GLOBAL-AFFINE AND LOCAL-SPECIFIC GENERATIVE ADVERSARIAL NETWORK FOR SEMANTIC-GUIDED IMAGE GENERATION

Susu Zhang, Jiancheng Ni*, Lijun Hou, Zili Zhou
Jie Hou and Feng Gao
Qufu Normal University, Qufu, China

(Communicated by Song Wang)

Abstract. The recent progress in learning image feature representations has opened the way for tasks such as label-to-image or text-to-image synthesis. However, one particular challenge widely observed in existing methods is the difficulty of synthesizing fine-grained textures and small-scale instances. In this paper, we propose a novel Global-Affine and Local-Specific Generative Adversarial Network (GALS-GAN) to explicitly construct global semantic layouts and learn distinct instance-level features. To achieve this, we adopt the graph convolutional network to calculate the instance locations and spatial relationships from scene graphs, which allows our model to obtain the high-fidelity semantic layouts. Also, a local-specific generator, where we introduce the feature filtering mechanism to separately learn semantic maps for different categories, is utilized to disentangle and generate specific visual features. Moreover, we especially apply a weight map predictor to better combine the global and local pathways considering the highly complementary between these two generation sub-networks. Extensive experiments on the COCO-Stuff and Visual Genome datasets demonstrate the superior generation performance of our model against previous methods, our approach is more capable of capturing photo-realistic local characteristics and rendering small-sized entities with more details.

1. Introduction. Generating high-resolution images that semantically consistent with various input types is a frontier but challenging task. It has tremendous practical applications, such as intelligent image editing [38], game generation [12], and face representation interpreting [29]. Recently, thanks to the generative adversarial networks (GANs), which have been at the helm of remarkable advances in image synthesis, GAN-based image generation methods [11, 24, 2] greatly drive research progress in multi-modal feature learning and visual distribution modeling.

Many existing methods have created stunning results in limited domains. However, most of them are trained on relatively simple datasets that only contain one instance at the center. Once the conditional inputs contain more complex scenarios where descriptions are with diverse objects and complex interactive relationships, the performance of these methods degrades drastically. This is likely because many

2020 Mathematics Subject Classification. Primary: 58F15, 58F17; Secondary: 53C35.

Key words and phrases. generative adversarial network, graph convolutional network, semantic layout construction, feature filtering mechanism, weight map predictor.

* Corresponding author: nijch@163.com.
generation models use the unstructured text as the input, thus its ambiguity property and unclear entity-relationship pairs will definitely increase the difficulty of cross-modal synthesis. There are also many other methods that directly input segmentation labels or images, yet it’s not flexible to manipulate these input types since they heavily rely on domain labels or paired data [5]. Additionally, a growing body of work has generated images directly from the input descriptions without using any intermediate semantic representations. As a result, the interactive relationships and locations of instances cannot be sufficiently utilized during the generation process, leading to unfaithful layouts and poor image quality.

We also observe that recent approaches mainly focus on image-level synthesis and predict category features by simply employing the identical network architecture, lack specific networks for learning various semantic classes. Thus, it is extremely detrimental to the generation of local details since all instance categories are treated equally. Moreover, different instance categories have various sizes and imbalanced numbers. For example, the sky, ocean, and road always occupy larger pixel areas in an image than the small-sized instances, such as traffic signs and telegraph poles. Also, the category number of training images is imbalanced, we note that the buses, cars, and automobiles in Dayton dataset [35] are extremely rare and less than 2% in terms of all pixels of training samples. In this case, it will inevitably make the feature learning of the generative model be dominated by the large-scale instance categories since the convolutional network usually shares the parameters at different convolutional positions, the small-sized instances could be easily ignored when synthesizing the final images, further making it merely impossible to generate images with sharp small-sized instances.

To make these aforementioned problems resolvable, we choose scene graphs, the more structured representations, as the inputs to avoid unclear and ambiguous descriptions. The graph nodes indicate objects and the edges represent the relationships between different objects. We then adopt the graph convolutional network (GCN) [14] to calculate the instance locations and spatial relationships so as to construct the global semantic layouts. Instead of synthesizing images directly from the inputs, we use the inferred layouts as guidance to learn the related instance features. Hence, the complexity of this generation task should be reduced and the layouts of generated images can be more reasonable as well.

Furthermore, considering the advantages of the multi-stage generation strategy [23], we specifically implement a global-affine and also a local-specific generator for image synthesis. In particular, the feature filtering mechanism proposed in the local-specific generator is employed to pay attention to all kinds of instances, especially the tiny ones, and learn the diverse and detailed features. Beyond that, we introduce a weight map predictor to combine these two generation branches. In this way, our model is able to take both the global and local contexts into account, make them complement each other, and output photo-realistic images.

As such, our contributions are summarized in the following aspects:

1) We leverage the graph convolutional network to calculate the instance embedding and further construct global semantic layouts containing bounding boxes and instance shape masks. The predicted layouts, as the intermediate constraints, are used to learn high-fidelity semantic information of scene classes.

2) To avoid the learning of a biased generation space, we put forward a feature filtering mechanism, which is beneficial to the synthesis of small-scaled instances. It specifically captures various category representations and ensures that our model
maintains high-fidelity alignment between the generated results and the predicted layouts.

3) We propose a weight map predictor that infers two pixel-level weight maps to integrate the global and local pathways in a weighted-combination. Thus our GALS-GAN is capable of combining the advantages of these two generators and synthesizing images with reasonable global appearances and rich local details.

2. Related work.

2.1. Generative adversarial networks. Generative Adversarial Networks (GANs) [8] have shown advanced ability in synthesizing realistic-looking images even based on complex datasets. Formulated as a two-player game, a vanilla GAN is comprised of a generator and a discriminator that both try to optimize the opposing objective function. The generator aims to synthesize the new samples that are indistinguishable from real images, while the discriminator focuses on recognizing the generated images from the real ones.

The GAN-based models have been widely studied for a variety of tasks, such as the image super-resolution [3], face editing task [4], video generation task [37], image-to-image translation [17], and image generation task [30]. Hence, considering the powerful capabilities of GAN, we also choose it as our basic network to perform high-quality image synthesis.

2.2. Multi-modal conditional image synthesis. There are a variety of input types in conditional image synthesis. Tang et al. [34] chose the original images as inputs and distilled them into low-dimensional latent vectors so as to output the target images. Analogously, Liu et al. [21] trained a class-conditional GAN with the input semantic labels to improve the image diversity. Besides, Sun et al. [31] focused on the layout-to-image generation and proposed an intuitive paradigm to bridge the gap between input labels and generated images. Given a scene sketch, Gao et al. [7] implemented a controllable image generation method to meet the specific requirements. Also, there are many generative models [18, 9, 23, 25] that take the text as the input for multi-modal text-to-image generation.

However, these leading approaches have struggled with complex descriptions containing many scene objects and unclear interactions, most existing text-to-image methods fail to output vivid instances. To this end, we explicitly convert the semantic information conveyed by a complex text into a scene graph. Then we adopt the GCN to process the structured representation and compute instance embedding vectors, which are further used to infer bounding boxes and instance masks, so as to form the global semantic layout for final image generation.

2.3. Image generation from Scene graphs. Johnson et al. [10] first proposed to generate images from scene graphs, they implemented the sg2im method to reason related objects and relationships. Then Vo et al. [36] adopted the scene structure in the conditional GAN network and put forward the stacking-GANs to infer visual-relation layouts. With the same input form, Li et al. [19] proposed the PasteGAN to crop objects from the external memory tank and paste them into correct locations of the final images. Another work by Li et al. [20] presented a general sequence-to-sequence framework in which a scene graph is transformed into a series of fragments to infer the semantic layout.

Different from the aforementioned methods, Dhamo et al. [6] focused on image manipulation and produced modified images from the edited scene graphs. Yet,
this method requires both the original image and the corresponding scene graph as
the supervision, it cannot generate target samples freely. A recent development in
this field has been exploited by Sylvain et al. [32] who showed a scene-graph sim-
ilarity module to improve the layout fidelity and a novel metric to better measure
the model performance. Despite these advances, the above models lack investiga-
tion in synthesizing semantically consistent and detailed features for each instance,
especially the small-scaled one that can be easily ignored by GAN.

We overcome these above two shortcomings by proposing an efficient global-affine
and local-specific generation network architecture, GALS-GAN, in which we espe-
cially construct a feature filtering mechanism to concentrate on each specific class
being, thus the model can largely get rid of the interference from the large-scale
instance classes. At the same time, we fully take into account the complementary
property of the global and local pathways and implement a weight map predictor.
By so doing, the predicted maps, act as priors, guide generators to learn corre-
sponding sharp image features during the joint optimization.

3. The proposed GALS-GAN. Our GALS-GAN model is based on the GAN
framework and mainly consists of two parts, an overview of this method is shown in
Fig. 1. The first part is semantic layout generation where we implement the graph
convolutional network to calculate object locations and shapes from the input scene
graphs. Conditioned on the inferred layouts, we especially adopt a weight-map
predictor in the second part to fuse the global-affine generator and the local-specific
generator so as to synthesize visual-realistic images.

![Figure 1. Overview of the proposed GALS-GAN.](image)

3.1. Semantic layout construction.

3.1.1. Scene graph embedding via GCN. The graph convolutional network (GCN)
can directly operate on graphs. Following [10, 36, 19], we also take the scene graph
as input and calculate new embedding vectors for each node and edge. Additionally,
we apply the same function on each graph convolutional layer, which ensures a single
layer can work with arbitrarily shaped graphs.

We first construct the semantic-related scene graph containing a set of instance
classes $\mathcal{C}$ and relationships $\mathcal{R}$. A scene graph is a tuple $(O, E)$ where $O = \{o_1, \cdots, o_T\}$
indicates a set of instances with each \( o_\ell \in C \), and \( E \subseteq O \times R \times O \) represents a set of edges of the form \( o_\ell, r_k, o_j \), the relationship \( r_k \in R \) and the instances \( o_\ell, o_j \in O \). In terms of each tuple \((o_\ell, r_k, o_j)\), we leverage a learned embedding layer which is similar to the embedding layer employed in neural language models, to convert each node and edge to the object embedding vector \( v_i \in R^{D_{in}} \) and the predicate embedding \( v_{r_k} \in R^{D_{in}} \). The obtained embedding vectors \( v_i, v_{r_k} \in R^{D_{in}} \) are further used as the inputs of GCN.

Fig. 2 illustrates a single graph convolution layer for calculating instance embedding vectors. Given input vectors \( v_i, v_{r_k} \in R^{D_{in}} \) with dimension \( D_{in} \) for the instance \( o_i \in O \) and the edge \((o_i, r_k, o_j) \in E \), we adopt three functions \( f_s, f_p, \) and \( f_o \) to calculate the output vectors \( v_i, v_{r_k} \) for the related graph node and edge.

We adopt the function \( f_p \) to calculate the output vector \( v_{r_k} \) for the input edge embedding \( v_{r_k} \):

\[
v_{r_k}' = f_p(v_i, v_{r_k}, v_j)
\]

where the function \( f_p \) is implemented with convolutional networks, the output vector \( v_{r_k}' \in R^{D_{out}} \) with dimension \( D_{out} \) indicates the new vector of the predicate \( r_k \).

However, it is more difficult to calculate the node vector since an instance may exist in many relationships, as such a node may participate in several edges. Considering this, we divide the edges connected to the node \( o_i \) into two parts: the subject edges starting from \( o_i \) and the predicate edge ending at \( o_i \).

When the entity \( o_i \) acts as the subject, we adopt the function \( f_s \) to calculate a candidate vector for each edge starting at \( o_i \), gathering all such candidates in the set \( V^s_i \):

\[
V^s_i = \{f_s(v_i, v_{r_k}, v_j) : (o_i, r_k, o_j) \in E\}
\]

where \((o_i, r_k, o_j) \in E\) is the edge with the entity \( o_i \) as the starting node and the entity \( o_j \) as the ending node, and the triple \((v_i, v_{r_k}, v_j)\) indicates the input vectors (calculated by the aforementioned embedding layer) respectively of the subject \( o_i \), predicate \( r_k \), and object \( o_j \).

Conversely, when the entity \( o_i \) acts as the object, its candidate vector set is calculated by:

\[
V^o_i = \{f_o(v_j, v_{r_k}, v_i) : (o_j, r_k, o_i) \in E\}
\]
where the implementation of function \( f_o \) is similar to that of function \( f_s \), both of which are completed by a single network to concatenate three input vectors, \((o_j, r_k, o_i) \in E \) represents the edge with the entity \( o_j \) as the starting node and \( o_i \) as the ending node. This is just the opposite of formula (2) in which the entity \( o_i \) appears as the subject. The triple \((v_j, v_r, v_i) \) indicates the input vectors of the subject \( o_j \), predicate \( r_k \), and object \( o_i \), respectively.

Then the candidate vectors \( V_s^i \) and \( V_o^i \) are fed into a symmetric pooling function \( h \) to output the new instance vector \( v'_i \) for the entity \( o_i \):

\[
v'_i = h(V_s^i \cup V_o^i)
\]

where the function \( h \) averages the input set of vectors, feeds the results to MLP shown in Fig. 3, and outputs the new vector \( v'_i \) which is a learned robust instance representation.

![Figure 3. Architecture of the MLP.](image)

As demonstrated in Fig. 3, we implement a Multi-Layer Perceptron (MLP) to convert the set of candidate vectors \( V_s^i \) and \( V_o^i \) into a single vector output vector \( v'_i \) of the dimension \( D_{out} \). We collect all of the candidate vectors for each instance in the graph, process them with the symmetric pooling function \( h \) illustrated in formula (4), and take \( V_s^i \) and \( V_o^i \) as input and performs an average operation. Then the averaged result is utilized to calculate the output vector \( v'_i \in \mathbb{R}^{D_{out}} \) for the entity \( o_i \).

In this way, we obtain the embedding vector for each instance after processing the whole input scene graph with a series of graph convolution layers. The output instance representation will be further reassembled into a semantic layout to indicate the image spatial configuration.

3.1.2. Semantic layout prediction. To generate the final image, we construct a semantic layout as the intermediate representation (a 2D spatial arrangement of instances) to reduce the gap between the graph domain and the image domain.

As shown in Fig. 1 (a), we employ a box predictor, which takes the final instance embedding calculated by GCN as the input, to infer the bounding box \( B_i = (x_0, y_0, x_1, y_1) \), \( B_i \in \mathbb{R}^4 \) for each instance where \( x_0, x_1 \) are the left and right coordinates and \( y_0, y_1 \) indicates the bottom and top coordinates of the bounding box. All coordinates are normalized to be in the range \([0, 1]\).

The mask predictor is responsible for predicting segmentation masks, it consists of a sequence of up-sampling and convolution layers and terminates in a sigmoid nonlinearity. We feed instance embedding into this predictor and apply masking operation so that regions outside the bounding box \( B_i \) are set to 0, if and only if the box contains the corresponding class-label we set its element to 1.
To transform the scene graph representations calculated by the GCN into a two-dimensional spatial semantic layout, we inject the shape mask inferred by the mask predictor for each instance into its corresponding bounding box and perform the mask embedding operation in it. The process of the mask predictor inferring shapes for multiple instances is depicted in Fig. 4.

![Figure 4. Inferring process of the mask predictor.](image)

For each instance, the mask predictor encodes its bounding box tensor $B_i$ into a binary one $B_i \in \{0, 1\}^{w \times h \times l_i}$, where $w \times h$ represents the entity size and $l_i$ indicates the category label. As depicted in Fig. 4, we down-sample the bounding box feature through a down-sampling module composed of a $3 \times 3$ convolution with stride 2, batch normalization, and ReLU activation. The down-sampled feature is then inputted into a bi-directional LSTM and cascaded with the random noise $z$. If and only if the bounding box contains the relevant instance label, the binary tensor is set to 1, and the other regions outside the bounding box are set to 0. After this mask operation, we input the obtained instance mask feature into a residual unit, which is composed of three $3 \times 3$ convolutions (stride 1) and a skip connection. The design of the skip connection enables the mask predictor to possess a deeper encoding ability. Finally, we adopt an up-sampling module containing a $4 \times 4$ deconvolution with stride 2, batch normalization, and ReLU activation to infer the instance mask $m_i$ with a pixel range of $(0,1)$, which can be further used for the related instance generation.

The inferred spatial mask $m_i \in \mathbb{R}^{M \times M}$ should match the location and class information of the box $B_i$. Also, it is supposed to add instance-wise constraints to ensure the alignment between each instance shape and its surrounding context. To this end, we use $B_i$ to project the instance mask $m_i$ in the proper 2D area. Hence, the projected mask region is filled with the related features while the remaining regions are padded with zeros. We repeat this process for all instances and aggregate these boxes and masks into a single semantic layout $L$ for further image generation.

### 3.2. Semantic-guided image generation.

#### 3.2.1. Global-affine generator

The large instance usually occupies a larger pixel area than the small-scaled one, also, the large-sized instance features are more prominent, all of these lead to its dominance in the feature learning process. Moreover, generative models tend to ignore or even omit the generation of small-sized entities. To solve these problems, we leverage the encoder $E$ to encode the input semantic layout $L$ and specifically decompose the generation process to prevent different-level generators from learning a biased generation space.

We employ two generators, a global-affine generator as well as a local-specific generator, to synthesize images from predicted layouts. As shown in Fig. 1 (b), we apply three different pathways and design an encoder $E$ that utilizes the shared-parameter in these pathways to ensure a compact backbone architecture. In these three branches: global-affine generator, weight map predictor, and local-specific
generator, the encoder $E$ shares parameters and plays a role of guidance to acquire the image features corresponding to each category spatially, which are further utilized in separate generation branches. By so doing, the gradients from all branches together contribute to the learning of the encoder, it thus can not only obtain local and global information but also obtain the corresponding relationships between them. Meanwhile, the encoder $E$ is endowed to encode rich feature representations from the predicted layout and learn diverse visual information of different entities. Therefore, the GALS-GAN has the capability of avoiding unfavorable interference of different branches, which makes the global-affine and local-specific generation benefit each other during the optimization process.

The encoded features of the predicted layout $L$ are denoted as $E(L)$ and then input into the global-affine generator $G_{g-a}$, which is implemented with the SPADE architecture \cite{24}, to affine the layout into the corresponding coarse-grained image $I_G$. Concretely, the global target image $I_G$ is synthesized by a feed-forward computation:

$$I_G = G_{g-a}(E(L))$$ (5)

### 3.2.2. Local-specific generator.

The imbalance of training data and the size diversity of instances make it difficult for the global-affine generator to synthesize fine-grained images. To this end, as illustrated in Fig. 1 (b), we propose a novel local-specific generator in which each category branch has independent parameters and focuses on the generation of each specific class.

**Feature filtering module** To avoid the interference from large instance categories, we specifically employ a feature filtering module in our local-specific generator. It separately generates each category and yields richer local details, the architecture of our local-specific generator $G_{l-s}$ is illustrated in Fig. 5. We feed the encoded features $E(L)$ into two deconvolutional layers to increase the spatial size and decrease the channel number by two times in order to obtain the scaled feature map $M' \in \mathbb{R}^{w \times h \times n}$, which is then multiplied by the semantic mask of each category $l_t$. In this way, we obtain a filtered local-specific feature map for each class. The semantic-layout guided feature filtering is represented as:

$$M_t = M' \times l_t (t = 1, 2, \cdots, T)$$ (6)

where the scaled feature map $M'$ is multiplied by the semantic mask of each category (i.e., $l_t$), to get filtered local-specific feature map for each class, $T$ indicates the number of instance categories, and $M_t \in \mathbb{R}^{w \times h \times n}$ ($w$, $h$, and $n$ are the width, height, and the number of channels, respectively) denotes the filtered feature of the $t^{th}$ semantic mask. We feed $M_t$ into convolutional layers to generate fine-grained features $I_{L_t}$ for the $t^{th}$ semantic class.

Finally, we perform convolutional operations on all of the local-specific results for the final image $I_L$ synthesis:

$$I_L = \text{conv}(I_{L_1} \oplus I_{L_2} \oplus \cdots \oplus I_{L_T})$$ (7)

where $\oplus$ denotes the channel-wise concatenation.

**Instance-level feature prediction module.** We observe that the filtered feature map $M_t$ tends to produce similar target images for different classes, especially for the small-sized categories. Besides, most of the existing methods do not sufficiently take the instance-level representations into consideration. To overcome this limitation, as shown in Fig. 5 (b), we propose an instance-level feature
prediction module with the aim of obtaining distinguishable features for different instance categories. We perform the element-wise addition of all filtered feature maps \(\{M_1, M_2, \cdots, M_T\}\) to obtain the packed semantic map \(M_p \in \mathbb{R}^{w \times h \times n \times T}\) where \(T\) denotes the number of instance categories.

Considering that bounding box overlap and instance pixel occlusion may occur when the defined scene structure contains multiple entities and interactions, we perform the maximum pooling operation on the pixel overlap areas and selects the most relevant instance feature vector in our local-specific generator. Further, we carry out the most relevant pixel representation on the overlapping position to make the generated visual features meet the constraints of semantic layout. Taking the \(i\)-th instance as an example, we multiply its instance mask feature vector \(m_i\) with the packed semantic map feature \(M_p\) to calculate a clear and distinguishable semantic feature map \(M'_p\):

\[
M'_p = \max_{1 \leq i \leq T} m_i \otimes M_p
\]

where \(\otimes\) represents the outer product of vectors. After this maximum pooling operation, the most relevant instance pixel is selected for the corresponding feature representation, thereby solving the problem of instance pixel overlap.

The commonly used global average pooling layer \cite{22} tends to wash away important spatial information, as a result, the learned image attributes cannot preserve crucial cues for disentangling diverse visual features. To address the above problem, we input \(M'_p\) into a semantic-guided pooling layer, which can better preserve semantic information, to learn the pooled feature with the dimension of \(T \times n \times 1 \times 1\). Then we apply a fully-connected layer to calculate the classification probabilities \(P_{cls} \in \mathbb{R}^{T \times T}\) for the \(T\) instance classes. Our model predicts a \(1 \times T\) one-hot vector for each category probability. Considering that the input filtered map only contains some specific categories, the attributes related to the void categories should not contribute to the classification loss. Therefore, we specifically adopt the cross-entropy loss \(\mathcal{L}_{\text{cross-entropy}}\) containing an invalid class indicator \(\beta_j\) to learn discriminative
and instance-level features for each input filtered map:

\[
\mathcal{L}_{\text{cross-entropy}} = -\sum_{j=1}^{T} \sum_{t=1}^{T} \beta_j \log(h_{\text{cls}}(M_t))
\]

where the invalid class indicator \( \beta = \{\beta_j\}_{j=1}^{T} \) is a one-hot vector, we set \( \beta_j = 1 \) for the valid category and \( \beta_j = 0 \) for the void category. \( P'_{\text{cls}} \) denotes a set of labels of all the instance classes. With the filtered feature map \( M_t \) as input, the function \( h_{\text{cls}}(\cdot) \) is responsible for computing the classification probability. \( 1(\cdot) \) functions as the indicator that returns 1 if the input semantic map contains the instance category \( (P'_{\text{cls}}(t) = t) \) otherwise returns 0.

3.2.3. Weight map predictor. As shown in Fig. 1 (b), We tackle the image composition task with the layered structure modeling in which we propose a weight map predictor \( G_{\text{weight}} \) to integrate the two generators. Our main idea resembles how a painter outlines the overall global structure and then embellish the local details to populate and perfect the final scene. Thus, GALS-GAN is allowed to handle the instance affine transformations and resolve the occlusion artifacts.

We implement two blocks in \( G_{\text{weight}} \) including a transposed convolution followed by InstanceNorm and ReLU and a convolution followed by InstanceNorm and ReLU. We set the kernel size as \( 1 \times 1 \) with stride 1 for the last layer and other layers are set as \( 3 \times 3 \) with stride 2 for weight map prediction. And we obtain the two-channel weight map \( w_p \) by:

\[
w_p = \text{softmax}(G_{\text{weight}}(E(L)))
\]

where the channel-wise function \( \text{softmax}(\cdot) \) is utilized for normalization, the sum of the weights for the same pixel thus should be equal to 1.

In this way, our model learns the aggregated representations without suffering the combination explosion. The two-channel weight map \( w_p \) is disentangled to obtain the weight maps \( w_G \) and \( w_L \) for the global-affine and local-specific branches, respectively. We then fuse the samples generated from two generators and the weight maps by element-wise multiplication to infer the final high-quality image \( I \):

\[
I = I_G \otimes w_G + I_L \otimes w_L
\]

where \( \otimes \) denotes an element-wise multiplication operation, \( I_G \) and \( I_L \) represents the global-affine generation and local-specific generation. By so doing, our global-affine generator \( G_{g-a} \) and local-specific generator \( G_{l-s} \) are enabled to directly take advantage of these two semantic-level weight maps and make the different-level generators contribute to each other during training optimization.

3.2.4. Multi-scale discriminators. We design a pair of discriminators \( D_{\text{img}} \) and \( D_{\text{inst}} \), as depicted in Fig. 6, to ensure plausible images. With the generated sample \( I \) as input, the image-level discriminator \( D_{\text{img}} \) adopts two convolutional blocks each with stride 2 to decrease spatial dimension and extract global features \( f_{\text{global}} \). In this case, \( D_{\text{img}} \) is trained to classify the image as real or fake by maximizing the following unconditional loss \( \mathcal{L}_{\text{uncon}} \):

\[
\mathcal{L}_{\text{uncon}} = \mathbb{E}_{I_{gt} \sim p_{\text{data}}} \left[ \log D(I_{gt}) \right] + \mathbb{E}_{I \sim p_G} \left[ \log(1 - D(I)) \right]
\]

where \( I_{gt} \) is the ground truth image, \( p_{\text{data}} \) represents the real image distribution. \( p_G \) defines the distribution of the generated sample \( I \).

Given the generated image \( I \) and the predicted semantic layout \( L \) which contains the bounding boxes \( B_{1:T} \) and the instance shapes \( m_{1:T} \), as shown in Fig. 6, our
instance-level discriminator $D_{inst}$ concatenates them in the channel dimension, then feed into image encoder to obtain the encoded features. To estimate the accuracy of inferred bounding boxes, in which the number of predicted boxes is set to be the number of entities in an image, we employ the pre-trained Faster R-CNN [27] with the ROI-align layer to extract the local-specific feature vector $f_i$:

$$\{f_i\}_{i=1}^T = FasterRCNN(I, m_{1:T}, B_{1:T})$$ (13)

where the instance feature vector $f_i$ is processed by a convolutional layer.

To determine whether the generated result is consistent with the ground truth, the instance-level discriminator $D_{inst}$ classifies its input as real or fake by maximizing the conditional compound loss $L_{con}$:

$$L_{con} = \lambda_1 L_{img}^1 + \lambda_2 L_{feature}^1 = ||I_{gt} - I||_1 + \sum_{t=1}^T ||v'_i - f_i||_1$$ (14)

where $\lambda_1$ and $\lambda_2$ denote weighting factors, $L_{img}^1$ penalizes the $L_1$ differences between the ground truth $I_{gt}$ and the synthesized image $I$. Similarly, $L_{feature}^1$ also utilizes the $L_1$ penalty to ensure that the feature $f_i$ re-extracted from the generated image conforms to the instance embedding $v'_i$ calculated by GCN.

3.3. Objective function. Combining the aforementioned cross-entropy, conditional, and unconditional losses, we define the final objective function as follows:

$$L = L_{cross-entropy} + \lambda_1 L_{img}^1 + \lambda_2 L_{feature}^1 + \lambda_3 L_{uncon}$$ (15)

where $\lambda_i$ are the hyper-parameters that balance different loss terms. In our implementation, the generators and multi-scale discriminators are trained adversarially to optimize the complete objective.

4. Experiments.

4.1. Experiment settings. We utilize Ubuntu 16.04.7 LTS operating system and PyTorch 1.0.1 framework, implement our model on a single Tesla P100 with 16GB video memory, to synthesize the final 64 x 64 and 128 x 128 images. GALS-GAN is trained in an end-to-end manner using Adam optimizer [13] with the recommended parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Considering that diverse learning rates are beneficial for stabilizing the architecture, we set the learning rate to $1e^{-4}$ for all components except the mask predictor, where we adopt a smaller learning rate of $1e^{-5}$. Besides, we use the batch size of 32 for 200 epochs, the hyper-parameters in the objective function are set to $\lambda_1 = \lambda_3 = 1$ and $\lambda_2 = 10$. 
Datasets. We evaluate our GALS-GAN qualitatively and quantitatively on two datasets, COCO-Stuff [1] and Visual Genome [15], which contain diverse entities and interactive relationships. The details of the datasets are shown in Table 1. COCO-Stuff is derived from the COCO dataset [16] and has pixel-level instance annotations. Following [10], the instances covering less than 2% of the whole image are ignored, we only make use of the images with 3 to 8 target instances.

Each image in the Visual Genome dataset is divided into several regions, in which the objects and relationships are extracted to form sub-graphs and then construct the complete scene graph. We also remove images that contain small-sized and infrequent objects and split the dataset into train, validation, and test sets at a ratio of 8:1:1. Thus, as shown in Table 1, we have left 62565 train, 5506 validation, and 5088 test images, and all the images contain a total of 178 instance types.

Table 1. Statistics of COCO-Stuff and Visual Genome datasets.

| datasets          | train | val  | test | categories | max | min |
|-------------------|-------|------|------|------------|-----|-----|
| COCO-Stuff        | 74121 | 1024 | 2048 | 171        | 8   | 3   |
| Visual Genome     | 62565 | 5506 | 5088 | 178        | 30  | 3   |

Evaluation metrics. To access the generated image quality and diversity, we adopt two commonly used metrics: Inception Score (IS, higher is better) and Fréchet Inception Distance (FID, lower is better), which are calculated by the pre-trained Inception network [33]. Moreover, we utilize the recently proposed Classification Accuracy Score (CAS) [26] to encourage the generation of recognizable instances. For this, we train a classifier on the generated images and validate it on the original test set. It will indicate that the trained data conforms to the real distribution if the classifier achieves high accuracy. To assess how well the inferred layouts match the ground truth, we employ R@τ, the recall with an IoU (Intersection-over-Union) threshold τ, to evaluate the quality of predicted bounding boxes. R@0.3 and R@0.5 (the thresholds are set to 0.3 and 0.5) are specifically reported in our experiment.

4.2. Qualitative results. The samples generated by our different-level generators are shown in Fig. 7. The results of stage I and stage II are synthesized by the global-affine generator and local-specific generator, respectively. The former simply converts the inferred semantic layouts into coarse-grained images, it fails to synthesize fine-grained attributes. By contrast, the local-specific generator especially employs a feature filtering mechanism in stage II to learn discriminative features for each category, so the produced samples have vivid details and fewer visual artifacts.

As demonstrated in Fig. 8, GALS-GAN generates recognizable and semantically meaningful images on the complex COCO-Stuff dataset. It learns instance-level information from the predicted semantic layouts and is capable of generating diverse scenes that contain multiple categories and instance interactions. Also, we conducted experiments on Visual Genome to prove the generalization performance of our model, the synthesized samples are shown in Fig. 9. All these qualitative examples maintain the semantic consistency with the given scene graphs.

We compare our model with three state-of-the-art methods (sg2im [10], stacking-GANs [36], and PasteGAN [19]) that allow scene-graph-to-image generation. Fig. 10 illustrates qualitative comparisons between these different methods. The sg2im infers reasonable layouts from the input scene graphs but fails to accurately render
the object attributes. Stacking-GANs has further improved image quality, but its generated instances still lack detailed features, especially the tiny ones. For example, compared with the sea and the sand, the pixels occupied by the two people (the third image of Fig. 10 (d)) are much smaller. This model omits these small-sized individuals when synthesizing the whole images, resulting in a lack of relevant object appearances. Hence, the images generated by stacking-GANs only maintain the image-level rather than the instance-level consistency with the input scene graphs. The PasteGAN constructs an external memory tank, which serves as the source materials, to provide the most compatible object crops for image generation. Yet, it is time-consuming and labor-intensive to build the tank. We also note that this method is semi-parametric and requires the selected ground truth crops, however, the selection operation is hard perform in other real-world datasets.

By contrast, there is no need to provide the additional crops in our method, our proposed feature filter module overcomes the dataset bias and object category imbalance. Therefore, GALS-GAN is able to focus on diverse instance types, even the small-sized or infrequent object classes. As shown in Fig. 10 (f), our approach successfully capture high-fidelity relationships and synthesize multiple recognizable instances.

As depicted in Fig. 11, we also achieved the novel task of image manipulation. Our model generates the corresponding image from the input (see Fig. 11 (a)). We define several modification modes and edit the original scene graph. By adding the object nodes, Fig. 11 (b) illustrates that GALS-GAN synthesizes semantically consistent instances in the target image and also keep the existing objects unchanged. This is mainly due to the instance-level feature prediction module learns the identifiable representations of the newly added objects, and the global-affine generator avoids the interference of random pixels. We also perform the removal operation by deleting the node and its related relational edges from the input scene graph, our model occludes and removes the corresponding regions in the source image.
In addition, we perform object replacement by assigning the source node to other category. As shown in Fig. 11 (e), we use the building to replace the sky. Instead of deleting the replaced node, we set its instance embedding vector to 0 since it does not describe the new instance. The primary challenge of this operation is that the replacement should be performed in the correct position, for this, we provide the original location to the new entity to make it compatible with the target image.

In Fig. 12, we display more examples of image manipulation, we show visual samples in two different settings: removal operation and instance replacement.
Compared with the former, the replacement operation is relatively difficult. Nevertheless, we still generate reasonable images which lays the foundation for our further image editing research.

4.3. **Quantitative results.** We calculate the IS and FID scores of our synthesized images and compare them with state-of-the-art methods. The quantitative comparisons on COCO-Stuff and Visual Genome are shown in Table 2 and 3, respectively. Since the PasteGAN does not generate the $128 \times 128$ images, we omit the corresponding scores. This model achieves the highest IS and lowest FID when
it directly applies the ground truth layouts for the $64 \times 64$ image synthesis. However, our results achieve the highest visual quality when all methods employ the calculated semantic layouts for the generation task.

As shown in Table 4, GALS-GAN outperforms the PasteGAN and achieves the highest classification accuracy on both datasets, indicating that our approach is capable of synthesizing discriminative images. We also utilize the entity recall R@τ to measure how the input scene structure was preserved in synthesized images. Table 5 demonstrates that GALS-GAN has the best performance, our predicted bounding boxes more accurately match the entity locations than these compared methods and the inferred semantic layouts more precisely agree with the ground
GALS-GAN FOR SCENE-GRAPH-TO-IMAGE SYNTHESIS

Figure 12. Example results of different image manipulation types.

Table 2. Quantitative comparison of images generated by different methods on the COCO-Stuff dataset.

| Methods                        | IS ↑                     | FID ↓                  |
|--------------------------------|--------------------------|------------------------|
|                                | 64 × 64                  | 128 × 128              |
|                                | 64 × 64                  | 128 × 128              |
| sg2im [10]                     | 6.7±0.1                  | 5.99±0.27              | 67.99  | 95.18   |
| stacking-GANs [36]             | 9.1±0.20                 | 12.01±0.40             | 50.94  | 39.78   |
| PasteGAN [19]                  | 9.2±0.32                 | -                      | 42.30  | -       |
| PasteGAN (GT layout) [19]      | 10.20±0.20               | -                      | 34.30  | -       |
| ours                           | 9.85±0.15                | 13.82±0.30             | 38.29  | 29.62   |

Table 3. Quantitative comparison of images generated by different methods on Visual Genome dataset.

| Methods                        | IS ↑                     | FID ↓                  |
|--------------------------------|--------------------------|------------------------|
|                                | 64 × 64                  | 128 × 128              |
|                                | 64 × 64                  | 128 × 128              |
| sg2im [10]                     | 5.5±0.10                 | 4.78±0.15              | 73.79  | 70.40   |
| stacking-GANs [36]             | 6.90±0.20                | 9.24±0.41              | 59.53  | 50.19   |
| PasteGAN [19]                  | 7.97±0.30                | -                      | 58.37  | -       |
| PasteGAN (GT layout) [19]      | 9.15±0.20                | -                      | 34.91  | -       |
| ours                           | 8.87±0.15                | 11.20±0.55             | 39.25  | 29.94   |

We remark that the scores of R@0.3 and R@0.5 on Visual Genome are lower than those of COCO-Stuff. This is mainly because each image in Visual Genome contains a larger average number of objects, which increases the difficulty of generating high-quality layouts.

4.4. Ablation study. We perform the ablation comparison to prove the necessity of our global-affine generator. Fig. 13 shows that the ablated version fails to convert the inferred semantic layouts into the corresponding images without using the global-affine generator (w/o $G_{g-a}$). The generated samples expose problems such as disordered locations and unreasonable layouts. In contrast, our full model successfully synthesizes reasonable layouts (see Fig. 13 (d)), the generated images (Fig. 13 (c)) are rendered with realistic attributes in the correct positions.
Table 4. Comparison of classification accuracy.

| Methods          | Classification Accuracy Score |
|------------------|------------------------------|
|                  | COCO-Stuff       | Visual Genome       |
|                  | 64 × 64          | 128 × 128           |
|                  | 64 × 64          | 128 × 128           |
| sg2im [10]       | 28.8            | 24.1                |
| stacking-GANs [36] | 33.9          | 31.2                |
| PasteGAN [19]    | 40.3            | -                   |
| ours             | 46.1            | 44.6                |
|                  | 45.4            | 43.5                |

Table 5. Quantitative comparison of predicted semantic layouts.

| Methods          | R@0.3 COCO-Stuff | Visual Genome |
|------------------|------------------|---------------|
|                  |                  | R@0.5 COCO-Stuff | Visual Genome |
| sg2im [10]       | 52.4             | 21.9          |
|                  | 32.2             | 10.6          |
| stacking-GANs [36] | 65.3          | 35.0          |
|                  | 49.1             | 23.2          |
| PasteGAN [19]    | 71.2             | 45.2          |
|                  | 62.4             | 33.8          |
| ours             | 80.7             | 48.4          |
|                  | 66.2             | 36.5          |

Fig. 13. Ablation study of the global-affine generator.

Fig. 14 illustrates the ablation study of the local-specific generator. After removing the local-specific generator \( G_{l-s} \) from our full model, the lack of the instance-level feature prediction module makes it difficult for the ablated model (w/o \( G_{l-s} \)) to obtain distinguishable visual features of different instance categories and cannot produce reasonable semantic maps, leading to a combination collapse when the weight map predictor integrates the weight feature maps of two-generation branches. As a result, the ablated version (w/o \( G_{l-s} \)) fails to output weight maps. In contrast, our full model has both the global-affine generator and the local-specific generator, which can directly leverage two semantic-level weight maps and complement each
Figure 14. Ablation study of the local-specific generator.

other during the optimization process. Therefore, the weight maps (Fig. 14(c)) inferred by the complete GAL-S-GAN present clear and reasonable instance shapes, the predicted layouts (Fig. 14(e)) also contain fine-grained information, and the generated images (Fig. 14(d)) contain high-quality visual features that are more vivid than the ablation model.

Table 6. Ablation study of GALS-GAN different architectures.

| Architectures   | IS ↑   | FID ↓   |
|-----------------|--------|---------|
| w/o \( G_{g-a} \) | 7.52±0.40 | 78.94   |
| w/o \( G_{l-s} \) | 11.30±0.12 | 46.83   |
| full model      | 13.82±0.30 | **29.62** |

We further validate the necessity of our all generators by comparing the image quality of GAL-S-GAN and two ablated architectures (w/o \( G_{g-a} \) and w/o \( G_{l-s} \)). As shown in Table 6, we can easily conclude that lacking any of the generators will definitely cause image quality degradation. Yet, the absence of the \( G_{g-a} \) causes unreasonable global layouts, thus it is difficult to convert these layouts into recognizable images. As a result, the image quality and model performance are worse than that of the local-specific generator.

5. Conclusion. In this paper, we have proposed a novel GAN-based architecture named GAL-S-GAN for scene-graph-to-image synthesis. We employ the graph convolutional network to predict the corresponding semantic layouts that adhere to the input scene graphs, a feature filter module to highlight the discriminative features
of small-sized instances, and also a weight map predictor to fuse the information of the global and local pathways for producing realistic-looking images. Experimental results on two real-world datasets demonstrate that GALS-GAN outperforms three advanced methods on both qualitative and quantitative evaluations. Our approach specifically learns the semantic layouts and models the instance-level features. However, the final results still heavily depend on the accuracy of inferred semantic maps. Our future work will focus on designing a more powerful architecture that is capable of reconfiguring scene layouts and controlling object-level styles for multi-modal image generation.

REFERENCES

[1] H. Caesar, J. Uijlings and V. Ferrari, COCO-Stuff: Thing and stuff classes in context, IEEE Conference on Computer Vision and Pattern Recognition, (2018), 1209–1218.
[2] W. L. Chen and J. Hays, Sketchygan: Towards diverse and realistic sketch to image synthesis, IEEE Conference on Computer Vision and Pattern Recognition, (2018), 9416–9425.
[3] B. Chen, T. Liu, K. Liu, H. Liu and S. Pei, Image Super-Resolution Using Complex Dense Block on Generative Adversarial Networks, IEEE International Conference on Image Processing, (2019), 2866–2870.
[4] Y. Choi, M. Choi, M. Kim, J. M. Ha, S. H. Kim and J. Choo, Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, IEEE Conference on Computer Vision and Pattern Recognition, (2018), 8789–8797.
[5] Y. Choi, Y. Uh, J. Yoo and J. W. Ha, StarGAN v2: Diverse image synthesis for multiple domains, IEEE Conference on Computer Vision and Pattern Recognition, (2020), 8185–8194.
[6] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari and C. Rupprecht, Semantic image manipulation using scene graphs, IEEE Conference on Computer Vision and Pattern Recognition, (2020), 5212–5221.
[7] C. Gao, Q. Liu, Q. Xu, L. Wang, J. Liu and C. Zou, SketchyCOCO: Image generation from freehand scene sketches, IEEE Conference on Computer Vision and Pattern Recognition, (2020), 5173–5182.
[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems, (2014), 2672–2680.
[9] S. Hong, D. Yang, J. Choi and H. Lee, Inferring semantic layout for hierarchical text-to-image synthesis, IEEE Conference on Computer Vision and Pattern Recognition, (2018), 7986–7994.
[10] J. Johnson, A. Gupta and F. F. Li, Image generation from scene graphs, IEEE Conference on Computer Vision and Pattern Recognition, (2018), 1219–1228.
[11] T. Kaneko, Y. Ushiku and T. Harada, Label-noise robust generative adversarial networks, IEEE Conference on Computer Vision and Pattern Recognition, (2019), 2462–2471.
[12] S. W. Kim, Y. Zhou, J. Philion, A. Torralba and S. Fidler, Learning to Simulate Dynamic Environments With GameGAN, IEEE Conference on Computer Vision and Pattern Recognition, (2020), 1228–1237.
[13] D. Kingma and J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations, 2019.
[14] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, preprint, arXiv:1609.02907.
[15] R. Krishna, et al., Visual genome: Connecting language and vision using crowdsourced dense image annotations, International Journal of Computer Vision, 123 (2017), 32–73.
[16] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar and C. L. Zitnick, Microsoft coco: Common objects in context, European Conference on Computer Vision, 8693 (2014), 740–755.
[17] M. Li, H. Huang, L. Ma, W. Liu, T. Zhang and Y. Jiang, Unsupervised image-to-image translation with stacked cycle-consistent adversarial networks, European Conference on Computer Vision, (2018), 186–201.
[18] W. Li, P. Zhang, L. Zhang, Q. Huang, X. He, S. Lyu and J. Gao, Object-driven text-to-image synthesis via adversarial training, IEEE Conference on Computer Vision and Pattern Recognition, (2019), 12166–12174.
[19] Y. Li, T. Ma, Y. Bai, N. Duan, S. Wei, and X. Wang, Pastegan: A semi-parametric method to generate image from scene graph, Advances in Neural Information Processing Systems, 2019.

[20] B. Li, B. Zhuang, M. Li and J. Gu, Seq-SG2SL: Inferring semantic layout from scene graph through sequence to sequence learning, IEEE International Conference on Computer Vision, (2019), 7434–7442.

[21] S. Liu, T. Wang, D. Bau, J. Y. Zhu and A. Torralba, Diverse Image Generation via Self-Conditioned GANs, IEEE Conference on Computer Vision and Pattern Recognition, (2020), 14274–14283.

[22] S. Liu, Y. Kim and S. J. Kim, Text-adaptive generative adversarial networks: Manipulating images with natural language, Advances in Neural Information Processing Systems, (2018), 42–51.

[23] J. C. Ni, S. S. Zhang, Z. L. Zhou, J. Hou and F. Gao, Instance Mask Embedding and Attribute-Adaptive Generative Adversarial Network for Text-to-Image Synthesis, IEEE Access, 8 (2020), 37697–37711.

[24] T. Park, M. Y. Liu, T. C. Wang and J. Y. Zhu, Semantic image synthesis with spatially-adaptive normalization, IEEE Conference on Computer Vision and Pattern Recognition, (2019), 2332–2341.

[25] T. Qiao, J. Zhang, D. Xu, and D. Tao, MIRrorgan: Learning text-to-image generation by redescription, IEEE Conference on Computer Vision and Pattern Recognition, (2019), 1505–1514.

[26] S. Ravuri and O. Vinyals, Classification accuracy score for conditional generative models, preprint, arXiv:1905.10887.

[27] S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2016), 1137–1149.

[28] N. N. Vo and J. Hays, Localizing and orienting street views using overhead imagery, European Conference on Computer Vision, (2016), 494–509.

[29] L. Z. Zhang, J. C. Wang, Y. S. Xu, J. Min, T. Wen, J. C. Gee and J. B. Shi, Nested Scale-Editing for Conditional Image Synthesis, IEEE Conference on Computer Vision and Pattern Recognition, (2020), 5476–5486.

Received September 2020; revised March 2021.

E-mail address: zhangss9719@163.com
E-mail address: nijch@163.com
E-mail address: hljqfnu@163.com
E-mail address: zljzhou999@163.com
E-mail address: isjieh@outlook.com
E-mail address: isfengg@gmail.com