Diffusing Objects as Points: One-Stage Object Detection with Diffusion Probabilistic Models

Namgi Kim
IPAI, SNU
rlaskarl77@snu.ac.kr

Inseo Lee
GSDS, SNU
ian.lee@snu.ac.kr

Hoyeon Moon
mhy991011@gmail.com

JooYoung Jang
GSCST, SNU
jyjang1090@snu.ac.kr

Abstract

Diffusion models are crossing the boundary of image generation and explosively adapted to other tasks like noise-to-segmentation, and noise-to-detection. DiffusionDet [4] was the first approach to adopting diffusion models to object detection tasks, but due to its two-stage architecture, there are some intrinsic problems with the diffusion process. In this work, we propose a one-stage object detection diffusion model, DiffusionPoint, which applies the diffusion process to the heatmap. Our approach outperforms the baseline [6] at mAP_{50} *by* 0.9%*, with faster inference speed, compared to DiffusionDet. This framework does not need heuristic postprocessing and enjoys the advantages of diffusion models like iterative refinement.*

1. Introduction

Object detection is a computer vision technology that detects the types and positions of objects present in an image. In recent years, diffusion models such as DDPM [13] have gained attention in the field of image generation. These models are capable of generating various samples, which has led to attempts to apply them to other fields. For instance, there have been efforts to incorporate diffusion into Text-to-image generation [15], super resolution [24], segmentation [1], and the first attempt to apply diffusion models to the field of object detection has also emerged with DiffusionDet [4].

DiffusionDet [4] has a two-stage structure where bounding boxes are extracted through the diffusion model and objects are detected through RoI pooling. At the inference step, the detection decoder improves the accuracy of its predictions by refining them based on Gaussian random boxes in an iterative process. The final predictions are obtained by combining the predictions from each sampling step using Non-Maximal Suppression(NMS).

Although DiffusionDet seems to be good model, there are still some limits. First, it needs a heuristic box-renewal process during inference, as using a fixed size of queries during training. The two-stage structure itself can be a problem, as making the detection process even slower, while applying diffusion. Finally, as the diffusion process is done separately with classification, it might be difficult for the model to balance the classification-localization tasks.

Therefore, in this work, we present a novel one-stage object detection diffusion model, which utilizes the *noise-to-heatmap* approach. Our framework, DiffusionPoint, does not need heuristic procedures like the box renewal and is faster as a one-staged model. In our framework, the diffusion process considers class priors, so it is easier to balance between classification and localization tasks.

2. Related Works

Object detection. Object detection is a computer vision technology that detects the types and locations of objects present in an image [33]. After convolutional neural networks (CNNs) [16] showed high performance on the ImageNet challenge, deep neural networks began to be used for object detection as well. Models such as RCNN [9], SPP-Net [12], Fast RCNN [8], and Faster RCNN [22] used CNN to extract features from images and use them to detect objects. These models detect objects in two stages to examine whether real objects exist in those areas. YOLO [21] is a model that detects objects with only convolutional operations. SSD [19], RetinaNet [17], which have emerged since then, also detect objects in one stage. In the case of one-stage models, the structure is relatively simple and fast, but there is a disadvantage that the location accuracy is somewhat lower than two-stage models. These are all anchor-

based models, while CenterNet [6] presents a method of detecting the center point of an object rather than a bounding box without anchors. A structure such as DETR [2] presents a method that applies the Transformer structure to the detection model.

Diffusion model. The Diffusion model is a generative model that injects random noise into data such as images and then learns the reverse process of finding the original data. Through this, it aims to enable the model to generate data with a desired shape from completely random noise. Recently, diffusion models such as DDPM [13] and DDIM [26] have been in the spotlight in the field of image generation, and many attempts have been made to incorporate them into other fields, paying attention to the point that diffusion models generate various samples. A growing number of computer vision fields are using the diffusion model, including Text-to-image generation [15], super resolution [24], segmentation [1], and object detection [4].

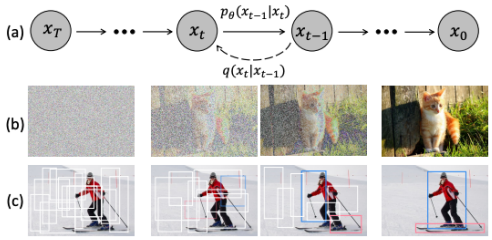

Figure 1. **Diffusion model for object detection.** The figure is brought from [4]. They formulated object detection as the denoising process of noisy bounding boxes.

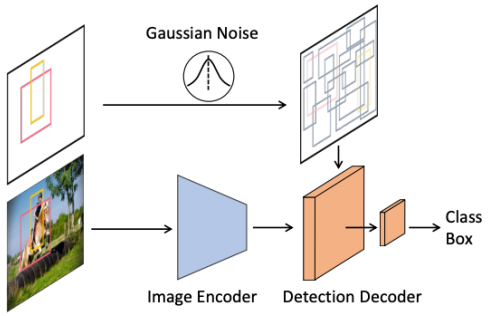

Figure 2. **DiffusionDet framework.** Figure from [4]. With extracted image features from the encoder, the detection decoder predicts class and box coordinates from noisy boxes.

Noise-to-Box Approach. Unlike image generation models, the diffusion process of object detection models does not happen in the pixel space or the latent space. Instead, they sample N samples of size$(x_{center}, y_{center}, width, height) \in R^4 \sim N(0, I)$. The method for detecting objects in DiffuseDet is the same as shown in Fig. 1. It uses a ResNet as a backbone to extract features from images and object detection is performed with bounding boxes that have gone through a backward diffusion process. The overall process is shown in Fig. 2. It starts with a random Gaussian noise and extracted feature representation of an input image. The denoising process of the decoder computes bounding boxes from noisy boxes, and successive classification heads predict their category and exact box coordinates. This process is called as *noise-to-box* approach. During the training, the noisy boxes are generated by adding Gaussian noise to the ground-truth boxes. In inference, the noisy boxes are randomly sampled from the Gaussian distribution.

3. Background

Anchor-free One-stage Detector Our framework is based on an anchor-free one-stage detection model. The common point is they are viewing objects as a combination of key points. This views objects as a set of disjoint points of the image: center position, box size, and offsets from the center position. The center position is constructed with class-wise heatmaps: if the value of (w, h, c) is high, it is likely that an object from class c is oriented at (w, h) of the image. Since the heat map can be easily drawn from GT and it is logical to think of injecting noise as passing the Gaussian kernel, we can apply the diffusion step to train this model. To be specific, we used Centernet [6], and VFNet [11] for our experiment. Both of these two networks have a heatmap head that shows the center location of the object. Then, by using heatmap head it can get center points of object $\hat{P}_c = \{(\hat{x}_i, \hat{y}_i\}_{i=1}^n$ of class c without any non-maximum suppression (NMS) or post-processing. Then produce a bounding box at a location

$$(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2, \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,$$

$$\hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2, \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2),$$

where $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i,\hat{y}_i}$ is the offset head's prediction and $(\hat{w}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i,\hat{y}_i}$ is the size head's prediction.

Diffusion process. In the diffusion model [25], the process of creating noise from the data through diffusion is called the forward diffusion process, while the process of the model restoring the original data format is called the backward denoising process. The forward diffusion process gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T$: defined as:

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

and backward denoising process is defined as:

$$p_\theta(x_{0:T}) := p(x_T)p_\theta \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$
$$p_\theta(x_{t-1}|x_t) := N(x_{t-1}; \mu_\theta(x_t, t), \sum \theta(x_t, t)) \quad (2)$$

In DDPM [13], the forward pass can be parameterized as:

$$\mu_\theta(\mathbf{x}_t, t) = \tilde{\mu}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t)\right)\right)$$
$$= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \quad (3)$$

Through DDPM, we can learn the generative Markov Chain Process that creates samples while denoising. However, this process has a limitation in that it requires a lot of steps in the form of sampling one by one. Therefore, in order to improve this, DDIM [26] generalized this process to non-markovian.

$$q_\delta(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q_\delta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q_\delta(\mathbf{x}_t|\mathbf{x}_0)}{q_\delta(\mathbf{x}_{t-1}|\mathbf{x}_0)} \quad (4)$$

DDIM's authors discovered that the objective of DDPM only depends on the marginal distribution and not directly on the joint distribution. Then, Eq. (4) can be derived from Bayes' rule.

It is known that adopting DDPM for training and DDIM for inference works best for many image-generation models. Therefore, we also trained our model by the DDPM approach and adopted DDIM for inference.

4. Method

4.1. Architecture

Since running the whole model iteratively for the diffusion process is computationally intractable, we followed the approach of DiffusionDet [4]. The *image encoder* runs only once to extract features from the input image, and the *detector head* inputs those features as conditions and goes through the diffusion process to make predictions. For the image encoder, we used conventional backbones like ResNet18 and ResNet50, which are often used as a backbone in object detection models. Our overall architecture is shown in Fig. 4 and Fig. 5.

For the detector head, we discussed multiple one-stage options. First, DETR-like approaches [2] [32] [30] can be a option. These can be easily modified to a DiffusionDet-like structure, but due to their heavy computation cost, we do not use this option. Anchor-free approaches [27] [6] [29] are structurally different from that of DiffusionDet and the diffusion process needs to be defined from scratch, but they are easy to implement and more intuitive in iterative refinement. We chose CenterNet [6] from Objects as Points, and VFNet [11] as the structure is simple but achieves high mAP with a smaller model size.

Noise-to-Heatmap approach. In this work, we present a *noise-to-heatmap* approach, which is applicable to various one-stage object detectors. The most challenging part of adopting the diffusion process to object detection task is defining the forward diffusion process. One naive approach is diffusing in pixel space, viewing the object detection task as an image-to-image translation task. Draw ground-truth bounding boxes on the image, inject noise, and denoise with UNet-like models. The bounding boxes can be retrieved by comparing the original image with the reconstructed image. However, this will take too much time to converge, as mere image generation tasks need vigorous training. Also, diffusing in pixel space is not intuitive, as the bounding boxes do not move spatially in the forward process.

Another possible solution is diffusing in the latent space. Each instance has five properties - class, x and y position, width and height. However, viewing objects as 5-dimensional vectors does not solve the problem. The class information should be considered separately from other properties since the class is categorical and the others are continuous. Moreover, there can be an arbitrary number of objects in one image. Therefore, an excessive number of proposals is mandatory for inference. This increases computations and also halts performance, as many false positives are generated.

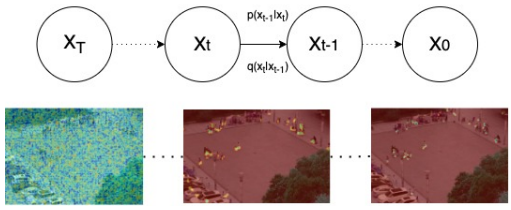

Figure 3. **Noise-to-Heatmap framework.** Start from random Gaussian noise, the heatmap is iteratively refined.

Therefore, our approach is to view objects as points, which go through a random walk positionally and categorically, and consider their probabilistic density function. In the forward process, each object randomly travels in $(height, width, class)$ space. For each position in $(height, width, class)$, it is equivalent to adding a small Gaussian noise. Thus, the forward process can be defined as injecting noise in the heatmaps - the probability distributions of objects.

Now the forward process can be defined as in training 1. At $t = 0$, the initial heatmap H_0 can be directly generated from the ground truth bounding boxes like in CenterNet, assigning high probability in $(height, width, class)$ when

Algorithm 1 Training

Input: total diffusion steps T, image features, ground truth heatmaps and bboxes $D = \{(I_k, H_k, B_k)\}_k^K$

repeat

Sample $(\mathbf{I}_i, \mathbf{H}_i) \sim D$

Sample $\epsilon \sim N(\mathbf{0}, \mathbf{I}_{h \times w \times c})$

Sample $t \sim Uniform(\{1, \dots, \mathbf{T}\})$

$\beta_t = 1 - \frac{\cos^2(\pi(t+1)/2(T+1))}{\cos^2(\pi t/2(T+1))}$

$\alpha_t = 1 - \beta_t$

$\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

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

$\hat{H}_i = normalize(H_i)$

$x_t = \sqrt{\bar{\alpha}_t} \cdot \hat{H}_i + \sqrt{1-\bar{\alpha}_t}\epsilon$

$x_0 = x_t - \epsilon_\theta(x_t, I_i, t)$

$\tilde{H}_0 = unnormalize(x_0)$

$x_{wh} = head_{wh}(x_0, I_i)$

$x_{offset} = head_{offset}(x_0, I_i)$

$L_{cls} = L_{focal}(\tilde{H}_t, H_0)$

$L_{wh} = L1(x_{wh}, B_i^{wh})$

$L_{offset} = L1(x_{offset}, B_i^{offset})$

$L_{det} = L_{cls} + \lambda_{wh} * L_{wh} + \lambda_{offset} * L_{offset}$

Take a gradient step on L_{det}

until convergence

there is an object of class $class$ near $(height, width)$ position. At $t = T$, the heatmap would be perfectly random, like Uniform distribution. As we are assuming the Gaussian noise injection, the heatmap H_i should be projected to be zero-centered and unit-variance. Therefore, $H_0 \in [0,1]^{(h \times w \times c)}$ is nomalized to $\hat{H}_0 = (H_0 * 2 - 1) * \sigma_{snr} \sim N(\mathbf{0}, \mathbf{I}^{(h \times w \times c)})$, where σ_{snr} is a signal-to-noise ratio.

At inference 2, we sample a Gaussian noise $x_T \sim N(\mathbf{0}, \mathbf{I}^{(h \times w \times c)})$, then compute x_0 with backward diffusion process. The predicted x_0 should be unnormalized to be a heatmap $\tilde{H}_0 = (x_0/\sigma_{snr} - 1)/2 \in [0,1]^{(h \times w \times c)}$.

We call this a *noise-to-heatmap* approach. This can work with an arbitrary number of objects, and this does not need the heuristic box-renewal process. Also, the class prior is considered when computing $\epsilon_\theta(x_t, I_i, t)$.

Detector head. The detector head directly follows that of CenterNet [6] and VFNet [11]. It takes image features as input and computes the heatmap $\tilde{Y}$, offset $\tilde{O}$, and object sizes $\tilde{S}$ for Centernet version, while the heatmap $\tilde{Y}$, offset $\tilde{0}$, and refined offset $\tilde{R}$ for VFNet. From a randomly noised heatmap, it iteratively refines to predict noise and the refined heatmap. This can be interpreted similarly to the coarse-to-fine approach - given an initial guess, finding more precise detections.

Feature fusion. In our approach, the detector head computes denoised heatmaps, conditioned with the previous heatmap and encoded image features. We can view this as computing successful heatmaps repeatedly, and therefore

Algorithm 2 Inference

Input: total diffusion steps T, image feature I

$x_T \sim N(\mathbf{0}, \mathbf{I}_{h \times w \times c})$

for $t = T, T-1, \dots, 1$ **do**

Sample $z \sim N(\mathbf{0}, \mathbf{I}_{h \times w \times c})$

$\beta_t = 1 - \frac{\cos^2(\pi(t+1)/2(T+1))}{\cos^2(\pi t/2(T+1))}$

$\alpha_t = 1 - \beta_t$

$\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

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

$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, I, t)) + \sigma_t z$

end for

$\tilde{H}_0 = unnormalize(x_0)$

$x_{wh} = head_{wh}(x_0, I)$

$x_{offset} = head_{offset}(x_0, I)$

Return Post-process $(\tilde{H}_0, x_{wh}, x_{offset})$.

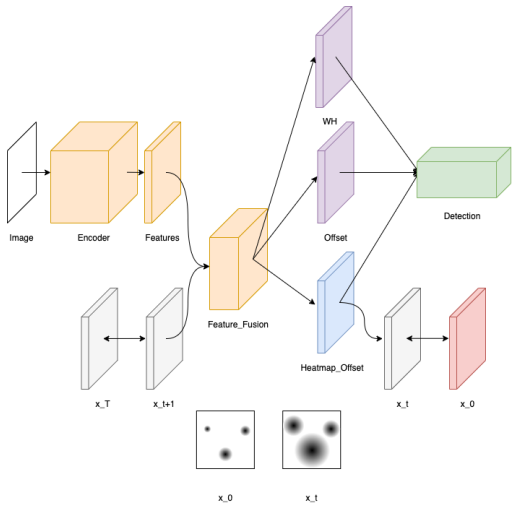

Figure 4. **Overall framework of Centernet-like head.** With the backbone image encoder, image features are extracted. The decoder head inputs those features and noised heatmap to predict a refined heatmap based on CenterNet.

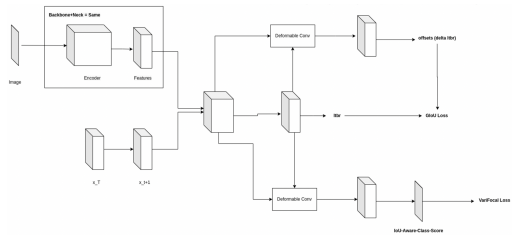

Figure 5. **Overall framework of VFNet-like head.** With the backbone image encoder, image features are extracted. The decoder head inputs those features and noised heatmap to predict a refined heatmap based on VFNet.

we can follow the structure of CenterTrack [31]. Different from tracking objects, our diffusion process lacks temporal consistency and spatial coherence, as random noise is repeatedly injected at every step. Therefore, we added the CBAM module [28] to handle this problem. After combining encoded features and the previous heatmap, three heads - heatmap head, offset head, and size head or heatmap head, offset head, and refined offset head- use the computed feature to generate predictions.

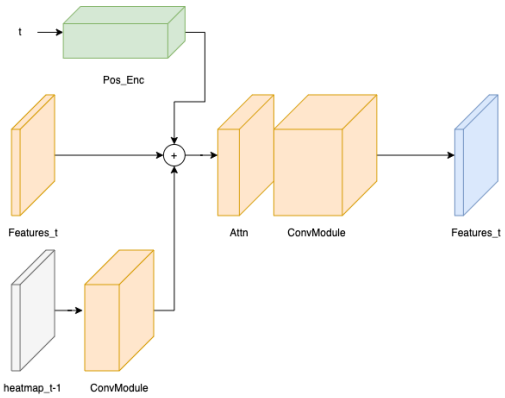

Figure 6. **Feature fusion module.** Incoming heatmap is added to encoded image features, passing through 1x1 convolution. Time embedding with randomly chosen t is added. The attention module followed by convolution layers computes the next feature to predict heatmap, sizes, and offsets.

4.2. Training

Forward Diffusion Step. Following Centernet [6], the ground truth heatmap $\hat{Y} \in [0,1]^{C \times \frac{W}{R} \times \frac{H}{R}}$ is generated, where (W, H) is the size of the image and R is the output stride, and C is the number of classes to predict. Sampled noise is added to the initial heatmap until the heatmap becomes a completely random distribution. The noise is scaled with α_t of time step t from decreasing cosine schedule, following [20].

We tried multinomial diffusion similar to [14] separately for the class dimension, but the result was not great. Therefore, we choose to follow the diffusion process of DDPM [13].

Training loss. Following the original CenterNet, we used focal loss [17] for the heatmap $\hat{Y}$ and used L1 loss for offset $\hat{O}$ and size $\hat{S}$. The total training objective is

$$L_{det} = L_{heatmap} + \lambda_{size} L_{size} + \lambda_{offset} L_{offset}. \quad (5)$$

4.3. Inference

Sampling step. We followed DDIM [26] for sampling. For every step, the predicted heatmap is used to estimate the heatmap of the proceeding heatmap. The predicted heatmap is directly used to generate detection results. Unlike DiffusionDet, we do not need a box renewal step, as we do not need to categorize each prediction into desired or undesired. This makes the sampling step more reasonable and corresponds with the original DDIM.

5. Experiments

we evaluated our approach on MS-COCO [18] and VisDrone [5]. We used VisDrone dataset to test our model on small objects. We used Resnet18 and Resnet50 for the image encoder, and CenterNet-based head for the detector head. The baseline was tested with implementation from MMDetection [3].

MS-COCO [18] Coco dataset is a widely used dataset in the field of object detection. It consists of 328,000 images containing everyday objects and 1.5 million object labels. We checked the box average precision over average IoU thresholds (AP), threshold 0.5 (AP_{50}).

VisDrone-DET2019 [5] VisDrone dataset contains images taken from drone-mounted-cameras. It contains 8.6K images and 540M labels. As the images are shooted from long range and high altitudes, the number of objects in one image is much larger compared to COCO, and also the sizes of the objects are relatively small.

5.1. Implementation Details

Training. We used ResNet18 pre-trained on ImageNet-1k. For the optimizer, we used Adam with the initial learning rate of 2.5×10^{-5} and weight decay of 10^{-4}, following DiffusionDet. The training step is a total of 450k iterations. We used only one GPU with a mini-batch size of 16. We used RandomResizedCrop of the size (512, 512) for our only augmentation.

Testing. After predicting a heatmap, the decoder head selects top-k points for inference. We used the top 100 points and the top 300 points, respectively, to generate a total of 100 predictions after NMS.

5.2. Experiment Result on MS-COCO

We compared our implementation with its baseline CenterNet [6] and VFNet [11] on table 1. Since we were not able to find an appropriate optimizer setting and the model is still in its training, the performance is not better than the baseline. However, with lower IoU, it outperformed the baseline by 1%. It is most probably due to the low convergence speed of the heatmap head, which is a common problem in CenterNet structure. Additionally, since COCO has a class imbalance problem, the heatmap style head might be not appropriate for this training.

5.3. Experiment Result on VisDrone-DET

We also conducted comparison experiments on the VisDrone validation set. The results are in table 2. As well as

Method	Backbone	AP	AP_{50}
CenterNet	ResNet18	25.9	42.6
VFNet	ResNet101	13.8	32.2
SSD	VGG16	26.8	46.5
DiffusionDet(reported)	ResNet50	45.5	65.1
DiffusionDet(reproduced)	ResNet50	32.0	49.0
DiffusionPoint(ours)	ResNet18	27.2	43.5
DiffusionPoint(ours)	ResNet50	26.7	49.5
DiffusionPoint(ours)	VFNet-ResNet101	11.6	28.8
DiffusionPoint(ours)	YoloXm	25.2	43.9
DiffusionPoint(ours)	YoloXl	25.2	44.3

Table 1. **MS-COCO validation set.** The CenterNet is the baseline from MMDetection [3] with ResNet18 backbone and without DCN. The others are our implementation. We applied no test-time augment.

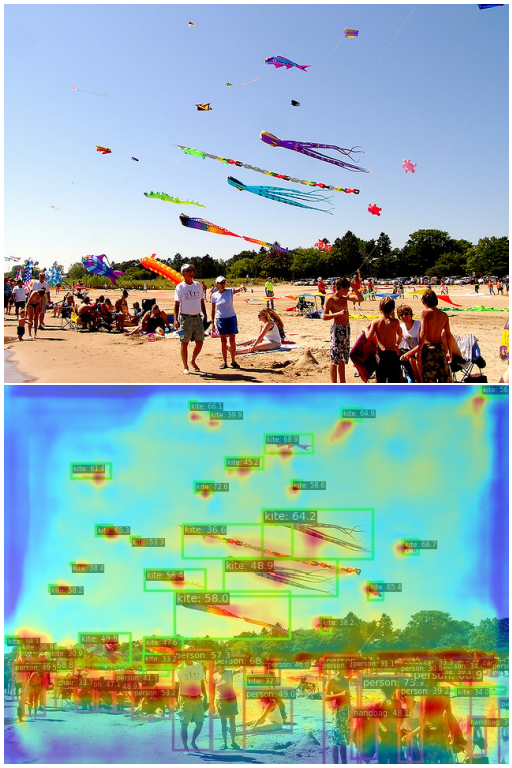

Figure 7. **Heatmap visualization on COCO.** The detector head finds small and dense objects as well.

Method	Backbone	AP	AP_{50}
CenterNet	ResNet18	24.7	49.22
CenterNet	ResNet50	28.13	59.51
RetinaNet [17]	ResNet50	13.9	23.0
FRCNN [22]	ResNet50	21.4	40.7
DiffusionDet	ResNet18	21.2	36.5
DiffusionPoint(ours)	ResNet18	13.3	30.3

Table 2. **VisDrone Validation set.** The CenterNet is the baseline from MMDetection [3] with ResNet18 backbone. The others are our implementation.

Model	Backbone	ms per image	FPS
CenterNet [6]	ResNet18	**8.1**	**123.4**
DiffusionDet [4]	ResNet18	37.7	26.5
DiffusionDet [4]	ResNet50	60.6	16.5
DiffusionPoint(ours)	ResNet18	10.2	98.0
DiffusionPoint(ours)	ResNet50	15.6	63.9
DiffusionPoint(ours)	YoloXm [7]	25.0	40.0
DiffusionPoint(ours)	YoloXl [7]	27.2	36.8
DiffusionPoint(ours)	VFNet-res50 [7]	33.9	29.5
DiffusionPoint(ours)	VFNet-res101 [7]	40.8	24.5

Table 3. **Inference speed.** The CenterNet is the baseline from MMDetection [3] and the others are our implementation. Single RTX3090 is used for inference.

in the COCO, we did not meet the performance of SOTA models, due to training issues. Remarkably, mAP at IoU 50 was much better than mAP from an average IoU of 0.5:0.95. This can be due to the slower convergence speed of the offset and size heads.

5.4. Speed test

We conducted an inference speed test between ours and the baselines. All baselines are modified to use the ResNet18 backbone, for a fair comparison. We applied no test-time augment. Compared to DiffusionDet, our approach was much faster, as we do not apply RoI pooling. Although using diffusion sampling for inference, our approach was not so much slower than the baseline.

5.5. Iterative Refinement

One of the advantages of adopting diffusion models is that the detection result can go through iterative refinement. In other detection models, no matter if they are one-stage or

Figure 8. **Iterative refinement.** The left figure is the visualization of heatmaps after a single inference. The right one is from the same image after 4 sequential inferences. Clearly, iteratively refining heatmaps made the result more clear and sharp.

two-stage, once the image features are used in the detector head, they are not used again and the detection result cannot be used as the next refinement stage. On the other hand, Diffusion models can adjust the number of denoising sampling steps to improve the quality of the output image. The above figure 8 shows the comparison of heatmap after single inference and multiple inferences. After repetitive refinement, the object proposal got more clear and more accurate. Using this, our architecture can accelerate the inference speed or iteratively refine detection results, similar to the coarse-to-fine approach. For example, for a high-resolution image, first downsize and compute a heatmap. Then resize and crop the computed heatmap, and use it as a noised heatmap for partial search. This can be applied to many other tasks like tracking [31], or hierarchical object search for large images, like in IterDet [23].

5.6. Limitations

One of the main problems to solve is that our model generates small objects which hurts mAP performance. Therefore, we need to figure out ways to filter those issues. Another problem is that the regression head doesn't converge but only the heatmap head. We also need to figure out this problem in the future. Finally, since we had lack of time and computational resources, we only used single GPU based training, resulting in decreased performance (compare Tab. 1 DiffusionDet reported vs. reporducted).

6. Conclusion and Future Works

In this work, we present a new one-stage detection model, DiffusionPoint, by applying the diffusion process to the heatmap. Through this, we showed that the current DiffusionDet, which is proposal-dependent and has a heuristic box renewal process could be implemented based on one-stage heatmap based Detector.

We experimented on famous object detection datasets such as MS-COCO [18] and VisDrone [5] and showed possibility of our approach. Although the performance of our framework was not good as expected, there is still room to improve. Moreover, this approach can be adapted to many other models.

In future work, we should try our approach in other datasets such as LVIS [10]. Last but not least, we could experiment with our model not only on object detection but also on other tasks, like key point estimation or tracking.

References

[1] Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. *arXiv preprint arXiv:2112.03126*, 2021. 1, 2

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. 2, 3

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5, 6

[4] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection, 2022. 1, 2, 3, 6

[5] Dawei Du, Pengfei Zhu, Longyin Wen, Xiao Bian, Haibin Lin, Qinghua Hu, Tao Peng, Jiayu Zheng, Xinyao Wang, Yue Zhang, et al. Visdrone-det2019: The vision meets drone object detection in image challenge results. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019. 5, 7

[6] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Int. Conf. Comput. Vis.*, 2019. 1, 2, 3, 4, 5, 6

[7] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021, 2021. 6

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

[9] 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

[10] 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. 7

[11] Feras Dayoub Niko Sünderhauf Haoyang Zhang, Ying Wang. Varifocalnet: An iou-aware dense object detector. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'21*. 2, 3, 4, 5

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 1

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 1, 2, 3, 5

[14] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021. 5

[15] Yuming Jiang, Shuai Yang, Haonan Qiu, Wayne Wu, Chen Change Loy, and Ziwei Liu. Text2human: Text-driven controllable human image generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022. 1, 2

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[17] 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, 5, 6

[18] 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 *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 5, 7

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. 1

[20] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 5

[21] 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

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 6

[23] Danila Rukhovich, Konstantin Sofiiuk, Danil Galeev, Olga Barinova, and Anton Konushin. Iterdet: iterative scheme for object detection in crowded environments. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, S+ SSPR 2020, Padua, Italy, January 21–22, 2021, Proceedings*, pages 344–354. Springer, 2021. 7

[24] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 2

[25] 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. 2

[26] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 3, 5

[27] 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. 3

[28] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 5

[29] Philipp Krähenbühl Xingyi Zhou, Vladlen Koltun. Probabilistic two-stage detection. In *archive*, 2021. 3

[30] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022. 3

[31] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV*, pages 474–490. Springer, 2020. 5, 7

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

[33] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023. 1

Appendix A.

Figure 9. **Qualitative Visualizations on COCO**

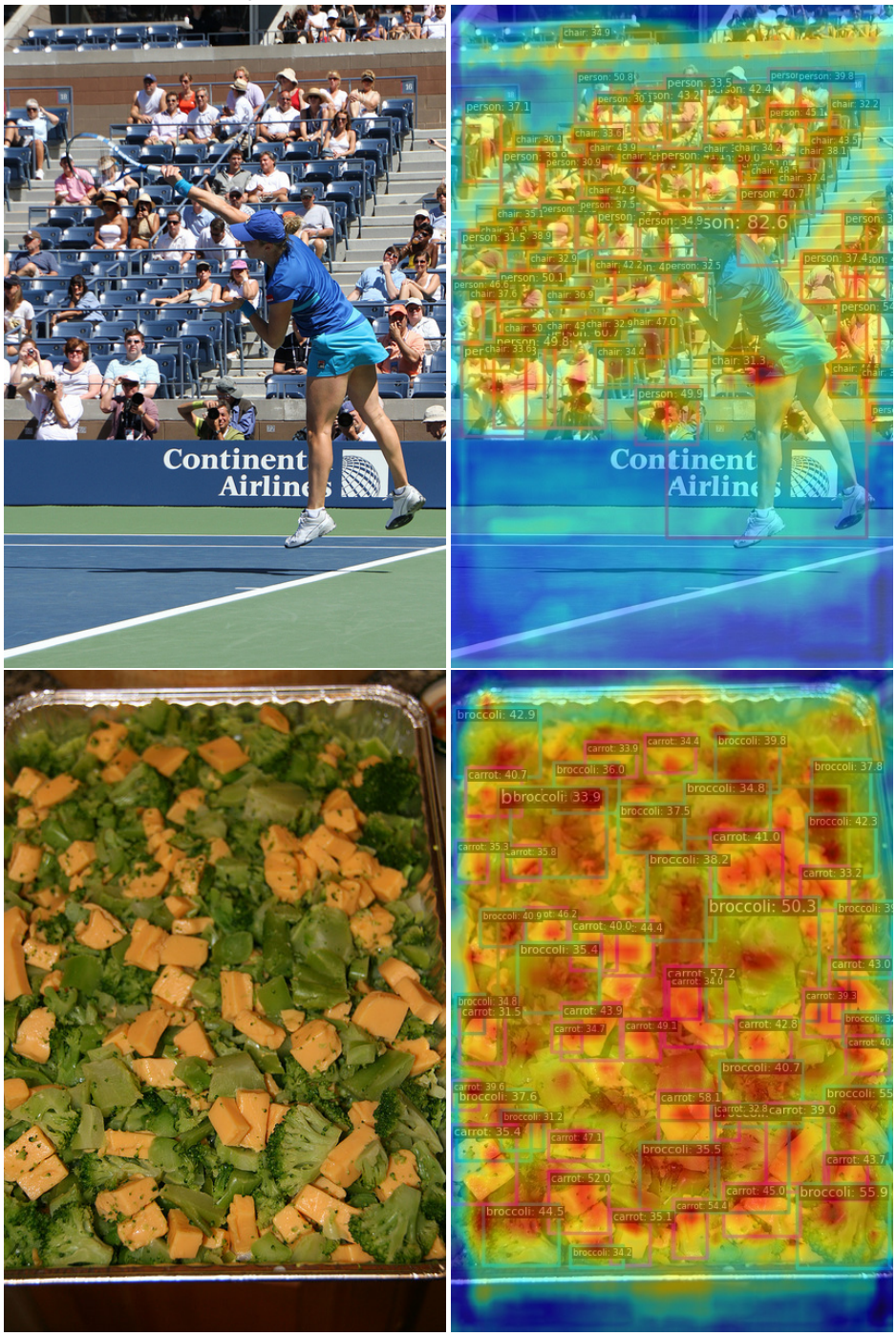

Figure 10. **Qualitative Visualizations on VisDrone**