Object-level Scene Context Prediction

Xiaotian Qiao, Quanlong Zheng, Ying Cao, and Rynson W.H. Lau

Abstract—Contextual information plays an important role in solving various image and scene understanding tasks. Prior works have focused on the extraction of contextual information from an image and use it to infer the properties of some object(s) in the image or understand the scene behind the image, e.g., context-based object detection, recognition and semantic segmentation. In this paper, we consider an inverse problem, i.e., how to hallucinate the missing contextual information from the properties of standalone objects. We refer to it as object-level scene context prediction. This problem is difficult, as it requires extensive knowledge of the complex and diverse relationships among objects in the scene. We propose a deep neural network, which takes as input the properties (i.e., category, shape, and position) of a few standalone objects to predict an object-level scene layout that compactly encodes the semantics and structure of the scene context where the given objects are. Quantitative experiments and user studies demonstrate that our model can generate more plausible scene contexts than the baselines. Our model also enables the synthesis of realistic scene images from partial scene layouts. Finally, we validate that our model internally learns useful features for scene recognition and fake scene detection.

Index Terms—Scene context, object inference, object properties, scene understanding.

1 Introduction

A scene context refers to how the objects of interest are related to the surrounding environment, *i.e.*, the spatial relations among the co-occurring objects around the objects of interest. Previous works have attempted to leverage the scene contextual information *already existed* in an image to infer the properties of some objects of interest in the image, *e.g.*, object detection [1], [2], recognition and segmentation [3], [4], [5], [6], and visual representation learning [7]. However, an unexplored problem is to hallucinate the *unknown* scene context of some given objects in the image (*i.e.*, to anticipate what and where the missing objects are). Given only a few foreground objects, humans are remarkably capable of inferring the *unknown* scene context, by relying on extensive commonsense knowledge of our visual world. For example, as shown in Figure 1, given a foreground object, we humans can reason about multiple plausible environments surrounding it. The properties of the given object, *i.e.*, object category, shape, size and position, provide strong hints about what and where the missing objects may appear in the scene.

Thus, we are interested in a fundamental question of whether machines can replicate such a scene context inference capability to predict what and where the missing objects are. We believe that such a capability can benefit the vision community and be applied to many scene generation and recognition tasks. For example, state-of-the-art methods for caption-conditioned image generation need the full scene context to generate realistic images. However, developing a model for scene context prediction can be challenging, as natural scenes often contain a rich variety of semantic objects with complex spatial relations among them. Objects can be at various locations in the image with different scales and shapes. In addition, this problem is inherently ambiguous, as the same objects can have multiple semantically plausible scene contexts. It is difficult to evaluate which scene context is more appropriate.

X. Qiao, Q. Zheng, Y. Cao, and R.W.H. Lau are with the Department of Computer Science, City University of Hong Kong.
Y. Cao and R.W.H. Lau are the corresponding authors.

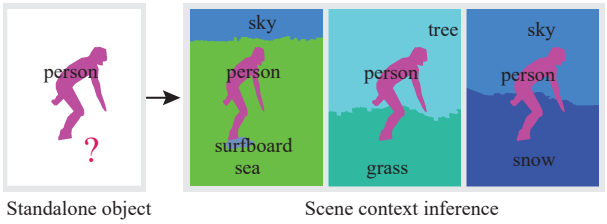


Fig. 1: Inferring scene contexts from a standalone object. A standalone object provides rich information for predicting its scene context (*i.e.*, other objects that co-occur with it and their spatial relations). While the pose and position of the person in the image suggest that the scene may be related to sports activities, the presence and position of the person provide hints as to what and where other objects can appear (*e.g.*, the sky in the upper part of the image, and the sea in the lower part of the image).

In this paper, we propose a new task of scene context prediction from standalone objects. Given the categories, shapes and positions of one or more foreground objects, we propose a novel model to predict the scene context for these objects. Instead of directly hallucinating low-level pixels, the context predicted by our model is in the form of an object-level scene layout. Although lacking detailed object appearances, such a mid-level representation can capture the important semantics shown to be sufficient to generate photo-realistic images [8] as well as build scene structures [9]. Our model has three modules: a shape generator, a region generator, and a compositor. The shape generator aims to generate object shapes of different categories. The region generator aims to generate object bounding boxes to indicate possible object positions and sizes. The compositor aims to generate a scene layout that represents the scene context in coherent with the input objects, by fusing the outputs from the two generators.

To evaluate the effectiveness of our model, we conduct quantitative experiments and user studies on the pre-processed COCO-Stuff dataset [10]. Experimental results show that our model can generate more plausible scene layouts that put the input objects

into the right context, as compared with the baselines. In addition, we demonstrate wide applications of our model. First, we show that our model enables an image synthesis approach that can generate full scene images from partial scene layouts. Second, we validate that learning to hallucinate scene contexts can internally obtain useful features for scene recognition. Finally, we show the benefits of our model on fake scene detection.

In summary, the main contributions of this paper include:

- To our knowledge, we make the first attempt to address the problem of predicting the unknown scene context where some objects of interest reside in.
- We propose a novel neural network model to predict object-level scene contexts from just standalone foreground objects.
- We demonstrate the benefits of scene context prediction on a variety of tasks, including image synthesis, scene recognition and fake scene detection.

A preliminary version of this work was presented as an oral paper in [11]. This submission extends [11] on both methodology and experiments in four aspects. First, we add a related problem on layout generation in Section , and restructured the section. Second, we design a zoom-in strategy to enlarge the feature maps for small objects, so that the network can obtain meaningful object features from the encoder. Third, we add a variety loss to improve the diversity of the predicted scene contexts. Fourth, we perform more experiments to evaluate the effectiveness of our model (including comparison to three more baseline methods, diversity evaluation, and additional ablation studies), and explore the application of our model to fake scene detection and feature visualization.

2 Related Work

In this section, we review related works on scene context modeling, layout generation, context-based representation learning, and context-based image manipulation.

2.1 Scene Context Modeling

The scene context of an image contains rich information about how objects and scenes are related to each other. Cognitive science studies have shown that contextual information plays a crucial role in human visual recognition [12], [13]. There are many types of context information, including visual context [14], global scene context [15], relative location [16], and layout [17].

Contextual information has been exploited in many vision tasks to learn semantic features and improve visual understanding performance. On the one hand, context is essential for feature learning. For example, Pathak *et al*. [7] proposed a context encoder to learn high-level semantic features for image inpainting. On the other hand, scene context has been shown to be effective in many vision tasks, such as recognition, detection and segmentation [4], [6]. Multiple contexts can also be combined to improve performance. Choi *et al*. [18] proposed a graphical model to exploit multiple contexts in order to identify the out-of-context objects in a scene. Izadinia *et al*. [19] encoded the scene category, the context-specific appearances of objects and their layouts in order to learn the scene structure. Chien *et al*. [20] proposed the ConvNet to predict the probability of a pedestrian located at some location in the image. Wang *et al*. [21] used a variational auto-encoder to show the possibility of reasonable nonexistent human poses in a scene.

All these works use the existing scene context of the image as an additional cue to reason about the properties of foreground objects of interest. In contrast, our objective is fundamentally different from these prior works. Conceptually, we are proposing to address an inverse problem, *i.e*., to infer the missing scene context from the properties of the given foreground objects.

2.2 Layout Generation

In recent years, we have witnessed a rising interest in scene layout generation in the graphics and vision community. Our work bears some high-level resemblance to the recent efforts on data-driven indoor scene synthesis. These works attempt to model object arrangements with undirected factor graphs [22], activity graphs [23], and stochastic grammars [24]. Unlike these works that built context information from pair-wise object relationships, Wang *et al*. [25] introduced a deep neural network to learn the priors of object placements for indoor scene synthesis. Similar to [25] from a high-level perspective, we also use deep neural networks to learn priors on the spatial structure of objects from image data in order to synthesize semantic layouts. However, unlike [25] which aimed to generate the arrangement of a sparse set of 3D objects, we aim to predict a dense, pixelwise scene layout. In addition, we deal with a more challenging problem as we only use the given objects as input but their method assumes the scene types to be known.

There are also few works on outdoor scene layout generation. In Li *et al*. [26], the LayoutGAN model was proposed to map a random layout (a set of elements with random class labels and geometric parameters) to a refined layout. In Jyothi *et al*. [27], the LayoutVAE model was proposed to generate stochastic scene layouts given an input label set, i.e., categories of all the elements. Unlike these works on outdoor scene synthesis, the input to our method only contains a few standalone objects, from which a complete layout needs to be inferred. In addition, our method is able to predict detailed and complex object shapes for pixel-level scene layouts, whereas the prior works simply model objects as geometric primitives (e.g., rectangles).

2.3 Context-based Representation Learning

There have been some efforts on visual representation learning via context prediction. Mikolov *et al*. [28] proposed the skip-gram model to learn a word representation by predicting surrounding words of a single word. Doersch *et al*. [29] learned an image representation via predicting the relative positions of patches in an image (*i.e*., spatial context). Vondrick *et al*. [30] learned to anticipate the visual representation in a future frame of an unlabeled video (*i.e*., temporal context). In our work, our ultimate goal is not to learn a visual representation, but to predict of the surrounding environment of some standalone objects.

2.4 Context-based Image Manipulation

A number of works have investigated how to use context for image manipulation tasks. Some works used context as a prior to retrieve and composite the assets. Tan *et al*. [31] used CNN features to capture local context for person composition. By jointly encoding the context of foreground objects and background scenes, Zhao *et al*. [32] learned an object representation for compatible foreground object retrieval based on a given background image. However, the quality of the generated image depends on the retrieval database. The retrieved assets may not fulfill the user's requirement and generate unrealistic compositions.

Other works represented the context as a scene layout and learned a generative network to manipulate the synthesized image.

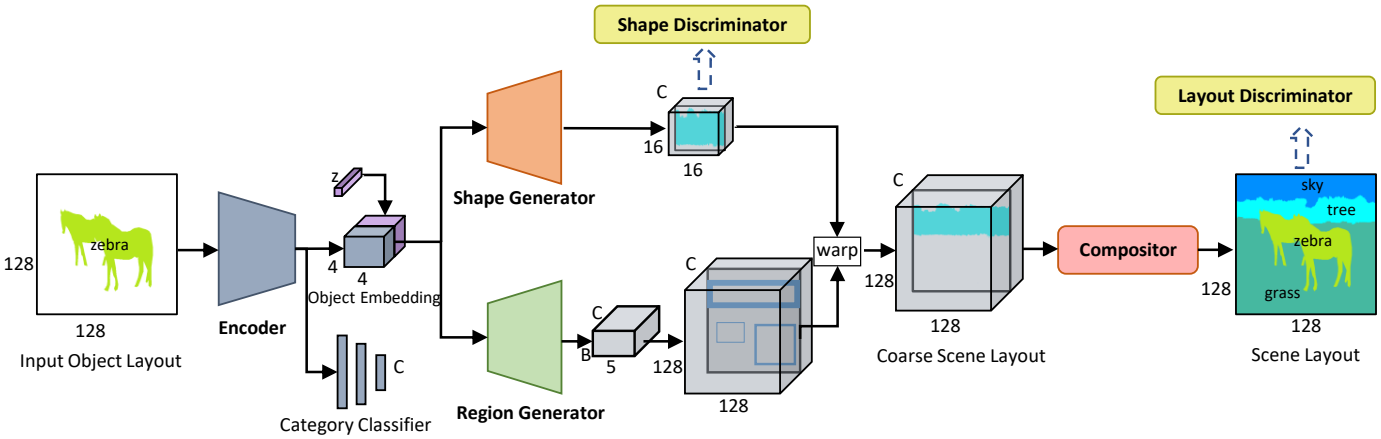


Fig. 2: Overview of our network architecture. Our model takes as input an object layout encoding the properties of input objects and generates a scene layout representing the scene context. We use the category classifier to pre-train the encoder to obtain the object representations. The object representations and a noise vector are concatenated and passed to the shape generator and the region generator. The shape generator generates the shapes of all C object categories, while the region generator generates the parameters and confidence values of B bounding boxes to represent the potential region proposals for each category. The boundary thickness of each bounding box indicates the confidence score of the box. The bounding boxes are then used to warp their corresponding shapes to a coarse scene layout, which is then refined by a compositor to output a final scene layout. Finally, a shape discriminator and a layout discriminator are introduced to classify the generated object shapes and scene layouts, respectively, as real or fake.

Wang *et al*. [33] proposed a GAN model to synthesize and manipulate high resolution images from the scene layout. Hong *et al*. [34] constructed a scene layout as an intermediate representation for image manipulation from text descriptions. Johnson *et al*. [35] proposed a graph convolutional neural network to generate images from scene graphs.

In contrast to these existing works, our proposed work can provide a new approach for users to specify the structure of the output image to be synthesized. Instead of asking users to specify all objects in the image (as a semantic map or a coarse layout map, or vaguely specifying the image content through text), they only need to specify the properties of the key objects that they concern about in the form of semantic objects. Our approach can generate diverse plausible scene layouts, which can be used to synthesize realistic full scene images. Thus, our model can be considered as complementary to existing image synthesis methods.

3 Methodology

3.1 Problem Formulation

Our goal is to develop a deep neural network that takes the properties of one or more standalone objects as input to generate a scene context around the given objects, which contains the other objects that are likely to co-occur with the given objects. The whole framework is shown in Figure 2. We encode both the input objects and the predicted scene context using object-level semantic layouts, which can compactly describe the classes, shapes, and positions of the objects in a scene layout. In other words, given an input object layout X_o, our model learns a function f to generate a full scene layout $X_s = f(X_o)$.

3.2 Scene Context Prediction Network

Figure 3 shows the structures of the modules used in the proposed network architecture.

Encoder. The input to the encoder is an object layout, $X_o \in \{0,1\}^{H\times W\times C}$, where H and W are the height and width of the layout, respectively, and C is the number of object categories. We encode each pixel in the input object layout as a one-hot vector to represent the object category of the pixel. A vector of all zeros represents that the pixel belongs to the unknown category. Each of the C channels in X_o specifies the shapes and positions of the objects from the category. The encoder contains five 3×3 stride-2 convolutions. All the convolutional layers are followed by batch normalization and Leaky-ReLU, except for the first and last layers where only Leaky-ReLU is applied. The output of the encoder is an object representation of size $4 \times 4 \times 512$ from X_o. In order to learn a useful object representation, we add a category classifier to predict the presence of each object category in the scene context. Similar to [36], our category classifier contains two fully connected layers, followed by a Sigmoid layer which outputs a C-dimensional vector.

We note that the encoder tends to extract a meaningless object representation if the input layout contains only a very small object. Such an object representation cannot be used to reliably predict a potential scene context. Inspired by the zoom-out and zoom-in architecture used in [37] for tiny object detection, we re-scale the input object layout to enlarge the spatial resolution of the small objects. However, unlike [37] that detects objects from a complete scene image, we only take an object layout as input. As the feature maps of an object layout contain much less information than those of a complete scene image, we cannot directly enlarge the feature maps as in [37]. Hence, we first divide the input object layout into 9×9 grid cells. Based on the cells overlapped by the object, we select a minimum rectangle region that contains the object. We then re-scale the selected region into the same size as that of the input layout, and pass it through another encoder that has the same architecture as the one described above to obtain the zoomed object representation. Finally, we concatenate the zoomed object representation and the original object representation, and use an additional layer with 3×3 stride-1 convolution to obtain the final object representation.

Shape Generator. To increase the diversity of the generated layout, we add a noise vector z_t on top of the object represen-

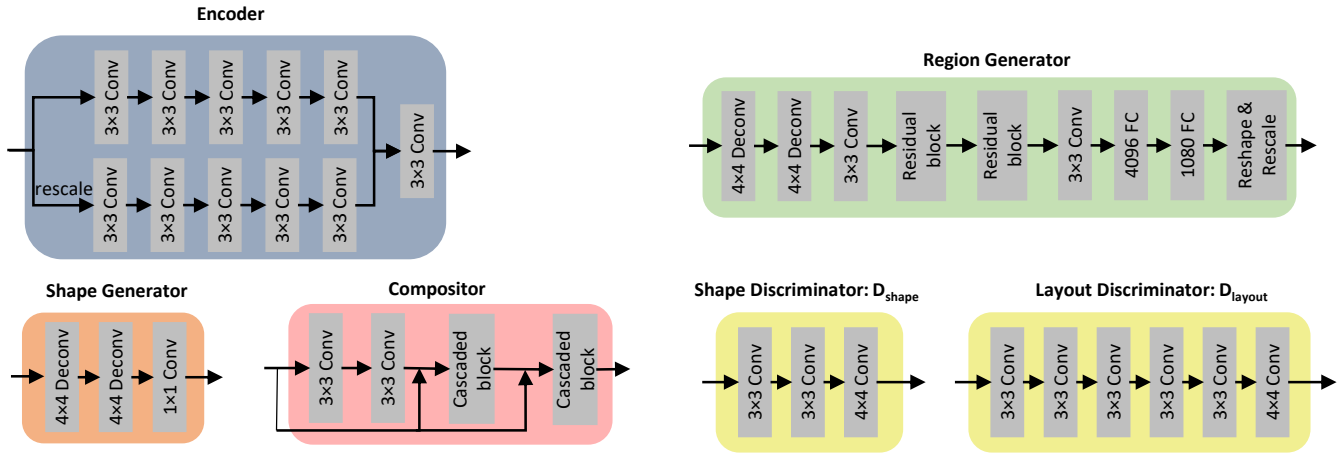


Fig. 3: Details of various modules used, including encoder, shape/region generators, compositor, and shape/layout discriminators.

tation by spatial duplication and feature channel concatenation, resulting in concatenated features F. We then feed F to the shape generator. The shape generator module is composed of three de-convolutional layers. Each of the first two layers is a 4×4 de-convolution with a stride of 2, followed by batch normalization and ReLU. The last layer is a 1×1 convolution followed by Sigmoid nonlinearity. The output is a soft binary mask $M \in [0,1]^{16\times16\times C}$, representing the shapes of all object categories. Note that our shape generator can naturally handle multiple occurrences of a single object class. In such a case, the shapes of these objects will appear as a group of connected or disjoint masks in the same channel of M.

Region Generator. The input to the region generator is the same as that of the shape generator. The region generator receives F and predicts B region proposals for each of the C object categories. Each region proposal is represented by a bounding box with four parameters (x, y, w, h) and a confidence score s. (x, y) refer to the center location of the box. (w, h) refer to the width and height of the box. The confidence value represents the probability that the bounding box covers an object. Thus, the output of the region generator is a tensor of size $B \times 5 \times C$, where 5 refers to the four parameters plus the confidence score. Note that to handle multiple instances of a single object category, we treat them as a group and use a single bounding box for them in training and testing. The region generator has two 4×4 stride-2 deconvolutions, several residual blocks and 3×3 convolutions, and two fully connected layers. Each residual block consists of a 1×1 convolution, a 3×3 convolution and a skip connection. The last fully connected layer has 1080 units that are reshaped to $B \times 5 \times 72$, which encodes the parameters of the bounding boxes of all the 72 categories (i.e., 5 parameters per bounding box $\times$ B bounding boxes per category).

Compositor. To combine the predicted shape masks and object bounding boxes coherently into a scene layout, for each object category, we warp the generated shape masks to the position of the corresponding bounding box using the bilinear interpolation operator in the spatial transformer network [38]. Note that some artifacts like unlabeled regions and tiny objects may exist in the fused coarse scene layout. In addition, the generated object bounding boxes may overlap with each other, causing occlusions among different objects. To address these problems, we further convert the coarse layout to a dense pixelwise scene layout using the cascaded block [39]. The compositor contains two 3×3 stride-2 convolutions, and two cascaded blocks [39]. Each cascaded block takes as input the feature map from the previous module and a rescaled coarse scene layout to produce a refined layout. It is composed of a bilinear upsampling and then two 3×3 convolutions. All the convolutional layers in the compositor are followed by batch normalization and Leaky-ReLU. The final output of the compositor is the refined scene layout of size $128 \times 128 \times 72$.

Discriminators. For a given input (*e.g.*, a standalone object on a canvas), there may be multiple scene layouts that are plausible and consistent with the input. To handle this multi-modal issue, we introduce two additional discriminators, as inspired by the recent success of adversarial learning approaches [8], [40]. One is a shape discriminator D_{shape}, and the other is a layout discriminator D_{layout}. The input to the shape discriminator D_{shape} is the generated shape masks $m_c^{'}$ or the real shape masks m_c. The input to the layout discriminator D_{layout} is a generated scene layout or a real scene layout. The shape discriminator first processes the input with a series of 3×3 stride-2 convolutions. All the convolutional layers are followed by batch normalization and Leaky-ReLU, except for the first layer where only Leaky-ReLU is applied. A 4×4 convolution and a fully connected layer are then applied to output the probability of the input being real or fake. Both layers are followed by Leaky-ReLU. The layout discriminator shares a similar architecture with the shape discriminator, but has a different number of convolutional layers.

3.3 Training

Due to the complexity of our network, it is difficult to directly train our model end-to-end. Thus, we pre-train the encoder and the category classifier to obtain the object representation. Since the zoom-in strategy is not differentiable, we first obtain the zoomed object representation via preprocessing as described in Section 3.2. We then concatenate the zoomed object features and the original object features as the final object representation from the encoder for training. Let $l = \{l_c \in \{0,1\}, c \in C\}$ be the ground truth object categories of an image. We use the cross-entropy loss L_{cls} for the category classifier as:

$$L_{cls} = -\sum_{c \in C} [l_c \log p_c + (1 - l_c) \log(1 - p_c)], \qquad (1)$$

where $l_c = 1$ if the image contains object category c. p_c is the predicted probability for c.

For the region generator, we use $t = (x, y, \sqrt{w}, \sqrt{h})^T$ and s to denote the parameters and the confidence score of the predicted bounding box, respectively. We define the loss as:

$$L_{box} = \sum_{i=1}^{C} \sum_{j \in B_i} (p_{i,j}^{obj} ||t_{i,j} - \hat{t}_{i,j}||^2 + \lambda(p_{i,j}^{obj}) ||s_{i,j} - \hat{s}_{i,j}||^2), \tag{2}$$

where i ranges over all object categories and j ranges over all the bounding boxes of category i. $\hat{t}_{i,j}$ and $\hat{s}_{i,j}$ are the parameters and the confidence score of the ground truth bounding box, respectively. $p_{i,j}^{obj}$ is an object indicator that is 1 when the bounding box covers a ground truth object and 0 otherwise. Since most generated bounding boxes do not cover any ground truth objects, we introduce a class re-balancing function $\lambda(p)$ to prevent the model from predicting a zero confidence score for most bounding boxes, where $\lambda(p) = 1$ if $p = 1$ and $\lambda(p) = 0.1$ if $p = 0$.

For the shape generator, let $m \sim p_{fake}(m)$ be the generated shapes. Its loss is defined as:

$$\begin{aligned} L_{shape}^{ori} =& E_{m \sim p_{fake}(m)}[L_{crs}(m, \hat{m})] + \\ & E_{m \sim p_{fake}(m)}[(D_{shape}(m) - 1)^2]. \end{aligned} \tag{3}$$

The first term penalizes the difference between each generated shape m and its ground truth $\hat{m}$ using a pixelwise cross-entropy loss L_{crs}. The second term encourages the shape generator to produce realistic shapes to fool the shape discriminator. We use L2 norm instead of $\log$, as in LSGAN [41], to stabilize our training.

Note that GAN-based frameworks often suffer from the mode collapse problem, where the generator only samples from a single or few modes of the data distribution and ignores other modes. To encourage the generator to produce more diverse results, we add a variety loss $L_{variety}$ as in [42]:

$$L_{shape}^{diversity} = -\frac{d_I(G(c, z_1), G(c, z_2))}{d_z(z_1, z_2)}, \tag{4}$$

where c is the object representation. z_1 and z_2 are two noise vectors. d_I is the L1 norm distance between two generator outputs in the image space. d_z is the L1 norm distance between the noise vectors in the latent space. Here, we aim to maximize the ratio of d_I to d_z. The loss induces a large penalty if two different latent vectors are mapped to similar outputs, and thus encourages the generator to spread its output distribution to cover the space of possible scene layouts. The final loss for the shape generator is:

$$L_{shape} = L_{shape}^{ori} + \lambda_d L_{shape}^{diversity}, \tag{5}$$

where λ_d is the weight to control the diversity.

For the shape discriminator, its adversarial loss is defined as:

$$\begin{aligned} L_{shape}^{D} =& E_{t \sim p_{real}(t)}[(D_{shape}(t) - 1)^2] + \\ & E_{m \sim p_{fake}(m)}[D_{shape}(m)^2], \end{aligned} \tag{6}$$

where $t \sim p_{real}(t)$ are real shapes.

Finally, the losses for the output scene layout and the layout discriminator are similar to Eq. 3 and 6, respectively, except that object shapes are replaced with scene layouts.

3.4 Implementation Details

The detailed network architecture is described in Section 3.2. The input object layout is resized to 128×128 by nearest interpolation. To help the network converge, we first train the category classifier to obtain the object representation. To obtain the values of the object indicator $p_{i,j}^{obj}$, which are used in the loss in Eq. 2, we follow the approach of YOLO [43]. In particular, in each iteration during training, we feed an input into our network to predict the region proposals for all object categories. A region proposal of a category is labeled as positive (*i.e.*, $p_{i,j}^{obj} = 1$) only if it has an intersection over union (IOU) of at least 0.5 with any ground truth bounding box of the same category, and labeled as negative (*i.e.*, $p_{i,j}^{obj} = 0$) otherwise. The ground truth confidence score $\hat{s}_{i,j}$ of a predicted bounding box is defined as the IOU between the predicted bounding box and the ground truth bounding box. Note that during training, we use the ground truth of object bounding boxes for the compositor to generate the scene layout. During inference, we choose the generated object bounding boxes based on the corresponding confidence values. We use the Adam optimizer [44] for optimization, and set $\beta_1 = 0.5$ and $\beta_2 = 0.9999$ to improve convergence during training. The learning rate is set to be $2e^{-4}$ and the batch size to 128 to stabilize training while meeting our memory budget. We train our model for 10,000 iterations until the model reaches convergence. In each iteration, we alternately update the parameters of the generators and discriminators, as in [40]. To find the optimal value of the variety loss weight λ_d in Eq. 5, we use grid search (by varying its value and choosing the one giving the best performance). We show the performances of our model under different values of λ_d in Table 1. According to the results, we use $\lambda_d = 0.5$ for all remaining experiments.

4 Experiments

In this section, we train our model to generate scene layouts on the COCO-Stuff [10] dataset. We aim to show that our model can generate plausible scene contexts from the input objects with diverse object properties. We show both qualitative and quantitative results of our method, in comparison with the baselines, and evaluate the plausibility and fitness of our generated scene contexts via a user study. Finally, we show the usability of our model in several applications, including realistic image synthesis from partial scene layouts, scene recognition, and fake scene detection.

4.1 Dataset

We perform experiments on the COCO-Stuff dataset, which augments a subset of the COCO dataset [47] with additional stuff categories. The dataset includes 40k training and 5k testing images with bounding boxes and semantic layouts for both indoor and outdoor scenes. It contains 80 *thing* categories and 91 *thing* categories in total. Our evaluation only focuses on outdoor scene images in the dataset, which have many complex and diverse scene structures and thus make the scene context prediction problem very challenging. Given the outdoor scene layouts, we only select those with 2 to 8 objects in them. To train our network, for each layout, we randomly select one or two objects and remove other regions to form an *object layout*, which together with the original semantic layout (referred to as *scene layout* in this work) form a training pair. If a selected object covers less than 5% of the image, we skip this object. As a result, we produce a dataset that contains 72 object categories (39 *thing* categories and 33 *stuff* categories), with a total of 52,803 training pairs and 1,934 test pairs.

4.2 Baselines

Since we are not aware of any previous works on scene context prediction, we compare our method with several semantic image

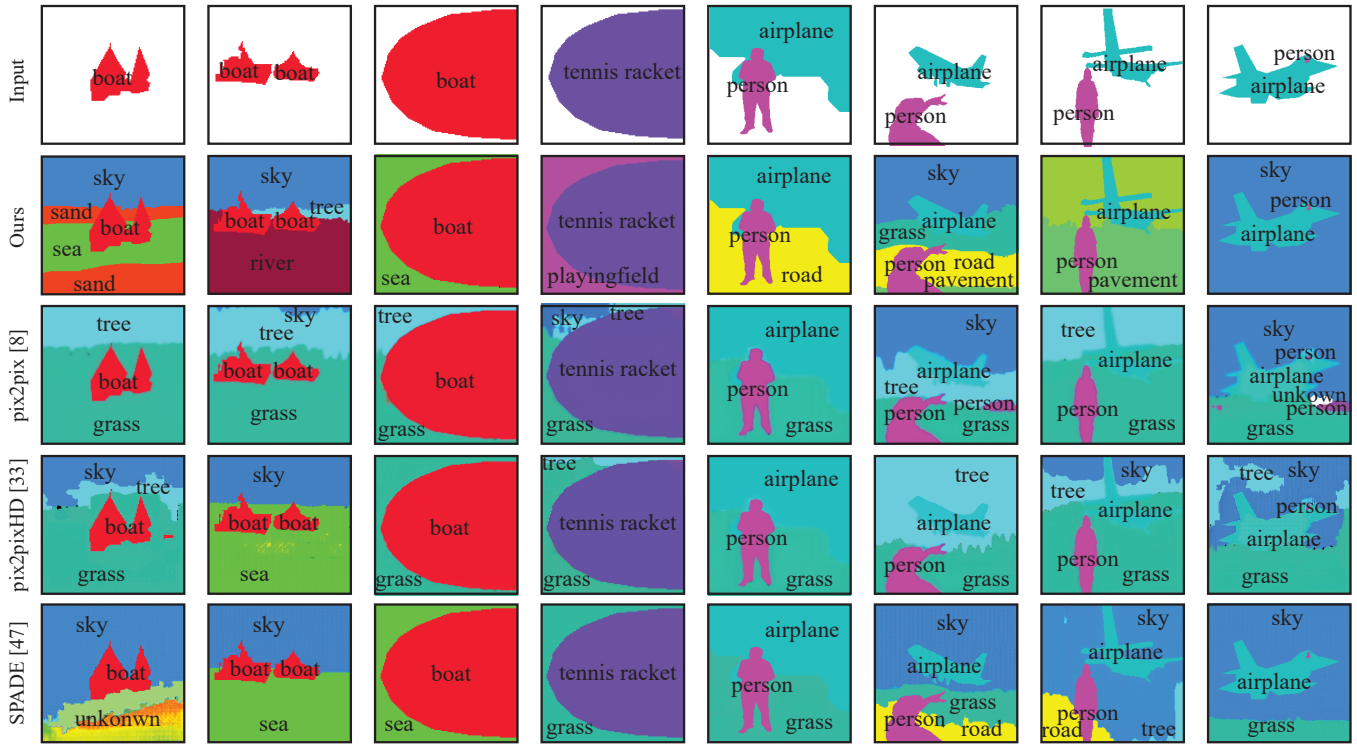


Fig. 4: Qualitative results from our model and the baselines. Given the input object layouts (1st row), which contain one or two standalone objects, we generate output scene layouts using our model (2nd row) and the baselines (3rd to 5th rows).

synthesis models: pix2pix [8], pix2pixHD [33], SPADE [45], and BicycleGAN [46]. pix2pix learns a generic mapping between an input image and an output image with aligned image pairs. pix2pixHD extends pix2pix to generate high-resolution photo-realistic images from semantic layouts. SPADE uses a decoder-only architecture and feeds the input into a spatially adaptive normalization to preserve semantic information in the input. BicycleGAN extends pix2pix to produce more diverse results. We train the baselines using our training dataset so that they can map an input object layout to an output scene layout, as in our model. Note that we have empirically found that the baselines tends to produce noise, *i.e.*, small artifacts, in their outputs. To address this problem, we refine their results via a simple post-processing step. Specifically, we first filter their initial results by a weighted median filter. As object boundary is important in a scene layout, we then apply a guided filter [48] to serve as an edge-preserving smoothing operator, with the filtered images as guided images to obtain the final results.

4.3 Qualitative Results

Figure 4 shows some qualitative results of our model, in comparison with those from the baselines. We make several observations as follows. First, our method can generate more visually diverse scene layouts than the baselines. For example, from the first to fourth columns, pix2pix and pix2pixHD always give similar object categories and positions (*e.g.*, grass, tree and sky), while our results have a higher diversity in terms of object category and position. In addition, although SPADE generates better results than other baselines, there are some artifacts that cannot be eliminated, *e.g.*, the unknown region appeared in the first column. Second, the scene layouts predicted by our model are more semantically plausible than those by the baselines. For example, in the fourth column, the baselines predict some unlikely spatial relations among the objects for the scene (*i.e.*, tennis racket in grass). In contrast, our method predicts a playing field given the tennis racket, which is more convincing. Third, our model is able to generate scene contexts that respect the input objects consistently, while the baselines fail to give reasonable results in some cases (*e.g.*, boat on top of the grass or the unknown region in the first column). We further investigate how the change in spatial relationship between the input objects may affect the outputs of different methods. In columns 5-8, when we change the spatial relationship of the person and the airplane from left/right (fifth column) to above/below (sixth column) or inside/surrounding (eighth column), the scene layouts by our model favorably adapt to the inputs, while the baselines tend to give similar results.

Figure 5 shows a diversity evaluation of our method against SPADE and BicycleGAN, which are designed to produce diverse outputs. Given the same input object layout, our model is able to generate diverse and realistic scene layouts that fit the inputs. We observe that SPADE and BicycleGAN can also produce diverse scene layouts, but cannot produce meaningful variations. For example, the input in the first row is a standing person. Our model can predict different categories, shapes and arrangements for the surrounding objects. However, although SPADE and BicycleGAN can also produce different surrounding object categories, such as snow and grass, object shapes and arrangements are similar across all the results. More importantly, results of SPADE and BicyleGAN do not semantically match the input, *i.e.*, the boat is predicted to be on the grass. One possible reason is that the generator architectures of SPADE and Bicycle are object-agnostic and tailored to model appearance changes, and thus have difficulty in handling our scene layout prediction task that involves large semantic structure variations. These results show that our method can produce more diverse scene layouts that fit the inputs, compared with SPADE and BicycyleGAN.

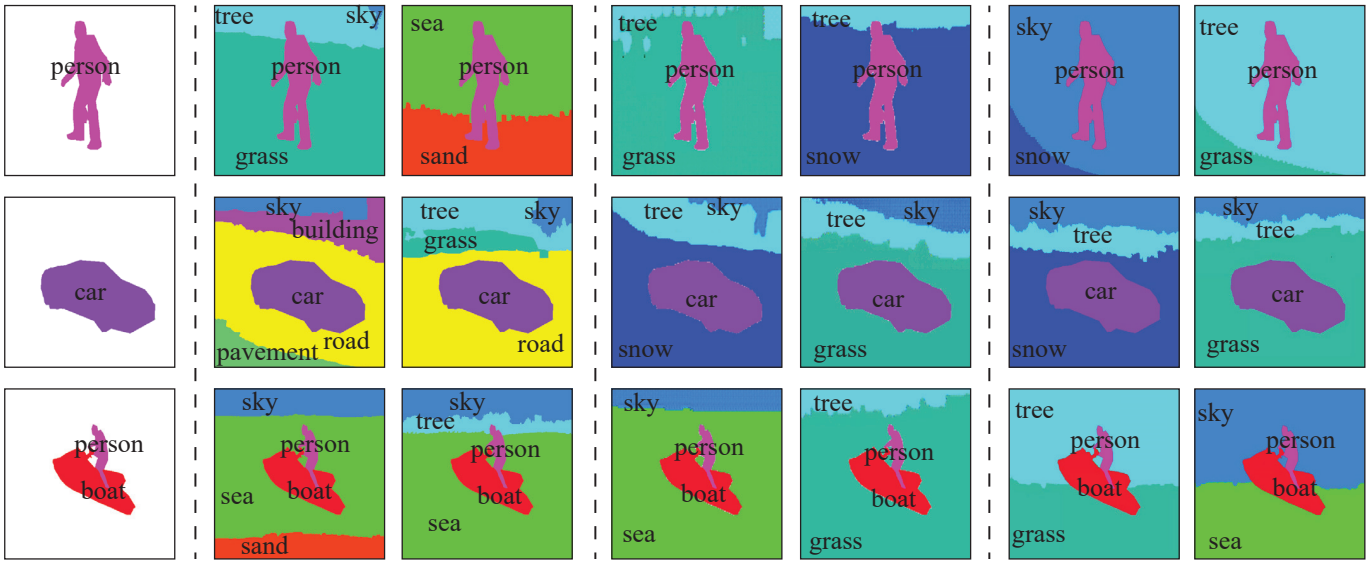


Fig. 5: Diversity evaluation of our method, compared with SPADE [45] and BicycleGAN [46]. Given an input object layout (left), we show two different scene layouts generated by our method, SPADE, and BicycleGAN respectively.

4.4 Quantitative Evaluation

We conduct evaluations using the following metrics using three metrics, NLL, LPIPS and FID, explained as follows.

NLL. We quantitatively evaluate the plausibility of our generated scene layouts using the object pairwise relationship priors, which have been widely used in indoor scene synthesis to characterize scene structure [25], [49]. In particular, we compute the probabilities of pairwise relations among object classes from a dataset of natural scene images, and evaluate the likelihood of each generated scene layout under the probabilities to measure its quality. Formally, let C and R be the object categories and pairwise spatial relations, respectively. For each pair of object classes $\langle u, v\rangle$, $u, v \in C$, we compute a probability of them being in a spatial relation $r \in R$ as $p(r|\langle u, v\rangle)$. Here, we consider six mutually exclusive spatial relationships, i.e., $R \in \{left, right, above, below, inside, outside\}$. Given a generated scene layout X, we define its negative log likelihood (NLL) as:

$$NLL = -\frac{\sum_{\langle u,r,v\rangle \in T} \log p(r|\langle u, v\rangle)}{|T|}, \tag{7}$$

where $\langle u, r, v\rangle$ iterates over all the possible class pairs denoted as T in the layout.

We use 2-fold cross-validation for this evaluation. In particular, we first split our training dataset uniformly into two folds. For each fold, we train a model on it, learn the priors from the other fold, and compute the NLL value on the test dataset against the priors. Finally, we use the mean NLL value (NLL_{all}) over the two folds as our metric for scene layout plausibility evaluation. In addition, we also compute the input-centric mean NLL value (NLL_{object}) to measure how well the predicted scene layouts fit the inputs. To do this, we only consider the class pairs where the inputs are involved in Eq. 7.

LPIPS. We evaluate the layout diversity by computing the average LPIPS distance between the generated layouts [50]. In particular, we randomly chose 50 input object layouts from our test dataset. For each input, we randomly sample 6 layouts from our model, which are then used to construct 3 layout pairs at random. For each layout pair, we calculate a LPIPS distance and average over all the pairs to get an average LPIPS distance. Specifically, we pass a pair of layouts through our layout discriminator, and use the features from different layers to calculate the LPIPS distance. Let X^l and Y^l be the feature maps of layer l for the two layouts. The feature maps are normalized on the channel dimension to unit length. The LPIPS score can be written as:

$$d_{LPIPS} = \sum_l \frac{1}{H_l W_l} \sum_{i,j} \left\| w_l (X^l_{i,j} - Y^l_{i,j}) \right\|_2^2, \tag{8}$$

where $X^l_{i,j}$ represents the feature responses of X^l at position (i, j). H_l and W_l are the height and width of X^l, respectively. w_l is a layer-specific weight. Here, we set $w_l = 1$. A higher LPIPS distance indicates a better diversity of the generated images.

FID. To evaluate the quality of the generated layouts, we compute the FID value [51] between the generated and ground truth layouts. We feed the generated layouts into the layout discriminator and use the features from the last convolution layer to calculate the FID value. A lower FID value indicates a better quality of the generated layout.

Table 1 compares the performance of our model with the baselines. Our method outperforms all baselines by a large margin on all the metrics. This again confirms the superior performance of our method in predicting plausible, fitting and diverse scene contexts, in comparison to the baselines.

4.5 Ablation Study

To investigate how different components in our network affect the generation performance, we compare several ablated versions of our model, using the mean NLL value introduced in Section 4.4:

- **No category classifier.** We remove the category classifier, so that there is no pre-training for the object representations.
- **No discriminators.** We remove both shape and layout discriminators, relying only on the pixelwise cross entropy losses for model learning.

Method	$NLL_{all} \downarrow$	$NLL_{object} \downarrow$	$LPIPS \uparrow$	$FID \downarrow$
pix2pix [8]	2.15	2.11	0.012	118
pix2pixHD [33]	2.01	1.95	0.053	112
SPADE [45]	1.81	1.72	0.083	106
BicycleGAN [46]	2.11	2.03	0.102	115
Ours (No category classifier)	1.72	1.63	0.108	104
Ours (No discriminators)	1.85	1.78	0.098	113
Ours (No shape discriminator)	1.68	1.55	0.118	101
Ours (No layout discriminator)	1.82	1.78	0.102	109
Ours (No zoom-in strategy)	1.69	1.57	0.091	88
Ours ($\lambda_d = 0$)	1.64	1.44	0.085	86
Ours ($\lambda_d = 0.25$)	1.64	1.44	0.107	83
Ours ($\lambda_d = 0.75$)	1.63	1.43	0.120	91
Ours ($\lambda_d = 1$)	1.63	1.43	0.118	98
Ours (Full model)	**1.63**	**1.42**	**0.121**	**81**

TABLE 1: Quantitative evaluation of the baselines (*i.e.*, pix2pix, pix2pixHD, SPADE, and BicycleGAN) and our model (ablated versions and full model). We evaluate the performance using negative log likelihood (NLL) of the generated layouts under pre-computed pairwise relation priors. NLL_{all} reflects the overall plausibility of an output layout, and NLL_{object} indicates the fitness between the input objects and an output layout. FID measures the visual quality of the generated layouts. $LPIPS$ evaluates the layout diversity.

- **No shape or layout discriminator.** We remove one of the discriminators.
- **No zoom-in strategy.** We remove the zoom-in strategy for small object layouts.
- **Different values of the variety loss weight.** We vary the value of λ_d in Eq. 5 to study its influence on the performance. When $\lambda_d = 0$, the variety loss is excluded.

From the results in Table 1, we can observe that compared with the ablated versions, our full model achieves the best performances on all metrics. This demonstrates the necessity of each component in our model.

4.6 User Studies

We use Amazon Mechanical Turk (AMT) to evaluate the quality of our results. We assess the quality of the generated scene layouts by conducting two user studies: plausibility study and fitness study. In the plausibility study, our goal is to evaluate whether the objects in the generated scene layouts have plausible spatial relations. In the fitness study, we aim to evaluate whether the generated scene layouts provide convincing contexts for the input object(s). In both studies, we use 50 input object layouts randomly chosen form our test dataset. For reference, we compare our method (Ours) with SPADE (Baseline), which performs the best quantitatively among all the baselines in Section 4.4.

Plausibility. We ask AMT workers to judge the generated layouts and ground truth (GT), by evaluating whether objects in the scene layouts have incorrect relationships (plausibility). They were given a sequence of scene layouts selected randomly from three sources (Ours, Baseline and GT), and asked to evaluate whether the objects in the scene layouts likely have such spatial relations. For those scene layouts that are regarded as implausible, they are asked to label at least a pair of objects that have a wrong spatial relation. Each Human Intelligence Task (HIT) contains 50 scene layouts, along with 10 duplicate layouts for consistency check. We discard the responses from a worker who has a consistency rate of less than 80% on the duplicate questions. We end up with 30 workers in our experiments, with each scene layout being evaluated by at least 10 workers.

For each scene layout, we compute the fraction of workers who have chosen it to be plausible as a plausibility score, and report the average score for each method in Table 2. Note that the average score of the ground truth represents an upper bound performance. The results show that our results are perceived to be significantly plausible than those by the baseline and much closer to the ground truth.

	Baseline	Ours	GT
Plausibility score ↑	0.59 ± 0.15	0.75 ± 0.12	0.88 ± 0.09

TABLE 2: Plausibility scores for the baseline (Baseline), our method (Ours), and the ground truth (GT).

	Ours over Baseline	Ours over GT	Baseline over GT
Fitness preference	64%	45%	21%

TABLE 3: Fitness preferences for the baseline (Baseline), our method (Ours), and the ground truth (GT). Each number shows the percentage of time that one method is preferred over another.

Fitness. In this experiment, the AMT workers are presented with an input object layout, along with two scene layouts generated from the input. They are asked to select which scene layout illustrates a more appropriate context for the input objects. In each comparison, we display two scene layouts chosen randomly from three sources (Ours, Baseline and GT) side by side in a randomized order. We conduct 150 pairwise comparisons (50 for Ours vs. Baseline, 50 for Ours vs. GT and 50 for Baseline vs. GT). We randomly divide these 150 comparisons into three HITs uniformly. In each HIT, we add 5 duplicate comparisons for consistency check. We discard the responses from a worker who has a consistency rate of less than 80% on the duplicate comparisons. We have a total of 9 workers in the experiment, and each comparison is evaluated by 3 workers.

The results in Table 3 show that our results are preferred by the workers most of time, compared with those by the baseline. In addition, the workers only show a slight preference for the ground truth layouts over our results. This implies that our results are considered to fit the input objects better than those of the baseline, and comparable to the ground truth.

4.7 Additional Analyses

We provide additional analyses on the performance of our model under different factors.

Effect of object shape, category and spatial relation. Figure 6 shows the results of our method by varying the category and shape of the input objects as well as spatial relation among the input objects. Our results favorably adapt to changes in the inputs. For example, when changing an input object from a car (third column) to an elephant (fourth column), our predicted context changes the region supporting the input object from road (third column) to grass (fourth column). In addition, when we change the spatial relation between a person and a surfboard from $\langle person, above, surfboard \rangle$ (fifth column) to $\langle person, right, surfboard \rangle$ (sixth column), the region below the person changes from sea to sand accordingly.

Effect of object size. We study how our quantitative performance varies as input object size changes. We first randomly select one input object category (*i.e.*, person, car or airplane), keep the corresponding object shape fixed, and vary the size of the object. We evaluate the performance using NLL defined in Eq. 7 and plot the results in Figure 7. We can see that our performance degrades if the input object size is too small or too large.

Generalization to unusual inputs. We are interested to find out how well our model can generalize to unusual inputs. To

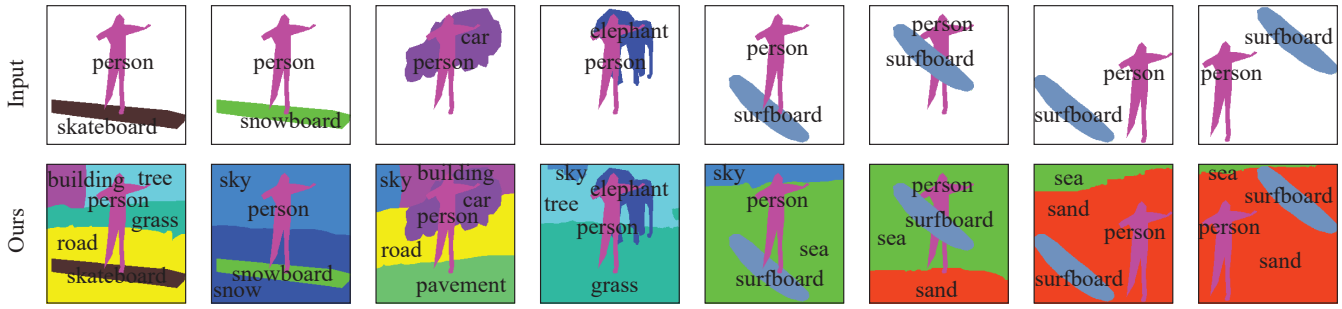


Fig. 6: Qualitative results of our model by varying the categories, shapes and spatial relation of the input objects.

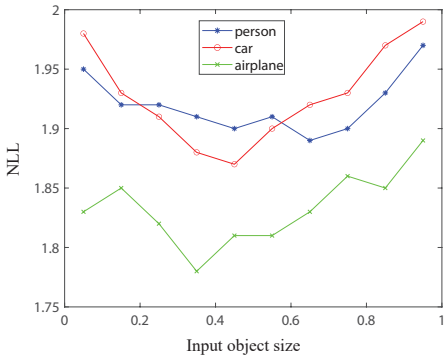


Fig. 7: Performance vs. input object size. We use NLL to evaluate the performance on three object categories (*i.e.*, person, car and airplane). The lower the score, the better the performance. The input object size is normalized with respect to the image size to range between 0 and 1).

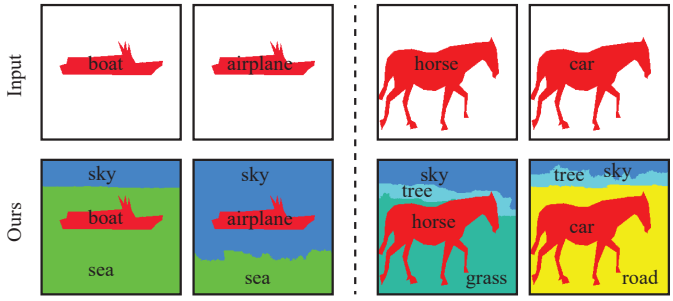


Fig. 8: Qualitative results of our model by changing the category of an input object to an unusual one while keeping the shape fixed.

Method	Accuracy
Chance	0.5%
ImageNet-CNN [55]	38.9%
Places-CNN [56]	49.8%
Ours + SVM	39.8%
Ours + Random Init	37.6%
Ours + Finetune	**52.4%**

TABLE 4: Accuracy of outdoor scene recognition on the SUN dataset [57]. We evaluate the representation learned by our layout discriminator for scene recognition. We compare the performance of directly using the learned representation with a SVM (Ours + SVM), randomly initializing the discriminator (Ours + Random Init) and fine-tuning the discriminator (Ours + Finetune). We also show the results from ImageNet-CNN and Places-CNN for a comparison.

investigate this, we show the results of our method in Figure 8 by changing the category of an input object to one that is unusual to its shape. We can see that our model can still predict proper scene contexts by considering both the input object shape and category. For example, when changing the category of boat to airplane, the predicted scene layout contains the sea for the boat shape, and the sky for the airplane category.

4.8 Image Synthesis

Layout-based Image Synthesis. We conduct experiments by using our model for image synthesis. Several recent promising works [33], [39], [52] on image synthesis have attempted to generate realistic images from scene layouts. While being able to synthesize stunning results, they all need a complete scene layout to start with. The ability of our model to infer scene context from only standalone foreground objects makes it possible to hallucinate a *full* scene image with just a *partial* semantic layout.

For this task, we leverage the state-of-art image synthesis method [52], which transforms a semantic layout into a realistic image. The image segments are extracted from our training dataset to generate the memory bank for image synthesis. Given a partial scene layout, we first use our model to predict a full scene layout, which is then fed into the image synthesis method [52] to produce an output image.

Sketch-based Image Synthesis. In addition to using a partial semantic layout as input, we also experiment with using a sketch as an input to our model for image synthesis. To do this, we first need to convert a sketch into a partial layout required by our model. In particular, given the images and their semantic layouts in our training dataset, we apply an edge detection step [53] to obtain the sketches of randomly chosen foreground objects, followed by some post-processing steps as in [54], including binarization, thinning, small component removal, erosion and spur removal. After that, we train a pix2pix network [8] to map the sketches to partial layouts. During the testing stage, given an input sketch, we first map it to a partial layout, and then transform it to a full scene image with the above image synthesis process.

Figure 9 shows some image synthesis results generated from partial semantic layouts and sketches. As can be seen, our method can synthesize complex and semantically meaningful full scene images from sparse user inputs.

4.9 Scene Recognition

We also test the representation learned by the layout discriminator for outdoor scene recognition [58] on the SUN dataset [57]. Note that we use the 220 outdoor scene categories in the dataset for evaluation since our model is only trained on outdoor scenes.

To do this, we first replace the output layer of our discriminator with a K-way softmax layer. We then construct a scene recognition model by using a pre-trained semantic segmentation model [59] to map an input color image to a scene layout, which is then fed into

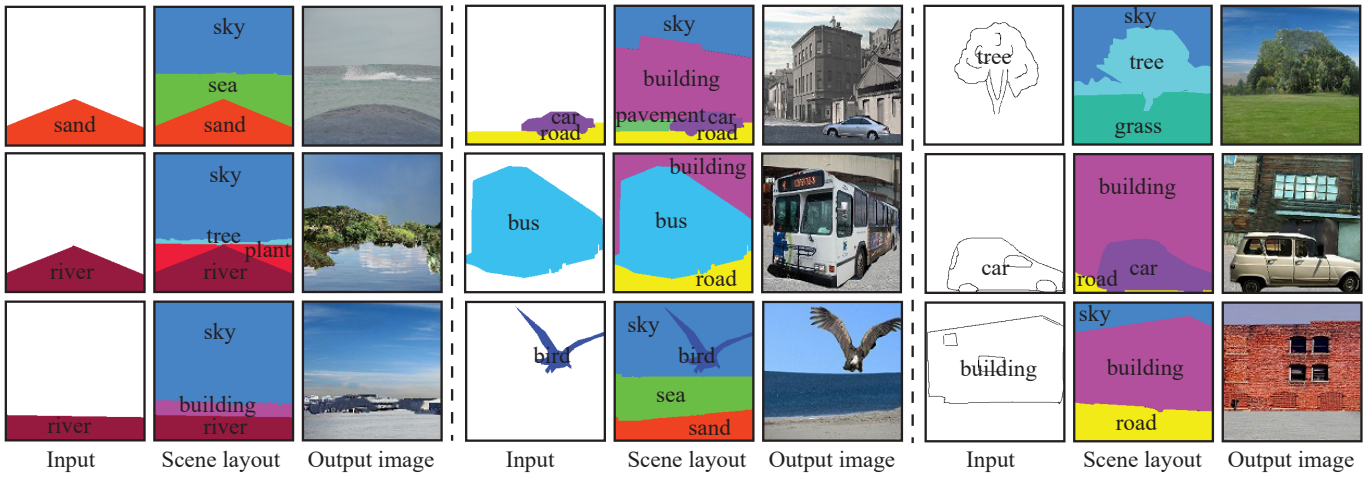


Fig. 9: Given a partial scene layout or a sketch as input, our method is able to generate a complete scene layout and further synthesize a realistic full scene image.

Method	Accuracy
Out-of-context [18]	72.9%
pix2pix [8]	55.3%
pix2pixHD [33]	57.8%
SPADE [45]	61.4%
BicycleGAN [46]	55.9%
Ours	**73.1%**

TABLE 5: Accuracy of fake scene detection.

our discriminator for classification. We fine-tune our discriminator using the training splits of the SUN dataset (Ours + Finetune). We also experiment with randomly initializing the discriminator of our recognition model (Ours + Random Init) instead of using learned weights of our discriminator, and directly using the outputs of the penultimate layer of the discriminator as features for a multi-class SVM (Ours + SVM). Note that since we are interested in exploring the representation of our discriminator, we fix the weights of the semantic segmentation model during the experiment.

We report the recognition accuracy in Table 4. SVM using our learned representation as features slightly outperforms AlexNet pre-trained on ImageNet [55], but is inferior to the pre-trained Places-CNN [56] which is designed for scene recognition. In addition, while our randomly initialized model performs worse than ImageNet-CNN and Places-CNN, the model initialized from the weights of our learned discriminator (Ours + Finetune) obtains better performance. This is possibly because in order to discriminate between real and fake scene layouts, our discriminator needs to learn a representation that captures complex semantic and spatial relationships among the objects in a scene layout, which is important to excellent scene recognition performance. These results suggest that learning to hallucinate object-level scene context helps learn useful features for scene recognition.

4.10 Fake Scene Detection

In this application, we are interested in detecting fake scene images with unusual objects that violate contextual relationships [18], [60], [61]. It is challenging because contextual violations can be detected only if the relationships among the objects are carefully and precisely modeled.

To do so, we need to identify if the contextual relationship among the objects in a scene is plausible or not. Since our layout discriminator can tell if an input scene layout is real or not, we use the output of the layout discriminator for this task. In particular, for each input scene image, we first use a pre-trained semantic segmentation model [59] to obtain its scene layout. We then feed the scene layout to our discriminator to classify it as real or fake.

We use 100 fake scene images from the out-of-context dataset introduced in [62] and another 100 real scene images from our test dataset for evaluation. We report the detection accuracy in Table 5. We compare our method with a fake scene detection method [18]. We also use the layout discriminators from other baselines, including pix2pix [8], pix2pixHD [33], SPADE [45], and BicycleGAN [46], for comparison. The out-of-context model [18] not only detects a fake scene, but also predicts which object violates the whole scene context by using a tree-structured graphical model. Our model implicitly learns the semantics necessary for recognizing scene layout plausibility, and achieves a better performance to [18], even though it is only trained on a small dataset without ground truth labels of interest. In addition, we can see that our layout discriminator outperforms those of the other baselines by a large margin. This demonstrate that our layout discriminator can learn more discriminative and semantically richer features than those of the baselines, possibly because our generator is more effective in synthesizing high quality and realistic layouts, which in turn causes a stronger discriminator to emerge.

4.11 Feature Visualization

We further look into the hidden units of the layout encoder and the layout discriminator to probe what semantics that has been learned by our model. We follow [63] to visualize layouts and parts of them that maximally activate specific hidden units in our networks. For the encoder, we feed object layouts into it and show input regions that most strongly activate a particular hidden unit in the last convolutional layer. For the layout discriminator, we feed the full scene layout into it and visualize its hidden units from the last layer (just before the output layer) in the same way.

The first row of Figure 10 shows the results of the encoder. We can see that some hidden units emerge to detect high-level objects (e.g., airplane and boat). The units strongly respond to some important object boundary segments, e.g., the nose and wing of the airplane in the left group of examples and the bottom of the boat in the right group of examples. These segments are indicative of object shapes and poses, which are important cues for reasoning the surrounding context of the objects. For example, a boat should

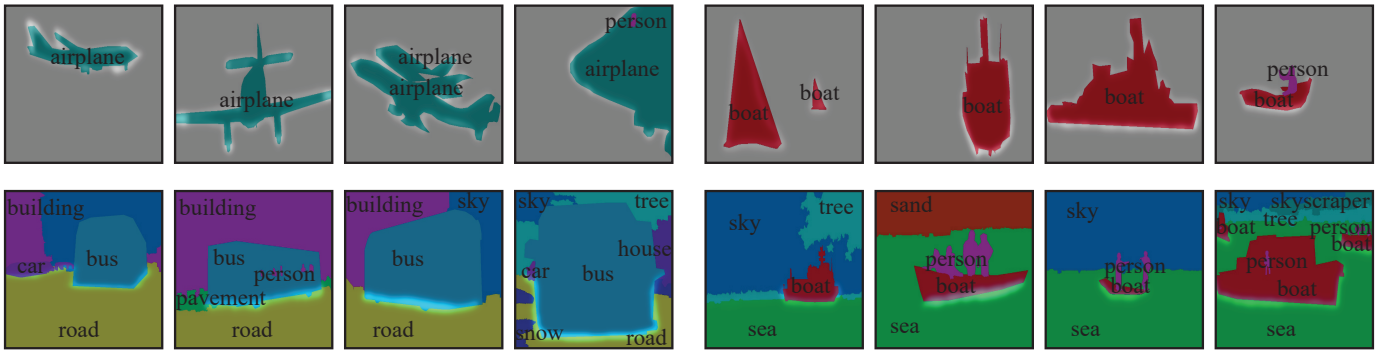


Fig. 10: Hidden unit visualization of the encoder (first row) and layout discriminator (second row). Top: for a specific hidden unit of the encoder, we show four object layouts and the regions of them that maximally activate it. Bottom: for a specific hidden unit of the layout discriminator, we show four scene layouts and the regions of them that maximally activate it.

be above the sea or a river, instead of grass. The second row of Figure 10 shows the results of the layout discriminator. We can see that some hidden units focus on the boundary between two objects, e.g., the boundary between road and bus, and the boundary between boat and sea. These boundary regions correspond to the relationships between two objects (e.g., bus is on the road and boat is on the sea). These results demonstrate that our encoder and layout discriminator have learned some semantics for scene context prediction.

5 Conclusion

In this paper, we make an effort to address the problem of reasoning about the missing environment from the properties of a few standalone objects. To this end, we propose a scene context prediction model that estimates scene layouts from input object layouts in an end-to-end manner. Extensive qualitative and quantitative results show that our model is able to generate more plausible and diverse scene layouts that can put the input objects into the right context, as compared with the baseline models.

We have demonstrated that the ability to predict scene contexts enables an image synthesis approach that can generate full scene images from only sparse, partial user inputs. We have also shown that learning to hallucinate scene contexts can be a promising supervisory signal to learn useful features for scene recognition and fake scene detection. The code is available at our website[1].

As a future work, it would be interesting to investigate how the complex interactions of multiple objects would determine the scene context, and study different ways for users to specify input objects and their relations (e.g., the user inputs can be in the form of a graph or a sketch).

Acknowledgments

This work was partially supported by a GRF grant from the Research Grants Council of Hong Kong (Ref. No.: 11205620).

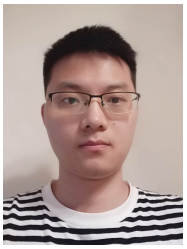
References

[1] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *CVPR*, 2018.
[2] Y. Liu, R. Wang, S. Shan, and X. Chen, "Structure inference net: Object detection using scene-level context and instance-level relationships," in *CVPR*, 2018.
[3] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, "An empirical study of context in object detection," in *CVPR*, 2009.
[4] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *CVPR*, 2014.
[5] A. Torralba, "Contextual priming for object detection," *IJCV*, vol. 53, no. 2, pp. 169–191, 2003.
[6] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE TPAMI*, vol. 32, no. 10, pp. 1744–1757, 2010.
[7] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.
[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017.
[9] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *CVPR*, 2017.
[10] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *CVPR*, 2018.
[11] X. Qiao, Q. Zheng, Y. Cao, and R. W. Lau, "Tell me where i am: Object-level scene context prediction," in *CVPR*, 2019.
[12] M. Bar, "Visual objects in context," *Nature Reviews Neuroscience*, vol. 5, no. 8, p. 617, 2004.
[13] M. M. Chun and Y. Jiang, "Contextual cueing: Implicit learning and memory of visual context guides spatial attention," *Cognitive psychology*, vol. 36, no. 1, pp. 28–71, 1998.
[14] N. Dvornik, J. Mairal, and C. Schmid, "Modeling visual context is key to augmenting object detection datasets," in *ECCV*, 2018.
[15] A. Torralba, K. Murphy, and W. Freeman, "Using the forest to see the trees: Object recognition in context," *Communications of the ACM*, 2010.
[16] C. Desai, D. Ramanan, and C. C. Fowlkes, "Discriminative models for multi-class object layout," *IJCV*, vol. 95, no. 1, pp. 1–12, 2011.
[17] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun, "Box in the box: Joint 3d layout and object reasoning from single images," in *ICCV*, 2013.
[18] M. J. Choi, A. Torralba, and A. S. Willsky, "Context models and out-of-context objects," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 853–862, 2012.
[19] H. Izadinia, F. Sadeghi, and A. Farhadi, "Incorporating scene context and object layout into appearance modeling," in *CVPR*, 2014.
[20] J.-T. Chien, C.-J. Chou, D.-J. Chen, and H.-T. Chen, "Detecting nonexistent pedestrians," in *ICCVW*, 2017.
[21] X. Wang, R. Girdhar, and A. Gupta, "Binge watching: Scaling affordance learning from sitcoms," in *CVPR*, 2017.
[22] Z. S. Kermani, Z. Liao, P. Tan, and H. Zhang, "Learning 3d scene synthesis from annotated rgb-d images," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 197–206.
[23] Q. Fu, X. Chen, X. Wang, S. Wen, B. Zhou, and H. Fu, "Adaptive synthesis of indoor scenes via activity-associated object relation graphs," *ACM TOG*, vol. 36, no. 6, p. 201, 2017.
[24] S. Qi, Y. Zhu, S. Huang, C. Jiang, and S.-C. Zhu, "Human-centric indoor scene synthesis using stochastic grammar," in *CVPR*, 2018.
[25] K. Wang, M. Savva, A. X. Chang, and D. Ritchie, "Deep convolutional priors for indoor scene synthesis," *ACM TOG*, vol. 37, no. 4, p. 70, 2018.
[26] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks," *IEEE TPAMI*, 2020.

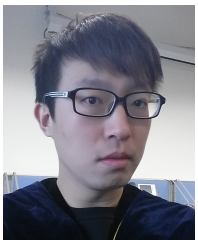
1. https://xtqiao.com/projects/object2scene/

[27] A. A. Jyothi, T. Durand, J. He, L. Sigal, and G. Mori, "Layoutvae: Stochastic scene layout generation from a label set," in *CVPR*, 2019, pp. 9895–9904.
[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013.
[29] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015.
[30] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating the future by watching unlabeled video," 2016.
[31] F. Tan, C. Bernier, B. Cohen, V. Ordonez, and C. Barnes, "Where and who? automatic semantic-aware person composition," in *WACV*, 2018.
[32] H. Zhao, X. Shen, Z. Lin, K. Sunkavalli, B. Price, and J. Jia, "Compositing-aware image search," in *ECCV*, 2018.
[33] 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 *CVPR*, 2018.
[34] S. Hong, D. Yang, J. Choi, and H. Lee, "Inferring semantic layout for hierarchical text-to-image synthesis," in *CVPR*, 2018.
[35] J. Johnson, A. Gupta, and L. Fei-Fei, "Image generation from scene graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1219–1228.
[36] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.
[37] J. Pang, C. Li, J. Shi, Z. Xu, and H. Feng, "Cnn: Fast tiny object detection in large-scale remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, 2019.
[38] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *NeurIPS*, 2015.
[39] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *ICCV*, 2017.
[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.
[41] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *ICCV*, 2017.
[42] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *CVPR*, 2019.
[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.
[45] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *CVPR*, 2019.
[46] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *NeurIPS*, 2017.
[47] 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.
[48] K. He, J. Sun, and X. Tang, "Guided image filtering," in *ECCV*, 2010.
[49] T. Liu, S. Chaudhuri, V. G. Kim, Q. Huang, N. J. Mitra, and T. Funkhouser, "Creating consistent scene graphs using a probabilistic grammar," *ACM TOG*, 2014.
[50] 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.
[51] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, 2017.
[52] X. Qi, Q. Chen, J. Jia, and V. Koltun, "Semi-parametric image synthesis," in *CVPR*, 2018.
[53] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015.
[54] W. Chen and J. Hays, "Sketchygan: Towards diverse and realistic sketch to image synthesis," in *CVPR*, 2018.
[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.
[56] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *NeurIPS*, 2014.
[57] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *CVPR*, 2010.
[58] K. Van De Sande, T. Gevers, and C. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE TPAMI*, 2009.
[59] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.
[60] P. Wang, L. Liu, C. Shen, Z. Huang, A. van den Hengel, and H. T. Shen, "What's wrong with that object? identifying images of unusual objects by modelling the detection score distribution," in *CVPR*, 2016.
[61] S. Aneja, C. Bregler, and M. Nießner, "Catching out-of-context misinformation with self-supervised learning," *arXiv:2101.06278*, 2021.
[62] M. J. Choi, A. Torralba, and A. S. Willsky, "A tree-based context model for object recognition," *IEEE TPAMI*, 2011.
[63] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *NeurIPS*, 2016, pp. 82–90.

Xiaotian Qiao received the B.Eng. and M.Sc. degrees in information and communication engineering from Zhejiang University, China, and the Ph.D. degree in computer science from City University of Hong Kong. He is now a Postdoctoral Researcher in the Department of Computer Science, City University of Hong Kong. His research interests include computer vision and computer graphics.



Quanlong Zheng received the B.S. degree from BUAA in 2015. He is currently a PhD student at City University of Hong Kong. His research interests include computer vision.



Ying Cao received the Ph.D. degree in computer science from the City University of Hong Kong, and the M.Sc. and B.Eng. degrees in software engineering from Northeastern University, China. His research generally lies in computer graphics and computer vision. His primary research interests is data-driven graphic design.

Rynson W.H. Lau received his Ph.D. degree from University of Cambridge. He was on the faculty of Durham University and is now with City University of Hong Kong.

Rynson serves on the Editorial Board of International Journal of Computer Vision (IJCV) and Computer Graphics Forum. He has served as the Guest Editor of a number of journal special issues, including ACM Trans. on Internet Technology, IEEE Trans. on Multimedia, IEEE Trans. on Visualization and Computer Graphics, and IEEE Computer Graphics & Applications. He has also served in the committee of a number of conferences, including Program Co-chair of ACM VRST 2004, ACM MTDL 2009, IEEE U-Media 2010, and Conference Co-chair of CASA 2005, ACM VRST 2005, ACM MDI 2009, ACM VRST 2014. Rynson's research interests include computer graphics and computer vision.