. 1, 2, 3, 6

[67] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019. 15

[68] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *ArXiv*, abs/2208.12242, 2022. 2

[69] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2

[70] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022. 2

[71] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 1

[72] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018. 4

[73] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 1

[74] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 2

[75] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 3

[76] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 3, 4

[77] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 3

[78] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020. 3

[79] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 1, 2

[80] Peize Sun, Yi Jiang, Enze Xie, Wenqi Shao, Zehuan Yuan, Changhu Wang, and Ping Luo. What makes for end-to-end

object detection? In *International Conference on Machine Learning*, pages 9934–9944. PMLR, 2021. 2

[81] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15

[82] Jaesung Tae, Hyeongju Kim, and Taesu Kim. Editts: Score-based editing for controllable text-to-speech. *arXiv preprint arXiv:2110.02584*, 2021. 2

[83] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11662–11671, 2020. 7

[84] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 2

[85] Brian L Trippe, Jason Yim, Doug Tischer, Tamara Broderick, David Baker, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022. 1, 2

[86] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[87] Xinggang Wang, Kaibing Chen, Zilong Huang, Cong Yao, and Wenyu Liu. Point linking network for object detection. *arXiv preprint arXiv:1706.03646*, 2017. 2

[88] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*, 2021. 8

[89] Julia Wolleb, Robin Sandkühler, Florentin Bieder, Philippe Valmaggia, and Philippe C Cattin. Diffusion models for implicit image segmentation ensembles. *arXiv preprint arXiv:2112.03145*, 2021. 1, 2

[90] Junfeng Wu, Qihao Liu, Yi Jiang, Song Bai, Alan Yuille, and Xiang Bai. In defense of online models for video instance segmentation. *arXiv preprint arXiv:2207.10661*, 2022. 4

[91] Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *arXiv preprint arXiv:2209.00865*, 2022. 2

[92] Shoule Wu and Ziqiang Shi. Itôtts and itôwave: Linear stochastic differential equation is all you need for audio generation. *arXiv e-prints*, pages arXiv–2105, 2021. 2

[93] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5, 6, 7

[94] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2021. 2

[95] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *arXiv preprint arXiv:2207.09983*, 2022. 2

[96] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion probabilistic modeling for video generation. *arXiv preprint arXiv:2203.09481*, 2022. 2

[97] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9657–9666, 2019. 1

[98] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 5

[99] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022. 2

[100] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. In *arXiv preprint arXiv:2103.07461*, 2021. 6, 7

[101] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1, 2

[102] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4, 5, 6, 13, 14, 15

A. Formulation of Diffusion Model

We provide a detailed review of the formulation of diffusion models, following the notion of [15, 35, 59]. 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 $\beta_t \in (0, 1)$:

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

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

As noted by Ho *et al*. [35], we can directly sample data $\boldsymbol{z}_t$ at an arbitrary timestep t without the need of applying q repeatedly:

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

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

where $\bar{\alpha}_t \coloneqq \prod_{s=0}^{t} \alpha_s$ and $\alpha_t \coloneqq 1 - \beta_t$. 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

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

and

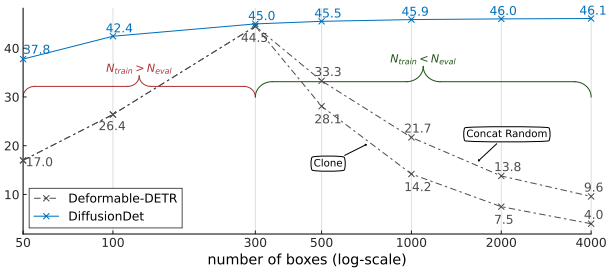
$$\tilde{\beta}_t \coloneqq \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{9}$$

are mean and variance of this Gaussian distribution.

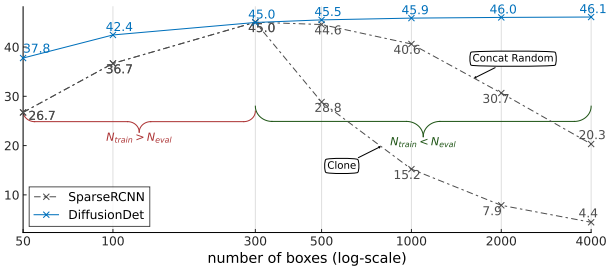
We could get a sample from $q(\boldsymbol{z}_0)$ by first sampling from $q(\boldsymbol{z}_T)$ and running the reversing steps $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ until $\boldsymbol{z}_0$. Besides, the distribution of $q(\boldsymbol{z}_T)$ is nearly an isotropic Gaussian distribution with a sufficiently 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:

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

Instead of directly parameterizing $\mu_\theta(\boldsymbol{z}_t, t)$, Ho *et al*. [35] 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 predicting $\boldsymbol{z}_0$ in this work.



(a) **Deformable DETR [102] vs. DiffusionDet**



(b) **Sparse R-CNN [81] vs. DiffusionDet**

Figure 6. **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. Dynamic Boxes

We further compare the dynamic box property of DiffusionDet with Deformable DETR [102] and Sparse R-CNN [81] in Figure 6. We directly use the provided models in their official code repositories.[1] [2]

We make Deformable DETR to 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 44.5 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 property of DiffusionDet.

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

C. Progressive Refinement

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)

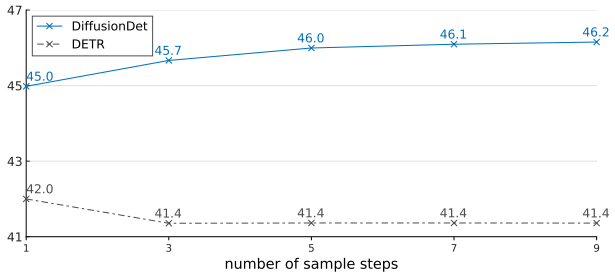
Table 4. **Progressive refinement.** [E] denotes ensembling predictions from multiple steps. NMS is adopted when using ensemble strategy. We show the performance differences of each method with respect to their own performance on step 1 by (-) or (+).

In Table 4 we compare the progressive refinement property of DiffusionDet with some previous approaches like DETR [10], Deformable DETR [102] and Sparse R-CNN [81]. All of these four models have 6 cascading stages as the detection decoder. The refinement refers to the output of the previous 6 stages is taken as the input of the next 6 stages. All model checkpoints are from Model Zoo in their official code repositories.

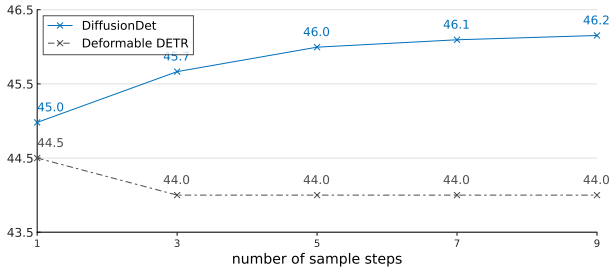
We experiment with two settings: (1) only use the output of the last reference step as final prediction; (2) use the ensemble of the output of multiple steps as final prediction. For the latter setting, we adopt NMS to remove duplicate predictions among different steps.

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. For example, DiffusionDet with five steps only has 0.05 AP drop. On the contrary, Deformable DETR has more than 1 AP drop, and Sparse R-CNN even totally failed (only gets 1.32 and 0.32 AP with three and five steps, respectively).

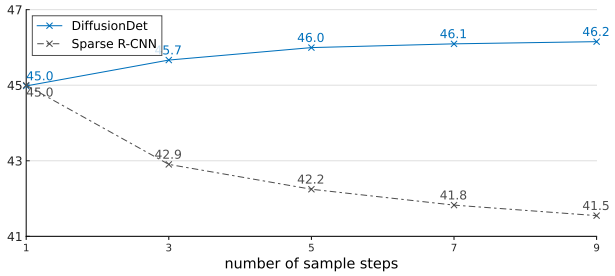
Adopting ensemble would mitigate the performance degradation except for DETR. Nevertheless, previous query-based based methods still have performance drop with more steps. In contrast, DiffusionDet turns performance down to up. Specifically, DiffusionDet has performance gains with more refinement steps. For example, DiffusionDet with five steps has 1.02 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 4 in Figure 7.



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



(b) **Deformable DETR [102] vs. DiffusionDet**



(c) **Sparse R-CNN [81] vs. DiffusionDet**

Figure 7. **Progressive refinement.** All models are the provided models in their official code repositories.

D. Experimental Settings

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

D.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 5. We note that 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 original one.

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

config	value
# object queries	300
optimizer	AdamW [55]
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 5. **DETR reproduction setting.**

config	value
optimizer	AdamW [55]
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 6. **COCO setting.**

config	value
optimizer	AdamW [55]
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 7. **LVIS setting.**

D.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 6 and Table 7, respectively.

E. Training Loss

We adopt set prediction loss [10, 81, 102] 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 [50] 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 [67]. λ_{cls}, λ_{L1} and $\lambda_{giou} \in \mathbb{R}$ are weight of each component to balance to overall multiple losses. Following [10, 81, 102], 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 [22, 23]. 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.

F. Visualization

We visualize our sampling step of DiffusionDet in Figure 8. The model is running with 300 boxes. For better visualization, we only draw 50 boxes in the image.

(a) Initial random boxes are input into the detection detector. The random boxes are sampled from the Gaussian distribution.

(b) The detection decoder predicts the category scores and box coordinates of the current step. In sub-figure (b), the color brightness is proportional to the score value, where the deep red is high score, and white is low score.

(c) DDIM estimates the boxes for the next step.

(d) Those boxes with lower scores than the threshold are dropped.

(e) New random boxes sampled from a Gaussian distribution are concatenated to the remaining boxes. This new set of boxes are input to the detection detector.

(f) After multiple steps of refinement, final predictions are obtained.

Figure 8. **Visualization** of sampling step in inference.