A. Implementation

A.1. Architecture

**Encoders** $E_O, E_R$ and **Classifiers** $C_O, C_R$ for SGE. Both the object embedding encoder $E_O$ and the relationship embedding encoder $E_R$ are word embedding codebooks. The object classifier $C_O$ has its weight shared with $E_O$. That is, when classifying an output feature $f^O_i$, the classifier will compare it to each object codevector in the codebook. The object class with the highest cosine similarity between the codevector and the feature is then returned as predicted label. The classifier $C_R$ has its weight shared with $E_R$ for the same reason.

**Encoders** $E_B, E_D$ and **Regressors** $R_B, R_D$ for G2L. Both the bounding box encoder $E_B$ and the disparity encoder $E_D$ are two-layer MLPs. The disparity regressor $R_D$ is a two-layer MLP while the bounding box regressor $R_B$ contains two distinct two-layer MLPs, $R_B^1$ and $R_B^2$ respectively. As mentioned in Sect. 3.3.2, one of the MLP $R_B^1$ is trained for predicting the bounding boxes for novel/masked objects and the another one $R_B^2$ is trained for predicting the boundary offset for existing objects.

**Encoder** $E_I$ for G2L & L2I. The architecture of the image encoder $E_I$ we deployed is detailed in Table A. The $E_I$ consists of five 2d convolution layers with batch normalization and leaky relu activation in each layer.

**Scene Graph Transformers for SGE & G2L.** The architecture overview of the SGT layer is shown in Figure A. Noted that the details of our node-level and edge-level attention are depicted in Sect. 3.2.1 and Sect. 3.3.2. All the SGT layer in $T_{SGE}$ and $T_{G2L}$ consists of attention hidden size $d_{atten} = 512$, feed forward size $d_{ff} = 2048$, multi-head number $n_{head} = 4$, and dropout = 0.1. Both $T_{SGE}$ and $T_{G2L}$ have 4 SGT layers.

**Generator** $G_{SPADE}$ for L2I. Our $G_{SPADE}$ is modified from AttSpade [2], with the ability to take input image as guidance for image outpainting work. The generator is comprised of seven SPADE Resnet blocks. Apart from the size of input channels, which is required to be adjusted due to the input image feature, the architectures and the parameters mostly follow their original setting with the same Pix2Pix models as image and object discriminators. Please refer to the official repository https://github.com/roelherz/CanonicalSg2Im for more details of the implementation of AttSpade.

**Baselines.** For the experiments in Table 1 and Table 2, the Transformer, LTNet and GTwE are all set to have the same aforementioned hyper-parameters, which are the attention hidden size $d_{atten} = 512$, feed forward size $d_{ff} = 2048$, multi-head number $n_{head} = 4$, and dropout = 0.1. For the experiments in Table 2, the hidden size of GCN is also set

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2d</td>
<td>c=64, k=5, pad=2</td>
</tr>
<tr>
<td>BatchNorm2d</td>
<td>c=64</td>
</tr>
<tr>
<td>LeackReLU</td>
<td>slope=0.2</td>
</tr>
<tr>
<td>Conv2d</td>
<td>c=128, k=3, stride=2, pad=1</td>
</tr>
<tr>
<td>BatchNorm2d</td>
<td>c=128</td>
</tr>
<tr>
<td>LeackReLU</td>
<td>slope=0.2</td>
</tr>
<tr>
<td>Conv2d</td>
<td>c=128, k=3, stride=2, pad=1</td>
</tr>
<tr>
<td>BatchNorm2d</td>
<td>c=128</td>
</tr>
<tr>
<td>LeackReLU</td>
<td>slope=0.2</td>
</tr>
<tr>
<td>Conv2d</td>
<td>c=256, k=3, stride=2, pad=1</td>
</tr>
<tr>
<td>BatchNorm2d</td>
<td>c=256</td>
</tr>
<tr>
<td>LeackReLU</td>
<td>slope=0.2</td>
</tr>
<tr>
<td>Conv2d</td>
<td>c=256, k=3, pad=1</td>
</tr>
<tr>
<td>BatchNorm2d</td>
<td>c=256</td>
</tr>
<tr>
<td>LeackReLU</td>
<td>slope=0.2</td>
</tr>
</tbody>
</table>
A.2. Datasets

COCO-stuff. The COCO dataset [6] provides images of 80 object categories with bounding box information. COCO-stuff [1] adds 91 more stuff categories (e.g., sky, snow, etc.), with 118K/50K images for training and validation.

VG-MSDN. The original Visual Genome dataset [4] contains more than 100K images with noisy labels, and thus we consider its subset of VG-MSDN [5], containing 150 object categories and 50 relationship categories with about 45K/10K for training/validation.

Cityscapes. The Cityscapes dataset [3] contains 2,975/500 images for training and validation. It provides about 40 object categories with bounding box information for each image.

A.3. Scene Graph Generation

For datasets with scene graph annotation, such as VG-MSDN, the labeled scene graphs are used as ground truth without any pruning. For datasets without scene graph annotation, such as COCO and CityScapes, we follow the processing technique of [2, 8] to generate ground truth scene graphs. That is, we utilize the ground truth objects & bounding boxes to construct rule-based scene graphs. This processing stage does not require extra data/label supervision.

A.4. Training and Inference

All our models are trained with Adam optimizer and the base learning rate $\gamma$ is set to 4e-4. We train the three models ($T_{SGE}$, $T_{G2L}$, and $G_{L2I}$) individually rather than end-to-end, since the errors of upper-stream models can greatly deteriorate the result of down-stream ones during training, i.e. incorrect generated novel object in SGE leads to impossible image reconstruction in L2I.

A.4.1 Scene Graph Expansion

We train our SGE model for a total of 10 epochs. We deploy the learning rate schedule as the following. The initial learning rate $\gamma_0$ is set to a tenth of $\gamma$. In the warmup stage, the learning rate grows linearly up to the maximum learning rate $\gamma$. Then the learning rate will hold for a short period of steps. Finally, the learning rate decay exponentially until the minimum learning rate $2.5 \times 10^{-5} \gamma$ is hit at the same time as the end of training. The warmup stage spans a tenth of our total training steps while the hold stage spans a thousandth. It takes 4-12 hours to train on a single NVIDIA-GTX 1080 depending on the datasets and different hyper-parameters.

For data processing on scene graphs, we also introduced a few dummy tokens besides [MASK] (the token for masked objects and relationships). A dummy object [IMAGE] is added to prevent null input. Each object will have an relationship [IN_IMAGE] between it and the dummy object [IMAGE]. Each object will have an relationship [SELF] between it and itself, i.e. $\forall i \in 1 : N, r_{ii}^{gt} = \text{SELF}$. Noted that when considering the objective function for training, such target labels are not excluded from computation.

We follow the setting in [8] where images with less than 3 objects are excluded while images with more than 8 objects will have the exceeding objects ignored. That is, if there are 10 objects in the given scene graph, we will randomly keep 8 of all. This rule is also applied to the training and inference on G2L, while on L2I the maximum number of objects is set to 30.

The objective function $\mathcal{L}_{SGE}$ has to be further adjusted since the distribution of the relationship labels is highly uneven, which can have a negative impact on training. That is, when computing the objective function $\mathcal{L}_{SGE} = \mathcal{L}_{CE}(\hat{r}_{ij}^{op}, r_{ij}^{gt})$, most of the target $r_{ij}^{gt}$ is no-relation due to the sparsity of the annotation in the datasets. To prevent biased training, weighted cross-entropy loss is deployed instead, and the weight of label no-relation is set to 0.05 while the weights of other labels are still set to 1.0.

A.4.2 Scene Graph to Layout

Each ground truth image $I^{gt}$ is first resized to 256 $\times$ 256 pixels and a sub-image of size 128 $\times$ 128 pixels is cropped.
Table B. Ablation studies of our SGT for scene graph expansion on VG-MSDN. Note that $L_{sym}$ exploits the converse relationship from the input node pair as described in Sect. 3.3.1.

<table>
<thead>
<tr>
<th></th>
<th>Obj</th>
<th>Rel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rAVG</td>
<td>Hit@1 / 5</td>
</tr>
<tr>
<td>node-level</td>
<td>9.32</td>
<td>35.7 / 64.8</td>
</tr>
<tr>
<td>+ edge-level</td>
<td>8.68</td>
<td>38.7 / 68.6</td>
</tr>
<tr>
<td>+ $L_{sym}$</td>
<td>8.96</td>
<td>38.2 / 67.8</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>8.38</strong></td>
<td><strong>39.7 / 68.9</strong></td>
</tr>
</tbody>
</table>

out as the input image $I^m$. The images then are normalized to the range $[-1, 1]$. The associated mask $M^m$ indicating which part of the $I^m$ is input guidance and which part is missing region will also be concatenated onto $I^m$ as the input to $E_I$. Objects that have their bounding boxes completely out of the cropped region will be considered masked and so are the associated relationships.

As for the input bounding boxes $b^m$, those of the masked objects will be set to $(b^r, b^i, b^w, b^h) = (0.5, 0.5, 0.0, 0.0)$. The bounding box of the dummy object [IMAGE] is set to $(0.5, 0.5, 1.0, 1.0)$. The rest of the objects will have their bounding boxes reduced to be in the area of the cropped region. All the input disparities $d^m$ will be calculated after the processing on $b^m$.

We train our G2L model for a total of 50 epochs. The learning rate schedule is the same as the one used for SGE. It takes about 2 days to train on a single NVIDIA-GTX 1080 depending on the datasets and different hyper-parameters.

A.4.3 Layout to Image

The image pre-processing is the same as G2L. We train our L2I model for a total of 30 epochs. No learning rate schedule is used. It takes about 7 days to train on a single NVIDIA-V100.

B. Ablation Studies

Due to the limitation of spaces, we report only a part of the scores for ablation studies in Sect. 4.2. Here we report the complete table with both rAVG and Hit@1/3 as metrics for ablation studies on SGE in Table. B. It is worth noting that, since scene graphs are often sparse, adding only edge-level attention w/o exploiting converse relationships, i.e. $L_{sym}$, tends to overfit the observed relationships thus with degraded results.

Additionally, we also conduct ablation studies for G2L to evaluate the effectiveness of our design. As shown in Table C, the introduction of edge-level attention is also beneficial to the layout prediction. Adding in visual features $f'$ from $E_I$ greatly enhances the performance on layout expansion with existing objects. Overall, with full objective implemented, our model achieve the best performance.

C. Quantitative Result

For image outpainting on VG-MSDN and COCO-stuff, we report the FID scores (the lower the better) for evaluation of the quality of outpainted images. As shown in Table D, our model surpass Boundless [7] by a great margin on VG-MSDN. It is also expected that our model is only slightly better than AttSpade due to the similarity between our L2I architecture and theirs. The main difference lies on the fact that our model is able to generate novel objects which itself does not directly enhance image quality but resulting in richer and more meaningful images. Our advantage is reflected better in Sect. D.4.

D. Visualization

D.1. t-SNE on relationship features

To examine the effect of our proposed feature converter $E_C$ for exploiting converse relationships on VG-MSDN, we project all the relationship features $E_R(y_i^R)$ and converse relationship features $E_R(\tilde{y}_i^R)$ for all $i = 1 : M$ onto a 2D plane with t-SNE, where each $y_i^R$ is a relationship label while $\tilde{y}_i^R$ is its pseudo converse label, and $M$ is the number of relationship labels.

As shown in Figure B(a), the feature of next_to and converse-next_to are close to each other since the converse relationship of next_to is itself. In Figure B(b), the features of below and under and the derived converse relationship features of their antonyms such as converse-on and
Figure B. t-SNE visualization of relationship features \( \mathbf{f_R} \).
(a) With the relationship feature of \textit{next to} highlighted in dark red triangle, its converse version (i.e., \textit{converse-next to}) is its synonym (shown in dark green square). Thus, their \( \mathbf{f_R} \) are close to each other. (b) With \textit{below} and \textit{under} as synonyms (i.e., the two dark red triangles), the derived converse relationship features of their antonyms such as \textit{converse-on}, \textit{converse-above}, \textit{converse-be on}, \textit{converse-stand on}, \textit{converse-on top of} and \textit{converse-lay on} would be nearby as expected. \textit{converse-above} are clustered as expected. Specifically, w/o exploiting the \( E_C \), the cosine similarity scores between “under” and its (1) synonym “below” and (2) antonym “above” are (1) 0.36 and (2) 0.15 respectively. By enforcing the converse property into our SG transformer, the associated scores (1) increase to 0.62 and (2) decrease to -0.93, which is consistent with the improved results reported in ablation studies of Table D. These experiments show that our design does exploit the information carried on converse relationship pairs.

D.2. Three-level image outpainting

In Figure C, we give more examples on how our three-stage semantic image outpainting is achieved, i.e. from the extrapolation on scene graphs, to the extrapolation on layouts, then to the generation of outpainted images.

D.3. Failure cases

We provide a few failure cases in Figure D, which are due to over-annotated data and thus prevent the learning of proper scene graphs.

D.4. Additional Visualization

Additional image outpainting results on VG-MSDN are shown in Figure E, F and G. For example, one can see the first and fourth rows of Fig. F, in which instances of novel/additional animal categories are introduced. In the fifth row of Fig. G, our model is able to synthesize the entire road region with traffic lines which are not presented in the input.

References

Figure C. **Visualization of our semantics-guided image outpainting on VG-MSDN.** From left to right: input image with layout \((I^\text{in}, L^\text{in})\), input scene graph \(S^\text{in}\), output scene graph \(S^{\text{op}}\), output layout \(L^{\text{op}}\) and output image \(I^{\text{op}}\). Note that only selected nodes from the input scene graphs are depicted for visualization purposes. Additionally, we note that when the SGE model is used for image outpainting, unlike the experiments in Figure 5(a) where only one novel object is generated, we do not enforce such constraint. That is, our SGE model is capable of generating multiple objects, e.g. the first example.

Figure D. **Two example failure cases.** From left to right: input image \(I^\text{in}\), our output image \(I^{\text{op}}\) and ground truth image \(I^{\text{gt}}\). Our model tends to generate images with noisy or repeating content (e.g. repeating “legs” or “windows” which are highlighted as blue bounding boxes) if the training images with similar visual concepts are over annotated. For example, training images of “zibra” are often annotated with multiple (more than 4) “legs”, or those of “building” are typically with a large number of “windows” annotated.
Figure E. **Example outpainting results on VG-MSDN.** From left to right: input image, output images produced by Boundless [7], AttSpade [2] & ours, and the ground truth image.
Figure F. Example outpainting results on VG-MSDN. From left to right: input image, output images produced by Boundless [7], AttSpade [2] & ours, and the ground truth image.
Figure G. **Example outpainting results on VG-MSDN.** From left to right: input image, output images produced by Boundless [7], AttSpade [2] & ours, and the ground truth image.