DiffusionDet: Diffusion Model for Object Detection

Shoufa Chen\textsuperscript{1} Peize Sun\textsuperscript{1} Yibing Song\textsuperscript{2,3} Ping Luo\textsuperscript{1,4}
\textsuperscript{1}The University of Hong Kong \textsuperscript{2}Tencent AI Lab
\textsuperscript{3}AI\textsuperscript{3} Institute, Fudan University \textsuperscript{4}Shanghai AI Laboratory
\{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 the training stage, object boxes diffuse from ground-truth boxes to random distribution, and the model learns to reverse this noisy process. In inference, the model refines a set of randomly generated boxes to the output results in a progressive way. Our work possesses an appealing property of flexibility, which enables the dynamic number of boxes and iterative evaluation. The extensive experiments on the standard benchmarks show that DiffusionDet achieves favorable performance compared to previous well-established detectors. For example, DiffusionDet achieves 5.3 AP and 4.8 AP gains when evaluated with more boxes and iteration steps, under a zero-shot transfer setting from COCO to CrowdHuman. 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 [36, 53], pose estimation [9, 22], action recognition [32, 81], object tracking [46, 65], and visual relationship detection [45, 62].

Modern object detection approaches have been evolving with the development of object candidates, i.e., from empirical object priors [27, 59, 72, 74] to learnable object queries [10, 90, 114]). Specifically, the majority of detectors solve detection tasks by defining surrogate regression and classification on empirically designed object candidates, such as sliding windows [28, 79], region proposals [27, 74], anchor boxes [56, 72] and reference points [19, 105, 112]. 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 [23, 51, 90, 114].

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 the training stage, we expect to gradually refine the positions and sizes of these boxes until they perfectly cover the targeted objects. This noise-to-box approach requires neither 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 [16, 38, 88], 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, 40, 71, 94] and start to be explored in perception tasks like image segmentation [1, 5, 6, 13, 31, 47, 97]. How-
ever, to the best of our knowledge, there is no prior art that successfully adopts it to object detection.

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 the training stage, Gaussian noise controlled by a variance schedule [38] is added to ground truth boxes to obtain noisy boxes. Then these noisy boxes are used to crop [36, 74] features of Region of Interest (RoI) from the output feature map of the backbone encoder, e.g., ResNet [37], Swin Transformer [60]. 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 the 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.

As a probabilistic model, DiffusionDet has an attractive superiority of flexibility, i.e., we can train the network once and use the same network parameters under diverse settings in the inference stage, mainly including: (1) Dynamic number of boxes. Leveraging random boxes as object candidates, we decouple the training and evaluation stage of DiffusionDet, i.e., we can train DiffusionDet with $N_{train}$ random boxes while evaluating it with $N_{eval}$ random boxes, where the $N_{eval}$ is arbitrary and does not need to be equal to $N_{train}$. (2) Iterative evaluation. Benefited by the iterative denoising property of diffusion models, DiffusionDet can reuse the whole detection head in an iterative way, further improving its performance.

The flexibility of DiffusionDet makes it a great advantage in detecting objects across different scenarios, e.g., sparse or crowded, without additional fine-tuning. Specifically, Table 1 shows that when directly evaluating COCO-pretrained models on CrowdHuman [80] dataset, which covers more crowded scenes, DiffusionDet achieves significant gains by adjusting the number of evaluation boxes and iteration steps. In contrast, previous methods only obtain marginal gains or even degraded performance. More detailed discussions are left in Section 4.

Besides, we evaluate DiffusionDet on COCO [57] dataset. With ResNet-50 [37] backbone, DiffusionDet achieves 45.8 AP using a single sampling step and 300 random boxes, which significantly outperforms Faster R-CNN [74] (40.2 AP), DETR [10] (42.0 AP) and on par with Sparse R-CNN [90] (45.0 AP). Besides, we can further improve DiffusionDet up to 46.8 AP by increasing the number of sampling steps and random boxes.

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 iterative evaluation.
- We conduct extensive experiments on COCO, CrowdHuman, and LVIS benchmarks. DiffusionDet achieves favorable performance against previous well-established detectors, especially zero-shot transferring across different scenarios.

2. Related Work

Object detection. Most modern object detection approaches perform box regression and category classification on empirical object priors, such as proposals [27, 74], anchors [56, 72, 73], points [93, 95, 112]. 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 [12, 24, 43, 51, 58, 64, 66, 89, 90, 107, 110, 114]. In this work, we push forward the development of the object detection pipeline further with DiffusionDet.

Diffusion model. As a class of deep generative models, diffusion models [38, 86, 88] 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, 21, 33, 35, 39, 68, 71, 75, 76, 82, 104, 108], nature language processing [3, 30, 52], audio processing [41, 48, 50, 70, 91, 100, 103], graph-related topics [42], interdisciplinary applications [2, 40, 44, 78, 94, 99, 102], etc. More applications of diffusion models can be found in recent surveys [8, 104].

Diffusion model for perception tasks. While Diffusion models have achieved great success in image generation [16, 38, 88], 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, 19831]
3. Approach

3.1. Preliminaries

**Object detection.** The learning objective of object detection is input-target pairs \((x, b, c)\), where \(x\) is the input image, \(b\) and \(c\) are a set of bounding boxes and category labels for objects in the image \(x\), respectively. More specifically, we formulate the \(i\)-th box in the set as \(b^i = (c_{x}^i, c_{y}^i, w^i, h^i)\), 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 [38, 83, 84, 86] are a class of likelihood-based models inspired by nonequilibrium thermodynamics [86, 87]. These models define a Markovian chain of diffusion forward process by gradually adding noise to sample data. The forward noise process is defined as

\[
q(z_t | z_0) = \mathcal{N}(z_t | \sqrt{\bar{\alpha}_t} z_0, (1 - \bar{\alpha}_t) I),
\]

which transforms data sample \(z_0\) to a latent noisy sample \(z_t\) for \(t \in \{0, 1, ..., T\}\) by adding noise to \(z_0\). \(\bar{\alpha}_t \defeq \prod_{s=0}^{t} \alpha_s = \prod_{s=0}^{t} (1 - \beta_s)\) and \(\beta_s\) represents the noise variance schedule [38]. During training, a neural network \(f_\theta(z_t, t)\) is trained to predict \(z_0\) from \(z_t\) by minimizing the training objective with \(\ell_2\) loss [38]:

\[
L_{\text{train}} = \frac{1}{2} \left| f_\theta(z_t, t) - z_0 \right|^2.
\]

At inference stage, data sample \(z_0\) is reconstructed from noise \(z_T\) with the model \(f_\theta\) and an updating rule [38, 84] in an iterative way, i.e., \(z_T \rightarrow z_{T-1} \rightarrow \cdots \rightarrow z_0\). 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 \(z_0 = b\), where \(b \in \mathbb{R}^{N \times 4}\) is a set of \(N\) boxes. A neural network \(f_\theta(z_t, t, x)\) is trained to predict \(z_0\) from noisy boxes \(z_t\), conditioned on the corresponding image \(x\). The corresponding category label \(c\) is produced accordingly.

3.2. Architecture

Since the diffusion model generates data samples iteratively, it needs to run model \(f_\theta\) multiple times at the inference stage. However, it would be computationally intractable to directly apply \(f_\theta\) 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 \(x\), and the latter takes this deep feature as condition, instead of the raw image, to progressively refine the box predictions from noisy boxes \(z_t\).

**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 [37] and Transformer-based models like Swin [60]. Feature Pyramid Network [55] is used to generate multi-scale feature maps for both ResNet and Swin backbones following [55, 60, 90].

**Detection decoder.** Borrowed from Sparse R-CNN [90], the detection decoder takes as input a set of proposal boxes to crop Rol-feature [36, 74] from feature map generated...
Algorithm 1 DiffusionDet Training

```python
def train_loss(images, gt_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
    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
```

by image encoder, and sends these RoI-features to detection head to obtain box regression and classification results. For DiffusionDet, these proposal boxes are disturbed from ground truth boxes at training stage and directly sampled from Gaussian distribution at evaluation stage. Following [10, 90, 114], our detection decoder is composed of 6 cascading stages (Figure 2b). 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 can re-use the detector head in an iterative way for evaluation and the parameters are shared across different steps, each of which is specified to the diffusion process by timestep embedding [38], which is called iterative evaluation, 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.

Ground truth boxes padding. For modern object detection benchmarks [20, 34, 57, 80], 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 \( \alpha_t \) (in Eq. (1)), which adopts the monotonically decreasing cosine schedule for \( \alpha_t \) in different time step \( t \), as proposed in [67]. 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 [13]. We observe that object detection favors a relatively higher signal scaling value than image generation task [14, 16, 38]. 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, 90, 114] 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 [18, 25, 26, 98].

Algorithm 2 DiffusionDet Sampling

```python
def infer(images, steps, T):
    # 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(linspace(-1, T, steps))
    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
```

`linspace`: generate evenly spaced values

19833
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 [84] 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.

**Flexible usage.** Thanks to the random boxes design, we can evaluate DiffusionDet with an arbitrary number of random boxes and the number of iteration times, which do not need to be equal to the training stage. As a comparison, previous approaches [10, 90, 114] rely on the same number of processed boxes during training and evaluation, and their detection decoders are used only once in the forward pass.

### 3.5. Discussion

We conduct a comparative analysis between DiffusionDet and previous multi-stage detectors [7, 10, 74, 90]. Cascade R-CNN adopts a three-stage prediction refinement process where the three stages do not share parameters and are used only once as a complete head during the inference phase. Recent works [10, 90, 114] have adopted a similar structure as Cascade R-CNN but with more stages (i.e., six), following the default setting of DETR [10]. While DiffusionDet also employs the six-stage structure within its head, the distinguishing feature is that DiffusionDet can reuse the entire head multiple times to achieve further performance gains. However, prior works could not improve performance by reusing the detection head in most cases or could only achieve limited performance gains. More detailed results are in Section 4.4.

### 4. Experiments

We first show the attractive flexibility of DiffusionDet. Then we compare DiffusionDet with previous well-established detectors on COCO [57] and CrowdHuman [80] dataset. Finally, we provide ablation studies on the components of DiffusionDet.

**COCO** [57] 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 (AP50) and 0.75 (AP75).

**LVIS v1.0** [34] 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 COCO, while its annotations capture the long-tailed distribution in 1203 categories. We adopt MS-COCO style box metric AP, AP50 and AP75 in LVIS evaluation. For LVIS, the training schedule is 210K, 250K, and 270K.

**CrowdHuman** [80] dataset is a large dataset covering various crowd scenarios. It has 15K training images and 4.4K validation images, including a total of 470K human instances and 22.6 persons per image. Following previous settings [54, 90, 109, 113], we adopt evaluation metrics as AP under IoU threshold 0.5.

#### 4.1. Implementation Details.

The ResNet and Swin backbone are initialized with pretrained weights on ImageNet-1K and ImageNet-21K [15], respectively. The newly added detection decoder is initialized with Xavier init [29]. We train DiffusionDet using AdamW [61] optimizer with the initial learning rate as $2.5 \times 10^{-5}$ and the weight decay as $10^{-4}$. All models are trained with a mini-batch size 16 on 8 GPUs. The default training schedule is 450K iterations, with the learning rate divided by 10 at 350K and 420K iterations. 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 [101], and random crop augmentations. We do not use EMA and some strong data augmentation like MixUp [106] or Mosaic [26].

At the inference stage, we report performances of DiffusionDet under diverse settings, which are combinations of different numbers of random boxes and iteration steps. The predictions at each sampling step are ensembled together by NMS to get the final predictions.

#### 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 the number of iteration steps in inference, as shown in Figure 3 and Table 1. Therefore, we can deploy a single DiffusionDet to multiple scenarios and obtain a desired speed-accuracy trade-off without re-training the network.

**Dynamic number of 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 of 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.

Since the learnable queries are fixed after training in the original setting of DETR, we propose a simple workaround to enable DETR work with a 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 clone existing \( N_{train} \) queries up to \( N_{eval} \) (a.k.a. clone). We equip DETR with NMS because cloned queries will produce similar detection results as the original queries. As shown in Figure 3a, the performance of DiffusionDet increases steadily with the number of boxes used for evaluation. For example, DiffusionDet can achieve 1.0 AP gain when the number of boxes increases from 300 to 4000. On the contrary, cloning more queries for DETR (\( N_{eval} > 300 \)) causes a slight decrease in DETR performance from 38.8 to 38.4 AP, which is then held constant when using more queries.

We also implement another method for DETR when \( N_{eval} > N_{train} \), concatenating extra \( N_{eval} - N_{train} \) randomly initialized queries (a.k.a. concat random). With this strategy, DETR has a clear performance drop when the \( N_{eval} \) is different from \( N_{train} \). 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).

**Iterative evaluation.** We further investigate the performance of our proposed approach by increasing the number of iterative steps from 1 to 8, and the corresponding results are illustrated in Figure 3b. Our findings indicate that the DiffusionDet models employing 100, 300, and 500 random boxes exhibit consistent performance improvements as the number of iterations increases. Moreover, we observe that DiffusionDet with fewer random boxes tends to achieve more substantial gains with refinement. For instance, the AP of DiffusionDet instance utilizing 100 random boxes improves from 41.9 (1 step) to 46.1 (8 steps), representing an absolute improvement of 4.2 AP.

**Zero-shot transferring.** To further validate the effectiveness of generalization, we conduct an evaluation of COCO-pretrained models on the CrowdHuman dataset, without any additional fine-tuning. Specifically, our focus is on the [person] class for the final average precision (AP) performance. The experimental results are presented in Table 1. Our observations indicate that when transferring to a new dataset with scenarios that are more densely populated than COCO, our proposed method, namely DiffusionDet, demonstrates a notable advantage by increasing the number of evaluation boxes or iteration steps. For instance, by increasing the number of boxes from 300 to 2000 and the iteration steps from 1 to 4, DiffusionDet achieves a significant AP gain of 5.3 and 4.8, respectively. In contrast, previous methods exhibit limited gain or serious performance degradation, with a decrease of 14.0 AP. The impressive flexibility of DiffusionDet implies that it is an invaluable asset for object detection tasks across a wide range of scenarios, including sparsely populated and densely crowded environments, without any additional fine-tuning requirements.

### 4.3. Benchmarking on Detection Datasets

In Table 2, we present a comparison of our DiffusionDet with several state-of-the-art detectors [7, 10, 56, 74, 90, 114] on the COCO dataset. For more comprehensive experimental settings, please refer to the Appendix. Notably, our Dif-
Our current model is still lagging behind some well-established methods such as Faster R-CNN, RetinaNet, DETR, and Sparse R-CNN by a considerable margin. Moreover, DiffusionDet can further enhance its superiority by increasing the number of iterations and evaluation boxes. Besides, DiffusionDet shows steady improvement when the backbone size scales up. DiffusionDet with ResNet-101 (1 @ 300) achieves 46.7. When using ImageNet-21k pre-trained Swin-Base [60] as the backbone, DiffusionDet obtains 52.5 AP, outperforming strong baselines such as Cascade R-CNN and Sparse R-CNN.

Experimental results on LVIS are presented in Table 3. We reproduce Faster R-CNN and Cascade R-CNN based on detectron2 [101] 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 3) with ResNet-50/101 backbone, respectively. Further, we boost their performance using the federated loss in [111]. Since images in LVIS are annotated in a federated way [34], the negative categories are sparsely annotated, which deteriorates the training gradients, especially for rare classes [92]. 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 [111], 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 evaluation steps, with both small and large backbones. Moreover, we note that iterative evaluation brings more gains on LVIS compared with COCO. For example, its performance increases from 45.8 to 46.6 (+0.8 AP) on COCO while from 29.4 to 31.5 (+2.1 AP) on LVIS, which demonstrates that our DiffusionDet would become more helpful for a more challenging benchmark.
4.4. Ablation Study

We conduct ablation experiments on COCO to study DiffusionDet in detail. All experiments use ResNet-50 with FPN as the backbone and 300 random boxes for 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 4a. Results demonstrate that the scaling factor of 2.0 achieves optimal AP performance, outperforming the standard value of 1.0 in image generation task [14, 38] and 0.1 used for panoptic segmentation [13]. 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, \( e.g., \) \( 512 \times 512 \) mask presentation in panoptic segmentation [14]. Therefore, DiffusionDet prefers an easier training objective with an increased signal-to-noise ratio compared to image generation and panoptic segmentation.

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 4b, including (1) repeating original ground truth boxes evenly until the total number reaches pre-defined value \( N_{\text{train}} \); (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 [90]. Concatenating Gaussian random boxes works best for DiffusionDet. We use this padding strategy as default.

Sampling strategy. We compare different sampling strategies in Table 4c. When evaluating DiffusionDet that does not use DDIM, we directly take the output prediction of the current step as input for the next step. We found that the AP of DiffusionDet degrades with more iteration steps when neither DDIM nor box renewal is adopted. Besides, only using DDIM or box renewal would bring slight benefits at 3 iteration 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.

Matching between \( N_{\text{train}} \) and \( N_{\text{eval}} \). 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 affects inference performance, we train DiffusionDet with \( N_{\text{train}} \in \{100, 300, 500\} \) random boxes separately and then evaluate each of these models with \( N_{\text{eval}} \in \{100, 300, 500, 1000, 2000\} \). The results are summarized in Table 5. First, no matter how many random boxes DiffusionDet uses for training, the accuracy increases steadily with the \( N_{\text{eval}} \) until the saturated point at around 2000 random boxes. Second, DiffusionDet tends to perform better when the \( N_{\text{train}} \) and \( N_{\text{eval}} \) matches with each other. For example, DiffusionDet trained with \( N_{\text{train}} = 100 \) boxes behaves better than \( N_{\text{train}} = 300 \) and 500 when \( N_{\text{eval}} = 100 \).

Running time vs. accuracy. We investigate the running time of DiffusionDet under multiple settings, which are evaluated on a single NVIDIA A100 GPU with a mini-batch size of 1. We utilize the notation \( \#\text{Stages} \times \#\text{Heads} \) to indicate the number of stages and heads utilized during the training and test phases, as depicted in Figure 2b and results of our investigation are presented in Table 6.

First, our findings indicate that DiffusionDet with a single iteration step and 300 evaluation boxes demonstrate a comparable speed to Sparse R-CNN, achieving 30 and 31 frames per second (FPS), respectively. DiffusionDet also showcases similar zero-shot transfer performance on CrowdHuman while outperforming Sparse R-CNN with an
45.8 AP as opposed to 45.0 AP on COCO. Besides, Sparse R-CNN’s utilization of the six stages twice results in a 1.4 AP drop (from 45.0 to 43.6) on COCO and a 6.0 AP drop (from 66.6 to 60.6) on CrowdHuman. Similarly, DETR experiences 0.4 performance drop on COCO but 1.2 performance gain on CrowdHuman.

When increasing the number of iteration steps, DiffusionDet achieves a 0.7 AP gain on COCO and a 3.1 AP gain on CrowdHuman. And DiffusionDet obtains clear performance gains with 1000 evaluation boxes. However, neither DETR nor Sparse R-CNN can achieve performance gains with additional iteration steps. Even if we expand the number of stages to 12, it can cause performance degradation for Sparse R-CNN.

It is worth noting that in this work, we have utilized the most fundamental diffusion strategy, DDIM, in our pioneering exploration of using generation models for perception tasks. Similar to the Diffusion model employed in generation tasks, DiffusionDet may suffer from a relatively slow sampling speed. Nonetheless, a series of recent works [17, 63, 77, 85] have been proposed to improve the sampling efficiency of the diffusion model. For instance, the most recent consistency models [85] have proposed a fast one-step generation method for the diffusion model. We believe that a more advanced diffusion strategy could potentially address the issue of decreased speed performance of DiffusionDet, which we plan to explore in future work.

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 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 [69, 96]. As shown in Figure 4, most evaluation results are distributed closely to 45.7 AP. Besides, the performance differences among different model instances are marginal, demonstrating that DiffusionDet is robust to the random boxes and produces reliable results.

4.5. Full-tuning on CrowdHuman

In addition to the cross-dataset generalization evaluation from COCO to CrowdHuman discussed in Section 4.2, we further full-tune DiffusionDet on CrowdHuman. The comparison results are shown in Table 7. We see that DiffusionDet achieves superior performance compared with previous methods. For example, with a single step and 1000 boxes, DiffusionDet obtains 90.1 AP50, outperforming Sparse R-CNN with 1000 boxes. Besides, further increasing boxes to 3000 and iteration steps can both bring performance gains.

5. Conclusion

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 the dynamic number of boxes and iterative evaluation, enabling us to use the same network parameters for flexible evaluation without re-training the model. Experiments on standard detection benchmarks show that DiffusionDet achieves favorable performance compared to well-established detectors.

Acknowledgement. This paper is partially supported by the National Key R&D Program of China No.2022ZD0161000 and the General Research Fund of Hong Kong No.17200622.
References


Matthieu Lin, Chuming Li, Xingyuan Bu, Ming Sun, Chen Lin, Junjie Yan, Wanli Ouyang, and Zhidong Deng. Detr for crowd pedestrian detection, 2021. 


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


Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenceng 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, 9


[114] 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, 4, 5, 6, 7, 9