DiffusionDet: Diffusion Model for Object Detection

Pavol Harar[1,2] 0000-0001-5206-1794

Deep Learning Seminar @dsUniVie

Vienna, 12. April 2023

[1]University of Vienna, [2]Brno University of Technology

Paper info

Authors: Shoufa Chen, Peize Sun, Yibing Song, Ping Luo

ArXiv preprint: 2211.09788

Repository: ShoufaChen/DiffusionDet with 1.7k stars and over 120 forks

Please note that the peer-reviewed version of this preprint (if exists) is not yet public.

Abstract

- DiffusionDet: New framework for object detection
- Formulates object detection as a denoising diffusion process
- During training: Object boxes diffuse from ground-truth to random distribution
- Inference: Model refines randomly generated boxes progressively
- Outperforms previous detectors on standard benchmarks
- Finding: Random boxes can be effective object candidates
- Finding: Object detection can be solved generatively

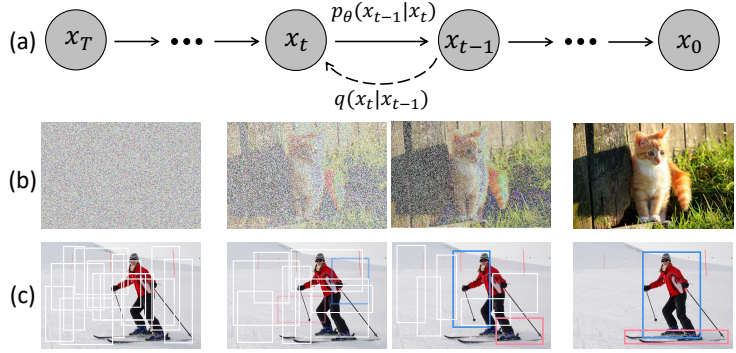

Figure 1: Diffusion model for object detection. (a) A diffusion model where q is the diffusion process and p_θ is the reverse process. (b) Diffusion model for image generation task. (c) We propose to formulate object detection as a denoising diffusion process from noisy boxes to object boxes.

Main contribution

- The first study to apply diffusion to object detection.
- Formulation of object detection as a generative denoising process.
- Provide extensive experiments on MS-COCO and LVIS benchmarks.
- Achieving favorable performance against previous well-established detectors.

Microsoft COCO

- MS COCO: Large dataset for object detection, segmentation, and captioning tasks
- 330,000+ everyday scene images with annotated object labels, masks, and keypoints
- More than 2 million annotated object instances
- Diverse in categories, styles, and scales
- Standard benchmark in computer vision research and competitions

LVIS

- LVIS: Large Vocabulary Instance Segmentation dataset.
- Developed by FAIR for object detection and instance segmentation tasks.
- Contains over 290,000 images with 2.9 million object instances.
- Annotated with fine-grained object categories, masks, and object relationships.
- Emphasizes on rare and fine-grained categories.
- Encourages research on long-tailed distribution of object instances in real-world.
- Benchmark for state-of-the-art object detection and instance segmentation models.

Method

Method

Objective: 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 (BB) and category labels for objects in the image $\boldsymbol{x}$, respectively.

Bounding-box: The i-th BB 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.

Data samples: 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.

Approach: 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.

Method

The forward noise process:

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

transforms data sample $\boldsymbol{z}_0$ to a latent noisy sample $\boldsymbol{z}_t$ for $t \in \{0, 1, ..., T\}$ by adding noise according to a noise schedule.

Training objective ℓ_2 loss:

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

Inference stage: Data sample $\boldsymbol{z}_0$ is reconstructed from noise $\boldsymbol{z}_T$ with the model f_θ and an updating rule in an iterative way, i.e., $\boldsymbol{z}_T \to \boldsymbol{z}_{T-\Delta} \to ... \to \boldsymbol{z}_0$.

Encoder decoder split

- Diffusion model generates data samples iteratively
- Running model f_θ multiple times at inference is comput. intractable on raw image
- Proposed solution: Separation of model into - image encoder and detection decoder
- Image encoder runs only once to extract deep feature representation from raw image $\boldsymbol{x}$
- Detection decoder refines box predictions from noisy boxes $\boldsymbol{z}_t$ using the deep feature representation as condition, instead of $\boldsymbol{x}$.

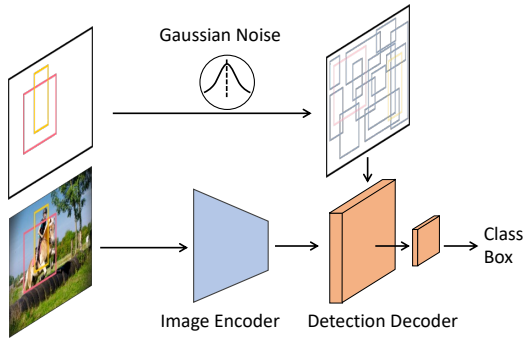

Figure 2: DiffusionDet framework. The image encoder extracts feature representation from an input image. The detection decoder takes noisy boxes as input and predicts category classification and box coordinates. During training, the noisy boxes are constructed by adding Gaussian noise to ground-truth boxes. In inference, the noisy boxes are randomly sampled from the Gaussian distribution.

Image encoder (IE)

Feature Pyramid Network is used to generate multi-scale feature maps for both ResNet (CNN) and Swin (Transformer-based) backbones.

Detection decoder (DD)

- Detection decoder borrowed from Sparse R-CNN
- Uses a set of input proposal boxes to crop RoI-features from IE feature map
- RoI-features go to detection head to obtain box regression and classification results
- Composed of 6 cascading stages, following prior work

Detection decoder (DD)

Differences between DiffusionDet decoder and Sparse R-CNN:

- DD begins from random boxes in inference, while Sparse R-CNN uses a fixed set of learned boxes
- Sparse R-CNN takes pairs of proposal boxes and corresponding proposal features as input, while DD only needs proposal boxes
- DD re-uses detector head in iterative sampling steps with shared parameters, each step specified by timestep embedding, while Sparse R-CNN uses the detection decoder only once in the forward pass.

Training & Inference

Algorithm 1 DiffusionDet Training

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

  # Encode image features
  feats = image_encoder(images)

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

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

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

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

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

  return loss
```

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

Algorithm 2 DiffusionDet Sampling

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

  # Encode image features
  feats = image_encoder(images)

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

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

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

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

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

    # Box renewal
    pb_t = box_renewal(pb_t)

  return pb_pred
```

`linespace`: generate evenly spaced values

Training & Inference

For more details head over to the paper page Section 3.3 and 3.4.

Results - Performance comparison

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

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

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

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

Conclusion

- DifusionDet is a novel object detection paradigm using denoising diffusion process
- DifusionDet has dynamic box and progressive refinement properties for speed-accuracy trade-off
- Shows favorable performance compared to established detectors
- Future work: Video-level tasks like object tracking and action recognition
- Future work: Extending to open-world or open-vocabulary object detection

Pavol Harar

pavol.harar.eu

HararPavol

Received an MSc in System Engineering and Informatics and a PhD in Machine Learning from Brno University of Technology. Gained experience in predictive modeling, signal processing, and parallel computing as a member of Brain Diseases Analysis Laboratory and Numerical Harmonic Analysis Group. At the time of presentation a postdoc at the Data Science Research Network @UniVie and a visiting postdoc at the Institute of Molecular Pathology in Vienna.