1. Belief Propagation Algorithm

1.1. Algorithm Description

Because belief propagation plays an important role in our method, we briefly explain its operation using the MRF in Figure 1 as an example. Suppose the probability distribution represented by the MRF is expressed by the following equation:

$$\begin{aligned} & -\log P(a_1, a_2, a_3) \\ & = v_1(a_2) + e_1(a_1, a_2) + e_2(a_2, a_3) + Z \end{aligned} \tag{1}$$

Belief propagation is a method of sending messages between the variables and factors of an MRF, as depicted in Figure 1, to obtain the marginal distributions and the normalization term Z. The rules for sending these messages are as follows.

Rule 1 A variable sends a specific factor the sum of all messages received from other factors.

Rule 2 A factor sends a specific variable the result of subtracting its own energy from the sum of the messages received from other variables and reducing the other variables by log-sum-exp.

Rule 3 A variable with only one connection sends 0, and a factor with only one connection sends the negative of its own energy.

Rule 4 The sum of all messages received from all connections, passed through an exponential function, is proportional to the marginal distribution of the variable.

Using these rules, we calculate the marginal distribution of a_3 as follows.

$$m_{a_1 \to e_1}(a_1) = 0 \tag{2}$$

$$m_{e_2 \to a_2}(a_2) = \log \sum_{a_1} \exp(m_{a_1 \to e_1}(a_1) - e_1(a_1, a_2)) \tag{3}$$

$$m_{v_1 \to a_2}(a_2) = -v_1(a_2) \tag{4}$$

$$m_{a_2 \to e_2}(a_2) = m_{e_2 \to a_2}(a_2) + m_{v_1 \to a_2}(a_2) \tag{5}$$

$$m_{e_2 \to a_3}(a_3) = \log \sum_{a_2} \exp(m_{a_2 \to e_2}(a_2) - e_2(a_2, a_3)) \tag{6}$$

Equation (5) is the result of applying Rule 1, Equations (3) and (6) are the results of applying Rule 2, and Equations (2) and (4) are the results of applying Rule 3. According to Rule 4, $\exp(m_{e_2 \to a_3}(a_3))$ is proportional to the marginal distribution of a_3. This can be confirmed by substituting Equations (2) to (5) and Equation (1) into Equation (6).

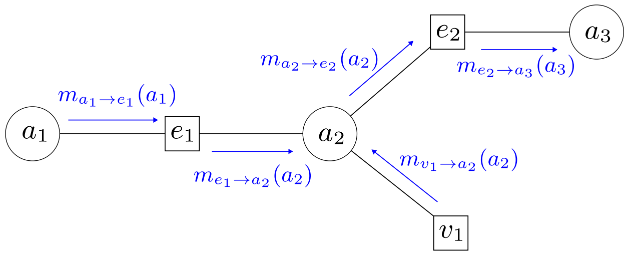


Figure 1. A simple MRF and the belief propagation algorithm. Circles represent variables, squares represent factors such as relationships or object labels, and blue arrows represent the flow of messages.

$$\begin{aligned} & \exp(m_{e_2 \to a_3}(a_3)) \\ & = \sum_{a_1, a_2} \exp(-v_1(a_2) - e_1(a_1, a_2) - e_2(a_2, a_3)) \\ & = \sum_{a_1, a_2} \exp(\log P(a_1, a_2, a_3) + Z) \\ & = \exp(Z) P(a_3) \end{aligned} \tag{7}$$

This is proportional to the marginal distribution of a_3. Considering that $\sum a_3 P(a_3) = 1$, we can calculate Z as follows.

$$Z = \log \sum_{a_3} \exp(m_{e_2 \to a_3}(a_3)) \tag{8}$$

Thus, we compute the marginal distribution of the variables and the normalization term Z for training and inference. This algorithm involves only the log-sum-exp operation and summation, which are differentiable and numerically stable, thereby enabling stable backpropagation.

1.2. Simplified Code

This section presents a simplified module for the belief propagation algorithm in PyTorch and demonstrates its simplicity and auto-differentiability. As a practical implementation note, we use `weakref` to prevent potential memory leaks that can arise from circular references.

```python
# bp.py

from weakref import ref as wref
import torch


class Variable:
  def __init__(self, dim):
    self.dim = dim
    self.adjs = []

  def get_message(self, caller=None):
    x = 0.
    for f in self.adjs:
      if f is not caller:
        x += f.get_message(self)
    return x


class Factor:
  def __init__(self, energy, variables):
    self.energy = energy
    self.adjs = list(map(wref, variables))
    for v in variables:
      v.adjs.append(self)

  def get_message(self, caller):
    x = - self.energy
    for v in self.adjs:
      x.transpose_(0, -1)
      if v() is not caller:
        x += v().get_message(self)
        x = x.logsumexp(dim=-1)
    return x
```

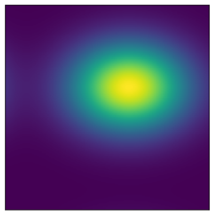
To illustrate the usage of the module, we applied it to the MRF depicted in Fig. 1. The following code demonstrates how to compute the marginal probability of variable a_3 the normalization term Z.

```
# example.py

from bp import Variable, Factor
import torch

# Define variables
a1 = Variable(2)
a2 = Variable(3)
a3 = Variable(4)

# Define energies
e1 = torch.rand(2, 3)
e2 = torch.rand(3, 4)
v1 = torch.rand(3)

# Connect
Factor(e1, [a1, a2])
Factor(e2, [a2, a3])
Factor(v1, [a2])

a3_message = a3.get_message()

Z = a3_message.logsumexp(dim=0)
print(Z)

marginal_prob_a3 = torch.exp(a3_message - Z)
print(marginal_prob_a3)
```

As is evident from the code, the belief propagation algorithm is not only easy to implement, but also compatible with neural network frameworks. We propose that belief propagation offers a viable approach for integrating graph structures into neural networks.

2. Fourier Transform of Our Positional Encoding

Our positional encoding method leverages Fourier transforms, enabling the encoding of positional relationships effectively. By applying an inverse Fourier transform to the encoding, we derive a function centered at (x, y) with a width and height of (σ_x, σ_y), as illustrated in Figure 2, which captures the spatial range. Additionally, the convolution property of Fourier transforms allows parallel movement and range expansion to be represented through simple multiplication operations. Furthermore, since the Fourier transform preserves inner products, the inner product of the encoded results reflects the inner product of the spatially spread functions, as shown in Figure 2. It indicates the degree overlap between ranges.


(a) $(x, y, \sigma_x, \sigma_y) = (0.6, 0.4, 0.2, 0.15)$

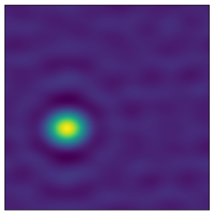

(b) $(x, y, \sigma_x, \sigma_y) = (0.3, 0.6, 0.05, 0.03)$

Figure 2. Fourier transform results of our positional encoding. The y-axis increases from top to bottom. In the case of a small range (Figure 2b), a pattern appears in the background, which is the result of randomly omitting high-frequency components.

3. Object Grounding Results for VG-PO Dataset

Model	Seen		Unseen	
	R@1	R@5	R@1	R@5
Florence-2-L [6]	20.7	-	20.9	-
MDETR [1]	26.2	47.1	26.4	45.7
GLIP-T [3]	30.2	51.5	29.5	48.4
GroundingDINO [4]	32.1	36.9	**31.6**	34.6
VL-MPAG [5] (w/o rel)	33.2	56.6	19.6	43.1
VL-MPAG [5]	39.9	66.9	29.0	**53.6**
SceneProp (Ours, w/o rel)	37.0	60.6	2.4	9.7
SceneProp (Ours)	**45.1**	**67.1**	25.8	46.0

Table 1. The result of VG-PO dataset. "Seen" denotes the performance on the 125 object categories used for training, and "Unseen" denotes the performance on the 25 new object categories introduced during testing.

This section presents our results on the VG-PO (Visual Genome Partially Observed) dataset (Table 1). Derived from Visual Genome [7], VG-PO is specifically designed to evaluate performance on unseen object categories and consists of 93K training and 40K test images. Its key feature is the vocabulary split: 125 object categories are used for training, while the test set introduces an additional 25 new categories.

Although SceneProp is not explicitly designed for open-vocabulary detection, it can infer the location of unseen objects purely through their relational context. This capability becomes more effective as queries become more descriptive. As shown in Figure 3, SceneProp's performance on unseen categories improves with the number of relationships in the query, eventually outperforming VL-MPAG on complex queries with more than two relationships.

Formally supporting open-vocabulary grounding, for instance by integrating a text encoder for category names, re-

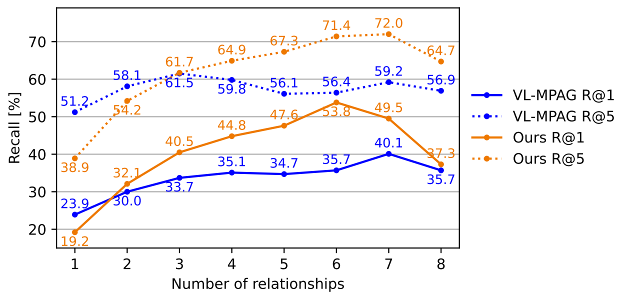


Figure 3. Impact of query graph size (number of relationships) on grounding performance on unseen category in VG-PO dataset. SceneProp's performance improves with more descriptive queries.

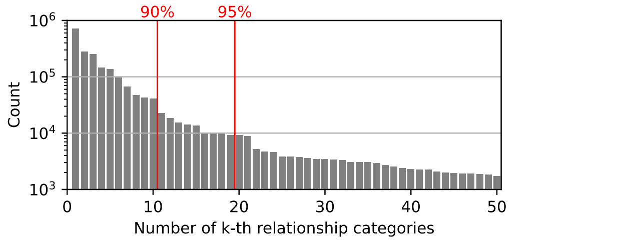
mains a promising direction for future work.

4. Comparison with Scene Graph Generation

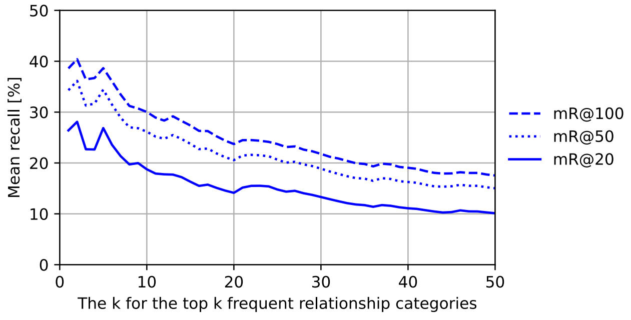
In this section, we compare the tolerance to the long-tail distribution of relationship categories between scene graph grounding and scene graph generation. We used SpeaQ [2], a state-of-the-art scene graph generation model, as a comparison target and evaluated its performance on the VG-150 dataset. We used the trained model provided by SpeaQ. We calculated the mean recall using the top k frequent relationship categories to show how the performance changed as the label distribution becames long-tailed. Note that the scores cannot be directly compared because of the different tasks. However, the comparison provides insights into the tolerance to the long-tail distributions.

Figure 4a shows the long-tailed distribution of relationship categories in the VG-150 dataset. The top 10 and 19 categories accounted for 90% and 95% of the relationships, respectively. Figure 4b and Figure 4c show the mean recall with the top k frequent relationship categories for SpeaQ and SceneProp, respectively. The SpeaQ showed a significant performance drop as the label distribution became long-tailed, whereas SceneProp maintained stable performance. Additionally, the recall (R@20, 50, and 100) for SpeaQ was 25.1, 32.1, and 35.5, respectively, whereas the mean recall (mR@20, 50, and 100) was 10.1, 15.0, and 17.5, respectively. This significant difference between recall and mean recall indicates that SpeaQ performs poorly in the less-frequent relationship categories. However, the differences in SceneProp between recall and mean recall are smaller than those in SpeaQ, as shown in Table 3, demonstrating stable performance even with a long-tailed category distribution.

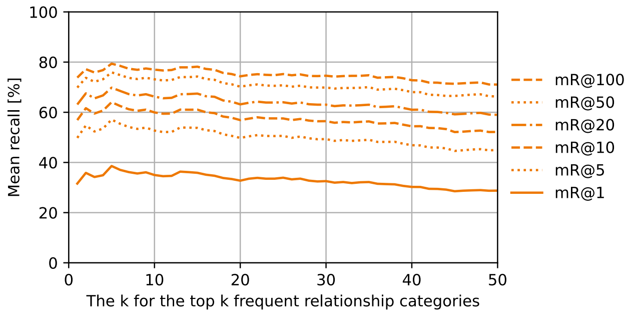
We also evaluated the performance of SceneProp on the GQA dataset. The result is shown in the main paper.



(a) Relationship category distribution in VG-150 dataset.



(b) SpeaQ [2] - Scene Graph Generation



(c) SceneProp - Scene Graph Grounding

Figure 4. Mean recall for the top k frequent relationship categories on the VG-150 dataset. The x-axis shows the value of k. As k increases, the distribution becomes more long-tailed. The right end of the graph represents the mean recall for all relationship categories.

5. Edge-removing Test for each number of relationships

In the main-text experiments, each query graph is paired with a different image and query difficulty varies across graphs, confounding the effect of the number of relationships. To control for this, we evaluate SceneProp on variants of the *same* query graph obtained by randomly deleting edges to reach specified #rels while keeping nodes (and labels) fixed. On VG-FO, Figure 5 shows a consistent drop in performance as edges are removed across all #rels buckets, supporting the claim that additional relations provide informative constraints that SceneProp effectively exploits.

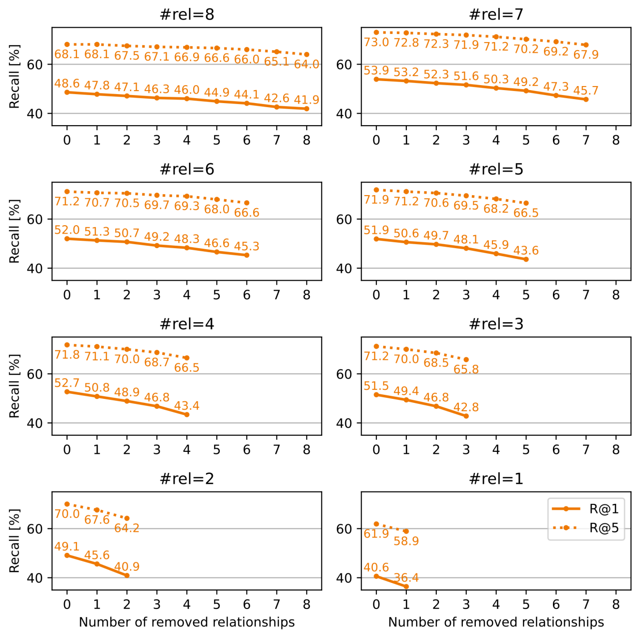


Figure 5. Performance on VG-FO dataset when randomly removing edges from the query graph. #rels denotes the number of relationships (i.e., edges) in the original query graph.

#rels	w/o loop		w/ loop	
	R@1	R@5	R@1	R@5
1	40.6	61.9	-	-
2	48.9	69.6	53.9	76.1
3	50.8	70.7	59.3	79.2
4	51.4	70.8	60.9	79.1
5	50.4	70.7	57.8	77.8
6	50.0	69.3	60.0	78.6
7	51.2	70.9	60.8	78.2
8	46.7	67.0	53.4	72.0
all	45.7	66.3	58.1	77.6

Table 2. Performance on acyclic (w/o loop) versus cyclic (w/ loop) query graphs in the VG-FO dataset. #rels denotes the number of relationships (i.e., edges) in the query graph.

6. Generalization to Cyclic Query Graphs

We report the full numeric breakdown on VG-FO by the number of relationships (#rels) in the query graph; see Table 2 for acyclic (w/o loop) versus cyclic (w/ loop) results. Algorithmic details on handling cycles (loopy belief propagation with a brief MCMC refinement) are given in the main text. Across most #rels and in the overall average, cyclic graphs outperform acyclic ones, consistent with the view that loops provide richer relational constraints (i.e., more relations per object) that SceneProp can leverage; note that loops are undefined at #rels=1, hence the "–" entries.

7. Additional Relationship Pair Grounding Results

This section presents the relationship pair grounding results for the VG-FO, VG-PO, COCOStuff, and GQA datasets, as shown in Table 3. Because of the difficulty in predicting relation pairs, we also provided the top 10, 20, 50, and 100 recalls, in addition to the top 1 and 5 recalls.

	VG-FO	VG-PO	COCO-Stuff	GQA	VG-150
R@1	31.3	14.6	46.8	32.5	35.9
R@5	46.6	27.7	70.0	46.7	54.1
R@10	53.4	35.1	77.3	52.2	61.2
R@20	59.7	42.8	82.8	56.8	67.2
R@50	66.9	53.1	88.2	61.7	73.7
R@100	71.3	60.0	91.5	64.4	77.4
mR@1	27.9	14.9	48.5	39.0	28.8
mR@5	43.1	26.6	71.0	53.2	44.9
mR@10	49.9	32.6	77.6	59.2	52.2
mR@20	56.5	40.2	83.4	65.0	59.0
mR@50	63.8	50.7	88.8	69.8	66.3
mR@100	68.6	56.9	92.1	72.8	71.1

Table 3. Relationship-pair grounding performance on VG-FO, VG-PO, COCOStuff, VG-150, and GQA datasets.

8. Sensitivity analysis of the hyperparameters

This section presents a the sensitivity analysis of the hyperparameters in SceneProp. We evaluated the performance of SceneProp with different hyperparameters using the VG-FO dataset.

First, we analyzed the sensitivity of the number of layers in the Transformer within the relationship feature extractor. Table 4 shows the recall at top-1 and top-5 with different numbers of layers. Performance was not significantly affected by the number of layers; however, two layers showed the best performance.

n layers	0	1	2	3	4
R@1	44.2	44.7	**46.6**	44.6	43.0
R@5	65.1	65.3	**67.1**	65.6	64.6

Table 4. Sensitivity analysis of the number of layers in the Transformer using the VG-FO dataset.

Next, we analyzed the sensitivity of the number of candidate bounding boxes. Table 5 shows the recall at top-1 and top-5 with different numbers of candidate bounding boxes. The number of candidate bounding boxes significantly affected the performance, with 512 candidates showing the best results. The decrease in performance with fewer candidates was owing to the lack of candidate bounding boxes for correct grounding.

n candidates	256	512	786	1024
R@1	29.0	**46.6**	45.6	44.3
R@5	47.9	**67.1**	66.6	66.2

Table 5. Sensitivity analysis of the number of candidate bounding boxes using the VG-FO dataset.

9. Unary vs. pairwise contributions.

In this section, we analyze the contributions of unary and pairwise energies in our model. Table 6 reports performance when using both unary and pairwise terms (full), only unary terms (w/o rels), and only pairwise terms (w/o objs). On VG-FO, VG-150, and GQA, performance is significantly higher with only unary terms than with only pairwise terms. This indicates that unary energies carry the larger share of evidence, consistent with the intuition that relations alone are insufficient for reliable grounding. However, on COCO-Stuff, performance with only pairwise terms is comparable to that with only unary terms. This likely reflects that COCO-Stuff is synthesized from object coordinates, making relative positions more informative than in the other datasets.

10. Additional Visual Examples

Examples of the grounding results for the VG-FO, VG-PO, COCOStuff, and GQA datasets are provided in Fig. 6, Fig. 7, Fig. 8, and Fig. 9. Figure 10 shows examples of failure cases in SceneProp.

References

[1] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr - modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1780–1790, 2021.

[2] Jongha Kim, Jihwan Park, Jinyoung Park, Jinyoung Kim, Sehyung Kim, and Hyunwoo J Kim. Groupwise query specialization and quality-aware multi-assignment for transformer-based visual relationship detection. In *CVPR*, 2024.

[3] Liunian Harold Li*, Pengchuan Zhang*, Haotian Zhang*, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *CVPR*, 2022.

[4] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[5] Aditay Tripathi, Anand Mishra, and Anirban Chakraborty. Grounding scene graphs on natural images via visio-lingual message passing. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4380–4389, 2023.

[6] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4818–4829, 2024.

[7] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

	full		w/o rels (unary only)		w/o objs (pairwise only)	
	R@1	R@5	R@1	R@5	R@1	R@5
VG-FO	46.6	67.1	40.0	64.6	26.0	46.6
VG150	43.7	68.4	37.4	64.1	24.9	43.5
COCO-Stuff	68.2	88.9	48.1	82.8	51.3	81.1
GQA	53.6	68.2	33.4	54.5	21.3	36.1

Table 6. Performance comparison of that using both unary and pairwise terms (full), only unary terms (w/o rels), and only pairwise terms (w/o objs).

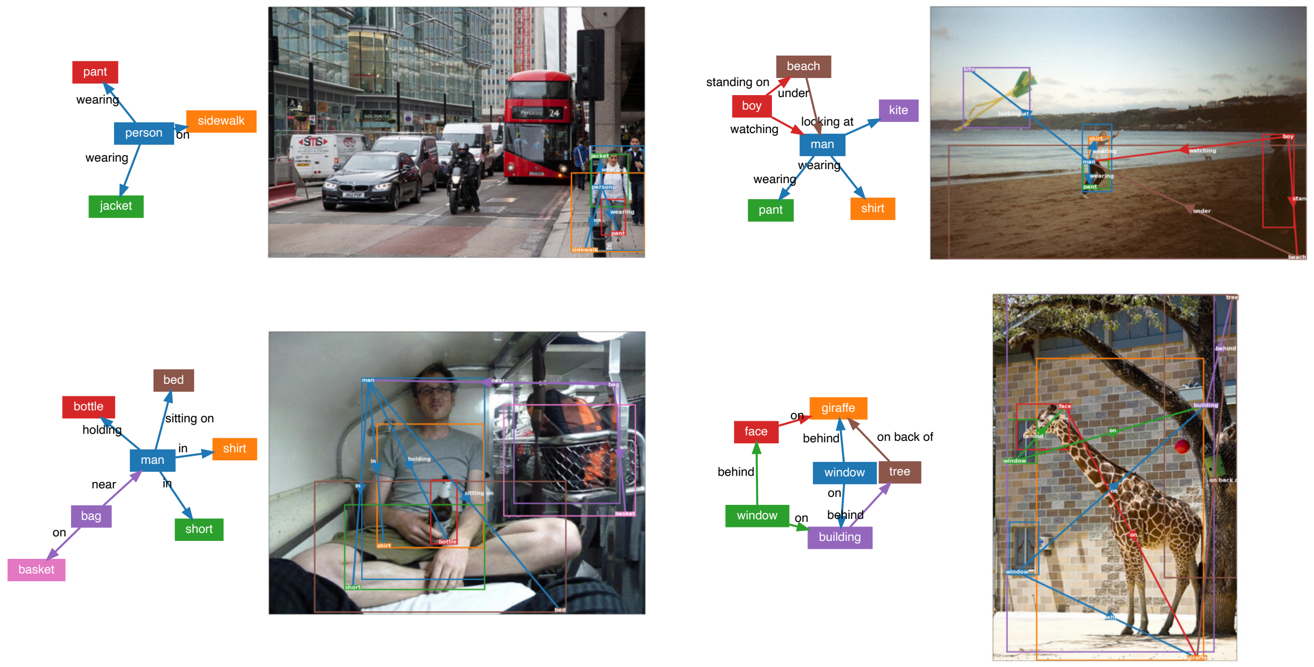


Figure 6. Grounding results on the VG-FO dataset. The right side of the query graph displays the grounding result. The colors of the nodes in the query graph match the colors of the bounding boxes in the grounding image.

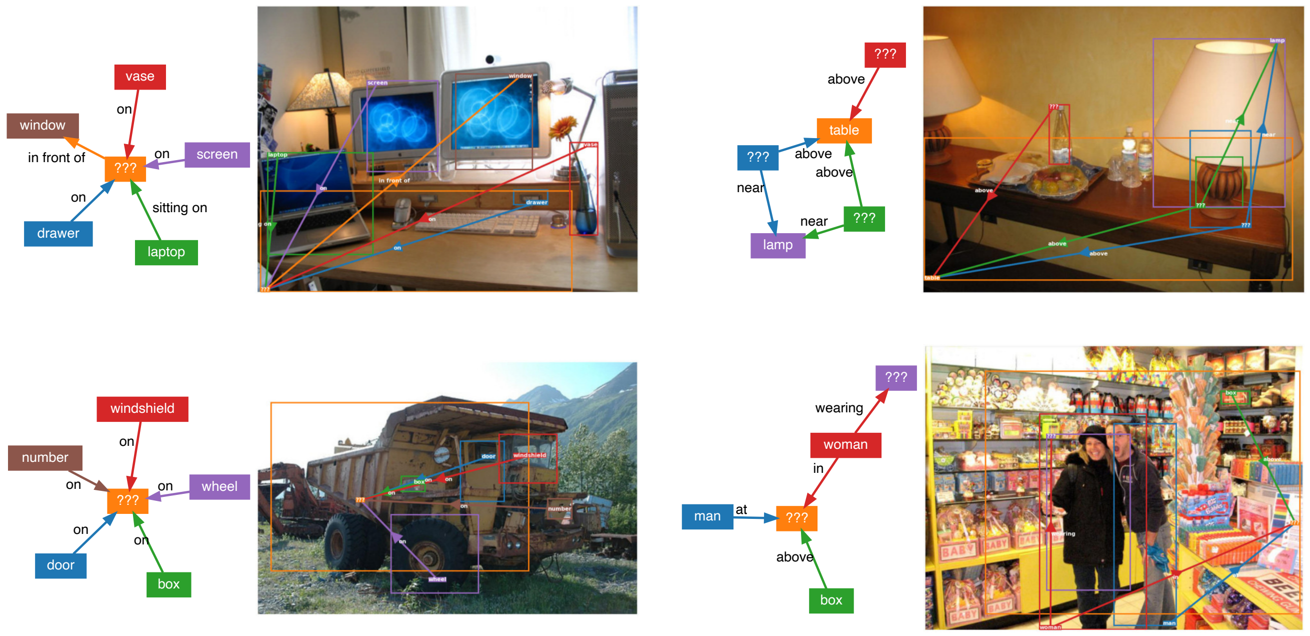


Figure 7. Grounding results on the VG-PO dataset. ”???” denotes object categories absent from the training set.

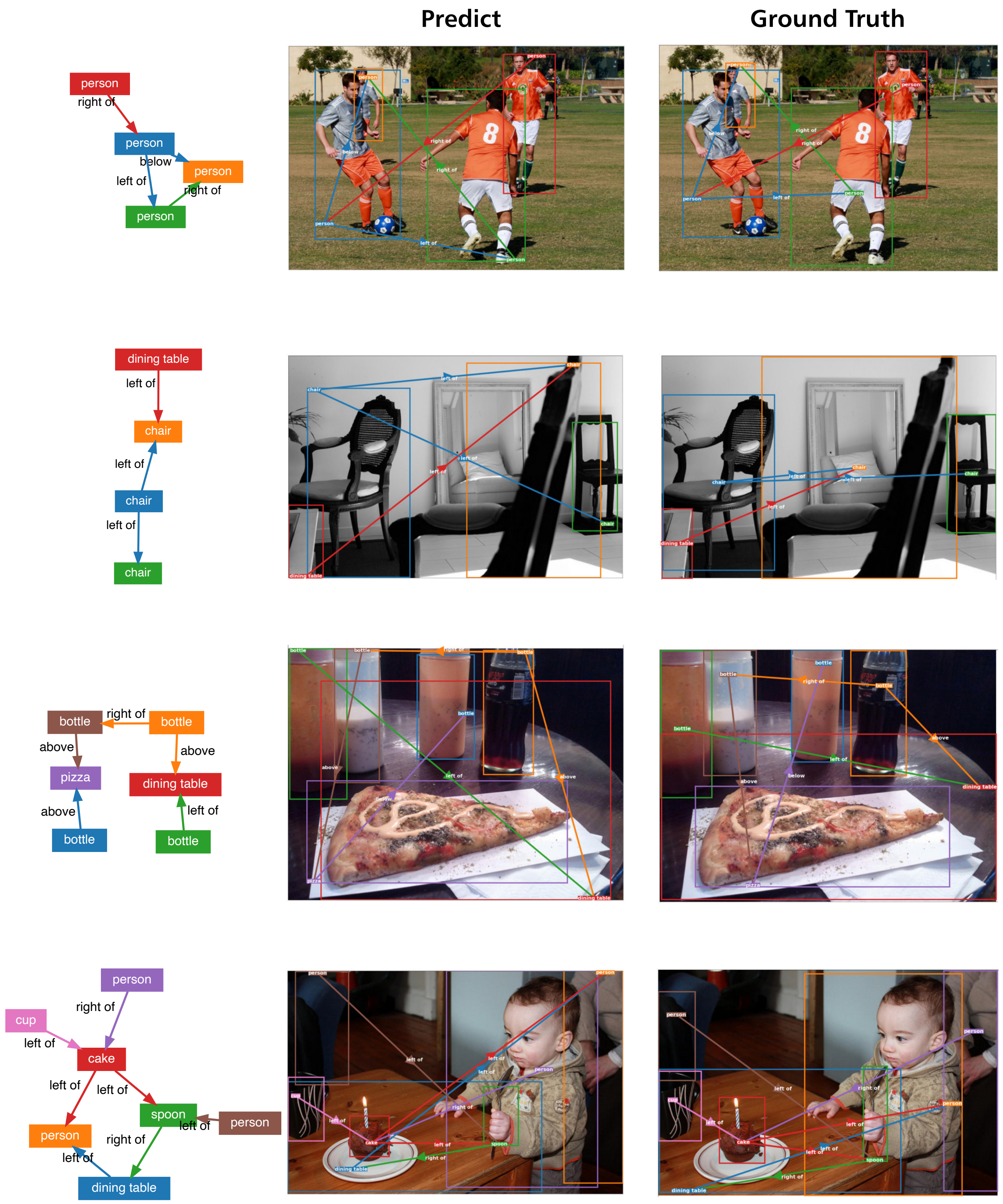


Figure 8. Grounding results on the COCOStuff dataset. Given the non-intuitive nature of scene graphs in the COCOStuff dataset, we include the ground truth images. The images on the right are the ground truth.

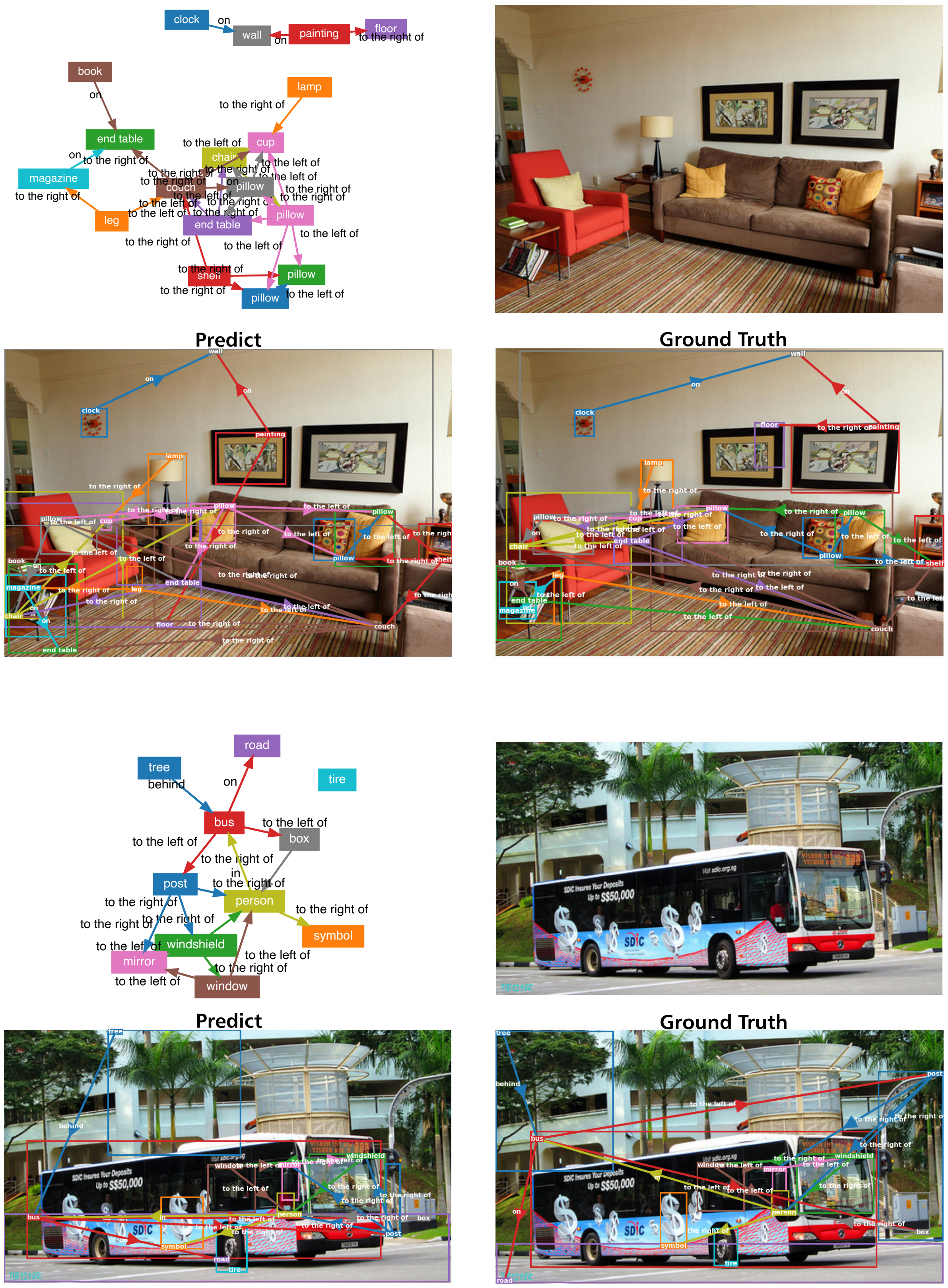


Figure 9. Grounding results on the GQA dataset. To facilitate better visibility of the original images, obscured by numerous bounding boxes, we also present images without bounding boxes. The lower left image represents the prediction result, while the lower right image is the ground truth.

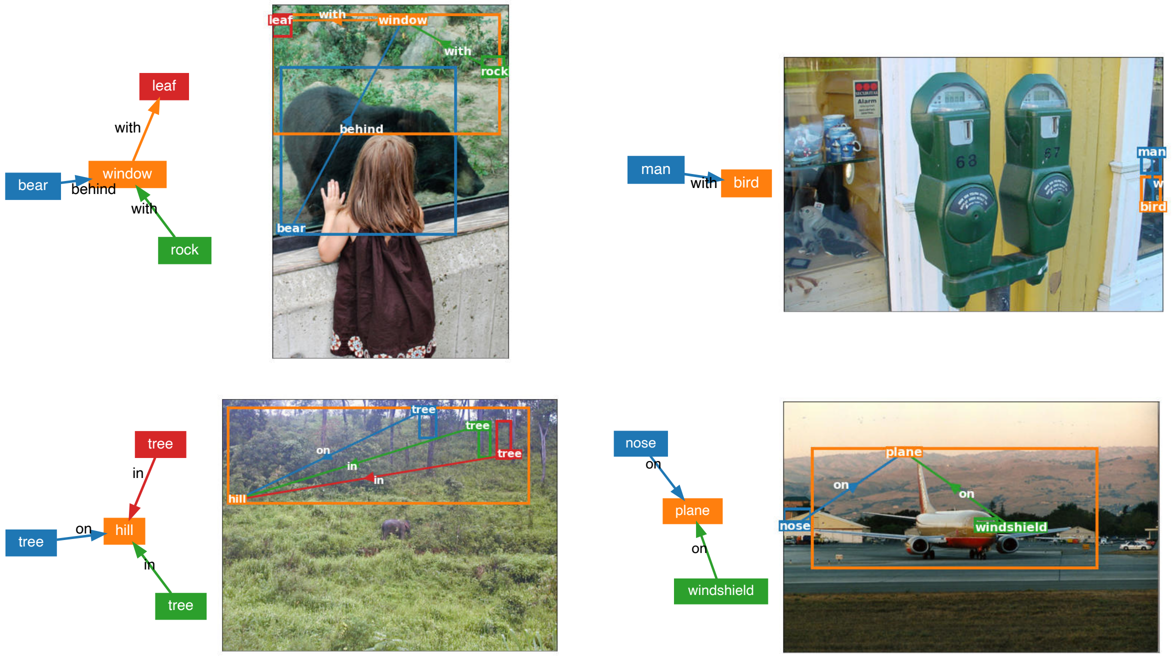


Figure 10. Examples of failure cases in SceneProp. Failures typically involve either (1) inaccurate localization for objects with ambiguous boundaries (left), or (2) incorrect grounding due to challenging visual conditions (right). In the top right example, the query asked for a ”man” and a ”bird” on the upper level of the left-side showcase, but these objects are small and have significant reflections, leading to incorrect grounding. In the bottom right example, the grounding failure appears to stem from the challenge of semantically associating the concept of a ”nose” with a ”plane.”