Data Augmentation for Object Detection via Differentiable Neural Rendering

Guanghan Ning[1], Guang Chen[1], Chaowei Tan[1], Si Luo[1], Liefeng Bo[1], Heng Huang[1, 2]
[1]JD Finance America Corporation [2]University of Pittsburgh

Abstract

It is challenging to train a robust object detector under the supervised learning setting when the annotated data are scarce. Thus, previous approaches tackling this problem are in two categories: semi-supervised learning models that interpolate labeled data from unlabeled data, and self-supervised learning approaches that exploit signals within unlabeled data via pretext tasks. To seamlessly integrate and enhance existing supervised object detection methods, in this work, we focus on addressing the data scarcity problem from a fundamental viewpoint without changing the supervised learning paradigm. We propose a new offline data augmentation method for object detection, which semantically interpolates the training data with novel views. Specifically, our new system generates controllable views of training images based on differentiable neural rendering, together with corresponding bounding box annotations which involve no human intervention. Firstly, we extract and project pixel-aligned image features into point clouds while estimating depth maps. We then re-project them with a target camera pose and render a novel-view 2d image. Objects in the form of keypoints are marked in point clouds to recover annotations in new views. Our new method is fully compatible with online data augmentation methods, such as affine transform, image mixup, etc. Extensive experiments show that our method, as a cost-free tool to enrich images and labels, can significantly boost the performance of object detection systems with scarce training data. Code is available at https://github.com/Guanghan/DANR.

1. Introduction

Given a monocular RGB image, an object detection system aims to determine whether there are any instances of semantic objects from pre-defined categories and, if present, to locate the instance and return the confidence. As a popular task, the object detection techniques have been widely deployed in real-world applications such as robotics and autonomous vehicles. However, in some applications,

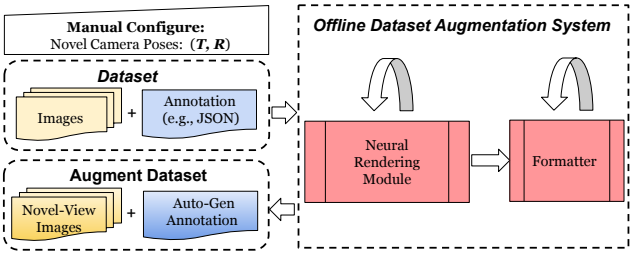


Figure 1. Overview of the proposed data augmentation system.

e.g., image content review, the data distribution is long-tailed, where labelled data of specific categories are naturally scarce. Training data-driven neural networks under such circumstances is quite challenging in the supervised learning paradigm. First of all, training object detectors with scarce annotated data is inherently difficult. Detectors need to handle objects that may occur in stochastic regions of an image, thus requiring a significant amount of data. Secondly, traditional online data augmentation methods, including random crop, image mixup and dozens of others, provide additional data diversity to avoid over-fitting but do not provide unseen semantics with novel locations. An offline approach to enrich data diversity is to create synthetic datasets via generative networks. However, it is unnatural to simply stick foreground objects onto arbitrary background images (alpha compositing), which introduces artifacts that may undermine the model. Besides, it may still require some additional labeling, *i.e.*, annotating foreground objects at pixel-level.

Several promising approaches were presented to tackle this problem, however, these methods are under different machine learning paradigms, *e.g.*, via semi-supervised learning that interpolates labeled data from unlabeled data, or self-supervised learning that exploit signals within unlabeled data via pretext tasks. Without changing the supervised learning paradigm, we propose a new approach of offline data augmentation for object detection via novel-view synthesis based on the differentiable neural rendering. The key motivation behind our approach is that novel views bring unseen yet realistic semantics and object locations, which are beneficial to a data-hungry object detector. Our system generates controllable views of training im-

ages given arbitrary novel camera poses, and produces corresponding bounding box annotations without human intervention. Specifically, we extract and project pixel-aligned image features into latent space point clouds based on the estimated depth maps, and mark keypoints of the annotations. We then re-project them with a target camera pose to render the corresponding novel-view 2D image. Keypoints in point clouds will recover annotated locations.

Our approach is advantageous in the following aspects: (1) It provides novel views of a training image, which feeds unseen semantics and novel locations to the detector. (2) The synthesized images are natural and have no artifacts. (3) Annotations are generated automatically for synthesized images; it requires no human labeling. (4) We can control the number of novel images to generate for each existing training sample, and have the full flexibility in controlling the camera pose that corresponds to the rendered image. This ensures diversity and is a great advantage over traditional GAN-based methods, where 3D attributes of a new scene cannot be explicitly controlled. (5) Our proposed method is fully compatible with any kind of online data augmentation methods, *e.g.*, affine transform, image mixup, *etc.* Thus, the combined models will further improve the performance.

In summary, we propose a cost-free approach to enrich images and labels, and show that it can significantly boost the performance of object detection systems suffering from scarce training data. Contributions of this work include:

- To the best of our knowledge, this is the first work to utilize the neural rendering for data augmentation in the object detection task. Our novel approach automates the annotation generation by rendering both images and annotations from cloud points.
- Our proposed method is 3D-aware and can augment long-tailed datasets in a controllable way. This method works for generic object detection and is compatible with online augmentation methods.
- Extensive experiments show that the proposed method can significantly boost the performance of object detection systems; the scarcity of training data is more severe, the improvement achieved by our method is larger.

2. Related Work

Convolutional Neural Networks (CNNs) have been successfully applied to computer vision tasks such as image classification [1], object detection [2], keypoint estimation [3], image segmentation [4] and visual tracking [5]. However, CNN Models with inadequate generalizability may overfit the training data and perform poorly on unseen data. Improving the generalization ability of CNN models is a most demanding challenge to solve, especially when annotated data is scarce, unbalanced (with skewed ratios of majority to minority samples), or semantically impoverished.

Existing approaches include semi-supervised learning methods [6, 7, 8] that interpolates labeled data from unlabeled data, self-supervised learning methods [9, 10] that exploit signals within unlabeled data via pretext tasks. Transfer Learning [11], One-shot and Zero-shot learning [12, 13] are other interesting paradigms for building models with extremely limited data. Conceptually similar to transfer learning, pre-training is training the network architecture on a big dataset such as ImageNet [14] or JFT-300M [15]. In this paper, we focus on data augmentation which tackles the root of the problem, the training dataset, without changing the standard supervised learning paradigm.

2.1. Data Augmentation

Data Augmentation [16] is a powerful method to mitigate the problem of data scarcity, as augmented data will represent a potentially more comprehensive set of data points, closing the gap between the training and testing sets. Usually, the methods can be classified as data warping and oversampling. Data warping augmentations transform existing images while preserving the labels. Oversampling augmentations create synthetic instances to add into the training set, often used to re-sample imbalanced class distributions. Methods can also be categorized into online and offline depending on when the augmentation process occur.

Online Methods: The earliest Data Augmentations started from simple transformations such as horizontal flipping, color space augmentations, and random cropping, followed by affine transformations, image mixup [1, 17], random erasing [18], feature space augmentation [19], etc. Applying meta-learning [20] concepts from *Neural Architecture Search* (NAS) [21] to data augmentation has become increasingly popular with works such as Neural Augmentation [22], Smart Augmentation [23], and AutoAugment [24], respectively. Optimal online augmentation policies can be automatically searched within a search space. A policy consists of various sub-policies, one of which is chosen for each image in a mini-batch.

Offline Methods: Following the introduction of GANs [25], data augmentation itself is augmented with methods such as adversarial training [26], GAN-based augmentation [27], neural style transfer [28]. Being computationally expensive, GAN-based methods usually offer offline augmentation, *i.e.*, constructing an extended dataset. Methods such as GANs and neural style transfer can 'imagine' alterations to images. However, such hallucination is artificial (often generated from random noise or limited conditional signals), the distribution of which may deviate from the original image, potentially leading the training process astray.

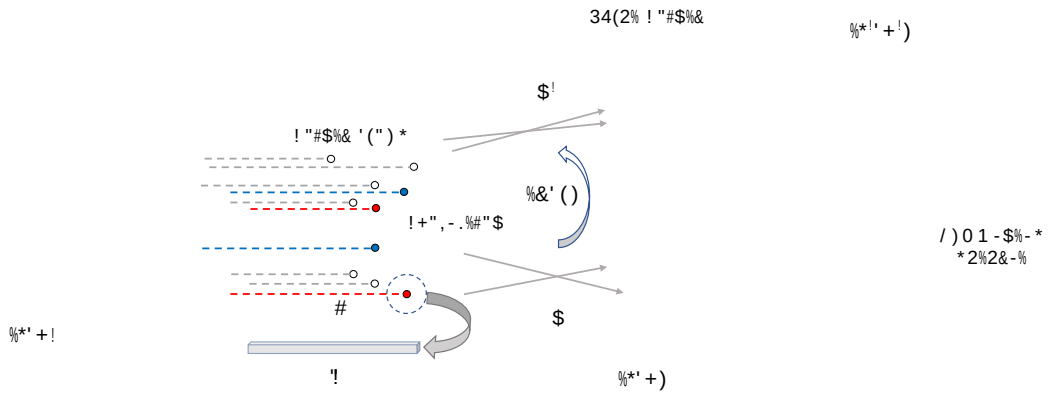


Figure 2. Neural renderer for object detection.

2.2. Neural Rendering

In cognitive computer vision domain, most existing tasks are related to perception, *i.e.*, perceive information from the image, video, voxel, mesh, or 3D point cloud. This is a process of either 2D or 3D reasoning. Typical tasks include object detection, human pose estimation, segmentation, 3D shape estimation, etc. In computer graphics domain, rendering is the process of generating images of 3D scenes defined by geometry, materials, scene lights and camera properties. The purpose of neural rendering [29, 30] is to bridge the gap between the 2D and 3D processing methods, allowing neural networks to optimize 3D entities while operating on 2D projections. Applications of differentiable neural rendering include: novel view synthesis [31, 32, 33, 34], semantic photo manipulation [35], facial and body reenactment [36, 37], re-lighting [38], free-viewpoint video [39], creating photo-realistic avatars [40] or simply generating high-quality images [41]. Our proposed augmentation method is inspired by novel-view synthesis [34], which is fully compatible with online data augmentation methods, and can be incorporated together to further inflate the dataset with novel semantics. GAN is described in [27] as a way to "unlock" additional information from a dataset. With neural rendering, we further unlock dataset information in highly controllable ways. The interpolation of data is non-linear and offers novel spatial semantics in 3D, which is extremely valuable for object detection tasks.

3. Proposed Method

In this section, we introduce the proposed Dataset Augmentation system based on Neural Rendering. We denote our method as *DANR*.

3.1. Overview

The overview of the DANR system is illustrated in Fig. 1. The purpose of the system is to augment an object detection dataset with novel view images to improve the performance of object detectors. The amount of augment images and degree of camera pose variations are both controllable. At its core, the system builds upon a novel view synthesis model, as shown in Fig. 2. The model takes as input an RGB image I and a series of 2D image keypoints B_i (representing bounding box annotations). The RGB image is simultaneously fed to two branches: (1) an hourglass network [42] for depth estimation, and (2) an Encoder with inception-resnet blocks [43] to extract visual features. The visual features are pixel-aligned, *i.e.*, each pixel on the feature maps carries a latent-space vector that corresponds to this particular spatial location. The depth estimation branch assigns depth value to each pixel, elevating latent-space vectors to a point cloud. Each point in the cloud resides in a 3D space with the 2.5D coordinate representation [44]: (u, v, d). Each point, however, may unfold its hidden N-dimensional latent-space vector after being projected to a target camera view. These feature vectors can be re-projected to an arbitrary camera coordinate system, based on a given projection matrix (translation vector T and rotation vector R for the corresponding camera of the original image). Given a target camera pose P', we splat the feature points following Wiles's method [34] to make the rendering process differentiable. After splatted on a palette of size $K \times K$, these points result in an N-dimentional feature map of our chosen resolution. The feature maps are then fed to a Decoder network with inception-residual blocks and upsampling layers, rendering the final novel view image.

In order to recover the bounding box annotations for object detection datasets, we mark the keypoints of the bounding boxes beforehand. When they are projected to 3D space with depth information and re-projected back to 2D in a novel camera view, the marks consistently follow corresponding pixel-aligned points. The whole system is end-to-end differentiable.

3.2. DANR

In this section, we describe the data augmentation system in details. Firstly, we elaborate the state-of-the-art networks we employ and the specific settings. Then we introduce

the point-cloud projection process and how to automate the generation of annotations in target views. Lastly, we depict the losses used to train the whole system.

Networks. We employ hourglass network as our depth estimator, as this UNet-like structure is proved advantageous in terms of exploring global interdependencies across multiple scales. Estimating the depth of a pixel requires understanding of the global features as well as the local features, which is important in perceiving relative depth. We stack two level-4 hourglass networks, with maximum channel number 256. We use a 1×1 filter at the last convolutional layer to produce the depth map. For the feature extractor network and refinement network after point splatting, we follow [34] to use Encoder-Decoder networks but makes several modifications: (1) when the input resolution is set to 512×512, we reduce the feature channel to 64; (2) we replace the basic resnet block with inception-resnet block, as the concatenation of features further consolidates the representative power.

Point Projection. We fix the intrinsic parameters for all images, assuming they are taken with a default camera. Specifically, we set focal lengths on both x and y axes to be 1 and set skew to be 0, while assuming principal point is at the origin of coordinates. Given a 2.5D point representation, we map it to the Camera Coordinate System (CCS) of the input image:

$$\begin{cases} X_c = d \times (u - c_x)/f_x, \\ Y_c = d \times (v - c_y)/f_y, \\ Z_c = d. \end{cases} \tag{1}$$

This CCS is now regarded as the World Coordinate System (WCS). Given a specific target view, we can determine the extrinsic parameters. Transforming from the WCS to a novel CCS is straight-forward with R and T:

$$P_c = R \cdot (P_w - T) \tag{2}$$

where $P_c = (X_c, Y_c, Z_c)$.

The points P_c are then splatted to a target camera palette with new pixel locations:

$$\begin{aligned} u' &= f_x \times X_c/Z_c + c_x, \\ v' &= f_y \times Y_c/Z_c + c_y. \end{aligned} \tag{3}$$

Within each bounding box i, points (at least the four points at corners) are marked with an index value i. These marked keypoints are a subset of $S \times S$ points, where S is the resolution of the extracted feature maps. When keypoints from the i_{th} bounding box are re-projected to the target view after the procedures described above, their new pixel locations are denoted as $P'_i = \{(u'_i, v'_i)\}$. The annotation for bounding box i in the target image is computed as:

$$\begin{cases} x_{min} = \min(\{u \mid (u', v') \in P'_i\}), \\ x_{max} = \max(\{u \mid (u', v') \in P'_i\}), \\ y_{min} = \min(\{v \mid (u', v') \in P'_i\}), \\ y_{max} = \max(\{v \mid (u', v') \in P'_i\}). \end{cases} \tag{4}$$

Loss Function. We follow [45] for the discriminator design, and use two multi-layer discriminators, the one with lower resolution serving as intermediate supervision to help the one with higher resolution learn better and faster. The overall loss is consisted of image-level L1 loss, feature-level content loss, and discriminator loss. The depth map is implicitly learned where we do not enforce supervised learning. The overall loss is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_D + \lambda_2 \mathcal{L}_{L1} + \lambda_3 \mathcal{L}_C. \tag{5}$$

3.3. Keypoint-based Object Detection

Keypoint estimation is naturally a regression problem [43]. The targets are represented by a series of heatmaps, each channel corresponding to a specific keypoint genre. Recent object detection methods such as CenterNet [2] , CornerNet [46] and ExtremeNet [47], begin to utilize keypoint estimation techniques. For instance, CenterNet transforms the task of object detection from generating and classifying proposals into predicting objects' centers (keypoints) and corresponding attributes. To object detection, the attributes are the width, height of the object, along with local offsets that recover pixel location in the original resolution from down-sampled heatmaps.

We use CenterNet as our baseline detection framework to conduct experiments and ablation studies, due to its simplicity. Because it is anchor-free and does not need NMS as a post-processing step, the ablation study can be decoupled from complex design choices that is not concerned with training data. **Anchor-free.** During training, an object does not need to be assigned with proper anchor(s). Instead, only heatmaps for the entire image are generated as the regression target. **NMS-free**. When the heatmaps for object centers are inferred, the local peaks are ranked based on their response, and only top K objects are extracted. With the center positions, corresponding attributes are extracted across their respective channels at the same 2D position. Since keypoints of different genres can occur at the same position, CenterNet is also naturally compatible with multi-label problems.

3.4. Keypoint-based Mixup

It is first demonstrated in [48] how mixing samples up could be developed into an effective augmentation strategy. Image mixup for the task of object detection has been described in [17], but it is restricted to bounding box mixup. We propose a straight-forward image mixup strategy for

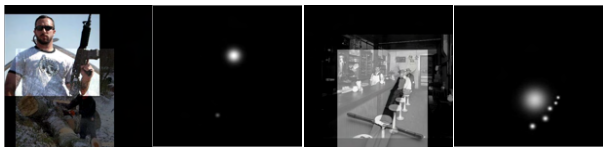


Figure 3. **Image mixup for keypoint-based object detection.** The radius of a heatmap blob is proportional to the size of the object, as illustrated in Equation 6. Objects with higher weights correspond to keypoints with stronger response.

keypoint-based object detection methods, where ground truth keypoints are splat onto a heatmap $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ using a Gaussian kernel:

$$Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2 + (y-\tilde{p}_y)^2}{2\sigma_p^2}\right) \quad (6)$$

σ_p is an adaptive standard deviation [46] that is proportional to the size of the object. During mixup, the confidence of keypoint heatmaps are applied with the same weights used on its corresponding image, as shown in Fig. 3. This mixup strategy can be applied to keypoint-based object detection methods such as CenterNet. Compatible with the proposed offline augmentation method, image mixup is used in all of our experiments as an online augmentation method.

4. Experiments

4.1. Dataset

Dataset: Data Augmentation. RealEstate10K [49] is a large dataset of camera parameters (intrinsic and extrinsic) corresponding to 10 million frames derived from about $80,000$ video clips, gathered from about $10,000$ YouTube videos. For each clip, the poses form a trajectory where each pose specifies the camera position and orientation along the trajectory. These poses are derived by running SLAM and bundle adjustment algorithms on a large set of videos. We download Youtube videos with the given URLs and then extract image frames. Note that we only use $10,758$ out of $82,315$ clips gathered from Youtube videos. We also trained the neural renderer with Matterport3D [50] and Replica[51], synthetic datasets in the format of Habitat [52]. However, we notice the model trained with Realestate10K has better generality as it is trained with real-world data instead of synthetic data. In the following experiments, we use numerous augmentation systems that are trained on Realestate10K to aid the training of object detection networks.

Dataset: Object Detection. To validate the effectiveness of DANR, we generate augmented datasets for Indoor object-detection [53] dataset to mimic the constrained environment that is similar to RealEstate10K where we train the neural renderer. We conduct experiments on Indoor dataset where we constrain to a subset of the dataset to simulate limited data problems. We also test on real-world data in an image content review system, which consists virtually any kind of images that users may upload. We organize these data into ICR-Weapon, ICR-Flag and ICR-Logo datasets. Although not captured in a constrained environment, ICR benefits drastically from the proposed DANR method because ICR datasets suffer from limited positive samples in contrast to massive negative samples, where we need to improve the recall of positive samples while maintaining a very high through rate. We further attempt to push boundaries on the generalized object detection dataset COCO [54]. COCO is neither long-tailed nor insufficient in samples, so we perform DANR on COCO to measure its scope of limitations.

4.2. Metrics

Metrics: Synthesis Quality. We use metrics peak-signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) to assess the objective image quality. Since the metrics of PSNR and SSIM are linked [55], we also resort to perceptual similarity (PSIM) [56] metric to compare the similarity of images. Although it remains challenging to determine the similarity of images when human judgement is involved [56], these metrics altogether provide a more robust estimation of relative quality of synthesized images. We extract feature stack from L layers and unit-normalize in the channel dimension, which we designate as $\hat{y}^l, \hat{y}^l_0 \in \mathbb{R}^{H_l \times W_l \times C_l}$ for layer l. We scale the activations channel-wise by vector $w^l \in \mathbb{R}^{C_l}$ and compute the ℓ_2 distance. Finally, we average spatially and sum channel-wise. Note that using $w_l = 1 \forall l$ is equivalent to computing cosine distance.

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||_2^2 \quad (7)$$

Metrics: Object Detection. We use Average Precision (AP) as object detection metrics. For real-world image review datasets: ICR-Weapon, ICR-Flag, ICR-Logo, we report AP as detection metric and AUC as review metric.

4.3. Implementation Details

For DANR, we crop input images to 512×512 for feature extraction and depth estimation. We also use a palette of the same size to splat points. We train on 4 V100 GPUs with batch size 8, randomly choosing samples for 500 iterations per epoch. The sampled viewpoints of a reference video frame and its paired video frame are within 30 frame range. The network is trained for a total of 275 epochs, which takes 3.5 days. Here are the corresponding metrics of the model, evaluated on RealEstate10K validation set: (1) the SSIM metric is 0.4386; (2) PSNR is 7.1564 and (3) perceptual score is 0.5964.

4.4. Baseline

We use CenterNet as our baseline detection framework to conduct experiments and ablation studies. We use iden-

Table 1. Datasets used for neural rendering and object detection.

Dataset	Type	Domain	Purpose	Annotation	#Category	#Boxes
RealEstate10K	Video	Indoor	Novel View Synthesis	-	-	-
Indoor-Objects	Image	Indoor	Object Detection	Bounding Box	7	4,595
COCO	Image	Indoor & Outdoor	Object Detection	Bounding Box	80	860,001
ICR-Weapon	Image	Real-world Online Data	Object Detection	Bounding Box	29	3,887
ICR-Flag	Image	Real-world Online Data	Object Detection	Bounding Box	1	2,002
ICR-Logo	Image	Real-world Online Data	Object Detection	Bounding Box	1	1,365
ICR-Norm10K	Image	Real-world Online Data	Evaluation	-	-	-

Auto-Generated Annotations

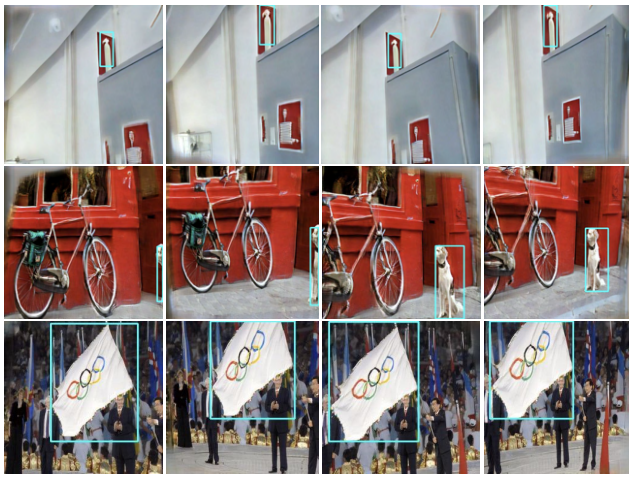


Figure 4. **Auto-generated annotations.** The bounding box annotations are precise for rendered images.

tical training settings for all ablation experiments. Regardless of backbones, the network performs affine transform to keep aspect ratio and use 512×512 image patch as input. Each network is trained for a total of 120 epochs. We use Adam optimizer [57]. Learning rate is initiated to 2.5×10^{-4}, which drops half twice at epoch 90 and 110, respectively. We do not perform any test-time augmentations, such as mirror flipping and multi-scale testing. For datasets except COCO, each image is augmented with four pre-defined views. They share the same translation vector $\begin{bmatrix}0 & 0 & 0.3\end{bmatrix}$. Their respective rotation vectors are: $\begin{bmatrix}-0.1 & -0.15 & 0\end{bmatrix}$, $\begin{bmatrix}0.1 & -0.15 & 0\end{bmatrix}$, $\begin{bmatrix}-0.1 & 0.15 & 0\end{bmatrix}$, $\begin{bmatrix}0.1 & 0.15 & 0\end{bmatrix}$. For COCO dataset, we augment each image by randomly choosing one of the pre-defined views.

4.5. Ablation Study

In this section, we ablate various choices in the use cases of our proposed *DANR*. For simplicity, all accuracy results here are for Indoor-Object validation set. Specifically, we show that the improvement is consistent regardless of detector configurations on various data distributions. We also discuss additional characteristics of augmentation such as the impact of image resolution. The following comparisons will be made. (1) Augmentation: online vs. offline; (2) render: high-resolution vs low-resolution; (3) Affine Transform: w/ affine vs. w/o affine; (4) data scarcity level: low vs. high; (5) detector: keypoints-based (one-stage) vs proposal-based (two-stage); (6) backbone: heavy vs. lightweight.

Aug-types	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	67.2	94.7	77.5	58.8	60.9	70.5
+ Online	69.4	95.4	78.4	56.5	59.1	72.8
+ Offline	77.8	96.9	90.4	54.8	59.9	81.5
+ Online + Offline	78.4	97.3	90.8	54.2	59.8	82.1

Table 2. **Compatibility: Online vs. Offline.** Performance for augmentation types on Indoor validation set.

Compatibility: We check the compatibility of the proposed DANR with online augmentation methods. In our baseline setting, we use CenterNet as our detection framework, with a pre-trained ResNeSt-50 backbone. We add a keypoint-based image mixup strategy to represent online augmentation, denoted as ***Online***. We add the data generated offline by the DANR to training, denoted as ***Offline***. In all of the following experiments, we only compare *+Online* vs. *+Online+Offline*, denoted as *N.A.* and *+Aug*, respectively. Table 2 shows that DANR is compatible with online mixup. Their augmentation strategies are complementary. DANR alone can boost the performance by around 10 percent.

Render Resolution: With DANR, the detection performance is significantly improved. However, we notice that the average precision for small objects suffer a performance drop. We hypothesize that it is due to the small render resolution (*e.g.* 256×256) compared to the detector input resolution (*e.g.* 512×512).

In order to generate images at higher resolution, we experiment with 3 different settings: (1) Without any further training, we apply the network trained on a smaller image size to a different size at test time. Specifically, we splat the 256^2 cloud points onto 512×512 palette maps, then render the maps to an image with a refinement network. This augment method increased image resolution by super-sampling of splatted points, and is denoted as ***512-pointSS***. (2) We first augment images at 256×256, and further upsample pixels at the image level with a pre-trained super-sampling network [58]. We denote this method as ***512-pixelSS***. (3) We re-train a network from scratch that takes in images at 512×512, generate feature maps at 512×512, and sample

Figure 5. **Augmentation with different rendering resolutions and techniques for Indoor dataset**. Columns from left to right: Original image, rendered images with: 256-Native, 512-PixelSS, 512-PointSS, 512-Native.

512^2 feature points in the cloud, which are splatted and rendered on the output image. This method naturally outputs images in 512×512 resolution, denoted as ***512-native***.

Render-Res	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
N.A.	69.4	95.4	78.4	56.5	59.1	72.8
256-native	78.4	97.3	90.8	54.2	59.8	82.1
512-pixelSS	79.0	97.5	91.4	63.2	65.5	82.5
512-pointSS	79.1	97.6	92.0	64.5	70.4	82.4
512-native	79.6	97.7	91.4	61.4	67.3	83.2

Table 3. **Render Resolution: High-Res vs. Low-Res.** Performance on Indoor validation set with augmentation at different resolutions.

As illustrated in Table 3, (1) augmentation with images rendered at different resolutions consistently boosts the detection performance; (2) synthesized images at low resolutions may potentially lose some details compared to real images, which does harm to the detection performance of very small objects; (3) uplifting the image resolution via super-sampling further improves the performance; (4) super-sampling from points may be a better choice than super-sampling from pixels; (5) training a network with dense features and points achieves the best performance. In the following experiments, we use 512-native as our resolution setting, as it achieves the highest AP and does not need any additional super-sampling networks.

Affine Transform: As illustrated in Table 4, (1) depriving affine transform from the default online augmentation during training impairs the performance; (2) offline data augmentation improves performance with or without affine transformation; (3) to some extent, offline augmentation can make up for the loss of affine transform during training.

Data Scarcity: We further experiment with three data-split settings: 5:5, 3:7 and 1:9. Highest degree of scarcity is 1:9. While splitting images to train and validation sets, an image is assigned along with its corresponding augmented images. Table 5 shows that the performance boost is closely related with the scarcity of training data. When annotated data is very limited, the DANR approach becomes more beneficial.

	AP		AP_{50}		AP_{75}	
Baselines	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug
w/ affine	69.4	79.6	95.4	97.7	78.4	91.4
w/o affine	57.7	79.0	93.0	98.0	63.9	91.3

Table 4. **Affine Transform: w/ affine vs. w/o affine.**

	AP		AP_{50}		AP_{75}	
Train/Val	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug
5/5	69.4	79.6	95.4	97.7	78.4	91.4
3/7	56.7	71.0	87.7	91.9	65.3	83.1
1/9	11.9	47.2	27.5	74.8	8.5	51.4

Table 5. **Scarcity Level: High vs. Low.** Performance (AP) on Indoor validation set with different split settings. The train:val ratio is 5:5, 3:7, 1:9, respectively.

	AP		AP_{50}		AP_{75}	
Backbone	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug
ResNet-50	53.2	77.4	77.0	96.4	60.3	90.7
ResNeSt-50	69.4	79.6	95.4	97.7	78.4	91.4

Table 6. **Backbone: ResNet-50 vs. ResNeSt-50.**

Backbone: We fix the detector to CenterNet while comparing different backbones. Table 6 shows that the proposed augmentation consistently boost the detection performance, regardless of the backbones used by the detector.

	AP		AP_{50}		AP_{75}	
Framework	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug
One-Stage	69.4	79.6	95.4	97.7	78.4	91.4
Two-stage	68.1	78.7	96.4	97.5	81.5	89.3

Table 7. **Detector: One-stage vs. Two-stage.** Performance for different detectors on Indoor validation set. We show that augmentation is consistently helpful regardless of detection frameworks. We use CenterNet as an example of one-stage method, while using Hit-Detector as a two-stage method.

Detector: Lastly, we compare one-stage and two-stage detection frameworks. We use CenterNet as an example of one-stage method, while using Hit-Detector [59] to represent the two-stage approach. We use the official code re-

	AP		AP_{50}		AP_{75}		AP_S		AP_M		AP_L		AUC	
Dataset	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug	N.A.	+Aug
Indoor	69.4	79.6	95.4	97.7	78.4	91.4	56.5	61.4	59.1	67.3	72.8	83.2	-	-
ICR-Weapon	-	-	-	-	-	-	-	-	-	-	-	-	0.68	0.75
ICR-Logo	-	-	-	-	-	-	-	-	-	-	-	-	0.73	0.96
ICR-Flag	-	-	-	-	-	-	-	-	-	-	-	-	0.72	0.97
COCO	33.4	33.1	52.8	52.6	34.8	34.5	12.8	12.6	38.0	37.6	51.6	51.8	-	-

Table 8. Performance on different data settings. We use CenterNet as our framework, with ResNeSt-50 backbone. N.A. denotes settings without DANR (with online augmentation); +Aug denotes settings with DANR.

Figure 6. **Image augmentation results for COCO dataset.** Columns from left to right: Original image, rendered image views 1 to 4. The extreme case is, when depth for each pixel is identical, the augmentation is equivalent to affine transform, which is linear.

lease of Hit-Detector to perform training and testing. Not surprisingly, *DANR* does not rely on the one-stage detection framework. It is beneficial to two-stage detectors as well.

4.6. Experimental Results

Summarizing the experimental results on various datasets, as shown in Table 8, we have come to the following insights: (1) Results on image content review datasets (ICR-Weapon, ICR-Flag and ICR-Logo) show that the proposed DANR is consistently helpful on real-world data when training data is scarce. (2) Results on indoor dataset show that the augmentation is highly beneficial with indoor data, even when training data is not scarce. (3) Results on COCO dataset have shown that DANR makes little contribution under circumstances where (a) the neural renderer in DANR fails to fully generalize to the semantics of the particular dataset, and (b) the training data is abundant.

5. Discussions

Overall, the role DANR plays in training an object detector on COCO dataset is neglectable, although the recall on large objects have improved over networks trained with the raw COCO dataset. However, the overall AP is lower due to inferior performance on small objects. Experimental results show that when annotated data is abundant, COCO dataset for instance, (1) augmented images with lower resolution may impair the network's ability in detecting small objects; (2) the augmentation quality becomes crucial in boosting object detection performance, which is highly correlated with the synthesis resolution as well as the accuracy in depth estimation. For models trained with indoor datasets, if augmenting COCO images in lower resolution, it does not provide COCO so much novel semantic data as to make up for its loss resulted from resolution degradation. In short, DANR needs more outdoor training data to improve its generality. This limitation will be mitigated with the maturity of future generative models.

6. Conclusion

In this paper, we have successfully developed a new method of data augmentation for object detection based on differentiable neural rendering. Compared with existing methods, neural rendering offers a more promising generative modeling technique for use in data augmentation, as it bridges the 2D and 3D domain, and offers highly controllable ways to manipulate images. Extensive experimental results show that the proposed method significantly improves the detection performance when annotated data is scarce, and is a very promising counterpart in dealing with long-tailed problems, along with other machine learning paradigms such as semi-supervised learning. In future research, we foresee the augmentation based on neural rendering effectively applied to other cognitive vision tasks.

References

[1] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *CVPR*, 2019. 2

[2] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019. 2, 4

[3] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019. 2

[4] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," in *CVPR*, 2020. 2

[5] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *CVPR*, 2019. 2

[6] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big self-supervised models are strong semi-supervised learners," *NIPS*, 2020. 2

[7] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *CVPR*, 2019. 2

[8] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. V. Le, "Rethinking pre-training and self-training," *NIPS*, 2020. 2

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020. 2

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020. 2

[11] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *arXiv preprint arXiv:1911.02685*, 2019. 2

[12] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019. 2

[13] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, 2020. 2

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009. 2

[15] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation learning," *arXiv preprint arXiv:1912.11370*, 2019. 2

[16] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, 2019. 2

[17] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of freebies for training object detection neural networks," *arXiv preprint arXiv:1902.04103*, 2019. 2, 4

[18] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation.," in *AAAI*, 2020. 2

[19] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," *arXiv preprint arXiv:1702.05538*, 2017. 2

[20] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018. 2

[21] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018. 2

[22] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017. 2

[23] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," *Ieee Access*, vol. 5, pp. 5858–5869, 2017. 2

[24] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018. 2

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014. 2

[26] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014. 2

[27] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert, "Gan augmentation: Augmenting training data using generative adversarial networks," *arXiv preprint arXiv:1810.10863*, 2018. 2, 3

[28] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015. 2

[29] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Differentiable rendering: A survey," *arXiv preprint arXiv:2006.12057*, 2020. 3

[30] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, *et al.*, "State of the art on neural rendering," *arXiv preprint arXiv:2004.03805*, 2020. 3

[31] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang, "3d photography using context-aware layered depth inpainting," in *CVPR*, 2020. 3

[32] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *arXiv*, 2020. 3

[33] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang, "3d photography using context-aware layered depth inpainting," in *CVPR*, 2020. 3

[34] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *CVPR*, 2020. 3, 4

[35] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural photo editing with introspective adversarial networks," *arXiv preprint arXiv:1609.07093*, 2016. 3

[36] S. Wu, C. Rupprecht, and A. Vedaldi, "Unsupervised learning of probably symmetric deformable 3d objects from images in the wild," in *CVPR*, 2020. 3

[37] L. Liu, W. Xu, M. Zollhoefer, H. Kim, F. Bernard, M. Habermann, W. Wang, and C. Theobalt, "Neural rendering and reenactment of human actor videos," *ACM Transactions on Graphics (TOG)*, 2019. 3

[38] Z. Chen, A. Chen, G. Zhang, C. Wang, Y. Ji, K. N. Kutulakos, and J. Yu, "A neural rendering framework for free-viewpoint relighting," in *CVPR*, 2020. 3

[39] R. Martin-Brualla, R. Pandey, S. Yang, P. Pidlypenskyi, J. Taylor, J. Valentin, S. Khamis, P. Davidson, A. Tkach, P. Lincoln, *et al.*, "Lookingood: Enhancing performance capture with real-time neural re-rendering," *arXiv preprint arXiv:1811.05029*, 2018. 3

[40] A. Shysheya, E. Zakharov, K.-A. Aliev, R. Bashirov, E. Burkov, K. Iskakov, A. Ivakhnenko, Y. Malkov, I. Pasechnik, D. Ulyanov, *et al.*, "Textured neural avatars," in *CVPR*, 2019. 3

[41] C. Gao, Q. Liu, Q. Xu, L. Wang, J. Liu, and C. Zou, "Sketchycoco: Image generation from freehand scene sketches," in *CVPR*, 2020. 3

[42] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016. 3

[43] G. Ning, Z. Zhang, and Z. He, "Knowledge-guided deep fractal neural networks for human pose estimation," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1246–1259, 2018. 3, 4

[44] U. Iqbal, P. Molchanov, T. Breuel Juergen Gall, and J. Kautz, "Hand pose estimation via latent 2.5 d heatmap regression," in *ECCV*, 2018. 3

[45] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018. 4

[46] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *ECCV*, 2018. 4, 5

[47] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *CVPR*, 2019. 4

[48] H. Inoue, "Data augmentation by pairing samples for images classification," *arXiv preprint arXiv:1801.02929*, 2018. 4

[49] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 37, 2018. 5

[50] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017. 5

[51] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019. 5

[52] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *ICCV*, 2019. 5

[53] B. Adhikari, J. Peltomaki, J. Puura, and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, 2018. 5

[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014. 5

[55] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*, IEEE, 2010. 5

[56] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018. 5

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 6

[58] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *ECCV*, 2018. 6

[59] J. Guo, K. Han, Y. Wang, C. Zhang, Z. Yang, H. Wu, X. Chen, and C. Xu, "Hit-detector: Hierarchical trinity architecture search for object detection," in *CVPR*, 2020. 7

DANR: Appendix

We give additional qualitative results in Section A, additional details of datasets in Section B, additional architectural details in Section C, and finally more information about the neural renderer in Section D.

A. Additional Qualitative Results

We give additional qualitative results, including rendered images with novel views on Indoor (Fig. 7), ICR (Fig. 8), and COCO (Fig. 9). We also show automatically generated bounding box annotations (Fig. 10 and Fig. 11).

B. Additional Details on Datasets

ICR dataset. Collected from an online image content review system, the dataset is extremely unbalanced as positive samples are very rare compared to the negative samples, with a ratio of 1:100. The positive samples are images potentially risky to exhibit in a system that seeks to retain certain types of visitors. The negative samples, on the other hand, are images safe to display.

One of the key metrics for evaluating such a review system is AUC score of the precision-recall curve. Specifically, the recall rate of positive samples and through rate of negative samples are measured for various thresholds. It is easy to overfit to the limited number of positive samples and mis-classify large amounts of negative images, because the visual features of the positive samples can be prevalently found in the negative samples. The ICR dataset is a much larger dataset compared to Indoor Objects dataset, in terms of the negative samples involved in training.

Simply feeding images with empty heatmaps, our baseline CenterNet is advantageous in training with negatives samples. On top of the baseline, CenterNet with the proposed DANR is proven extremely helpful in improving the AUC score on ICR datasets, including ICR-Flag, ICR-Logo and ICR-Weapon.

Figure 7. Additional results of augmented images on Indoor Objects. Zoom in for details.

C. Additional Details on Network Architecture

Here we illustrate the precise details of network architectures used to build the components of DANR.

Inception-Residual blocks. We first introduce the inception-residual block, the basic module that incorporates both inception design and ResNet design, i.e.,

Figure 8. Additional results of augmented images on ICR-Flag, ICR-Logo and ICR-Weapon. The images are carefully selected due to the sensitivity of the data. Zoom in for details.

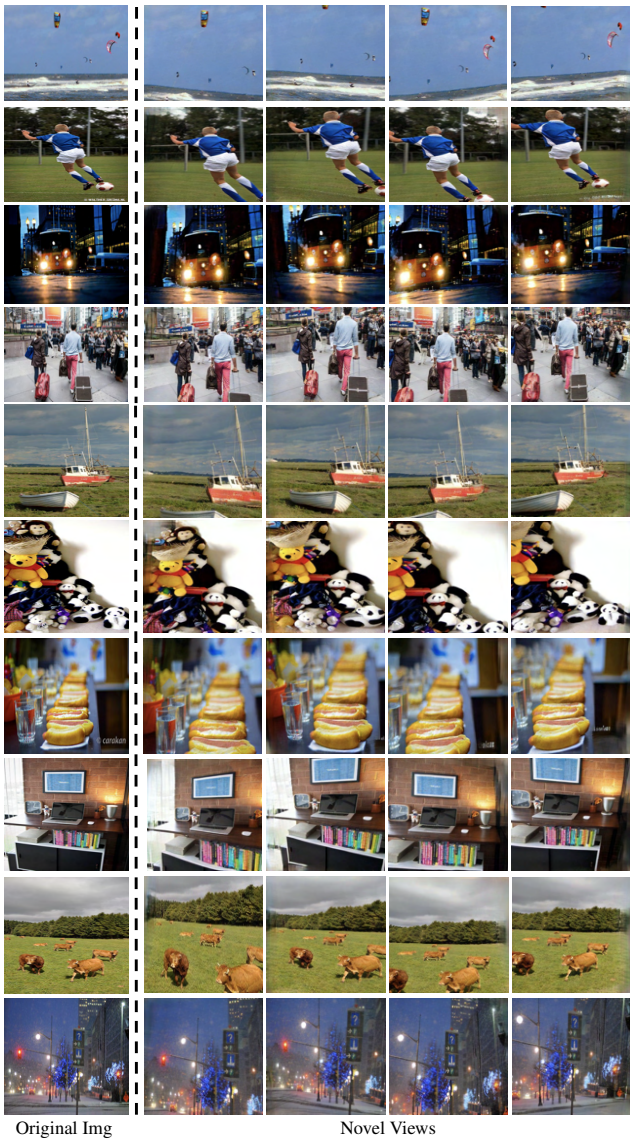


Figure 9. Additional results of augmented images on COCO. Zoom in for details.

channel-wise concatenation of two branches and pixel-wise addition with an identity branch. As illustrated in Fig. 12, the block can be configured to maintain, increase or decrease the resolution of features, with an identity layer (Fig. 12(a)), an upsampling layer (Fig. 12(b)) and an average pooling layer (Fig. 12(c), respectively.

Encoder. Inception-residual blocks are stacked together to form the embedding network. Specifically, our setup is illustrated in Fig. 13(a).

Decoder. Similarly, inception-residual blocks are stacked together to form the decoder network. Our specific setup is shown in Fig. 13(b). The resolution of input features of the Decoder (H', W') is by default identical to the features extracted by the Encoder. When the 3D points are splatted to a larger palette, e.g., $\times 2$ resolution, the feature map is upsampled by the scale of 2, which will uplift the image render resolution. We refer to this super-resolution method as ***PointSS***, in contrast to ***PixelSS*** where the rendered image are fed to an external super-resolution network.

Figure 10. Additional results of automatic annotations on Indoor dataset. Zoom in for details.

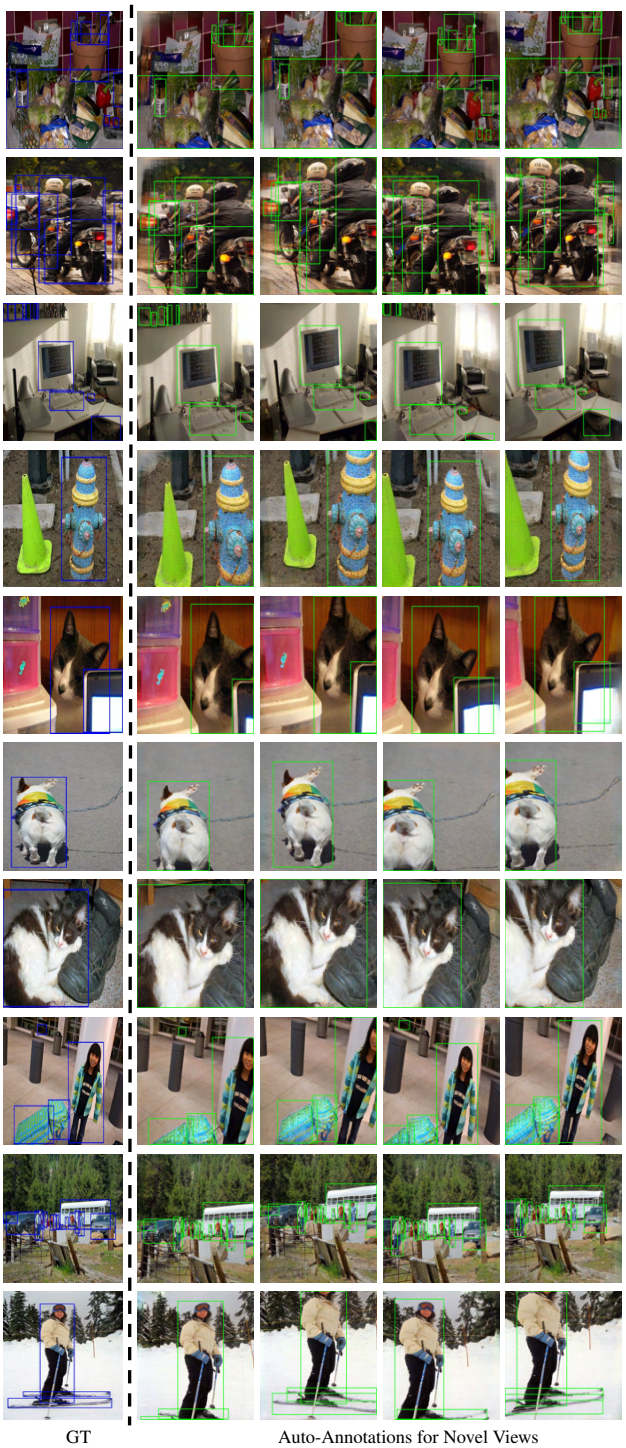
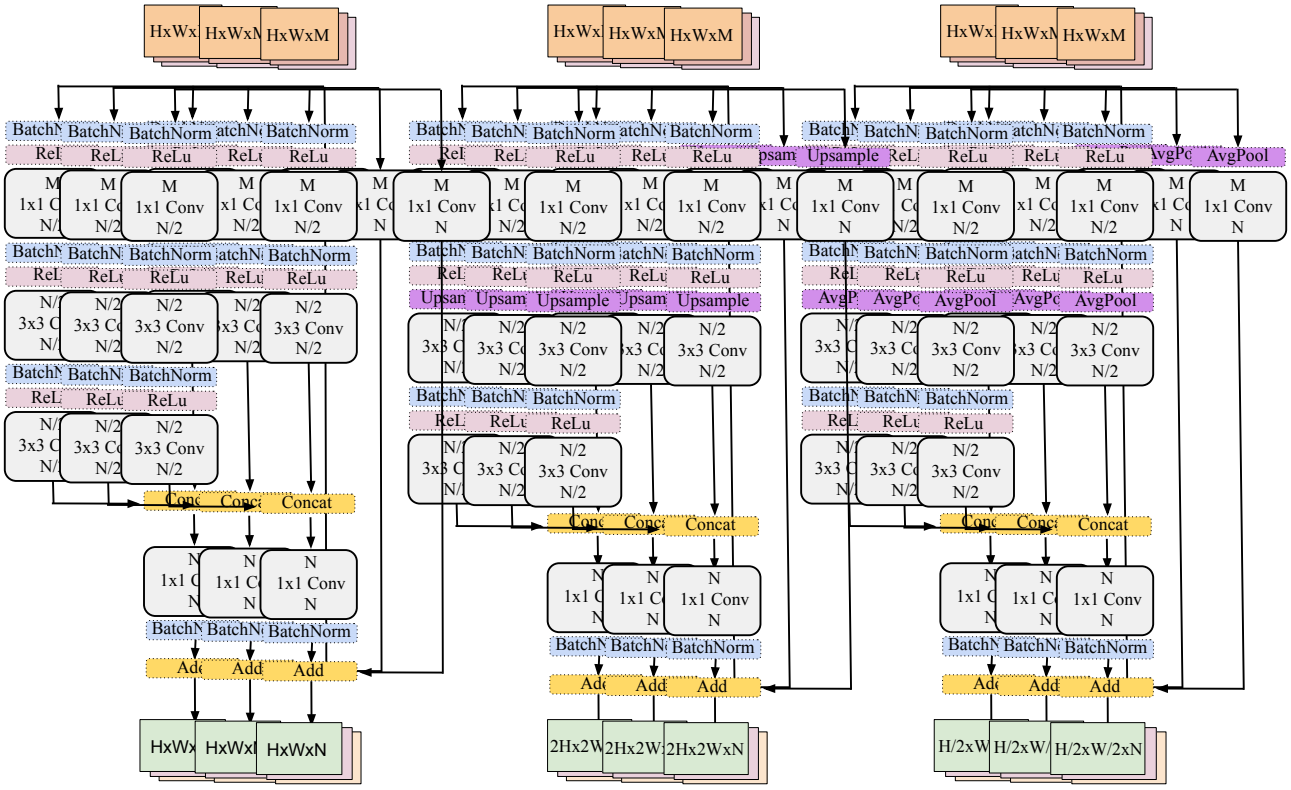


Figure 11. Additional results of automatic annotations on COCO dataset. Zoom in for details.

Depth estimator. The depth estimation network uses an hourglass architecture, as illustrated in Fig. 14. We simplify the original hourglass network by replacing the *pooling layer + resnet blocks* scheme with a *Down-sampling Block* that consists of a sequence of Leaky ReLU, convolution (stride 2 or 4, padding 1, kernel size 4), and batch normalization layers. A *Up-sampling Block* consists of a sequence of ReLU, a 2x or 4x bilinear upsampling, convo-



(a) Inception-Residual block.

(b) Inception-Residual block with upsampling layers.

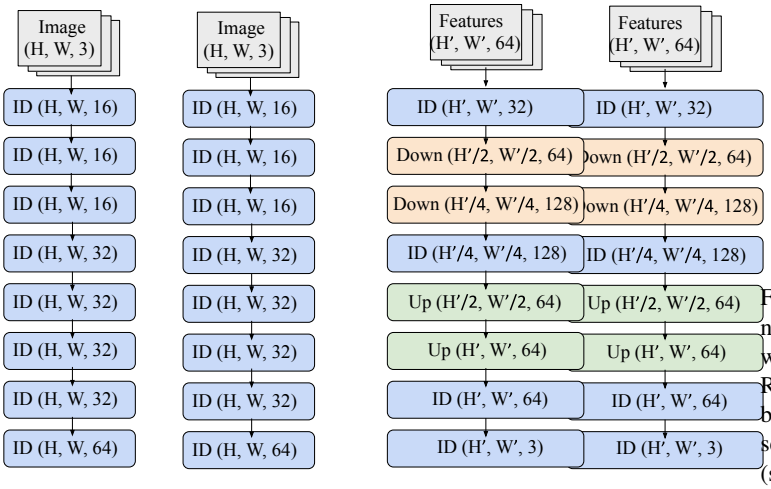
(c) Inception-Residual block with average pooling layers.

Figure 12. An overview of Inception-Residual blocks. We show the basic block in (a) that maintains the resolution, in (b) when we uplift the resolution, and in (c) when we downscale the resolution.

lution (stride 1, padding 1, kernel size 3), and batch normalization layers. We concatenate the output feature maps of stacked hourglass sub-networks and regress depth at the last layer (no batch normalization layer).

D. Additional Details on Differentiable Rendering

Before being splatted onto a palette, 3D cloud points in the new view are sorted in depth using a z-buffer. Naively, for all points in the new view, the nearest point in depth (by poping the z-buffer) is chosen to color that point. We follow the differentiable renderer in order to provide gradients with respect to the point cloud positions. In the differentiable renderer, K nearest points for each pixel are splatted; each of these points influences a range that originates from the splatted pixel with radius r, the influence of which is proportional to the Euclidean distance from the center. The sorted points are then accumulated using linear alpha over-compositing. In this way, the hard z-buffer becomes differentiable. An example is shown in Fig. 15. In our usage case of DANR, $K = 128$, $r = 4$.



(a) Encoder. (b) Decoder.

Figure 13. Encoder-Decoder Networks built with aforementioned inception-residual blocks: Identity (ID), Upsampling (Up), Average Pooling (Down). The resolution of input features of the Decoder (H', W') is by default identical to the features extracted by the Encoder. When the 3D points are splatted to a larger palette, e.g., $\times 2$ resolution, the feature map is upsampled by the scale of 2, which will uplift the image render resolution. We refer to this super-resolution method as ***PointSS***, in contrast to ***PixelSS*** where the rendered image are fed to an external super-resolution network.

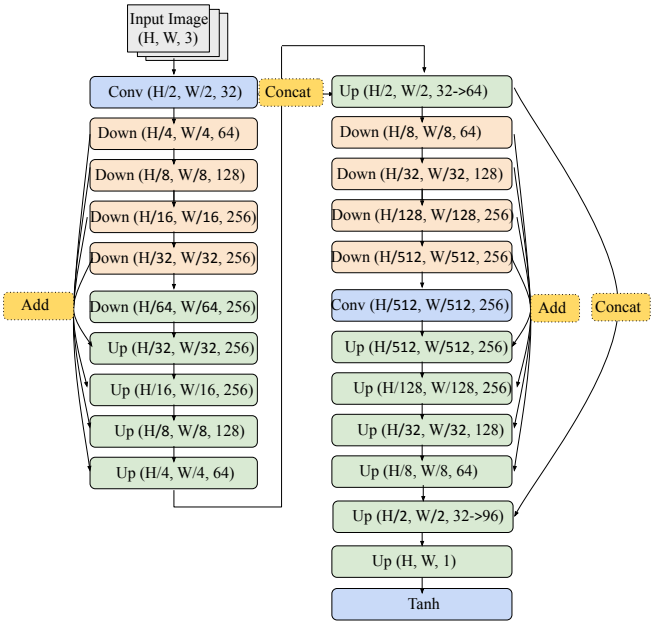


Figure 14. Depth estimation network. We simplify the hourglass network by replacing the *pooling layer + resnet blocks* scheme with a *Down-sampling Block* that consists of a sequence of Leaky ReLU, convolution (stride 2 or 4, padding 1, kernel size 4), and batch normalization layers. A *Up-sampling Block* consists of a sequence of ReLU, a 2x or 4x bilinear upsampling, convolution (stride 1, padding 1, kernel size 3), and batch normalization layers. We concatenate the output feature maps of stacked hourglass sub-networks and regress depth at the last layer (no batch normalization layer).

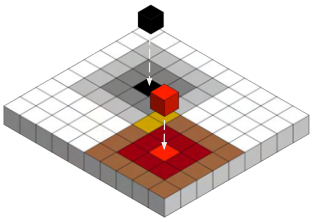


Figure 15. Differentiable renderer. In this example, two points from the z-buffer are splatted onto a palette with an influence range of radius 3.