Topic G - Object detection and tracking with DiffusionDet

Antoine Ratouchniak

21/01/2023

Abstract

Diffusion models [1] have shown very impressive results with applications such as text-to-image with Imagen [2], DALL · E 2, image generation [3] or high-quality image restoration [4]. In addition, object tracking is a field of computer vision that has been extensively studied for academic and industrial purposes [12].

1 Introduction

In this work, we propose the use of diffusion models for object detection and tracking. We summarize how DiffusionDet [9] works and we try to do some experiments to evaluate its performance on images and videos. We also try to implement SIFT [13] as an object tracker system.

2 Diffusion model

2.1 Reminder on diffusion models

Diffusion models [1] work by noising the data and denoising it in a specific way in order to reconstruct/edit it. The model can be broken down into two stages: the forward diffusion process and the reverse diffusion process. The first stage noises the data (blurs if it is an image) as follows: we let $x_0 \sim q(x_0)$ be the original data distribution and $(x_1, ...x_T)$ be the data produced by the forward diffusion process at step $t \in \{1, ...T\}$. We introduce $(\beta_1, ..., \beta_T)$ the variance scheduler rates and $q(x_t|x_{t-1})$ a conditional probability distribution. The probability distribution associated with the forward diffusion process is the distribution $q(x_t|x_{t-1}) \sim \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t \boldsymbol{I})$. Finally, we define the data $(x_1, ..., x_T)$ as $x_t \sim q(x_t|x_{t-1})$. This stage can be done in one step with the reparameterization trick [5] giving $q(x_t|x_0) \sim \mathcal{N}(\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha})\boldsymbol{I})$ by setting $\overline{\alpha}_t = \prod_{i=1}^{T} \alpha_i$ and $\alpha_t = 1 - \beta_t$ (2.1.1). The second stage reconstructs the original data from the noisy data. Starting from $x_T \sim p(x_T)$ where $p(x_T) \sim \mathcal{N}(0, \boldsymbol{I})$, we are looking for x_0. We introduce the conditional probability distribution $p(x_{t-1}|x_t) \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ where μ_θ and Σ_θ are learned by the network. Skipping the calculations, with Bayes' rule, and to avoid getting a too high variance, we condition q with x_0 to get the a posteriori distribution as follows: $q(x_{t-1}|x_t, x_0) \sim \mathcal{N}(\tilde{\mu}_\theta(x_t, x_0), \tilde{\beta}_\theta \boldsymbol{I})$ where $\tilde{\mu}_\theta(x_t, x_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}}x_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}x_t$ and $\tilde{\beta}_\theta = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$. For the training, the model minimizes the variational lower bound instead of $p_\theta(x_0)$ which is not directly computable. By setting good values for T and $(\beta_1, ..., \beta_T)$ (generally increasing with a cosine rate [6]), we can reconstruct the data. Obviously, at the inference stage, we do not have x_0, this is why we set $x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\epsilon_t)$ thanks to (2.1.1) where ϵ_t is the network architecture at step t. The architecture used for the reverse diffusion process is generally a U-Net [7] with cross-attention layers [8]. Finally, the network has to learn $\tilde{\mu}(x_t, x_0)$ if we suppose Σ_θ is fixed, since we know all the β_t.

2.2 Adaptation to DiffusionDet

Inspired by diffusion models, DiffusionDet [9] uses the same process to detect objects bounding boxes. Indeed, starting from ground truth boxes, the bounding boxes' coordinates are sampled from a Gaussian distribution $\mathcal{N}(0, 1)$. Then, following similar steps as diffusion models 2.1, DiffusionDet refines bounding boxes and keeps the best ones. However, there are some differences we need to mention. DiffusionDet is decomposed into two parts, the first part, the encoder, extracts high-level features, and creates multi-scale versions of the image with Feature Pyramid Network [10]. The second part, the

decoder, takes as inputs the features from the encoder, which are bounding boxes and crops region of interest (RoI). These RoI are sent to the head of the network iteratively 6 times to refine the bounding box coordinates and classify the object if there is one. At inference stage, since ground truth boxes are unknown, random boxes with coordinates following a Gaussian distribution are generated. Finally, the reverse diffusion process is applied. The bounding box coordinates are sent 6 times to the head and this process can be done multiple times if we modify the sampling timesteps.

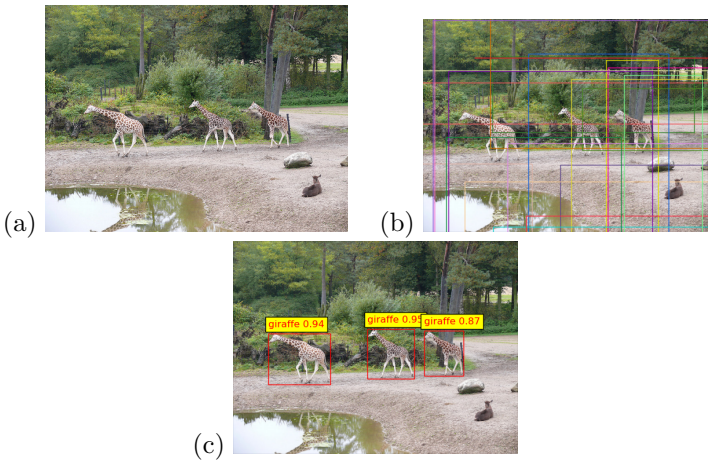

Figure 1: (a): input image, (b): random bounding boxes sampled from a Gaussian, (c): final prediction

Steps \ Precision	AP	AP_{50}	AP_{75}	AP_s
DiffusionDet (1 step)	46.5	66.3	50.0	30.0
DiffusionDet (4 steps)	46.9	66.9	50.4	30.6
DiffusionDet (8 steps)	47.1	67.1	50.5	30.4
	AP_m	AP_l		
DiffusionDet (1 step)	49.5	62.1		
DiffusionDet (4 steps)	49.6	62.6		
DiffusionDet (8 steps)	50.0	62.5		

Table 1: Precision metrics on MS-COCO validation dataset with different timesteps with DiffusionDet

The results are very close to the paper. Indeed, since bounding boxes are randomly generated, the results might be a bit different. One thing we can notice is that the inference stage according to the number of box proposals is linear.

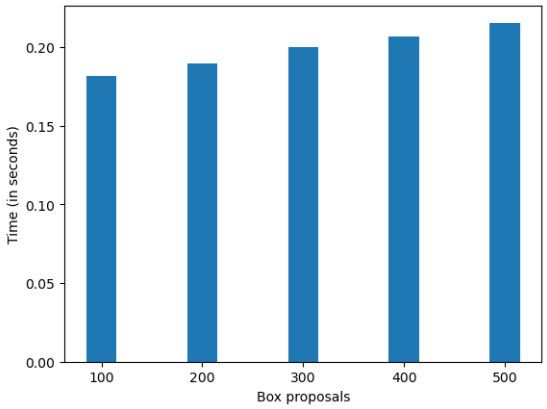

Figure 2: Execution time for different number of box proposals on a 480x640 image with a GeForce RTX 2060

3 Experiments

3.1 Object detection

The first thing we do is to evaluate DiffusionDet on MS-COCO validation dataset [11] to see if we get similar results as the article. We use DiffusionDet weights trained on the MS-COCO training dataset with Resnet-101 as a backbone.

3.2 Object tracking

We remind the two scores to measure an object tracking system:
$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDSW_t}{\sum_t |GT_t|}$ and $MOTP = \frac{\sum_t d_{t,i}}{\sum_t c_t}$
where FN = False Negative = object not detected, FP = False Positive = object that should not be detected, and $IDSW$ = Identity Switch = confusion between two objects or if the ID is not the same between two frames when it should have been the same. Finally, $|GT|$ denotes the number of bounding boxes in the frame and c is the number of matches

while t denotes the frame number. However, in the MOT16 dataset [12], we do not know the identities of objects, therefore, we use $MOTA = 1 - \frac{\sum_t FN_t + FP_t}{\sum_t |GT_t|}$. A question one might ask is whether the number of sampling timesteps will also increase $MOTP$ and $MOTA$ in the same way as the different average precisions for object detection. Here are the results for different sampling timesteps and different numbers of box proposals.

Steps \ Box proposals	500	400	300	200	100
DiffusionDet (1 step)	63.8	63.4	62.6	59.8	35.7
DiffusionDet (4 steps)	63.7	63.5	62.4	59.7	36.0
DiffusionDet (8 steps)	**63.9**	63.2	62.4	60.1	35.5

Table 2: $MOTA$ metric on MOT16 dataset with different timesteps and box proposals with DiffusionDet

In comparison, here are the results with state-of-the-art trackers according to the MOT challenge website.

Tracker	MOTA
1. ppbytemot	77.7
2. GMOTv2	76.6
3. VAITracker2	75.9
4. SacMOT	75.8
5. STRTrach	75.2

Table 3: $MOTA$ metric on MOT16 dataset with different trackers

Obviously, the comparison is biased since we removed the $IDSW$ from the calculations.

The $MOTP$ metric with DiffusionDet gave almost the same results ($\approx$ 73.9) for all the cells and is therefore not relevant to add. The results for the $MOTA$ metric are all quite similar even when the number of timesteps increases. Contrary to what we expected, it seems that the number of timesteps does not increase the accuracy. One of the reasons could be that the MOT16 dataset contains a lot of objects, therefore leading to random bounding boxes to find an object with a high probability.

One other question one may ask is if starting from a prior distribution at a frame t would help to detect objects at a frame $t+1$. However, we need to fix a proportion of prior boxes. Keeping 1 timestep, we get the following results.

Prior prop. \ Box prop.	500	400	300	200	100
50%	**65.9**	64.5	61.7	54.9	27.7
75%	63.9	59.0	54.1	43.6	19.0
95%	35.5	29.0	23.9	15.9	5.0

Table 4: $MOTA$ metric on MOT16 dataset with different prior proportions and box proposals with DiffusionDet

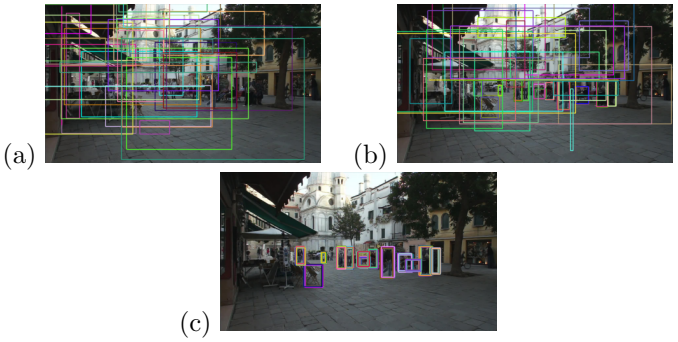

Figure 3: Example of a priori boxes between two consecutive frames; (a): random boxes at frame t, (b): random boxes + a priori boxes on frame $t+1$, (c): a priori boxes only on frame $t+1$

Once again, $MOTP$ scores are not relevant to add as they are all extremely close by row ($\approx 73.9, 73.6, 73.9$). We notice that we get a new higher $MOTA$ score with 500 boxes and 50% of the boxes set to boxes from the previous frame. In addition to that, using 400 boxes and 50% prior still give a better $MOTA$ score than without prior distribution, saving 100 boxes. This is a really good improvement even if the process is linear in terms of the number of boxes. We could also expect to get similar improvements with other backbones. However, we clearly see a rapid decrease in accuracy when the prior proportion increases. This is probably due to a bad trade-off between exploration and exploitation. Indeed, it is important to still have some boxes able to detect new objects on the video.

Finally, we could also ask ourselves how many frames the previous box prior might be worth keeping. We got almost the same results for $t+1$, $t+5$, $t+10$, and $t+20$ frames so we decide not to report the result here. The results can be found in the notebook.

3.3 SIFT implementation

The object tracking system implemented with DiffusionDet is known as Jaccard index. The way it works is that we consider that two objects are the same from a frame t and $t+1$ if the two classes are the same and the intersection over union is greater than a threshold, 0.6 in DiffusionDet. In this subsection, we show how we tried to implement SIFT [13] for object tracking. This approach has already been implemented in other studies ([14], [15], [16]). The idea is to consider two objects identical from a frame t to $t+1$ if there are of the same class and they share a good "similarity score". It works as follows: for each untracked object by the tracking algorithm implemented with DiffusionDet, we compute its SIFT descriptors. If an object has not found a match from an object in the previous frame, we check among objects that did not find a match either. If two objects are of the same class, we compare their descriptors using Lowe's second nearest neighbor. The algorithm returns the nearest 2 neighbors obtained with the L^1 metric and we finally decide to consider a match if the ratio between the best match and the second best match is below a threshold. We consider the object is the same from a frame t to $t+1$ if they share enough matches (potentially more matches for a big object and less for a little object). In the example below, if we denote $matches$ the total number of matches, we consider two objects are the same if $matches > min(length(old\_keypoints), length(keypoints)$ where $old\_keypoints$ indicates the number of keypoints in the previous object and keypoints the number of keypoints in the actual detected object. Even though this method seems interesting, it is difficult to make it robust. Indeed, parameters such as the contrast threshold, the edge threshold, the metric to measure distances between descriptors, the threshold for the ratio test, and the number of matches to consider the same object... make it difficult to parameterize.

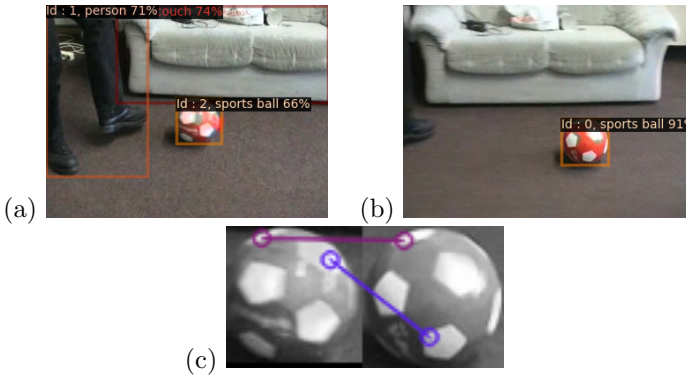

Figure 4: Example of SIFT use between two frames from BoBoT benchmark dataset [17]; (a): detections at frame t, (b): detection at frame $t+1$, (c): SIFT matches between the ball from frame t and $t+1$

4 Conclusion

This work proposed the use of diffusion models for object detection and tracking. DiffusionDet gives really good results on object detection and tracking. We were even able to improve the $MOTA$ score by using prior distributions. In addition, we tried to implement SIFT for object tracking, which would deserve further investigations. However, DiffusionDet is computationally too expensive as we can only run 5 FPS with 500 boxes on a GeForce RTX 2060.

References

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in Neural Information Processing Systems 33 (2020): 6840-6851.

[2] Saharia, Chitwan, et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." arXiv preprint arXiv:2205.11487 (2022).

[3] Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in Neural Information Processing Systems 34 (2021): 8780-8794.

[4] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[5] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[6] Nichol, Alexander Quinn, and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models." International Conference on Machine Learning. PMLR, 2021.

[7] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[8] Chen, Chun-Fu Richard, Quanfu Fan, and Rameswar Panda. "Crossvit: Cross-attention multi-scale vision transformer for image classification." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[9] Chen, Shoufa, et al. "Diffusiondet: Diffusion model for object detection." arXiv preprint arXiv:2211.09788 (2022).

[10] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[11] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

[12] Luo, Wenhan, et al. "Multiple object tracking: A literature review." Artificial Intelligence 293 (2021): 103448.

[13] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[14] Zhou, Huiyu, Yuan Yuan, and Chunmei Shi. "Object tracking using SIFT features and mean shift." Computer vision and image understanding 113.3 (2009): 345-352.

[15] Deori, Barga, and Dalton Meitei Thounaojam. "A survey on moving object tracking in video." International Journal on Information Theory (IJIT) 3.3 (2014): 31-46.

[16] Kim, Young Min. "Object tracking in a video sequence." CS 229 (2007): 366-384.

[17] Dubuisson, Séverine, and Christophe Gonzales. "A survey of datasets for visual tracking." Machine Vision and Applications 27.1 (2016): 23-52.