Supplementary Material: Fully Convolutional Scene Graph Generation

Hengyue Liu[1]* Ning Yan[2] Masood Mortazavi[2] Bir Bhanu[1]
[1]University of California, Riverside [2]Futurewei Technologies Inc.
{hliu087, bir.bhanu}@ucr.edu, {nyan, masood.mortazavi}@futurewei.com

Abstract

This supplementary document is organized as follows: 1) more training details in Section 1; 2) details on network inference in Section 2; 3) detailed architectures in Section 3; 4) qualitative studies in Section 4. All notations are followed from the paper.

1. Training

We defined the center heatmaps in Equation 1 from the paper such that the centers are converted into 2D Gaussian masks. The standard deviations σ_x and σ_y are computed based on the desired radii a^i and b^i along x-axis and y-axis, respectively:

$$(\sigma_x, \sigma_y) = \frac{1}{3}(a^i, b^i) = \frac{1}{3}\lfloor(\sqrt{2}-1)\,\mathbf{s}^i/\tau + 1\rfloor. \quad (1)$$

The values of a^i and b^i are determined by Eq.(1) such that for any point within the ellipse region of radii a^i and b^i, when using it as a center to create a bounding box of object size $\mathbf{s}^i$, the bounding box has at least 0.5 intersection over union (IoU) with the GT bounding box.

Then the prediction of center heatmaps can be supervised by forms of distance losses such as Gaussian focal loss [5, 7, 13] with hyper-parameters $\alpha = 2$ and $\gamma = 4$ for weight balancing. Let $\hat{\mathbf{O}}$ be the predicted center heatmap, then the pixel-wise loss $\mathcal{L}_{\mathbf{O}_{c,x,y}}$ is defined as:

$$\mathcal{L}_{\mathbf{O}_{c,x,y}} = \begin{cases} (1-\hat{\mathbf{O}}_{c,x,y})^{\alpha}\log(\hat{\mathbf{O}}_{c,x,y}) & \text{if } \mathbf{O}_{c,x,y}=1 \\ (\hat{\mathbf{O}}_{c,x,y})^{\alpha}(1-\mathbf{O}_{c,x,y})^{\gamma}\log(1-\hat{\mathbf{O}}_{c,x,y}) & \text{o/w.} \end{cases} \quad (2)$$

2. Inference

In this section, we provide more details on inference and post-processing of our proposed fully convolutional scene graph generation (FCSGG). Our model outputs four dense feature maps: center heatmaps $\hat{\mathbf{O}}$, center offsets $\hat{\mathbf{\Delta}}$, object sizes $\hat{\mathbf{S}}$ and relation affinity fields (RAFs) $\hat{\mathbf{F}}$. To get the object centers, we follow the same step in [13]. Specifically, a in-place sigmoid function is applied to the predicted center heatmaps $\hat{\mathbf{O}}$ such that their values are mapped into the range of $[0, 1]$. Then a 3×3 max pooling is applied to center heatmaps for filtering duplicate detections. For a point $\mathbf{p} = (x, y)$, the value of $\hat{\mathbf{O}}_{c,x,y}$ is considered as the measurement of the center detection score for object class c. Then peaks in center heatmaps are extract for each object class independently. We keep the top 100 peaks by their scores and get a set of object centers $\{\mathbf{o}^i\}_{i=1}^{100}$ with object classes $\{c^i\}_{i=1}^{100}$.

To get the corresponding center offset and object size given a detected object center $\hat{\mathbf{o}}^i = (\hat{x}^i, \hat{y}^i)$, we simply gather the values from $\hat{\mathbf{\Delta}}$ and $\hat{\mathbf{S}}$ at $\hat{\mathbf{o}}^i$. We can get the center offset $\hat{\boldsymbol{\delta}}^i = \hat{\mathbf{\Delta}}_{\hat{x}^i,\hat{y}^i} = (\hat{\delta}^i_x, \hat{\delta}^i_y)$, and the object size $\hat{\mathbf{s}}^i = \hat{\mathbf{S}}_{\hat{x}^i,\hat{y}^i} = (\hat{w}^i_x, \hat{h}^i_y)$. Finally, the bounding box $(\hat{x}^i_0, \hat{y}^i_0, \hat{x}^i_1, \hat{y}^i_1)$ of object $\hat{b}^i$ can be recovered by

$$\begin{aligned} \hat{x}^i_0 &= \left(\hat{x}^i + \hat{\delta}^i_x - \hat{w}^i/2\right)\cdot\tau \\ \hat{y}^i_0 &= \left(\hat{y}^i + \hat{\delta}^i_y - \hat{h}^i/2\right)\cdot\tau \\ \hat{x}^i_1 &= \left(\hat{x}^i + \hat{\delta}^i_x + \hat{w}^i/2\right)\cdot\tau \\ \hat{y}^i_1 &= \left(\hat{y}^i + \hat{\delta}^i_y + \hat{h}^i/2\right)\cdot\tau, \end{aligned} \quad (3)$$

where τ is the stride of the output features.

As for multi-scale prediction, we gather the top 100 detected objects for each scale, then perform a per-class non-maximum suppression (NMS) and keep the top 100 boxes from all the detections, *e.g.*, if there are 5 scales, we will keep the top 100 boxes from the 500 boxes across all the five scales.

2.1. Path Integral

For multi-scale RAFs, we select the valid object pairs from the kept 100 detections $\{\mathbf{o}^i\}_{i=1}^{100}$ following the rule defined in Section 4.2 from the paper. For example, we define

*Work done in part as an intern at Futurewei Technologies Inc.

the predicted 5-scale RAFs as $\{\hat{\mathbf{F}}_k\}_{k=3}^{7}$. If the distance between $\hat{\mathbf{o}}^i$ and $\hat{\mathbf{o}}^j$ is within $[0, 64]$, then the path integral from $\hat{\mathbf{o}}^i$ to $\hat{\mathbf{o}}^j$ will be only performed on $\hat{\mathbf{F}}_3$. We gather the top 100 relationships from each scale, and keep the top 100 relationships across all scales for evaluation.

Mentioned in the paper, the path integral is performed using matrix multiplication in practice. Specifically, we determine the longest integral "length" $m_{\max} = \lceil \text{MAX}(\{m^{i \to j} | \forall i \neq j\}) \rceil$ among the predicted object centers $\{\hat{\mathbf{o}}^i\}$. In other words, there will be $m_{\max}$ sampled points along the integral path for each pair of object centers regardless of their distance.

2.2. Performance Upper Bound

One may concern the relation affinity field representation can actually work and reconstruct relationship successfully by path integral. We analyze the performance upper bound by using the ground-truth of objects and RAFs for evaluation on the test set. It achieves 91.13 R@20 and 86.85 mR@20, which proves that our proposed method is capable of recovering scene graphs from our definition of RAFs. It is worth noting that it is not possible to get 100% recall since there exist multiple edges between nodes in some ground-truth annotations.

3. Detailed Architectures

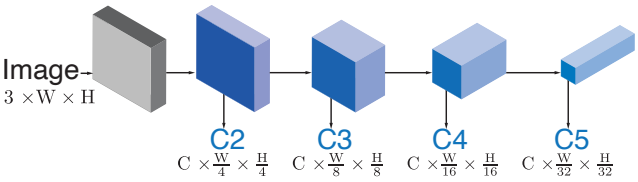
In this section, we provide more details of the proposed fully convolutional scene graph generation model. Our codebase is based on Detectron2 [12] and Tang *et al*. [9]. We list the models mentioned in the paper in Table 1 again for convenience. The number of parameters (#Params) of each network module is also listed in Table 1. As shown in the table, the backbone network has the largest number of parameters while the heads are relatively small.

3.1. Backbone

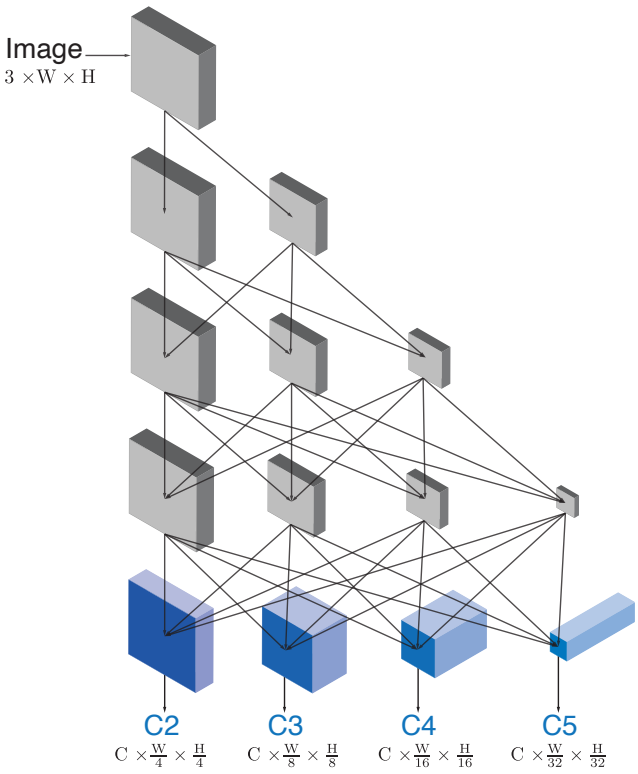
The backbone network serves as a feature extraction module in most of the deep learning applications. We choose the widely used network ResNet [3] and a recent successful alternative named HRNet [11]. ResNet is a representative of deep networks such that the resolution of the feature maps is downsampled while the number of channels is increased, sequentially. ResNet can be divided into stages after the "stem" (first several convolutional layers of the backbone), and the output features of each stage is named as C2, C3, C4, and C5 respectively. On the other hand, HRNet maintains a higher feature resolution all the way to the network output, and constructs several branches of features with lower resolutions. Features from each branch will be fused for exchanging information repeatedly. The output features of each branch are named as C2, C3, C4, and C5 respectively for convenience. The conceptual architecture diagrams of ResNet and HRNet are shown in the figure 1. We do not change the architecture and hyper-parameters of the backbone network with respect to the original papers [3, 11].

3.2. Neck

As presented in the paper, the neck networks serves as a module for constructing multiple scales of features that can be used for later predictions. We use feature pyramid network (FPN) [6] as the neck for ResNet, which is widely used for object detection [2, 7, 10]. FPN allows information exchange across different scales of features after backbone feature extraction. By up-sampling higher level of features (*e.g*. C5) then summing with lower level of features (*e.g*. output features from C4 after a 1 convolution) consecutively, a pyramid of feature maps (with the same num-



(a) An example of ResNet, where C = 256 for ResNet-50 and ResNet-101.



(b) An example of HRNet, where C = 32 for HRNetW32 and C = 48 for HRNetW48.

Figure 1: Conceptual backbone architectures.

FCSGG	Backbone	#Params	Neck	#Params	Object detection heads	Relation detection head	#Params
HRNetW32-1S	Figure 1b, C=32	29.3M	Figure 2c	0.0M	256 - 256 - 256 - 256	512↘ - 512 - 512 - 512 - ↗	18.0M
HRNetW48-1S	Figure 1b, C=48	65.3M	Figure 2c	0.0M	256 - 256 - 256 - 256	512↘ - 512 - 512 - 512 - ↗	20.8M
ResNet50-4S-FPN$_{\times 2}$	Figure 1a, C=256	23.6M	Figure 2a	11.4M	64 - 64 - 64 - 64	64 - 64 - 64 - 64	1.1M
HRNetW48-5S-FPN$_{\times 2}$	Figure 1b, C=48	65.3M	Figure 2b	6.3M	256 - 256 - 256 - 256	512 - 512 - 512 - 512	15.5M

Table 1: FCSGG model zoo. The detailed architectures are described in corresponding figures and tables. The symbol C represents the number of feature channels in C2. Columns 6 and 7 show the number of channels for each convolution in heads. Notation ↘ denotes that the convolution is of stride 2, and ↗ is bilinear interpolation for upsampling the features to the target stride ($\frac{1}{4}$ of the input image size for HRNetW32-1S and HRNetW48-1S).

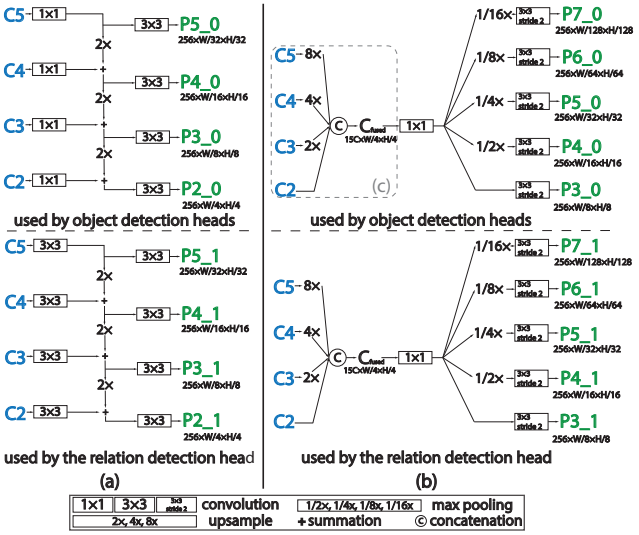


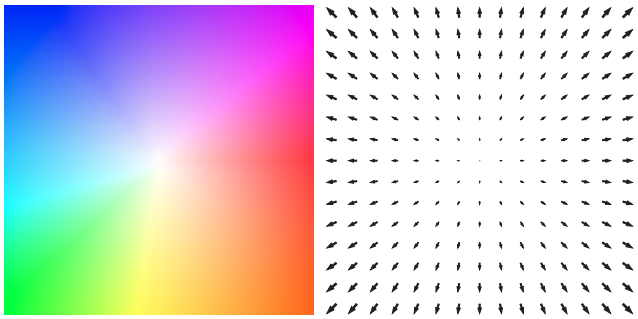
Figure 2: Conceptual neck architectures. The left sub-figure (a) represents the FPN, and the right sub-figure (b) represents HRNetV2p with five scales of features. The dashed block (c) represents the feature fusion module for the single-scale HRNet.

ber of channels) is built and called {P2, P3, P4, P5}. For ResNet50-4S-FPN$_{\times 2}$, we use a modified version of FPN called bidirectional FPN (BiFPN [8]) which allows more connections among each scale. A detailed illustration of FPN as a neck is shown in Figure 2a.

As for HRNet as backbone, we follow Wang *et al*. [11] and use the HRNetV2 for single-scale prediction, and HRNetV2p network for multi-scale feature representations. It should be addressed that even for single-scale models like HRNetW32-1S and HRNetW48-1S, a neck is applied for merging features from all branches. In this case, the neck is simply a feature fusion module without any trainable parameters. Features of C2, C3, C4, and C5 will be upsampled to the resolution of C2 via bilinear interpolation then concatenated. We name the fused features as C_{fused}. Different from HRNetV2 [11], we did not use 1×1 convolution after fusion, and the resulting features will be fed into the heads. As for HRNetV2p, the first step is the same as HRNetV2, then a max pooling and a 3×3 convolution are applied on C_{fused} with different strides to construct multiple scales of feature maps. The designs of necks for HRNet are shown in Figure 2b.

3.3. Heads

There are in total of four heads, and each head is responsible for the task of predicting center heatmaps, center offsets, object size and relation affinity fields respectively. We name the first three heads as object detection heads, and the last one as the relation detection head. As stated in the paper, all heads are small network with four convolutional blocks, each of which consists of a 3×3 convolution, a normalization layer of choice such as group normalization (GN), batch normalization (BN) or multi-scale batch normalization (MS-BN), and a ReLU activation layer. Then, for each head, there is a 1×1 convolution as the output layer, and the number of channels is $\mathcal{C}$, 2, 2 and $\mathcal{P}$ respectively. The number of channels are the same among object detection heads except the output layer, while we increase the number of channels for relation detection head in some models. We list the number of channels in each block for object detection heads and relation detection head as shown in Table 1 (the output convolution is omitted).



(a) Field color coding. (b) The corresponding vectors.

Figure 3: The color coding for RAFs.

4. Qualitative Results

We adopt the visualization method for optical flow [1] on visualizing relation affinity fields. Shown in Figure 3, the vector orientation is represented by color hue while the vector length is encoded by color saturation. We only visualize the bounding boxes with predicted scores over 0.2, and relationships with scores over 0.1.

The visualizations of scene graph detection (SGDet) results for several test images from Visual Genome dataset [4] are shown in Figure 5. From the figure, FCSGG has strong object detection performance, especially on the localization of bounding boxes. In Figure 5a, `pole` can be recognized even though it is not annotated in the ground-truth. In Figure 5f, FCSGG detects more and accurate objects compared to the ground-truth.

However, there are two challenges for training on Visual Genome dataset: object class ambiguity and predicate ambiguity. For the first challenge, as an example of Figure 5a, Jacket is misclassified as `coat` which is reasonable since the semantic difference between the two is subtle. Meanwhile, it also detects the person instance with even better bounding box than the ground-truth. However, it misclassified `man` as `woman`. As a result, all the relationships associated with `man` will be false detections. Similarly in Figure 5e, `woman` is misclassified as `lady`. We argue that these person-centric relationships take a large proportion in the Visual Genome dataset, and it is difficult to visually distinguish among person entities of similar semantics such as `woman` / `lady`, `boy` / `kid`, `man` / `men`, and `person` / `people`. Even though, FCSGG achieves superior object detection performance on Visual Genome dataset.

The other challenge is the predicate ambiguity. Even though VG-150 only keeps the top 50 frequent predicates, there are still predicates with similar semantics (*e.g.* `OF` / `PART OF`, `WEARING` / `WEARS`, `LAYING ON` / `LYING ON`), or with vague and trivial meanings (*e.g.* `OF`, `TO`, `NEAR`, `WITH`). FCSGG is still able to capture similar semantics with similar responses. For example, we see similar RAFs predictions between `WEARING` and `WEARS` (Figure 5a and 5d). Since we do not add any predicate-specific hyper-parameters or statistic bias during training, there will always be some loss contributed by the predicate ambiguities that causes the training even harder. However, FCSGG achieves strong generalization on relationship prediction. In Figure 5c, it predicts <`man`, `RIDING` ,`motorcycle`>, <`person`, `SITTTING ON`, `motorcycle`>, and `motorcycle`, `UNDER`, `person`> concurrently though neither of these are annotated in the dataset.

It should be addressed that multiple predicates could be all valid for a pair of objects, *e.g.* both <`person`, `ON`, `street`> and <`person`, `STANDING ON`, `street`> can represent the correct relationship. Our proposed RAFs are suitable for multi-class problem so that our no-graph constraint results are much improved. More importantly, <`street`, `UNDER`, `person`> is actually true even though we rarely describe this way due to language bias. Interestingly, FCSGG generalizes the relationships and learns the reciprocal correlations between predicates. From the visualizations in Figure 5a and 5d, `ABOVE` and `UNDER` will have responses with similar vector magnitudes but opposite directions. We can also see the similar pattern between `OF` and `HAS`.

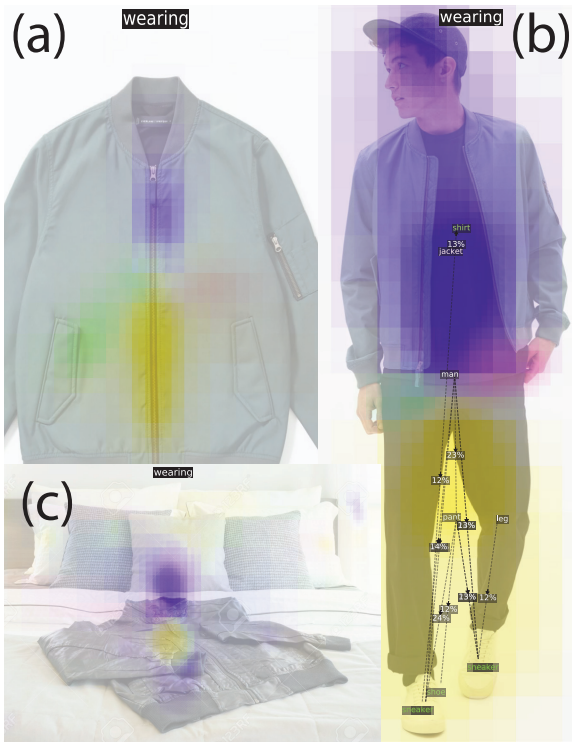


Figure 4: The RAF visualizations of `WEARING` for wild images. Among the images, (b) has the largest RAF vector norm before normalization.

We then tested on wild images for examining the applicability of our approach. For testing predicate `WEARING`, we downloaded 3 online images containing `jacket` shown in Figure 4 but in different scenarios: a product photo of `jacket` (4a); `man WEARING jacket` (4b); `jacket ON bed` (4c). The RAF predictions for `WEARING` are visualized, and the maximum unnormalized RAF vector norm is 0.79, 4.02, and 0.67 for 4a, 4b, and 4c, respectively. The predicate `WEARING` gets the strongest response in Figure 4b among the three images, demonstrating that RAFs successfully capture the semantics. It also exposes the problem of learning bias, such that even no person presents in 4a and 4c, there are responses of `WEARING` since `jacket` and `WEARING` often coexist.

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011. 4

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[4] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017. 4

[5] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 1

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2

[8] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 3

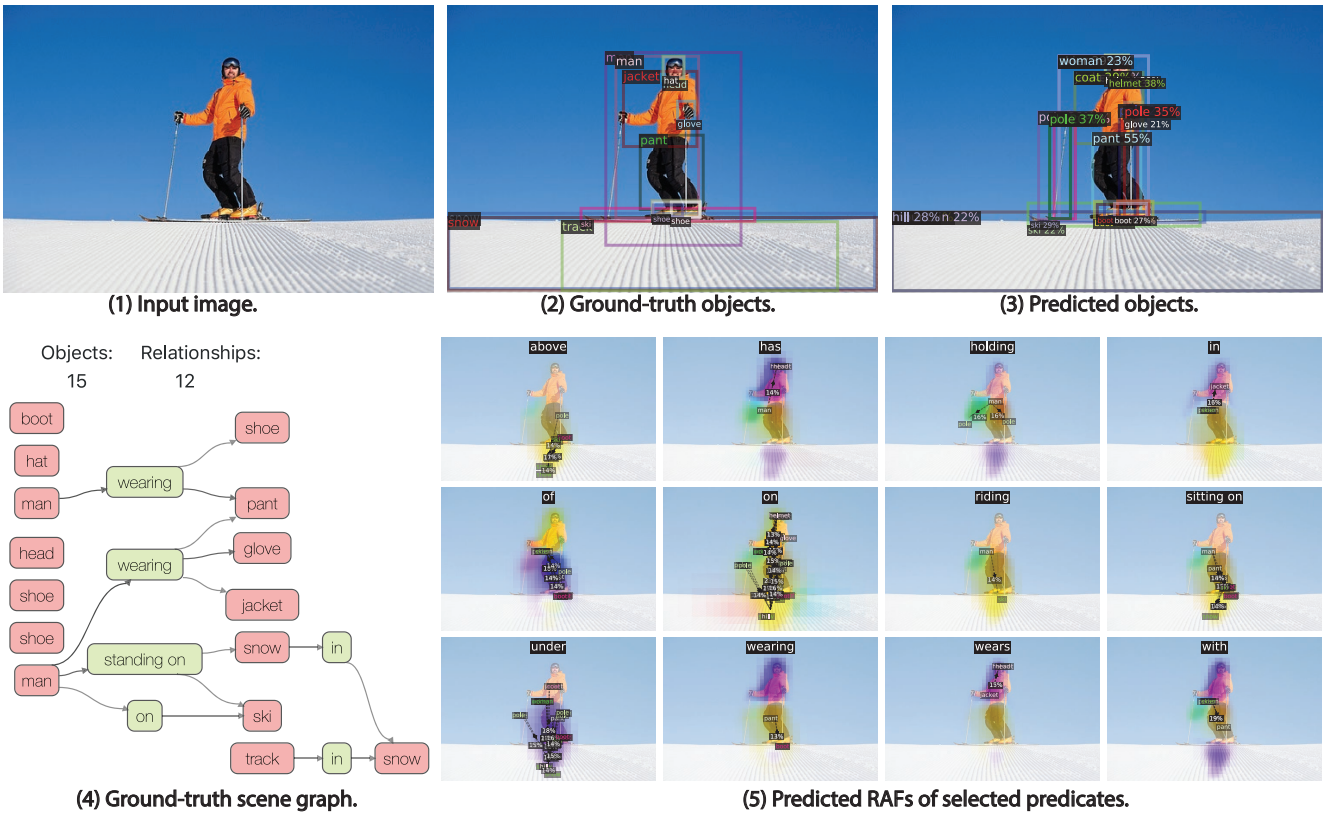
[9] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3716–3725, 2020. 2

[10] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 9627–9636, 2019. 2

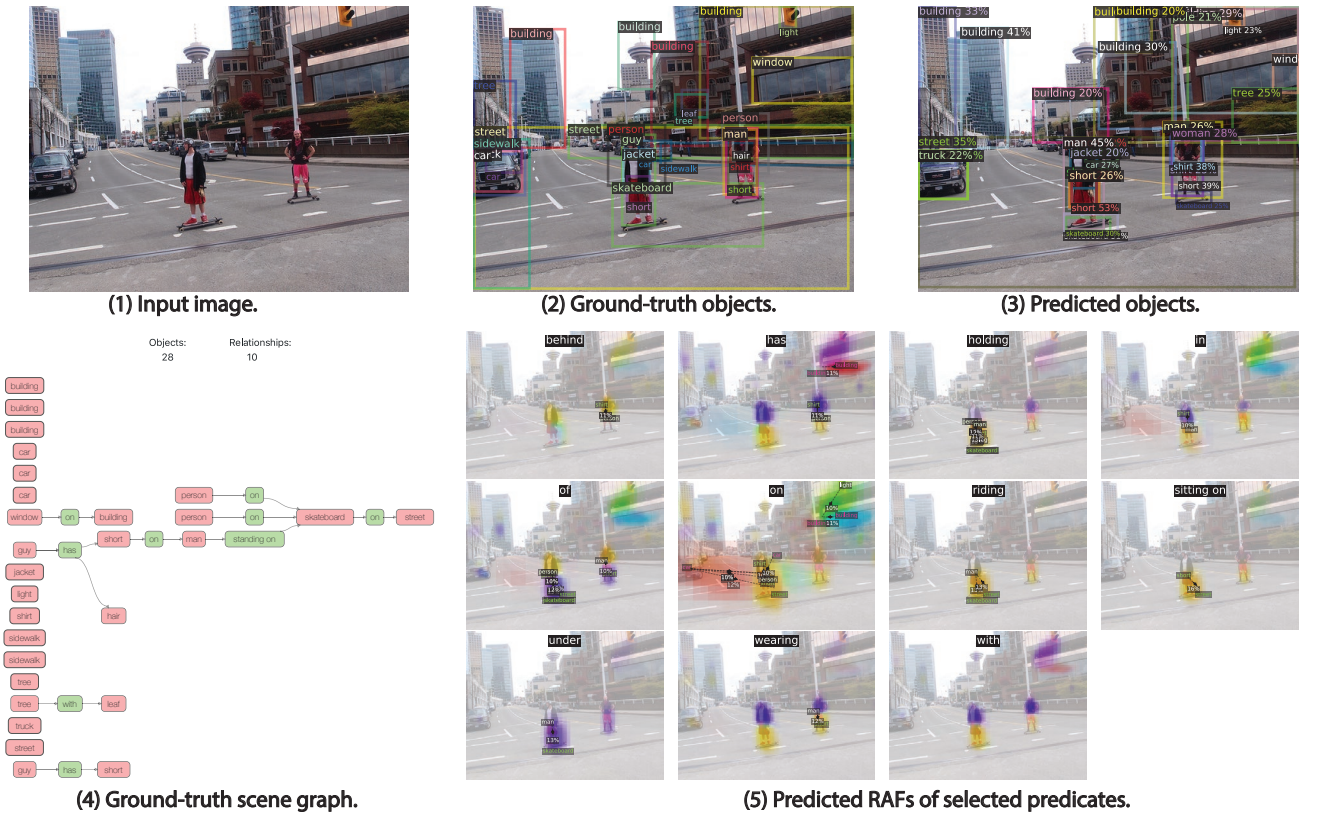
[11] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2, 3

[12] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 2

[13] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1

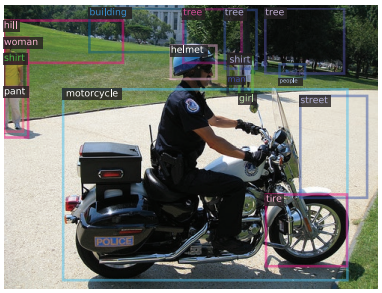


(a) Visualizations of test image `2343157.jpg`.

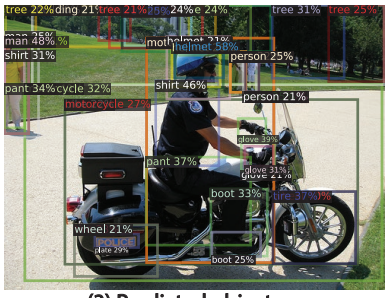


(b) Visualizations of test image `2343530.jpg`.

(1) Input image.

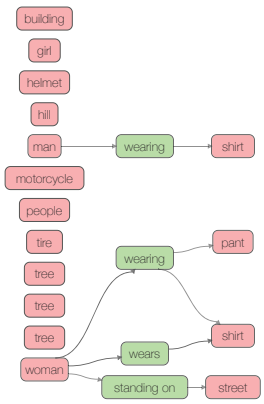


(2) Ground-truth objects.

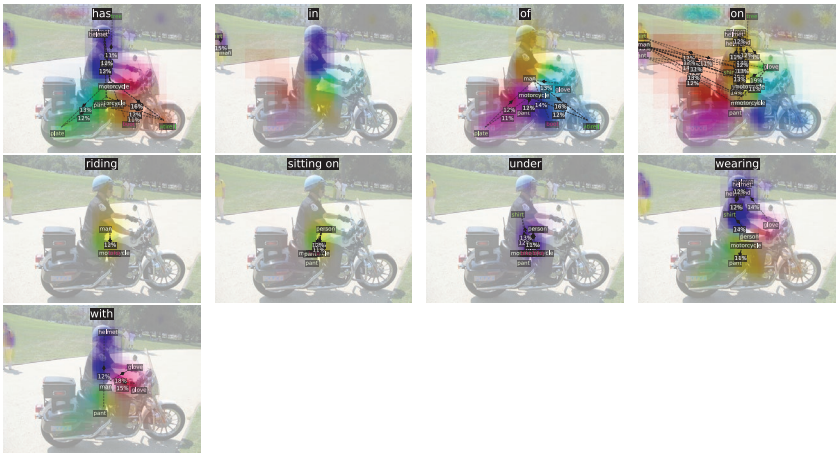


(3) Predicted objects.

Objects: 16 Relationships: 6



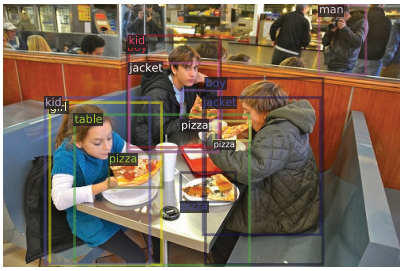
(4) Ground-truth scene graph.



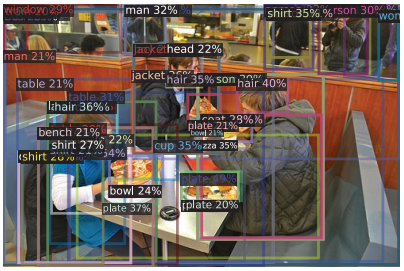
(5) Predicted RAFs of selected predicates.

(c) Visualizations of test image `2343342.jpg`.

(1) Input image.

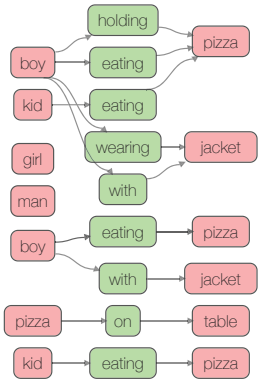


(2) Ground-truth objects.



(3) Predicted objects.

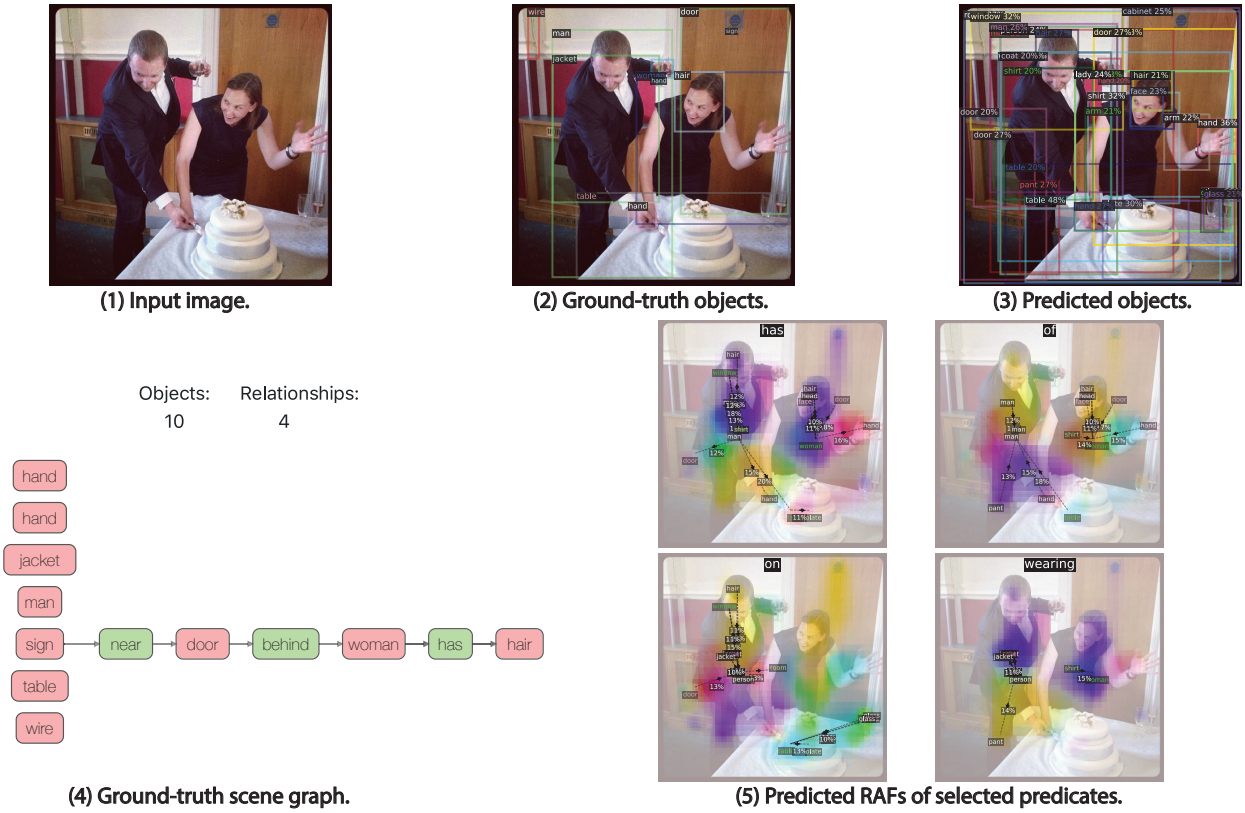
Objects: 13 Relationships: 10



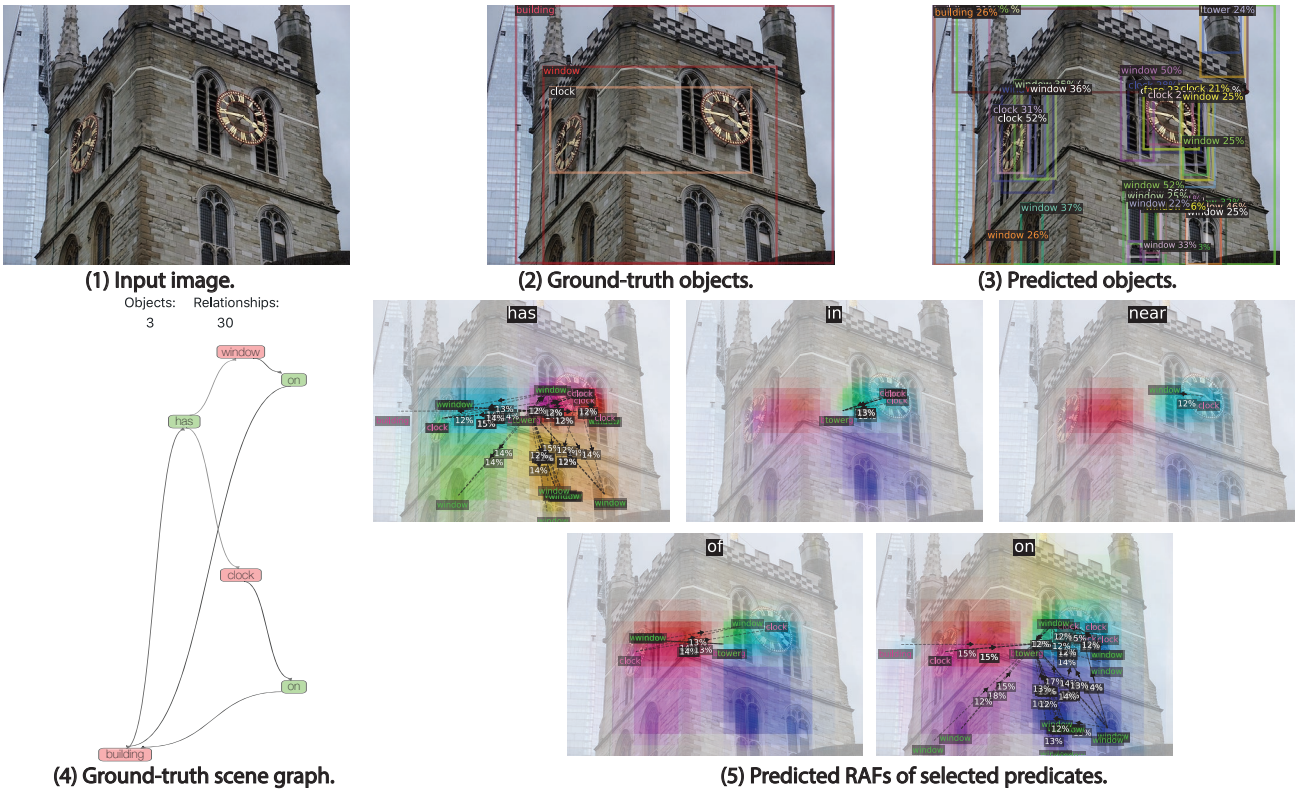
(4) Ground-truth scene graph.

(5) Predicted RAFs of selected predicates.

(d) Visualizations of test image `2342735.jpg`.



(e) Visualizations of test image `2342618.jpg`.



(f) Visualizations of test image `2343159.jpg` (there are duplicate relationships in the annotation).

Figure 5: Visualizations of results on VG-150 test set.