GEMS: Scene Expansion using Generative Models of Graphs - Supplementary

Rishi Agarwal[1*] Tirupati Saketh Chandra[2*] Vaidehi Patil[3*]
Aniruddha Mahapatra[4*] Kuldeep Kulkarni[5] Vishwa Vinay[5]
Stanford University, USA[1] IIT Bombay, India[2] UNC Chapel Hill, USA[3]
Carnegie Mellon University, USA[4] Adobe Research, India[5]
rishia@stanford.edu tsaketh@iitb.ac.in vaidehi@cs.unc.edu
amahapat@andrew.cmu.edu {kulkulka, vvinay}@adobe.com

1. Training and Testing Algorithms

In this section, we present the pseudo code for training 1 and inference 2. We use the notation as described in our main paper under problem description. $f_{node2edge}$ is an additional neural network (multi-layer perceptron) that is used to transfer information from f_{node} to f_{edge}, i.e. the hidden state of f_{edge} is initialized from the hidden state of f_{node} using this transformation.

Algorithm 1 GEMS Training Algorithm

Input: Dataset of Graphs $\mathbb{G} = \{G_1, G_2, \ldots, G_n\}$
Output: Learned functions $f_{node}, f_{edge}, f_{node2edge}$
  Initialize $f_{node}, f_{edge}, f_{node2edge}$
  for epoch in epochs **do**
    for $G \in \mathbb{G}$ **do**
      $S \leftarrow S^{\pi}(G)$
      $S_0, h^0_{node} \leftarrow SOS, h_{init}$
      for $i \in 1 \ldots n_G + 1$ **do** // $n_G + 1$ for EOS token
        $p_{\hat{v}_i}, h^i_{node} \leftarrow f_{node}(S_{i-1}, h^{i-1}_{node})$
        $q_i \leftarrow \min_q KL(q, p_{\hat{v}_i}) - \mathbb{E}_{v \sim q}(f(v, v_i))$
        $start \leftarrow max(1, i - k)$ // k is max_prev_node
        $h^{(start-1)}_{edge} \leftarrow f_{node2edge}(h^i_{node})$ Initialize hidden state of f_{edge}
        $L^i_{edge} \leftarrow 0$
        for $j \in start \ldots i-1$ **do**
          $p_{\hat{e}_{i,j}}, h^{temp}_{edge} \leftarrow f_{edge}(v_i, v_j, h^{j-1}_{edge})$
          $p_{\hat{e}_{j,i}}, h^{j}_{edge} \leftarrow f_{edge}(v_j, v_i, h^{temp}_{edge})$
          $L^i_{edge} \leftarrow L^i_{edge} + l_{edge}(p_{e_{i,j}}, p_{\hat{e}_{i,j}})$
          $L^i_{edge} \leftarrow L^i_{edge} + l_{edge}(p_{e_{j,i}}, p_{\hat{e}_{j,i}})$
        end for
        $L^i_{node} \leftarrow l_{node}(p_{v_i}, p_{\hat{v}_i})$
      end for
      $\mathcal{L}(G) \leftarrow L^{n_G+1}_{node} + \sum_{i \in 1 \ldots n_G} L^i_{node} + L^i_{edge}$
    end for
    Back-propagate loss and update weights of $f_{node}, f_{edge}, f_{node2edge}$
  end for // typically when validation loss is minimized

2. Selecting value of max_prev_node

Our method GEMS (and the baseline adapted from GraphRNN) requires the EdgeRNN to predict edges between the newly generated node v_i and the previous nodes in the graph sequence. The number k, referred to as max_prev_node, controls how many of the previously chosen nodes to take into account for predictions, and is a hyperparameter of the underlying method. If the value of k is set to a very large number, the model generates edges much larger in number than that is expected in training distribution (see count predicted and count reference rows in Table 1 in the main paper). If k is calculated to using the method described by GraphRNN [10], we obtain $k = 37$ for the Visual Genome dataset. This leads to generated graphs containing large number of edges (29.53) compared to reference number of edges in training set (11.09), and most of the edges are either degenerate or describe irrelevant relationships between the subject and object. To address this problem we compute the value of k as the maximum degree (out-degree + in-degree) of a node in a graph such that 99% of nodes in all graphs in the training set have degree less than or equal to k. By setting max_prev_node to k chosen in this manner, we lose only a few distant relationships between nodes in a graph. However, the use of BFS on maximal connected subgraphs in our Cluster-Aware BFS algorithm ensures that the objects that are closely related in the observed set of graphs appear together in sequence, further reducing the chances of not retaining edges between nodes which are related. As observed from Table 1 in main paper GraphRNN*, which has $max\_prev\_node = 6$ for Visual Genome, replicates the count of predicted edges in the generated graphs much more faithfully that GraphRNN. Hence, we gain efficiency and improve performance at the cost of a negligible number of relationships. From the plots of the

*The first four authors contributed equally to this work and was done while authors were at Adobe Research.

cumulative degree distribution in Figure 1, we observe that $k = 6$ for Visual Genome $k = 7$ for VRD.

3. GraphRNN [10] for Scene Graph Expansion (Baseline)

We use GraphRNN [10] as the baseline to compare against our method, GEMS, for the task of scene graph expansion. GraphRNN is an auto-regressive model that converts a graph to a sequence, and then successively predicts nodes and edges according to the sequence. The sequencing method used in their work is BFS starting from a randomly chosen node in the graph. This sequencing method was possible because the graphs which they work on are connected graphs, for which there is a valid BFS tracersal. However, scene graphs used in our task (both from Visual Genome [5] and VRD [7]) tend to have disconnected components. Additionally, the scene graphs that we work with, contain bi-directional edges between nodes (the datasets that GraphRNN was evaluated on contain only undirected or unidirectional edges). To use their sequencing method we convert the disconnected scene graphs to connected scene graphs. More specifically, we add a dummy node labelled $image$ to each of the graphs, and connect this $image$ node with all other nodes (v_i) in the graphs using bidirectional dummy edges in_image (from v_i to $image$) and $contains$ (from $image$ to v_i) and then perform BFS starting from any node in the graph. At inference time we generate the graph using these dummy nodes and edges, and then remove the dummy nodes and edges for evaluation. We modify the original GraphRNN code to predict bidirectional edges sequentially, i.e., for a newly generated node v_i and a previously generated node v_j in the graph, the EdgeRNN of GraphRNN first predicts the edge from v_i to v_j and then the edge from v_j to v_i.

4. SceneGraphGen for Scene Graph Expansion (Baseline)

We compare our method GEMS to SceneGraphGen [2], which was primarily designed for unconditional generation of scene graphs, for the task of scene graph expansion. Since SceneGraphGen also predicts nodes and the corresponding edges from previously generated nodes in an auto-regressive manner, they require an ordering method for converting graph to a sequence. The authors suggest a random ordering to flatten the graph into a sequence, and show that this performs best compared to other ordering methods – we therefore use the same ordering strategy while evaluating SceneGraphGen. For both Visual Genome and VRD we use $num\_node\_categories = 153$, $num\_edge\_categories = 55$ and $num\_node\_categories = 102$, $num\_edge\_categories = 73$ respectively, based on the dataset statistics. The other parameters for the models were retained as provided in the official github repository of SceneGraphGen https://scenegraphgen.github.io/. (Code is currently not available).

Algorithm 2 GEMS Inference Algorithm

Input: Seed graph G_s; $f_{node}, f_{edge}, f_{node2edge}$
Output: Expanded graph $\hat{G}_s$

$S \leftarrow S^{\pi}(G_s)$; $\hat{S}_0, h^0_{node} \leftarrow SOS, h_{init}$
for $i \in 1 \ldots n_{G_s}$ **do** // Conditioning on the input
  $p_{\hat{v}_i}, h^i_{node} \leftarrow f_{node}(\hat{S}_{i-1}, h^{i-1}_{node})$
  $\hat{S}_i \leftarrow S_i$
end for
repeat // Expanding the seed graph
  $i \leftarrow i + 1$
  $p_{\hat{v}_i}, h^i_{node} \leftarrow f_{node}(\hat{S}_{i-1}, h^{i-1}_{node})$
  $\hat{v}_i \sim p_{\hat{v}_i}$
  $start \leftarrow max(1, i - k)$ // k is max_prev_node
  $h^{(start-1)}_{edge} \leftarrow f_{node2edge}(h^i_{node})$
  for $j \in start \ldots i - 1$ **do**
    $p_{\hat{e}_{i,j}}, h^{temp}_{edge} \leftarrow f_{edge}(\hat{v}_i, \hat{v}_j, h^{j-1}_{edge})$
    $p_{\hat{e}_{j,i}}, h^j_{edge} \leftarrow f_{edge}(\hat{v}_j, \hat{v}_i, h^{temp}_{edge})$
    $\hat{e}_{i,j} \sim p_{\hat{e}_{i,j}}, \hat{e}_{j,i} \sim p_{\hat{e}_{j,i}}$
    $\hat{\mathcal{E}}_{i,j} \leftarrow (\hat{e}_{i,j}, \hat{e}_{j,i})$
  end for
  if $\hat{v}_i$ is not EOS **then**
    $\hat{S}_i \leftarrow (\hat{v}_i, \hat{\mathcal{E}}_i)$
  end if
until $\hat{v}_i$ is EOS
Convert $\hat{S}$ into graph $\hat{G}_s$

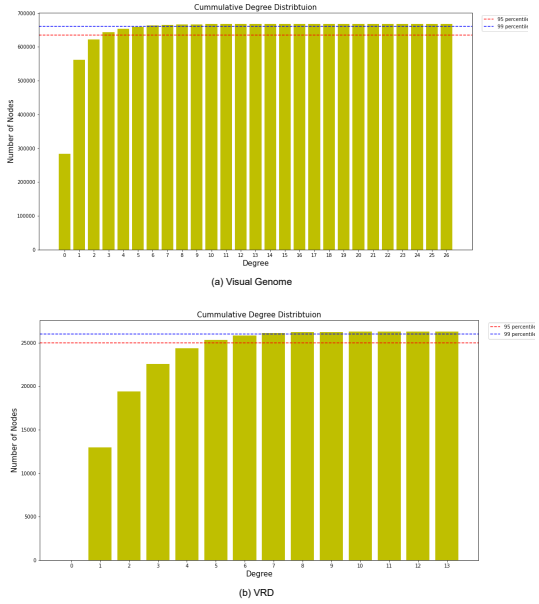

Figure 1. Cumulative degree distribution for the two datasets, Visual-Genome and VRD.

		Visual Genome			
		GraphRNN	**GraphRNN***	**GEMS (GloVe)**	**GEMS (cskg)**
MMD	Degree (x10^2) ↓	47.47	16.44	**2.11**	16.71
	Clustering (x10^2) ↓	18.63	4.05	**0.86**	1.31
	NSPDK* (x10^3) ↓	22.6	5.10	**1.21**	2.71
	Node Label (x10^4) ↓	5.44	5.26	5.19	**5.18**
	Edge Label (x10^2) ↓	22.38	6.19	**1.13**	3.34
Node Metrics	Count Reference	11.09			
	Count Predicted	29.53	13.84	10.17	10.97
	$(Obj)_K$ (x10^2) ↑	83.7	86.9	**92.9**	85.9
Edge Metrics	Count Reference	5.01			
	Count Predicted	57.95	11.86	7.45	13.50
	MEP (x10^2) ↑	22.4	24.52	**35.81**	35.12
Novelty (x10^2) ↑		12.26	57.59	**75.75**	68.96

Table 1. Comparison of 2 variants of our method, GEMS with glove embeddings and common-sense knowledge graph (cskg) [11] in input and external knowledge loss with the baselines of GraphRNN [10] and GraphRNN* on Visual Genome dataset. For all MMD based metrics, lower is better (↓). For the rest of the metrics, larger is better (↑). GraphRNN* refers to GraphRNN with $max\_prev\_node = 6$ (Visual Genome)

		Visual Genome				VRD			
		GraphRNN	**GraphRNN***	**SceneGraphGen**	**GEMS**	**GraphRNN**	**GraphRNN***	**SceneGraphGen**	**GEMS**
Edge Metrics	$(Trip)_K$ (x10^2) ↑	34.6	44.7	**72.9**	**52.8**	35.1	37.3	**43.9**	**38.1**
	ZSEP (x10^2) ↑	3.19	**3.41**	**10.1**	3.14	2.76	2.93	**6.2**	**3.18**

Table 2. Comparison of our method (GEMS) with the baselines of GraphRNN [10], GraphRNN* and SceneGraphGen [2] on Visual Genome [5] and VRD [7] datasets on additional metrics, Top-K Triplet Co-occurrence ($(Trip)_K$) and Zero-shot Edge Precision (ZSEP) . For both the metrics, upper is better (↑). GraphRNN* refers to GraphRNN with $max\_prev\_node = 6$ and 7 for Visual Genome and VRD respectively in Table 2. **Note:** **red** represents best and **blue** represents second best scores.

5. Common Sense Knowledge Graph Embeddings

We wanted to design a method that is agnostic to the type of embeddings (both at the input level as well as in the loss function as external knowledge) used to represent nodes and relationships. Hence, we experiment with the use of embeddings derived from the Common-Sense Knowledge Graph (CSKG, proposed in [11]), whose vocabulary is the same as that of objects and relations in Visual Genome dataset. To obtain the knowledge graph embeddings, we train ComplEx model [9] using the OpenIE framework [3] on the CSKG. Our hypothesis was that the nature of the information carried in the knowledge graph and word embeddings would be very different (co-occurrence in graph context, versus in unstructured text). However, we did not observe significant differences in the end results obtained using CSKG embeddings compared to GloVe embeddings [8]. Table 1 shows comparison in terms of metric values computed on graphs generated by GEMS (using glove and common-sense knowledge graph embeddings) with GraphRNN and GraphRNN*.

6. Additional Proposed Metrics

In this section we describe the novel metrics that we have proposed for the evaluation of expanded scene graphs. We also provide the metric values computed on baselines GraphRNN, GraphRNN* and SceneGraphGen, and our model (GEMS) for Visual Genome [5] and VRD [7] datasets.

6.1. Top-K Triplet Co-occurrence

$(Trip)_k$ The co-occurrence of a triple (Subject, Predicate, Object) in a set of graphs is calculated as the conditional probability of the predicate connecting the subject, object pair given that the pair is present. We compare the co-occurrence of the K-most commonly observed triples in the test set with the co-occurrence of the corresponding triples

in the generated set of graphs as follows:

$$(Trip)_K = 1 - \frac{1}{K} \sum_{\substack{v_i, e_j, v_k \in \\ top_k(P_{test})}} \mid P_{test}[i,j,k] - P_{gen}[i,j,k] \mid \quad (1)$$

Here, P_{test} (P_{gen}) is a matrix such that entry (i, j, k) is the co-occurrence of the triple (v_i, e_j, v_k) in the test set (generated set respectively). In combination with the other metrics, $(Trip)_K$ rewards a model that generates graphs containing coherent relations between two objects with similar probabilistic distribution as observed in training set. Note that a trigram MMD metric that compares the triplet distributions in the real and generated sets would achieve the same purpose. However, given the computational complexity of computing such an MMD-based metric, we propose the metric as defined above.

6.2. Zero-Shot Edge Precision

$ZSEP$ This is a metric inspired from zero shot learning [1, 6] to compute the relevance of the novel edges being generated by the model. [7] proposed a metric along these lines for zero shot visual relationship detection, which was based on recall. Our proposed metric computes the fraction of novel edges generated by the model which are present in the test data. The presence of a relationship in the test data implies that it is realistic, hence this fraction is a surrogate measure of how well the model learns about unseen relationships by leveraging similar relationships which it has already seen in the training distribution.

$$ZSEP = \frac{\sum_{e \in (G_E \setminus D_E) \cap T_E} 1}{\sum_{e \in (G_E \setminus D_E)} 1} \quad (2)$$

Here, G_E, D_E and T_E refer to the set of directed edges present in the generated graph G, the training set and the test set respectively.

7. Additional Qualitative Results

In figures 2, 3 and 4, 5 we show additional results of graphs expanded by our model (GEMS) on Visual Genome seed graphs and compare them with baselines GraphRNN* (GraphRNN with $max\_prev\_node = 6$) and SceneGraphGen [2] respectively.

In figure 6, 7 we show additional results of graphs expanded by our model (GEMS) on Visual Genome seed graphs and the corresponding 64x64 images generated by sg2im [4] (with the pretrained model for Visual Genome dataset) using these expanded scene graphs against baselines of GraphRNN* (GraphRNN with $max\_prev\_node = 6$) and SceneGraphGen [2].

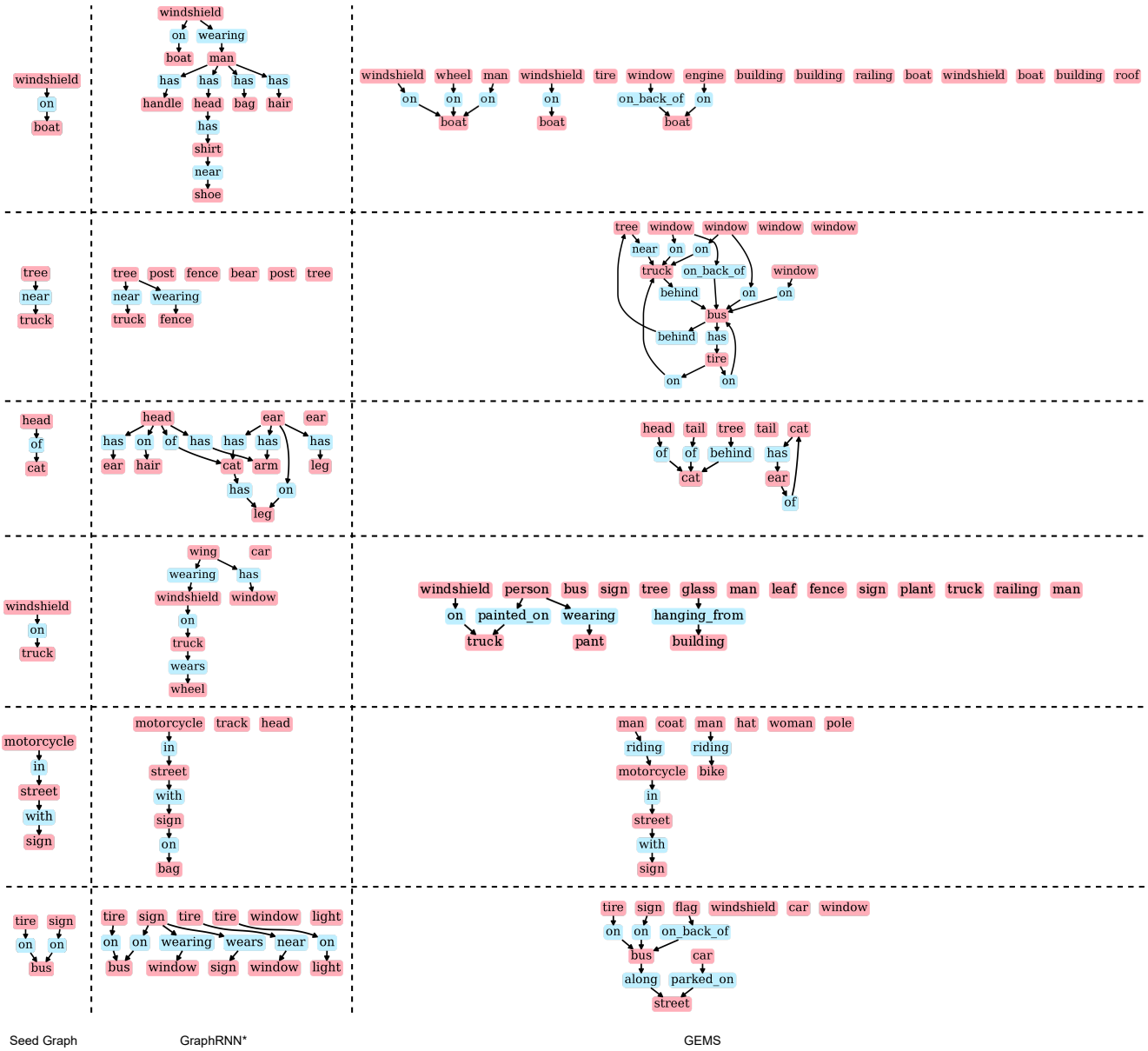

Figure 2. Additional comparison of expanded graphs generated by our model (GEMS) v/s baseline GraphRNN* (GraphRNN with $max\_prev\_node = 6$) on Visual Genome seed graphs.

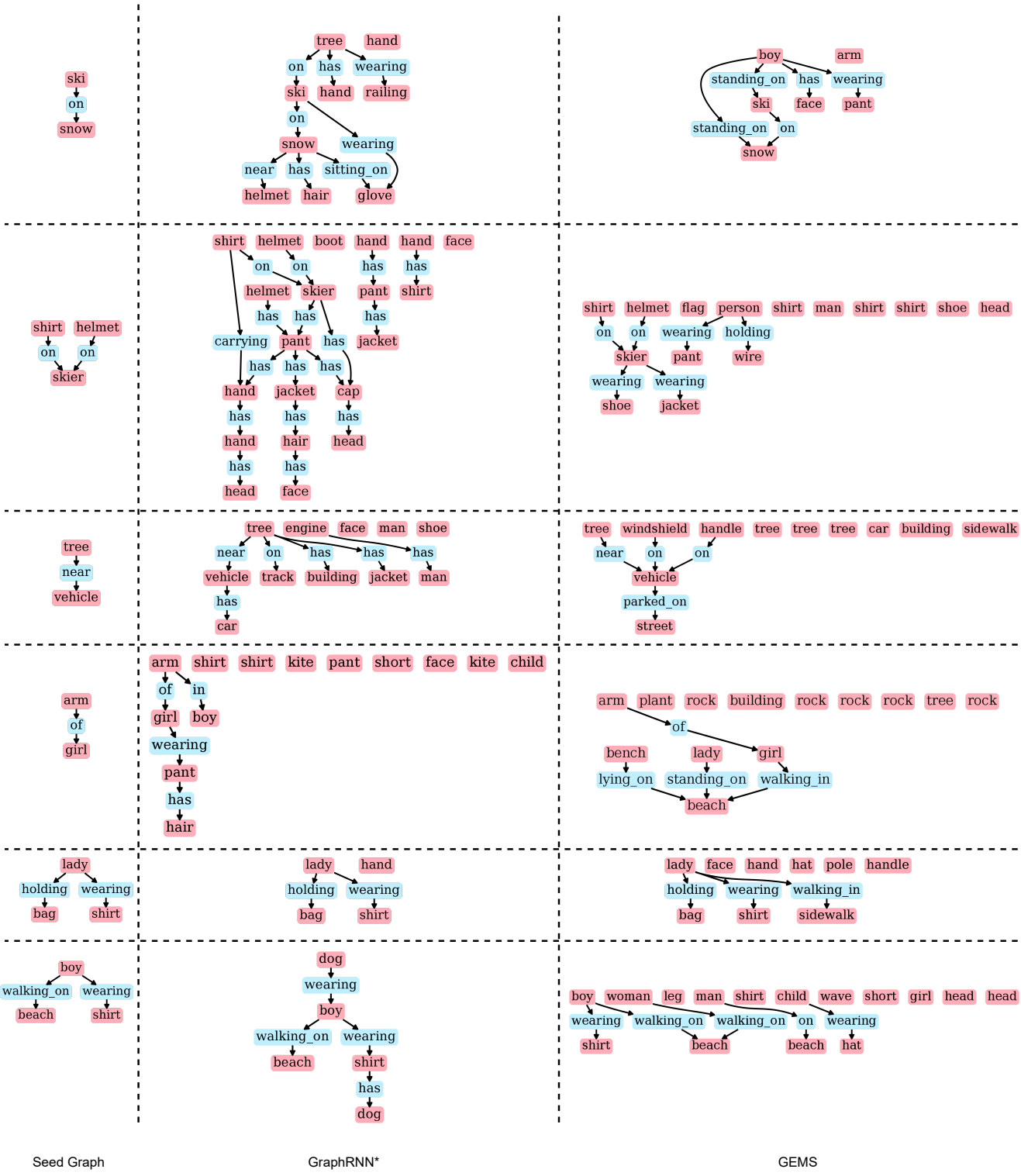

Figure 3. Additional comparison of expanded graphs generated by our model (GEMS) v/s baseline GraphRNN* (GraphRNN with $max\_prev\_node = 6$) on Visual Genome seed graphs.

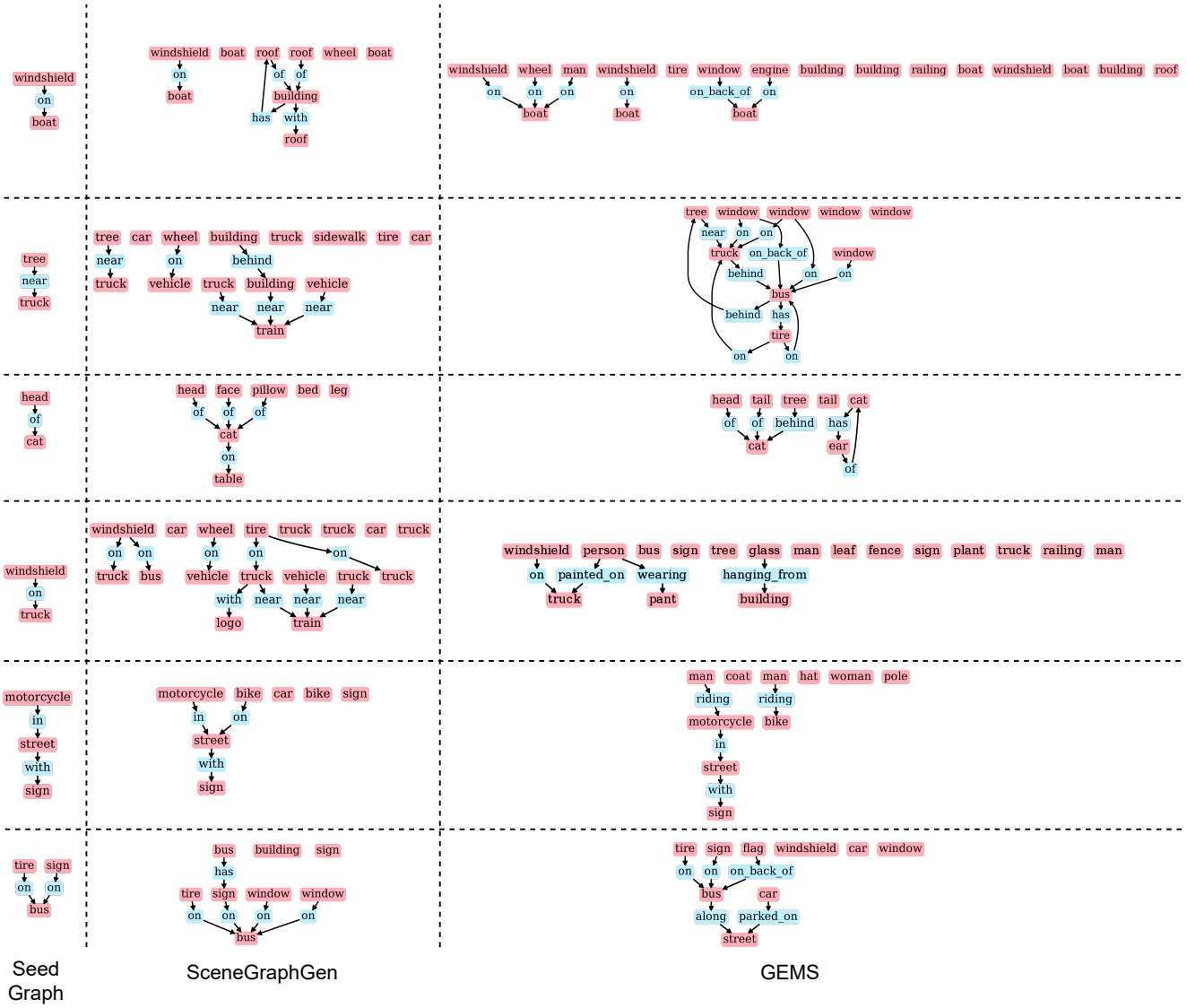

Figure 4. Additional comparison of expanded graphs generated by our model (GEMS) v/s baseline SceneGraphGen on Visual Genome seed graphs.

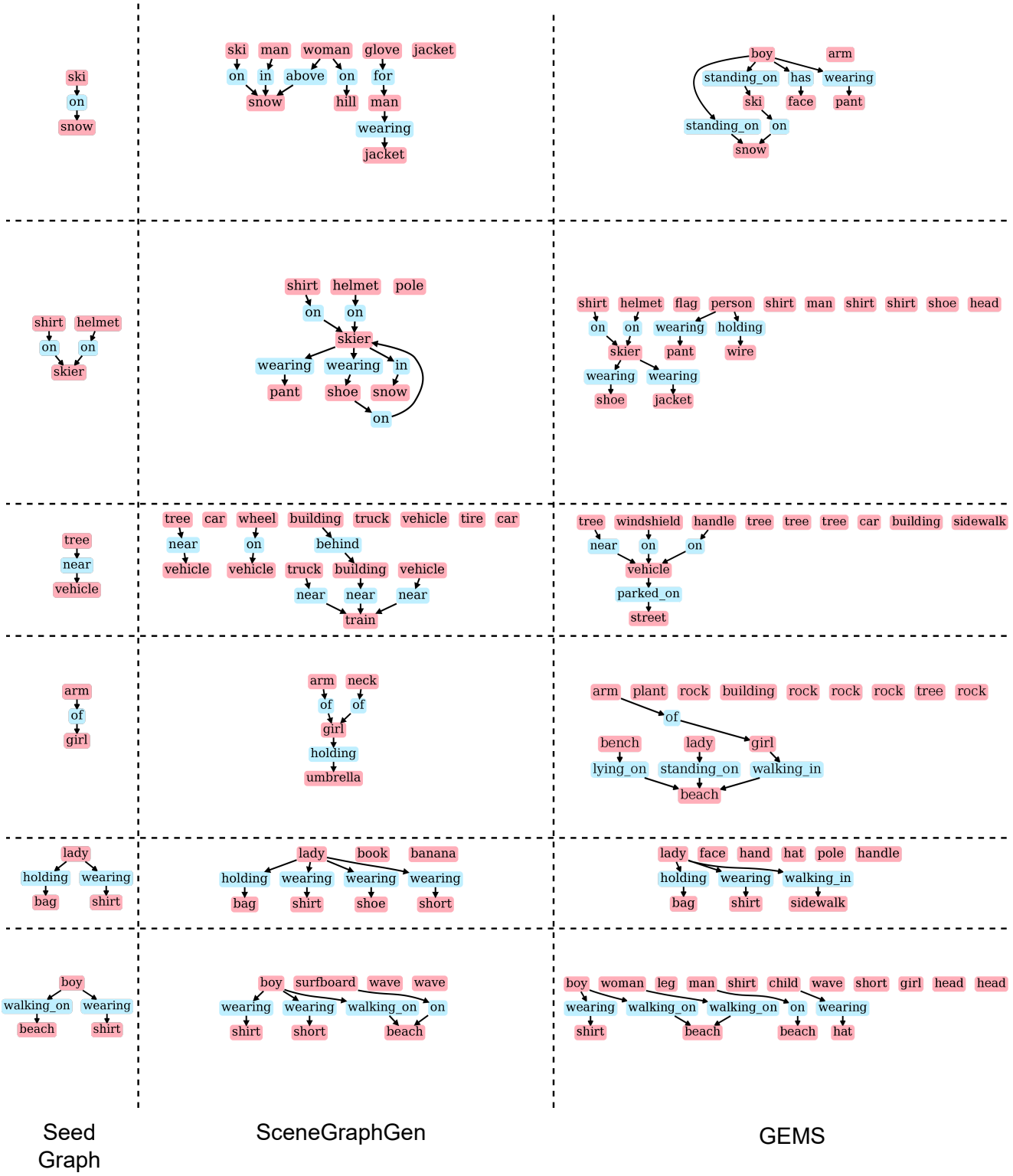

Figure 5. Additional comparison of expanded graphs generated by our model (GEMS) v/s baseline SceneGraphGen on Visual Genome seed graphs.

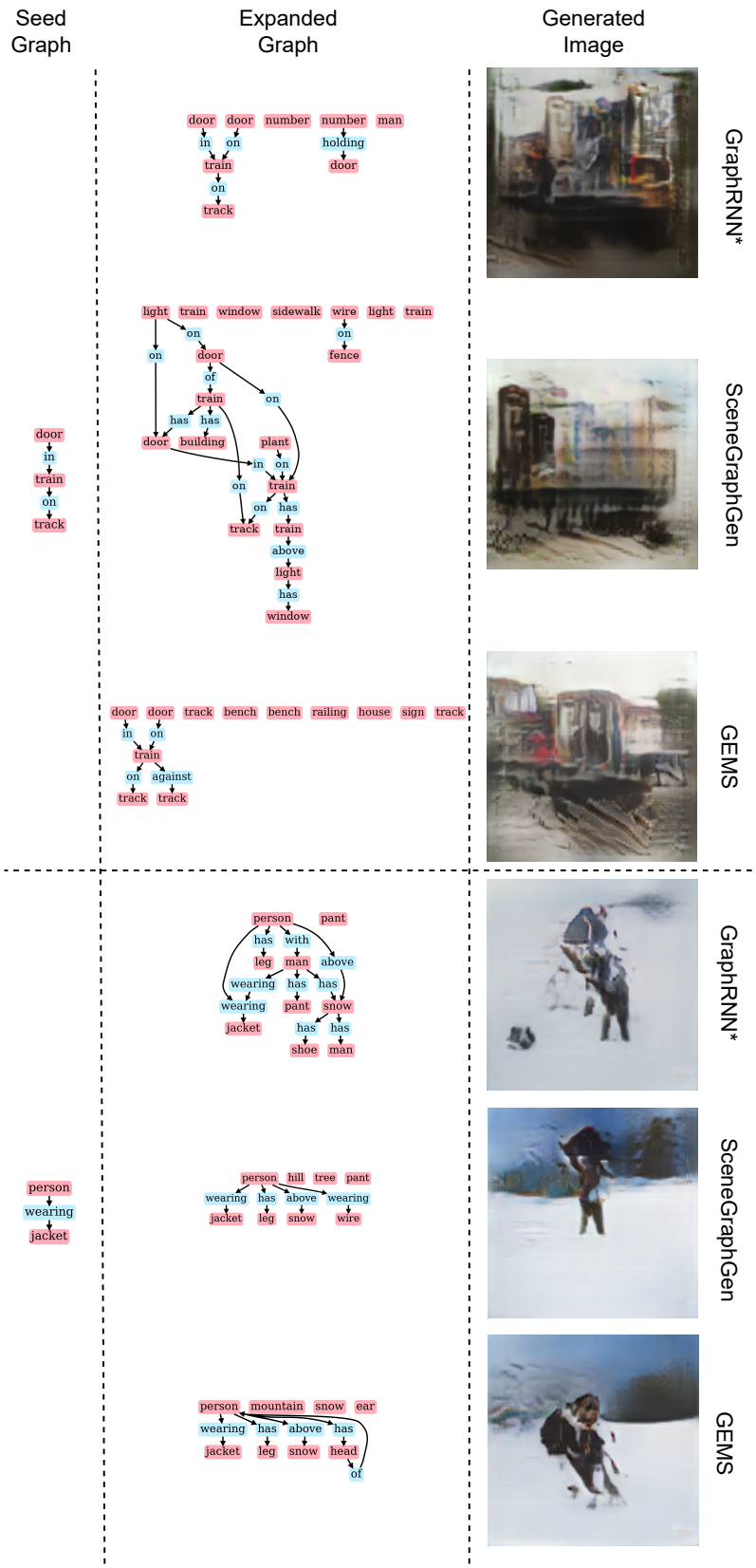

Figure 6. Additional comparison of images generated by sg2im [4] using expanded scene graphs from seed graphs using our method (GEMS) baselines GraphRNN* and ScenegraphGen.

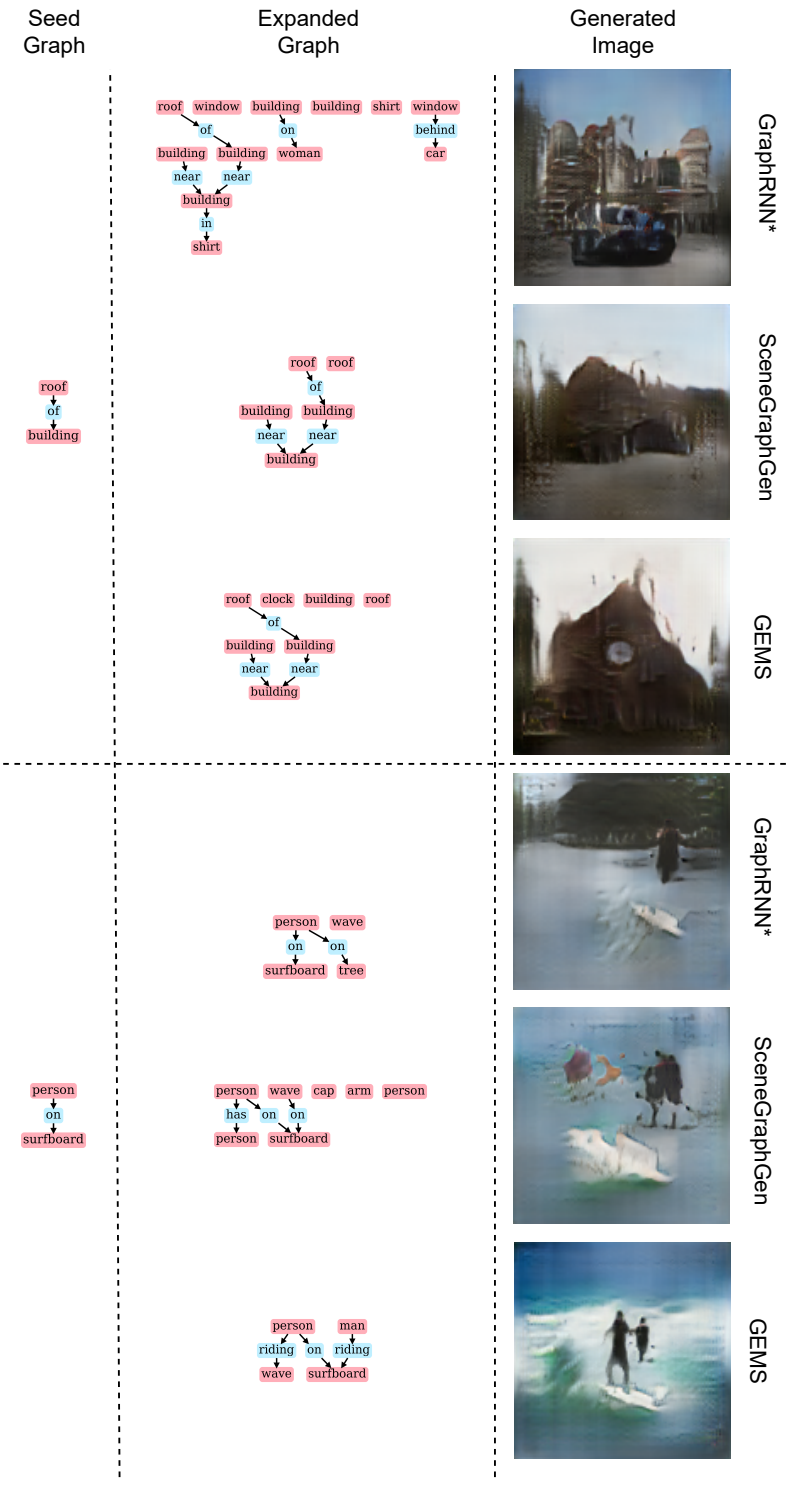

Figure 7. Additional comparison of images generated by sg2im [4] using expanded scene graphs from seed graphs using our method (GEMS) baselines GraphRNN* and ScenegraphGen.

References

[1] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. Importance of semantic representation: Dataless classification. In *AAAI*, volume 2, pages 830–835, 2008.

[2] Sarthak Garg, Helisa Dhamo, Azade Farshad, Sabrina Musatian, Nassir Navab, and Federico Tombari. Unconditional scene graph generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[3] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.

[4] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018.

[5] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[6] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008.

[7] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 852–869, Cham, 2016. Springer International Publishing.

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[9] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[10] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5708–5717. PMLR, 10–15 Jul 2018.

[11] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. Bridging knowledge graphs to generate scene graphs. In *ECCV*, 2020.