Scene-Aware Location Modeling for Data Augmentation in Automotive Object Detection

Anonymous ICCV submission

Paper ID 13838

Abstract

Generative image models are increasingly being used for training data augmentation in vision tasks. In the context of automotive object detection, methods usually focus on producing augmented frames that look as realistic as possible, for example by replacing real objects with generated ones. Others try to maximize the diversity of augmented frames, for example by pasting lots of generated objects onto existing backgrounds. Both perspectives pay little attention to the locations of objects in the scene. Frame layouts are either reused with little or no modification, or they are random and disregard realism entirely. In this work, we argue that optimal data augmentation should also include realistic augmentation of layouts. We introduce a scene-aware probabilistic location model that predicts where new objects can realistically be placed in an existing scene. By then inpainting objects in these locations with a generative model, we obtain much stronger augmentation performance than existing approaches. We set a new state of the art for generative data augmentation on two automotive object detection tasks, achieving up to $2\times$ higher gains than the best competing approach ($+1.4$ vs. $+0.7$ mAP boost). We also demonstrate significant improvements for instance segmentation.

1. Introduction

Generative Data Augmentation describes the use of generative models to create synthetic data that extends the training corpus of a learning model. The appeal of "free" training data has long motivated related work [3, 17], but with the recent progress in large generative image models [20, 34, 35] the interest in this field has increased drastically, with promising successes in image classification [19, 50] and object detection [15, 36, 49]. This includes automotive scenes [6, 14], the focus of this work, where the benefit of generative data augmentation is especially large, as edge case scenarios are often safety-critical and costly to acquire. Existing methods for training data augmentation usually concern themselves with improving the quality of generated objects, but they often neglect reasoning about their locations. Some approaches reuse *original* object locations from real frames [14, 27, 36], possibly with minor modifications [6, 38], which results in augmented frames that are visual variations of the same scene. Alternatively, other methods add new objects in *random* locations [49], completely ignoring the original scene composition, which results in unrealistic generations (Fig. 1).

In this work, we argue that object locations should also be considered a key component of data augmentation. To demonstrate this, we propose a scene-aware probabilistic location model that, given an existing scene, predicts where a new object should be placed. Specifically, our model parses the scene to extract depth and drivable space, and it factorizes the joint probability of object categories, their locations, and their dimensions into a series of simpler conditional densities, which can be sampled from with ancestral sampling. We then combine our location model with an inpainting diffusion model [34] to render objects in the predicted locations, yielding augmented frames that are both realistic and different from existing scenes. The result is a generative data augmentation technique that outperforms state-of-the-art approaches by a large margin, with a performance boost of up to $2\times$ w.r.t. the best competing approach ($+1.4$ vs. $+0.7$ mAP boost). By modifying the inpainting model to produce both RGB and instance masks, we further demonstrate substantial gains in the instance segmentation setting.

In summary, our contributions are the following:

- We propose a scene-aware probabilistic location model that augments street scene layouts by placing new objects in realistic locations.
- We combine our location model with an inpainting diffusion model to produce augmented frames for object detector training, where our performance boost is up to $2.8\times$ higher compared to state-of-the-art approaches.
- By enabling the diffusion model to predict instance masks for generated objects, we further demonstrate substantial performance gains on instance segmentation.

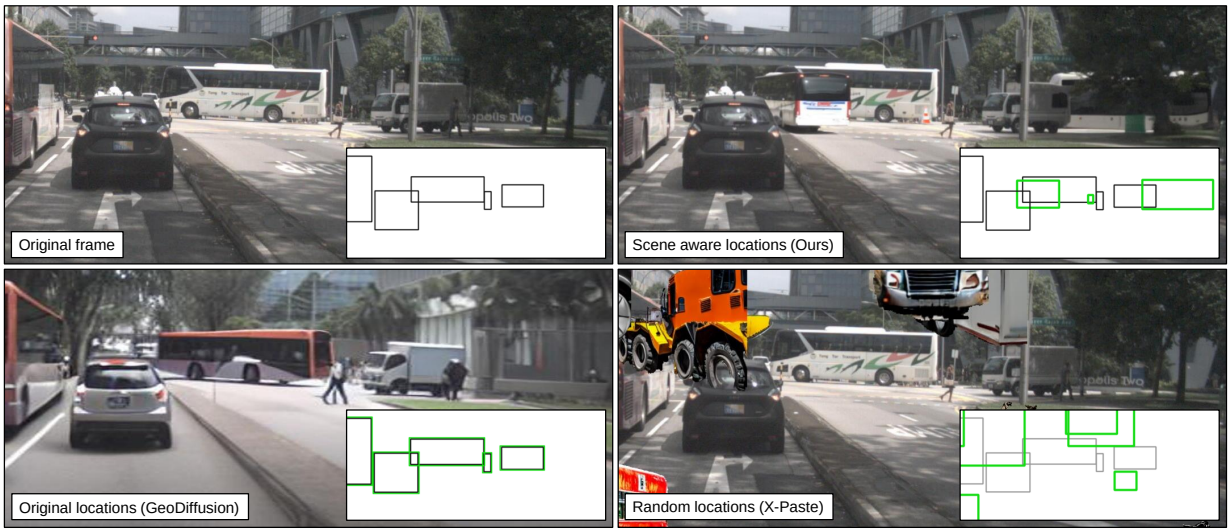


Figure 1. An original scene (top left) and three augmented frames using different location modeling and augmentation strategies. Generated objects are indicated by green bounding boxes. Our approach proposes locations that fit the original scene, resulting in novel compositions with high visual realism and challenging occlusion cases. Approaches that reuse original locations, even with minor modifications such as in GeoDiffusion [6], generate frames with visual appearance diversity but limited location diversity. Approaches that add objects in random locations such as X-Paste [49] disregard the realism of the resulting layout and, in turn, of the generated frames.

2. Related Work

Generative data augmentation. The use of synthetic instances to augment training data for vision tasks has recently become a common strategy [1, 15, 39, 48]. For object detector augmentation specifically, synthetic objects need to be rendered precisely in specified locations in a frame. Most works reuse object locations from real data [27, 36], change only the background while leaving objects intact [30], or perform minor modifications such as translation or removal of bounding boxes [6, 13, 14]. The focus of these works is therefore mostly on improving the realism of generated data.

Another popular augmentation strategy is "cut-and-paste" [11]: placing segmented or generated objects in real backgrounds in random locations. Cut-and-paste approaches have shown to be very effective in both object detection and instance segmentation [12, 15, 40, 49], despite the low realism of the resulting image. These results indicate that adding new objects in new locations may be just as important as creating realistic images, echoing earlier findings [10]. Our work achieves both by adding *new* objects in *realistic* locations to augment frames.

Layout generation and location modeling. Predicting object locations is related to layout completion and generation. Dedicated methods typically solve these tasks by modeling interactions at the bounding box level without additional context [16, 22, 23, 26, 41], and are usually applied to design documents or other highly structured data. Approaches that take scene information into account to determine object locations do exist [45, 51], but often require paired training datasets of (empty) images and feasible object placements, which are not readily available for automotive scenes. They may also require an image of the segmented object to place [46, 51], or have only been shown to work for specific object categories such as cars and pedestrians [29]. In this work, we instead want to determine the location before such an object is available. Finally, some approaches reason about object locations in 3D space [9, 24, 37], but this requires detailed 3D annotations, which are usually much harder to obtain than 2D annotations.

3. Method

Our goal is to augment street scenes by determining where new objects can be placed. Using these locations, we augment frames for detector training, as illustrated in Fig. 2. We first describe our proposed scene-aware location model, a probabilistic approach that factorizes the likelihood of new object locations into a sequence of simple conditional likelihoods. We then describe our strategy to render these objects into the existing scene to obtain augmented frames.

3.1. A factorized scene-aware location model

Each generated object is described by a class label c and a 2D bounding box b specifying its location and dimensions in the given scene. The procedure for placing a new object into an existing scene can be thought of as a two-step process: 1) decide *what* object to place and 2) determine *where* to place it (and what size it should have).

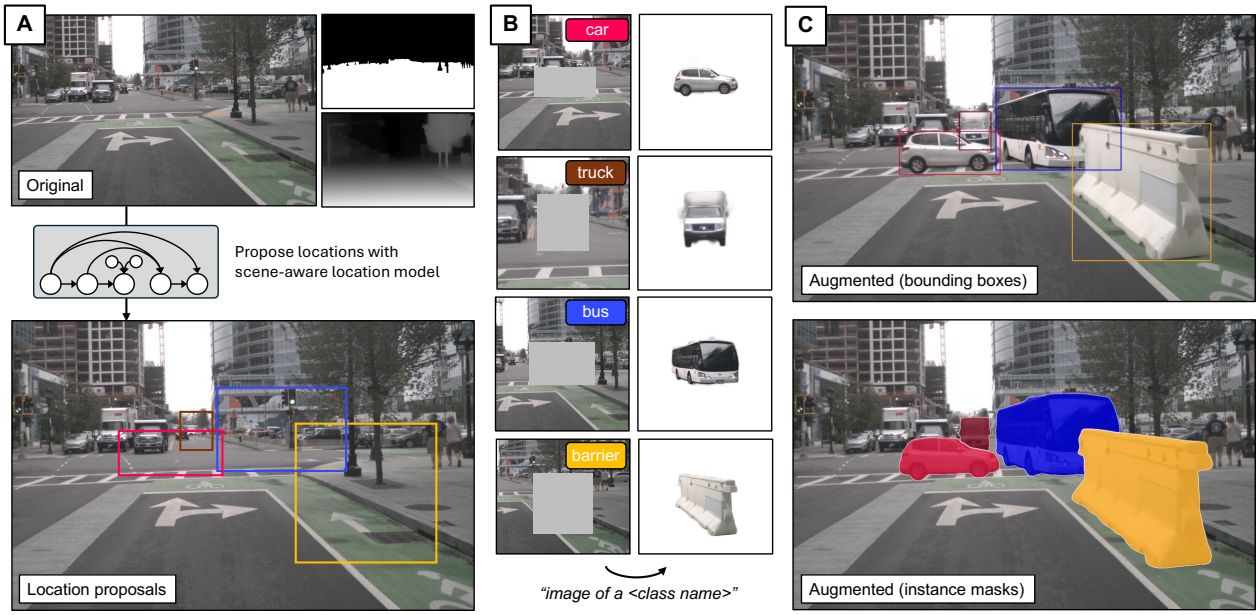


Figure 2. Overview of our augmentation pipeline. (A) We first use the location model to predict realistic bounding box locations for new objects, using depth and drivable space segmentation. (B) We then generate an object and corresponding instance mask using an inpainting model. (C) This allows us to create pseudo-labels for object detection and instance segmentation. Our approach scales to high resolution images, and creates realistic and challenging occlusion cases.

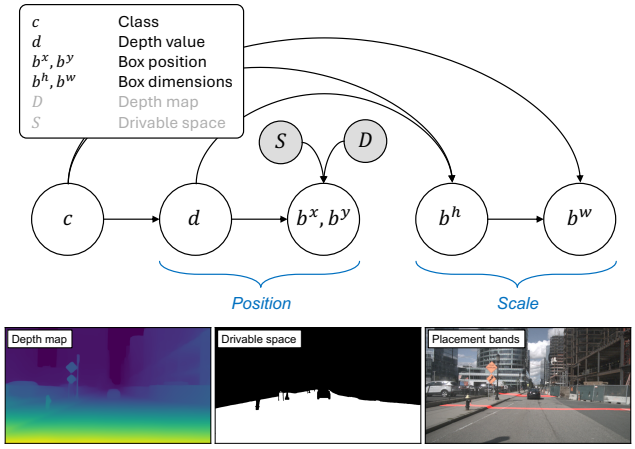


Figure 3. (Top) Our location model factorizes object placement into a series of conditional likelihoods, each of which is easy to approximate or parametrize. (Bottom) We sample a desired distance to the object, d, and determine admissible locations for this depth (red lines are two separate examples of such *placement bands*).

We make these choices explicit in a likelihood model $\hat{p}$ that approximates the true probability density p of object categories, locations and scales. Since the distribution of plausible object placements is highly dependent on the constraints imposed by the given scene, we condition our model on high level scene descriptions. Specifically, we assume to have scene representations in the form of a depth map $\mathbf{D}$ and a *drivable space* semantic map[1] $\mathbf{S}$. The depth map tells us something about the structure of the 3D scene and distance to existing objects, while the semantic map tells us exactly where the ground plane is.

We are now interested in predicting plausible objects and their locations. We approximate the likelihood of class c, the distance to the object d, and bounding box center, height and width (b^x, b^y, b^w, b^h) by factorizing it as follows:

$$p(c, b^x, b^y, b^w, b^h, d \,|\, \mathbf{D}, \mathbf{S}) \approx \\ \hat{p}(b^w|b^h, c)\, \hat{p}(b^h|d, c)\, \hat{p}(b^x, b^y|d, \mathbf{D}, \mathbf{S})\, \hat{p}(d|c)\, \hat{p}(c), \quad (1)$$

where d is a sampled depth value, used only as an intermediate variable. The corresponding graphical model is visualized in Fig. 3. This factorization is chosen such that individual terms in Eq. (1) are easy to approximate with empirical or simple parametric distributions. Using these approximations, we can use ancestral sampling to generate realistic object location proposals for the scene:

1. **Sample a class.** We first sample a class from the multinomial $\hat{p}(c)$, which we choose to have uniform probabilities to oversample rare classes.
2. **Sample a depth.** To sample objects at realistic distances, we collect observed object depths per class from training data, and approximate $p(d|c)$ with a log-normal distribution. In comparison, we observed that directly choosing

[1]We define semantic categories "road", "terrain", "sidewalk" to be drivable space, as this is where objects of interest typically appear.

Figure 4. Example bounding box proposals from our location model, separated by class.

a random location in the drivable space would result in oversampling of objects at short distances.

3. **Sample a location.** Using semantic map $\mathbf{S}$ and depth map $\mathbf{D}$ we select b^x, b^y uniformly at random from the scene's drivable space, limited to locations with depths that are within a threshold τ_d to the sampled distance d. Fig. 3 shows examples of such "placement bands".
4. **Sample a height.** We collect statistics of object heights at different depth intervals, and approximate them with log-normal distributions $\hat{p}(b^h|d, c)$.
5. **Sample a width.** We collect aspect ratios of objects, independent of depth, and use the resulting empirical distributions (*i.e.* the histograms) for sampling object widths $b^w \sim \hat{p}(b^w|b^h, c)$.

Examples of boxes generated by our model are shown in Fig. 4. We provide more details on the steps and evaluate the quality of the approximations in the supplementary material.

3.2. Generative augmentation

In the previous section we introduced a probabilistic location model that places new objects, parametrized by class and bounding box, into an existing scene (Fig. 2 (A)). In order to render the desired objects, we use a diffusion model for inpainting, namely Stable Diffusion 2 (SD2)[2] [34]. As operating at high resolution with diffusion models is not straightforward, we extract square patches centered in the proposed locations. Every patch has a resolution of $m \times m$ pixels, where $m = 2 \times max(b^h, b^w)$, and is resized to a fixed resolution of 512×512 for inpainting. Examples of such patches are shown in Fig. 2 (B). Prior work utilizes large language models to craft complex textual descriptions [39], but we found that simple text prompts in the format "`image of a <class name>`" are sufficient for realistic generations. We finetune the diffusion model on the domain of interest, for which we tried both direct finetuning and ControlNet [47]. We report scores with ControlNet, but the two options perform on par (see Appendix). Finetuning benefits the inpainting model in three ways. First, it allows it to adapt to the pixel-level statistics of the target dataset, *i.e.* to generate objects that look natural in terms of saturation and contrast. Second, it resolves ambiguities in textual category labels: for example, the class "rider" can be interpreted by SD2 as "horse rider", whereas in the BDD100K dataset it represents only "motorcycle riders". Third, finetuning forces objects to fit more tightly in the provided bounding box. We show examples of this in the Appendix.

Obtaining object masks. To augment data for instance segmentation tasks, we need instance masks for every synthetic object. To this end, we equip the inpainting model with a simple mask decoding module $\mathcal{M}$, responsible for providing a segmentation mask for the objects it generates.

The mask decoder $\mathcal{M}$ is created as a lightweight copy of the SD2 UNet-decoder, with 4x fewer channels per layer. It receives multi-scale features from the SD2 UNet-encoder as input, as its representations are rich in semantic information about objects being generated [39, 48]. Specifically, whenever generating an object, we pass the "denoised" latent variable $\mathbf{z}_0$ to the UNet and extract representations $\{\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_d\}$ at multiple resolutions, before each downsampling layer in the architecture. These features then undergo a simple multi-scale aggregation phase, before being fed to $\mathcal{M}$ to decode an alpha mask $\hat{s}$. To train $\mathcal{M}$, we assume access to crops with available groundtruth instance masks s and optimize a simple binary cross-entropy loss. More details are given in the Appendix.

[2] https://huggingface.co/stabilityai/stable-diffusion-2-inpainting

Figure 5. Example of nuImages frames augmented with our approach. We show the bounding boxes for all added objects. In diverse scenarios, the location and scale of added objects are realistic and thus result in realistic augmented images.

Besides enabling instance segmentation augmentation, we found that these masks allow more realistic handling of occlusions between generated objects, without artifacts (see Fig. 2 (C), occlusion between bus and barrier). Moreover, it allows us to refine the bounding box size in the case the inpainting model generates an object that is smaller than the input bounding box. We provide more details on mask decoding in the supplementary material.

4. Experiments

In this section we present experimental results for generative data augmentation. We show main experiments in Sec. 4.1, where we augment data for both object detection and instance segmentation. We then investigate the influence of different design choices in more detail in Sec. 4.2.

Datasets and evaluation. We conduct experiments on two public automotive object detection benchmarks: nuImages [2] and BDD100K [44]. The nuImages dataset contains 67.279 training images and 16.445 validation images, at resolution 1600×900. It is published by Motional AD Inc. under a CC BY-NC-SA 4.0 license. The BDD100K dataset contains 70.000 training and 10.000 validation images, at resolution 1280×720. To compare data augmentation strategies, we always use all available real training images and equally many augmented frames during each training epoch. Following standard practice, we evaluate object detectors through the mean Average Precision (mAP) on real validation images. For object detection experiments we use a Faster R-CNN [33] with a ResNet-50 backbone (pretrained on ImageNet [8]). For instance segmentation experiments, we use a Mask R-CNN [18] with the same backbone.

Baselines. To measure the effectiveness of our location model, we compare against two baselines that use the same generator (as described in Sec. 3.2), but use a different placement strategy. The first baseline re-uses the original object locations, *i.e.* it generates instances in existing locations, effectively replacing original objects with synthetic ones. This approach is similar to prior work on object detector augmentation [27, 36]. The main differences are that Gen2Det [36] uses an unspecified closed-source diffusion model, whereas Kupyn and Rupprecht [27] use additional depth and edge conditioning for generation. We refer to this baseline as "Replacement" in the following. The second baseline adds new objects in random locations, independently of the scene, which we call "Random Loc.". For the latter, bounding box *sizes* follow the training distribution, while the *locations* are uniformly sampled in the frame.

Additionally, we compare to two state-of-the-art augmentation methods. X-Paste [49]—a reference work in cut-and-paste augmentation—generates synthetic objects with Stable Diffusion [34], segments them, and then pastes them on an existing frame with random location and scale. We use the publicly released code[3] to generate 100.000 objects, and we paste up to 10 per frame, selected at random. We further compare to results reported by GeoDiffusion [6], a layout-to-image method that renders synthetic frames from slightly perturbed object locations. For this model, we report metrics from the original publication, as the authors only released inference code and lower-resolution models. Finally, we also reimplemented background augmentation from [30], but only observed negative augmentation performance, so we provide those results in the supplementary material.

[3] https://github.com/yoctta/XPaste

Table 1. Augmenting Faster R-CNN object detection on nuImages. While all augmentation methods improve over the base model, our proposed data augmentation with layout augmentation outperforms the other methods by a significant margin. "Replacement" is similar to [27, 36]. The bottom row shows the percentage of instances belonging to each category in real training frames.

		Locations	mAP	car	truck	trailer	bus	const.	bicycle	motor.	ped.	cone	barrier
800×456	Baseline	-	37.8	53.6	41.8	17.2	43.1	25.5	45.4	46.9	32.0	32.8	39.3
	Replacement	original	38.5 +0.7	**54.2**	43.3	17.7	44.5	26.1	46.1	47.8	**32.2**	**33.1**	**40.1**
	Random Loc.	random	38.2 +0.4	53.4	42.8	15.8	44.6	27.0	46.4	48.4	31.5	32.6	39.4
	X-Paste [49]	random	38.2 +0.4	53.7	42.9	16.0	44.1	26.4	46.4	48.7	31.8	32.7	39.5
	GeoDiffusion [6]	original	38.3 +0.5	53.2	43.8	18.3	45.0	27.6	45.3	46.9	30.5	32.1	39.8
	Ours	scene-aware	**39.2** +1.4	53.9	**44.0**	**18.6**	**46.1**	**27.7**	**47.0**	**49.4**	32.0	32.9	39.9
1600×900	Baseline	-	50.4	66.4	55.5	21.7	55.9	35.0	55.6	58.2	49.7	54.2	51.8
	Replacement	original	50.7 +0.3	66.8	55.4	22.9	56.5	34.7	55.3	58.9	49.6	54.3	52.1
	Random Loc.	random	51.3 +0.9	66.4	56.3	23.0	58.0	36.4	56.8	60.2	49.7	54.8	51.3
	X-Paste [49]	random	51.5 +1.1	**66.9**	56.7	23.6	58.0	35.9	57.0	60.3	**50.0**	54.7	52.2
	GeoDiffusion [6]	original	–	–	–	–	–	–	–	–	–	–	–
	Ours	scene-aware	**52.0** +1.6	**66.9**	**56.9**	**25.3**	**58.8**	**37.5**	**57.1**	**60.7**	49.8	**54.9**	**52.6**
	Real data [%]			37.1	5.4	0.6	1.2	0.9	2.5	2.5	24.4	12.6	12.8

Implementation details. We augment frames by generating 12 new objects per frame, and randomly showing each with 0.5 probability whenever the frame is chosen during training. For the inpainting model, we finetune for 300,000 iterations at a batchsize of 16, using ControlNet [47] with masked crops as conditioning input. All detector trainings are performed using the mmdetection library [5] and the default configurations, except for the number of training epochs that is set to 36 for all datasets and models. To enable a fair comparison to GeoDiffusion on nuImages, we follow the protocol in the original paper and further train (and evaluate) the detector at a reduced resolution (800×456 pixels); for this experiment exclusively, we reduce the number of epochs to 12 to match their setup. When training on BDD100K we do not augment the *traffic sign* and *traffic light* categories, as our location model is better suited for objects on the ground. The mAP is however computed on all classes. For our scene-aware location model, we extract scene representations from off-the-shelf models for depth estimation [42] and semantic segmentation [43]. We train our mask decoder on nuImages, as the dataset provides precise instance masks.

4.1. Training data augmentation

Object detection on nuImages. We first analyze the performance of different augmentation strategies for object detection on nuImages, for which we report both mAP and class specific scores in Tab. 1. Examples of augmented frames from our method are shown in Fig. 5. Although all augmentation methods improve over the baseline (trained on real data only), we can make the following observations.

At low resolution (800×456), methods using random locations improve mAP marginally (up to +0.4 points), whereas techniques leveraging original locations prove more successful (up to +0.7 points). This is especially true for classes like *pedestrian*, *cone* and *barrier*, which are generally smaller than other objects and therefore harder to detect: diversifying their appearance while keeping the location unchanged clearly helps detector training. However, this observation is reversed at full resolution, where the replacement strategy underperforms with respect to random locations. In both cases, our augmentation method proves to be the best approach, significantly increasing the mAP of the detector by 1.4 and 1.6 in low and high resolution, respectively.

Looking at per-category results, we can see how our approach improves the most on classes that are underrepresented in the dataset, such as *trailer* (0.6% of training instances), *construction vehicle* (0.9%), *bus* (1.2%), *bicycle* (2.5%) and *motorcycle* (2.5%). In contrast, strategies relying on original object locations (*e.g.* replacement) tend to work well on categories that are already well represented in the training data, such as *car* and *pedestrian*. This is likely due to the fact that they cannot easily oversample rare classes, unlike our method and approaches that use random locations. This finding suggests that approaches that allow oversampling of rare classes, such as ours, are the superior choice for problems where categories follow a long-tailed distribution. Arguably, this applies to many real-world problems.

Instance segmentation on nuImages. Next, we test our approach on instance segmentation, by using the strategy described in Sec. 3.2 to obtain pseudo-groundtruth masks for synthetic objects. We compare our method to replacement and random locations by assessing mAP both on bounding

Table 2. Augmenting Mask R-CNN instance segmentation and detection on nuImages at full 1600 × 900 resolution. All approaches use the same inpainting strategy and only differ by the locations in which objects are inpainted to the scene.

	Locations	Bounding box evaluation						Instance mask evaluation					
		mAP	mAP_{50}	mAP_{75}	small	med.	large	mAP	mAP_{50}	mAP_{75}	small	med.	large
Baseline	-	51.2	77.8	55.8	31.3	49.5	64.2	41.5	71.2	42.2	19.6	40.5	57.6
Replacement	original	51.5 $^{+0.3}$	77.9	56.1	**31.4**	50.0	64.6	41.6 $^{+0.1}$	71.6	42.3	**19.7**	40.8	57.8
Random Loc.	random	51.9 $^{+0.7}$	77.9	56.2	31.0	50.2	65.2	42.1 $^{+0.6}$	71.9	42.9	19.4	40.9	58.3
Ours	scene-aware	**52.6** $^{+1.4}$	**78.8**	**57.0**	31.1	**50.8**	**66.1**	**42.4** $^{+0.9}$	**72.4**	**43.1**	19.4	**41.0**	**58.9**

Table 3. Faster R-CNN object detection augmentation results on BDD100K at full 1280 × 720 resolution.

	Locations	mAP	ped.	rider	car	truck	bus	train	motor.	bicycle	tr.light	tr.sign
Baseline	-	31.4	34.5	26.3	50.8	46.2	46.9	0.0	24.6	25.9	21.9	**37.1**
Replacement	original	31.6 $^{+0.2}$	34.4	26.5	51.1	46.2	48.4	0.0	24.9	25.5	21.7	36.8
Random Loc.	random	32.1 $^{+0.7}$	34.6	27.2	51.0	47.3	48.9	0.0	25.9	26.8	**22.1**	37.0
X-Paste [49]	random	32.3 $^{+0.9}$	34.8	27.5	50.9	**47.9**	49.2	**3.4**	24.9	26.5	21.8	36.6
Ours	scene-aware	**32.7** $^{+1.3}$	**35.0**	**28.0**	**51.2**	47.6	**49.9**	0.8	**27.6**	**27.9**	21.9	**37.1**

boxes and on instance masks, and we report results in Tab. 2. Our approach outperforms both baselines in both metrics, highlighting the benefit of scene-aware location modeling. The baselines differ from our approach only in the chosen object locations, while the generation model is the same.

We also observe that the performance on small objects remains largely unchanged, regardless of the augmentation method. Moreover, the improvement over the baseline offered by our method seems to increase with object size. A potential reason for this behavior is that small objects are more common in the real data, leaving less room for improvement compared to the relatively rare large objects. The bottom row of Tab. 1 shows the distribution of object counts in the data.

Object detection on BDD100K. We repeat the object detection augmentation experiment on the BDD100K dataset, and report results in Tab. 3. Overall, the table shows lower scores than on nuImages, and we ascribe this behaviour to BDD100K being a more challenging dataset (*e.g.* it includes data filmed through the windshield and thus it shows a lot of reflections and dirt). Scene-aware location modeling again outperforms both replacement and random location strategies, improving the baseline detector by 1.3 points in mAP (31.4 → 32.7).

X-Paste also performs reasonably well on this dataset, and it notably outperforms our method in the *train* category. We believe this result is due to the scarcity of data for this class, which features only 15 instances in the validation set, and for which the AP is extremely low for all methods. We suspect that in such extreme long-tailed cases the high diversity of generated objects offered by X-Paste might prove more beneficial than realistic placement and generation.

4.2. Ablations

We use this section to investigate the influence of individual model components and design choices. The results are compiled in Tab. 4, we address them one-by-one.

Finetuning. First, we ablate the decision to finetune the inpainting model on the target dataset, for which we explored both direct finetuning and ControlNet [47] (see Appendix for a comparison). While the SD2 base model sometimes creates convincing objects, we find that on average, finetuning leads to a) better visual coherence with the surroundings and b) objects that better fill the provided bounding box. We show visual examples in the Appendix. Consequently, performance without finetuning is much lower, and hardly improves over the baseline model. We leave an exploration of other finetuning techniques [21, 32] for future work.

Mask prediction, SAM masks. At inference time, we generate objects and masks jointly. This is not strictly necessary for detector augmentation, and mainly serves to extend our approach to instance segmentation. However, we can also use the generated masks to refine the bounding boxes and improve foreground-background blending. The effect is strongest at high IoU thresholds (see Appendix), but it significantly influences mAP as a whole. Interestingly, using SAM [25] to extract segmentation masks for generated objects works only slightly less well at low resolution, but leads to a large performance drop at full resolution. We suspect that even though SAM was trained for broad applicability, there is still a distribution mismatch, and our model benefits from nuImages training. While this could be remedied with finetuning SAM, our mask decoder is orders of magnitude

Table 4. Ablation of the effect of design choices in our approach on Faster R-CNN detector performance (mAP) on nuImages.

	800×456	1600×900
Baseline	37.8	50.4
Ours	39.2	52.0
without finetuning	37.9 -1.3	50.8 -1.2
without mask pred.	38.7 -0.5	51.4 -0.6
with SAM masks	39.0 -0.2	51.4 -0.6
model loc., rand. scale	38.6 -0.6	51.6 -0.4
rand. loc., model scale	38.5 -0.7	51.6 -0.4
X-Paste [49]	38.2	51.5
with our location model	38.4 +0.2	51.6 +0.1

smaller than it (we use the *sam-vit-huge* checkpoint here). This result highlights the benefit of leveraging the diffusion model representations for mask generation.

Randomizing location or scale. To understand if object *locations* or object *scales* are more important, we perform two sets of experiments: one samples the object location according to our model, but samples the scale unconditionally from the empirical data distribution; the other samples a location uniformly at random, but samples the scale according to our model (conditioned on the location). We find that the augmentation gain in each case is roughly half of our model's total, indicating that location and scale are equally relevant, and that both need to be realistic to achieve an optimal performance gain.

Combining our location model and X-Paste. The purpose of our location model is to allow placing objects in a realistic context within a given scene. It is intuitive that good locations and scale matter here, as the inpainting model can take advantage of these and *e.g.* create challenging occlusion scenarios, but it may perform less well if locations are unrealistic. An open question is whether improving locations has a similar effect on cut-and-paste type approaches. Since X-Paste [49] pastes pre-generated objects into the frames ignoring the context entirely, the realism of their scale or location should not matter. Surprisingly, we still see a small performance boost when combining our location model with X-Paste. However, augmented frames still appear entirely unrealistic, and the performance gain may only be due to an improved location and scale bias of the detector.

Additional analyses. In the supplementary material, we attempt to quantify the realism and diversity of augmented frames, where our method yields numbers that are comparable to adding completely new data. We further analyze the effect of bounding box refinement, where we use the predicted instance masks to refine bounding boxes. While negligible at lower IoU thresholds, this has a significant influence at high thresholds. Finally, we show examples of failure modes, which occur when the depth or drivable space predictions are incorrect, or for more complex scene geometries.

5. Conclusions

In this work, we demonstrate that generative data augmentation benefits from adding objects in new and realistic locations. We first propose a scene-aware probabilistic location model that predicts new object locations for existing scenes. To fully take advantage of this location model, we then adapt a diffusion model to jointly inpaint objects in the proposed locations and to produce instance masks for them. Using this approach, we are able to generate realistic and challenging augmented frames, *e.g.* with object occlusions, which set a new state of the art in data augmentation for object detectors on two street scene datasets, outperforming the mAP gains achieved by existing methods by a large margin. We also demonstrate significant gains in data augmentation for instance segmentation. Crucially, using the same augmentation strategy but with completely random object placement, or only reusing existing object locations, performs much worse than our approach, highlighting the benefit of augmentation that places objects in new and realistic locations.

Limitations. The probabilistic factorization of our location model takes advantage of the high regularity of street scenes. We suspect that more for diverse scenes, as for example in COCO [31], a fully learned location model may be required. At the same time, we expect that a more advanced object placement strategy would improve performance even further. Our location model is also somewhat tied to the augmentation strategy, it requires an inpainting model to fully take advantage of the predicted locations. The benefit for cut-and-paste approaches is limited. It also depends on the quality of the depth estimation and the segmentation of drivable space (we show failure cases in the Appendix).

An opportunity for future work is combining the augmentation approaches discussed in this work. In particular, object replacement, full-frame synthesis, and our object placement in new locations are in principle complementary. However, identifying the right way to combine synthetic data from these sources is a non-trivial problem [36]. Finally, we expect all generative data augmentation to improve with the quality of the underlying generator. We test our approach using only Stable Diffusion 2 [34], as it is a commonly used open-source model, but other promising open-source or open-weight models have been released since [4, 28]. As generative models progress, generative data augmentation will likely play an increasingly important role in the training of task-specific models.

References

[1] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J Fleet. Synthetic data from diffusion models improves imagenet classification. *arXiv preprint arXiv:2304.08466*, 2023. 2

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2020. 5

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 1

[4] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. *arXiv preprint arXiv:2403.04692*, 2024. 8

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

[6] Kai Chen, Enze Xie, Zhe Chen, Yibo Wang, Lanqing Hong, Zhenguo Li, and Dit-Yan Yeung. Geodiffusion: Text-prompted geometric control for object detection data generation. *International Conference on Learning Representations*, 2024. 1, 2, 5, 6, 3, 11

[7] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 3

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2009. 5

[9] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *Proceedings of the European Conference on Computer Vision*, 2020. 2

[10] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. On the importance of visual context for data augmentation in scene understanding. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):2014–2028, 2019. 2, 3

[11] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1301–1310, 2017. 2

[12] Chengxiang Fan, Muzhi Zhu, Hao Chen, Yang Liu, Weijia Wu, Huaqi Zhang, and Chunhua Shen. Divergen: Improving instance segmentation by learning wider data distribution with more diverse generative data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3986–3995, 2024. 2

[13] Ruiyuan Gao, Kai Chen, Zhihao Li, Lanqing Hong, Zhenguo Li, and Qiang Xu. Magicdrive3d: Controllable 3d generation for any-view rendering in street scenes. *arXiv preprint arXiv:2405.14475*, 2024. 2

[14] Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. MagicDrive: Street view generation with diverse 3d geometry control. In *International Conference on Learning Representations*, 2024. 1, 2

[15] Yunhao Ge, Jiashu Xu, Brian Nlong Zhao, Neel Joshi, Laurent Itti, and Vibhav Vineet. Dall-e for detection: Language-driven compositional image synthesis for object detection. *arXiv preprint arXiv:2206.09592*, 2022. 1, 2

[16] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021. 2

[17] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee, 2008. 1

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, 2017. 5

[19] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? In *International Conference on Learning Representations*, 2023. 1

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Neural Information Processing Systems*, 2020. 1

[21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022. 7, 4

[22] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Layoutdm: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10167–10176, 2023. 2

[23] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9895–9904, 2019. 2

[24] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *IEEE International Conference on Computer Vision*, 2019. 2

[25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment any-

thing. *IEEE International Conference on Computer Vision*, 2023. 7

[26] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. Blt: Bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision*, pages 474–490. Springer, 2022. 2

[27] Orest Kupyn and Christian Rupprecht. Dataset enhancement with instance-level augmentations. *arXiv preprint arXiv:2406.08249*, 2024. 1, 2, 5, 6

[28] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 8

[29] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. In *Neural Information Processing Systems*, 2018. 2

[30] Yuhang Li, Xin Dong, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. A simple background augmentation method for object detection with diffusion model. In *European Conference on Computer Vision*, pages 462–479. Springer, 2024. 2, 5

[31] 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 *Proceedings of the European Conference on Computer Vision*, 2014. 8

[32] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *The AAAI Conference on Artificial Intelligence*, 2024. 7, 4

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Neural Information Processing Systems*, 2015. 5

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022. 1, 4, 5, 8

[35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Neural Information Processing Systems*, 2022. 1

[36] Saksham Suri, Fanyi Xiao, Animesh Sinha, Sean Chang Culatana, Raghuraman Krishnamoorthi, Chenchen Zhu, and Abhinav Shrivastava. Gen2det: Generate to detect. *arXiv preprint arXiv:2312.04566*, 2023. 1, 2, 5, 6, 8, 3

[37] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2021. 2

[38] Yibo Wang, Ruiyuan Gao, Kai Chen, Kaiqiang Zhou, Yingjie Cai, Lanqing Hong, Zhenguo Li, Lihui Jiang, Dit-Yan Yeung, Qiang Xu, et al. Detdiffusion: Synergizing generative and perceptive models for enhanced data generation and perception. *arXiv preprint arXiv:2403.13304*, 2024. 1

[39] Weijia Wu, Yuzhong Zhao, Hao Chen, Yuchao Gu, Rui Zhao, Yefei He, Hong Zhou, Mike Zheng Shou, and Chunhua Shen. Datasetdm: Synthesizing data with perception annotations using diffusion models. *Neural Information Processing Systems*, 2023. 2, 4

[40] Jiahao Xie, Wei Li, Xiangtai Li, Ziwei Liu, Yew Soon Ong, and Chen Change Loy. Mosaicfusion: Diffusion models as data augmenters for large vocabulary instance segmentation. *International Journal of Computer Vision*, 2024. 2

[41] Cheng-Fu Yang, Wan-Cyuan Fan, Fu-En Yang, and Yu-Chiang Frank Wang. Layouttransformer: Scene layout generation with conceptual and spatial diversity. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3732–3741, 2021. 2

[42] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2024. 6, 1, 5

[43] Yung-Hsu Yang, Thomas E Huang, Min Sun, Samuel Rota Bulò, Peter Kontschieder, and Fisher Yu. Dense prediction with attentive feature aggregation. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023. 6

[44] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2020. 5

[45] Jooyeol Yun, Davide Abati, Mohamed Omran, Jaegul Choo, Amirhossein Habibian, and Auke Wiggers. Generative location modeling for spatially aware object insertion. *arXiv preprint arXiv:2410.13564*, 2024. 2

[46] Lingzhi Zhang, Tarmily Wen, Jie Min, Jiancong Wang, David Han, and Jianbo Shi. Learning object placement by inpainting for compositional data augmentation. In *Proceedings of the European Conference on Computer Vision*, 2020. 2

[47] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 4, 6, 7

[48] Manlin Zhang, Jie Wu, Yuxi Ren, Ming Li, Jie Qin, Xuefeng Xiao, Wei Liu, Rui Wang, Min Zheng, and Andy J Ma. Diffusionengine: Diffusion model is scalable data engine for object detection. *arXiv preprint arXiv:2309.03893*, 2023. 2, 4

[49] Hanqing Zhao, Dianmo Sheng, Jianmin Bao, Dongdong Chen, Dong Chen, Fang Wen, Lu Yuan, Ce Liu, Wenbo Zhou, Qi Chu, et al. X-paste: Revisiting scalable copy-paste for instance segmentation using clip and stablediffusion. *International Conference on Machine Learning*, 2023. 1, 2, 5, 6, 7, 8, 10

[50] Yongchao Zhou, Hshmat Sahak, and Jimmy Ba. Using synthetic data for data augmentation to improve classification accuracy. In *ICML Workshop on Deployable Generative AI*, 2023. 1

[51] Sijie Zhu, Zhe Lin, Scott Cohen, Jason Kuen, Zhifei Zhang, and Chen Chen. Topnet: Transformer-based object placement network for image compositing. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023. 2

Scene-Aware Location Modeling for Data Augmentation in Automotive Object Detection

Supplementary Material

A. Method Details

A.1. Location model

In this section we provide more details in our location model, specifically the individual sampling steps and how we approximate the corresponding likelihoods. The accompanying figure is Fig. 7, where we show "car", "bus", and "pedestrian" as representative classes, using data from the front camera in the nuImages dataset (nuImages uses 6 cameras, and we treat them separately).

Depth sampling We use DepthAnything [42] as an off-the-shelf depth estimator. While the model technically outputs what the authors call *disparity* ($disparity \propto 1/depth$), we still refer to it as depth, as we believe our description is easier to understand this way. Figure 7a shows depth histograms for the three classes, along with the log-normal approximation we use. We prefer an easy-to-use parametric distribution over one that optimizes data fit, and in this case a log-normal is clearly a good enough choice. Note that larger depth values are closer to the camera, so most objects are comparatively far away.

Location sampling Once we have sampled a depth value, we select all pixels from the drivable space which are within $\tau_d = 5$ of this value. This typically results in a band of possible locations, two examples of which are shown in Fig. 7b. We then select a pixel from this band at random and use it as the bottom-center location for the bounding box. Should no pixels in the drivable space be within the allowed depth interval, we reset the depth value to the closest allowed one. By first sampling the depth, and only then a location from the resulting band, we avoid oversampling close objects, because logically there are more pixels closer to the camera than further away.

Height sampling Sampling the height is arguably the most complicated part in our model, as it is conditioned on the depth. Figure 7c shows example histograms at different depths. We also approximate these with log-normals, but the approximation is clearly not as good as in the case of the depth. Nevertheless, we find that on average it results in realistic object heights. To get the mean $\mu_h(d)$ and standard deviation $\sigma_h(d)$ of the log-normal for a given depth, we build such histograms for all possible disparities, evaluated in windows of width 2, and then calculate the parameters in each case (*i.e.* the mean and standard deviation of the log-data). We then fit a simple parametric model of the form $y = a + b \cdot x^c$ to be able to interpolate mean $\mu_h(d)$ and standard deviation $\sigma_h(d)$ for a given depth at sampling time. The interpolation is visualized in Fig. 7d and fits the data fairly well. Only at high depth values, *i.e.* close to the camera, do we find significant deviation from the underlying data. As these depths have very low likelihoods anyway, we accept this tradeoff.

Width sampling We sample the width conditioned on the height via the distribution of aspect ratios for the given class, visualized in Fig. 7e. These are independent of the depth. Unfortunately, the aspect ratio histograms follow a more complex pattern, and we were unable to find a good parametric approximation. This is likely due to objects living in 3D space with almost arbitrary rotations (in terms of yaw), whereas we only work with 2D bounding boxes. This effect is very pronounced for cars and buses, but less so for pedestrians. As a result, we use the empirical likelihoods directly, *i.e.* the histogram bins are normalized to sum to 1 and then taken as likelihoods for the corresponding bin intervals.

A.2. Mask generation

As mentioned in Sec. 3.2, we design a simple mask decoding module $\mathcal{M}$ to plug into the SD2 inpainting model, responsible for creating segmentation masks for every generated object. As explained in the main text, its architecture is mirroring the one of the image (VAE) decoder, with two notable exceptions. First, all its layers feature 4x fewer channels, as we assume that decoding a binary mask requires a lot less capacity than to decode an RGB image. Secondly, it does not use the clean latents $\mathbf{z}_0$ directly, but rather multi-scale features from the encoder part of the SD2 UNet, generated by feeding $\mathbf{z}_0$ to it. This step is akin to running the SD2 denoiser for an additional diffusion step, and extracting rep-

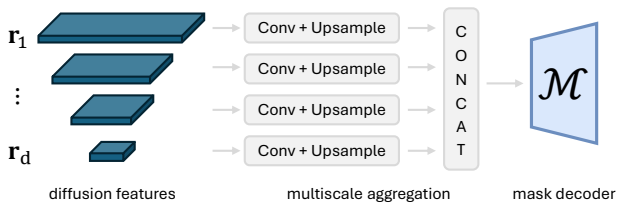
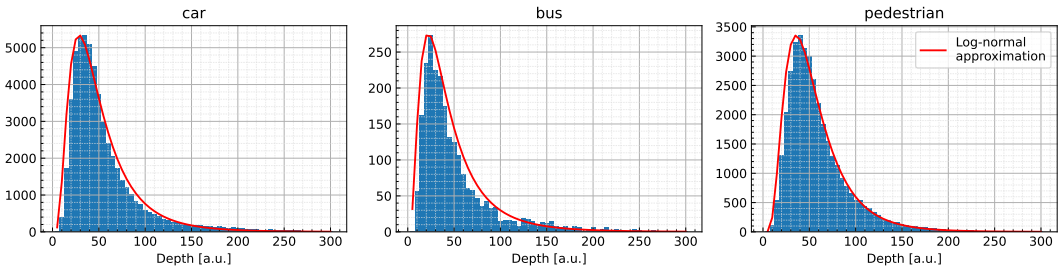
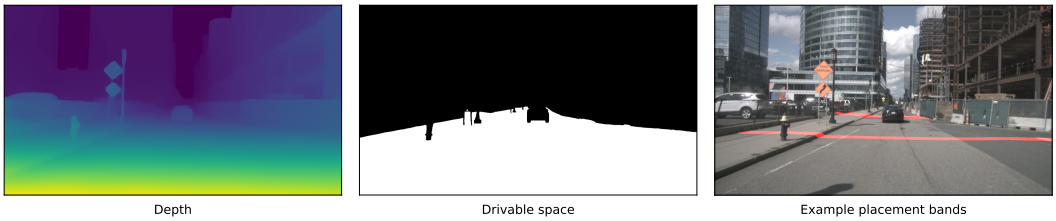


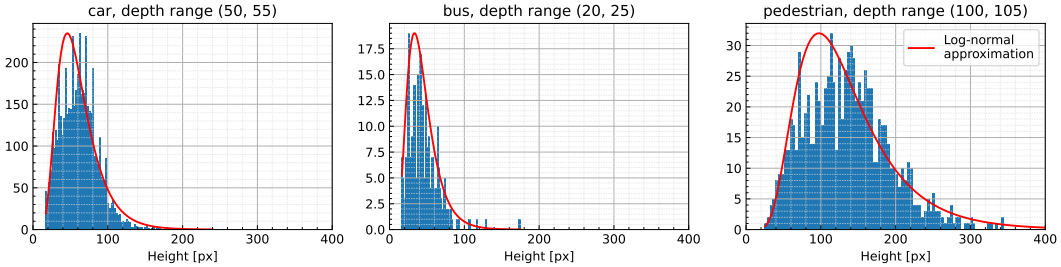
Figure 6. Representation of the multiscale aggregation module for feeding multi-scale UNet features to the proposed mask decoder.



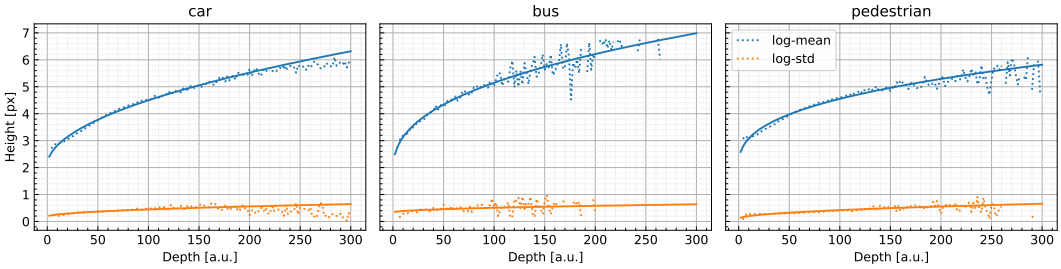
(a) Depth histogram for three representative classes, along with the log-normal approximation we use. Note that the depth estimator we use outputs values where higher means closer to the camera.



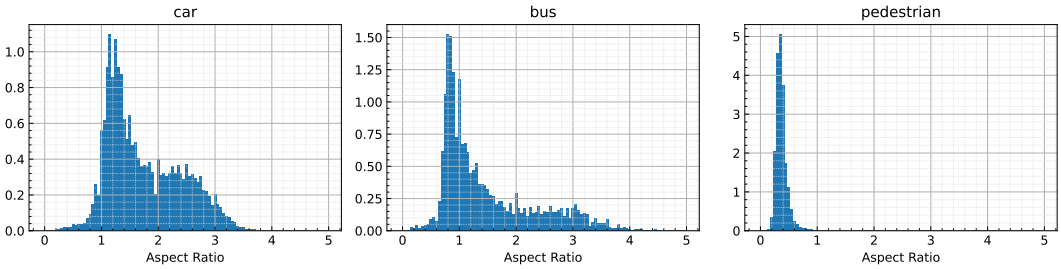
(b) Example frame with depth map and drivable space. We first draw a depth value and then select an area from the drivable space that is within a threshold $\tau_d = 5$ around that value. This results in placement bands from which a location is selected at random.



(c) Height histogram for three representative classes, and three example depth ranges. We also approximate the resulting height distributions with log-normals.



(d) To be able to parametrize a height distribution, given a depth value, we compute mean and standard deviation of height histograms for different depths in the dataset (using a window size of 2). We then fit a curve to approximate $\mu_h(d)$ and $\sigma_h(d)$.



(e) We get the object width for a given height via the aspect ratio. Because we could not find a satisfactory parametric distribution, we work with the empirical distributions directly, in this case.

Figure 7. Overview of the different sampling steps in our location model, and what approximations we use.

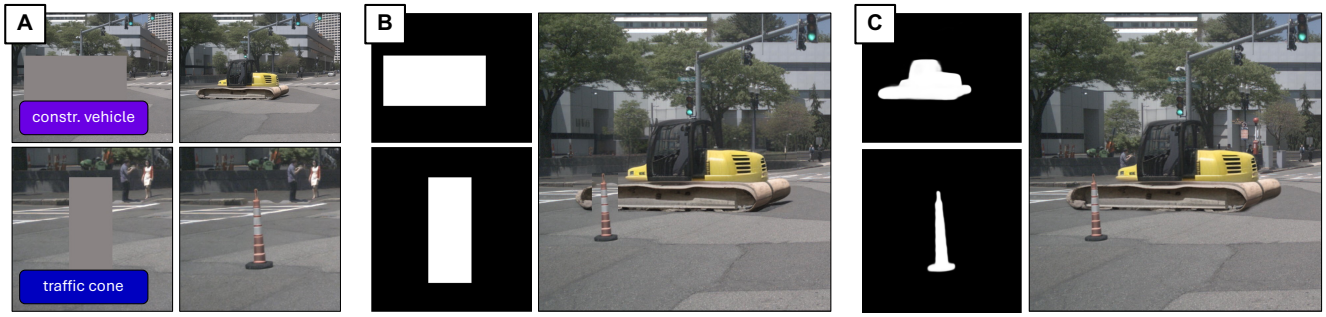


Figure 8. Producing object masks is crucial for seamless inpainting. (A) Two example generations and their original inpainted areas. (B) Pasting the synthetic objects back in the frame by using bounding boxes as masks can result in severe boundary artifacts, especially in the case of occlusions. (C) Using generated instance masks yields more realistic results.

resentations from several layers in its UNet (specifically, all downsampling layers). Given the fact that the input to the $\mathcal{M}$ is not a tensor but rather a set of tensors at different resolutions (which we name $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_d\}$), we run a simple multi-scale aggregation module (represented in Fig. 6), which upscales all features to the same resolution and concatenates them.

We train our mask decoder on nuImages, as all its labeled objects come with an instance mask that we can use for supervision. We optimize $\mathcal{M}$ after the finetuning stage of SD2, and observed no benefits in training both jointly. As BDD100k does not provide any segmentation masks, we use the mask decoder trained on nuImages when generating objects for this dataset. We observed that generated masks are of comparable quality, even for unseen classes that are not available in nuImages (such as 'rider' and 'train').

Besides providing pseudo-labels for instance segmentation augmentation, the masks decoded by the mask decoder prove useful in the case generated objects occlude themselves. Consider, for instance, the case represented in Fig. 8, where instances of a construction vehicle and of a traffic cone are generated independently, in close locations. When pasting the crops back to the original frame using full bounding box areas, it is impossible to avoid visible artifacts. However, by using precise pixel masks, occlusions are handled successfully, increasing the overall realism of the augmented frame. We believe such cases are one of the main factors explaining the drop in mAP that we observe when not using masks, as reported in Tab. 4 and Fig. 10.

B. Additional Results & Examples

B.1. Realism and diversity

In this section, we evaluate and compare the realism and diversity of frames augmented with our method and other methods, as these aspects are often mentioned as key factors for good data augmentation performance [10, 36]. Here, realism refers to the visual quality of augmented samples,

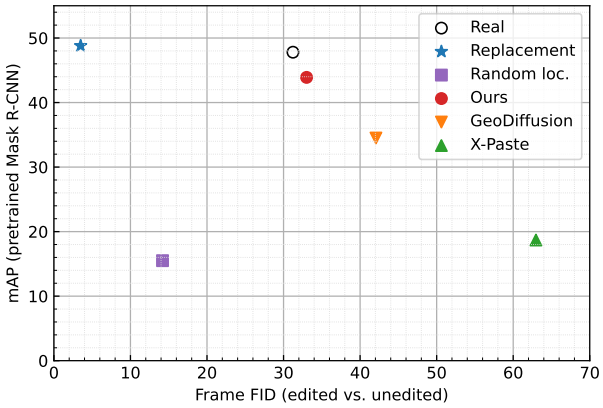


Figure 9. Pretrained MaskRCNN mAP vs. FID between edited and unedited frames. High mAP indicates high realism, high frame FID indicates high frame variability.

whereas diversity describes how different augmented frames are from the already available training frames. We argue that the current methods relying on original or random object locations maximize one aspect while neglecting the other: generating from original locations guarantees realistic scene composition, yet variations are limited to visual appearance, whereas using random locations yields very diverse scene layouts and scale for new objects, at the expense of realism.

This can be seen qualitatively in the examples in Fig. 1. To measure the realism of augmented frames quantitatively, we adopt the approach of GeoDiffusion [6] and evaluate a pretrained Mask R-CNN[4] on the augmented frames. To measure diversity, we compute the FID between paired sets of unedited and edited *full frames*, using 1000 images for both. A method that does not change the data at all should obtain a FID score of zero, and a method that produces high diversity samples should obtain higher FID. Importantly, generating unrealistic frames (*e.g.* random noise) can result in very high FID scores: as such, we argue that the optimal

[4]The ImageNet-pretrained R-50 model from mmdetection3d [7].

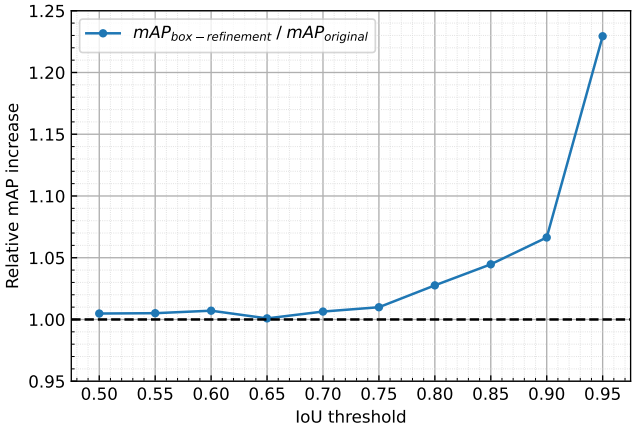


Figure 10. We use predicted instance masks to refine bounding boxes of inpainted objects. The graph shows the relative mAP improvement (on nuImages, full resolution) of this step, compared to not performing it, for different IoU thresholds. Up to a threshold of ca. 0.75 there is hardly a difference, but at higher thresholds this refinement results in up to 23% improvement.

value for this diversity metric is not necessarily the lowest or highest score, but rather a score that is comparable to that achieved by a different set of real frames.

Figure 9 shows realism and diversity scores for several augmentation methods. The replacement baseline achieves low frame FID and high mAP: the augmented frames are realistic, but hardly changed. GeoDiffusion and X-Paste have much higher frame FID, meaning they add a lot of diversity, but this comes at the price of lower realism. GeoDiffusion likely obtains high frame FID because it generates both background and objects. Our approach changes parts of the scene but it leaves other areas of the frame untouched, and achieves high realism according to the pretrained detector. Importantly, it achieves comparable frame FID to a set of *Real* data, meaning our augmented frames are as dissimilar to the originals as a set of new real frames. We argue that this is a very good operating point in the realism-diversity tradeoff. Finally, we see that the mAP of the random location baseline is similar to that of X-Paste (which also uses random locations), but with lower frame FID. We suspect that this is because the inpainting model sometimes struggles to generate objects in unrealistic locations, and instead just completes the crop to look realistic.

B.2. Effect of bounding box refinement

We adapt the inpainting model so that it also predicts instance masks for the generated objects. We use these masks to refine the bounding boxes, so that they fit the generated objects more closely, which leads to a significant performance improvement (see ablations in Tab. 4). As one might expect, this bounding box refinement has a stronger influence when the the IoU threshold for a successful detection is higher. In Fig. 10 we show the mAP of an augmented Faster R-CNN on nuImages (full resolution) for different IoU thresholds. Up to a threshold of around 0.75 there is hardly a difference between refined and normal bounding box use. But at higher thresholds the effect becomes more apparent, with an improvement of 23% at the highest threshold 0.95.

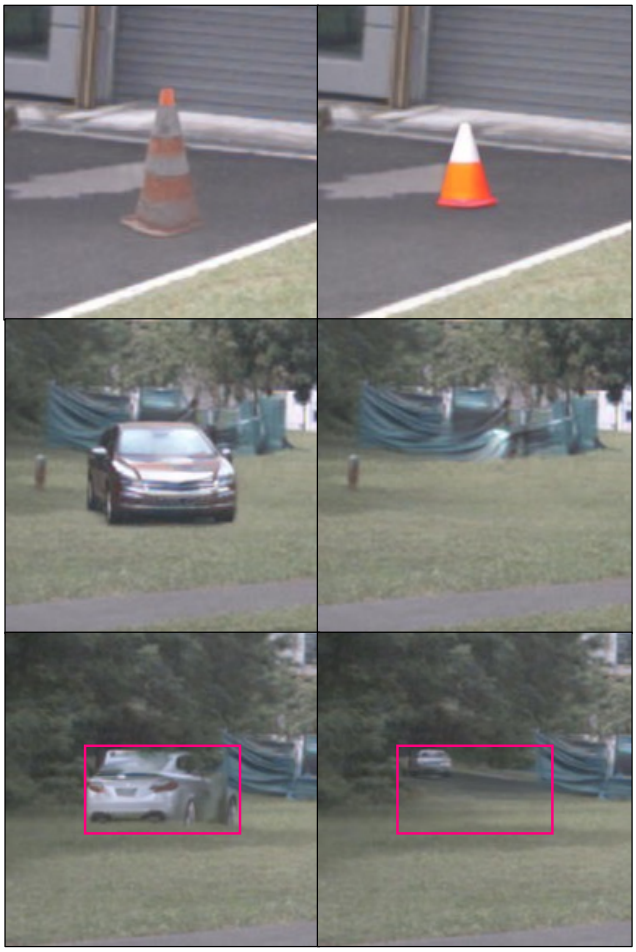


Figure 11. Comparison of finetuning with ControlNet (ours, left column) and using the SD2 pretrained inpainting checkpoint (right column). With finetuning, generated objects will often blend better with the surroundings in terms of color and saturation (top row). Without finetuning, the generator will sometimes not produce and object at all (middle row), or one that doesn't fully fill the provided bounding box (bottom row).

B.3. Effect of finetuning

We choose to finetune the SD2 inpainting model [34], for which we tried both direct finetuning and ControlNet [47] and observed similar performance (see Tab. 5, results in this work are from ControlNet). While we expect that other methods of finetuning [21, 32] work similarly well, we showed in the ablations that finetuning in general has a strong effect on augmentation performance. Examples of the main dif-

Table 5. Further results for Faster R-CNN augmentation on nuImages (800 × 456). As there is no public code available for [30], we reimplemented the method.

	Locations	mAP	car	truck	trailer	bus	const.	bicycle	motor.	ped.	cone	barrier
Baseline	-	37.8	53.6	41.8	17.2	43.1	25.5	45.4	46.9	32.0	32.8	39.3
Background [30]	original	36.6 $^{-1.2}$	53.0	41.9	13.6	42.1	23.9	43.7	46.3	30.0	31.7	39.4
Ours (ControlNet)	scene-aware	39.2 $^{+1.4}$	53.9	44.0	18.6	46.1	27.7	47.0	49.4	32.0	32.9	39.9
Ours (direct FT)	scene-aware	39.2 $^{+1.4}$	54.1	44.2	19.5	45.7	27.4	46.9	49.5	31.7	32.8	39.8

Figure 12. Examples of background augmentation [30].

ferences we observe are shown in Fig. 11. In many cases, with finetuning the generated objects visually fit their context better, *i.e.* they have more realistic colors and saturation (top row in Fig. 11). We also find that in a significant number of cases, the non-finetuned model doesn't produce an object at all, instead just completing the area to look realistic (middle row in Fig. 11). In other cases, the non-finetuned model does produce an object, but it only fills part of the provided bounding box, whereas the finetuned model tends to fill the desired box almost fully (bottom row in Fig. 11).

B.4. Further ablations & baselines

In Tab. 5 we show the performance of background augmentation [30] and compare the ControlNet-finetuned (default) and directly finetuned versions of our approach. Using our approach with ControlNet and direct finetuning lead to the same overall mAP, with only small differences in individual classes. This suggest that the specific method of finetuning is not important, only that finetuning is performed at all. As there is no official code release for background augmentation, we reimplemented the method ourselves. While we are confident in our implementation, there is a possiblity that we missed some important detail, leading to poor performance. With augmented data from this method, mAP is reduce by 1.2 points. Another possible explanation is a data mismatch between the model and the nuImages dataset—the method produces convincing results on some frames and fails completely on others (see Fig. 12). We suspect that with finetuning or quality filtering the approach could be improved significantly.

B.5. Failure cases

We could identify some failure cases in our augmentation approach, examples of which are shown in Fig. 13.

The first is a result of inpainting with instance masks. In most cases, our mask decoder predicts object masks that don't include shadows. Depending on the lighting in the original scene, this can give our objects a "floating" appearance, which hurts visual realism (top row in Fig. 13). The reason for this behaviour is that the instance masks in the original data, which our mask decoder is trained with, do not include shadows either. As a result, we were unable to find a way to test if this actually hurts augmentation performance or only visual realism.

The second failure case occurs when the depth estimate for the scene is wrong. We use DepthAnything [42], which produces relative depth estimates (or disparity to be more precise, see Appendix A) that are normalized to the distance value of the most distant parts of the scene. This is usually the sky, but if there is no sky in the scene, *e.g.* when the vehicle faces a wall, the depth is normalized to a closer value, which in turn leads to our model producing objects that are smaller than they should be (middle row in Fig. 13). We experimented with a version of DepthAnything finetuned for metric depth estimation, but still observed the same behaviour.

Finally, our location model relies on segmentation of drivable space to place objects. Mistakes in the segmentation map will sometimes lead object placement that is visually unrealistic. In the example in Fig. 13 (bottom row), the segmentation model identifies both the grass and the barrier

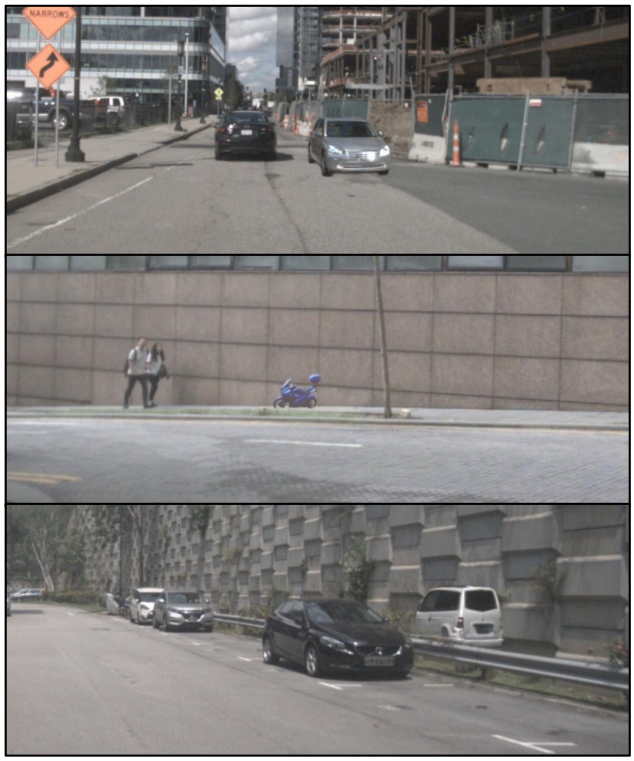


Figure 13. Some failure cases that occur in our augmentation approach. (top) Our mask decoder often fails to include shadows in the mask. (middle) If the depth estimate is wrong, our model produces objects with the wrong scale. (bottom) If the drivable space segmentation is wrong, our model puts objects in unrealistic locations.

above it as "terrain", so that the silver car is rendered in a location where it doesn't fit geometrically. This issue should disappear with better segmentation models, but we were unable to test how this influences augmentation performance.

B.6. Qualitative examples

We show more examples of augmented frames for all methods compared in this work. Examples for ours are shown in Fig. 14, for object replacement in Fig. 15, for random placement in Fig. 16, for X-Paste [49] in Fig. 17, and for GeoDiffusion [6] in Fig. 18.

Figure 14. Additional frames augmented with our approach. We only show annotations for generated objects. Some frames are zoomed in for better visibility.

Figure 15. Additional frames augmented with object replacement. We only show annotations for generated objects. Zoom factors are chosen to match Fig. 14.

Figure 16. Additional frames augmented with random placement. We only show annotations for generated objects. Zoom factors are chosen to match Fig. 14.

Figure 17. Additional frames augmented with X-Paste [49]. We only show annotations for generated objects. Zoom factors are chosen to match Fig. 14.

Figure 18. Additional frames augmented with GeoDiffusion [6]. Zoom factors are chosen to match Fig. 14.