Supplementary: Iterative Scene Graph Generation with Generative Transformers

Comparison with other Transformer-based and graph generative models. While many works use transformers, such as Cong *et al.* [1], in their internal mechanism, our work differs in several key aspects. We do not use transformers as in encoder-decoder for feature contextualization and edge labeling. Other iterative models such as Khandelwal *et al.* [2] first sample a graph and further refine them in a multi-stage process. Our iterative generation is based on sampling the graph structure and populating the sites and edges based on labeling. We begin with object proposals (hypotheses) that are used to generate an interaction graph between entities. This requires joint reasoning over the visual and semantic features to identify objects that share plausible semantic relationships. Our novelty lies in identifying the entity interactions based on visual semantic features and their use in the iterative generation of predicate labels. Similarly, we take inspiration from current graph generation models, which directly predict the graph from an input image. They do not generate, refine and reject hypothetical nodes to generate the graph structure. Further, we generate directed graphs with labeled edges, while other graph generative approaches work with undirected graphs with unlabeled edges, which increases the expressiveness of the generated graph.

Hypothesis Generation. The key contribution of our approach is the ability to accept/reject proposals. The graph sampling module (Sec 3.2) can reject entities from the final graph if there are no interactions present i.e., its adjacency is null. Similarly, the relation predictor (Sec 3.3) can reject the relation (edge) hypotheses proposed by Sec 3.2 by predicting a background class if the likelihood of a named predicate is low. Hence, the number of detected entities from the underlying object detector does not necessarily dictate the number of nodes in the final graph due to the dual-stage hypotheses generation and evaluation process, unlike current SGG which constructs edges based on pairwise comparisons and, with unbiasing, the correlation between co-occurrences of predicates. Additionally, the iterative graph sampling (Sec 3.2) allows us to progressively refine interactions by generating *complete* adjacency lists for each node to model possible interactions with other nodes. Hence, the adjacency matrix is not triangular since the edges are directed. To handle the sparsity in the underlying adjacency matrix, during training we allow the ground truth to have bidirectional edges, i.e., all present edges are also mirrored to avoid overfitting to the background "no edge" class. The node hypothesis provides location and semantic features to influence the edge direction.

Comparison with unbiased SGG. The focus of this work is to demonstrate that sampling an interaction graph can improve scene graph generation beyond specialized training for unbiasing. While we use a naïvely weighted softmax function to tackle the long-tail distribution, it is not considered "unbiased" SGG in the conventional sense, since according to the definition in the seminal paper by Tang et al [29], unbiasing involves "using the context and content" to de-bias the predicate learning "beyond reweighting with class frequency", which can "fail to generalize to unseen relationships, i.e., zero-shot SGG." The interaction graph sampling (Section 3.2) allows us to model the context and content during generation to reduce the need for a separate unbiasing step. Our work (see Table 2 in the main paper) outperforms all approaches, even unbiased SGG models, in zero-shot prediction, indicating that the need for unbiasing is reduced when modeling the interaction graph. Similarly, we provide a considerable improvement over early unbiasing models such as EBML and TDE as well as recent ones such as RU-NET. The performance gap is further reduced when considering the much harder SGCls and SGDet tasks, thus validating our original hypothesis. Without the weighted softmax, we achieve mR@50/100 of 23.7 and 27.1 on PredCls, 13.6 and 15.2 on SGCls and 7.8 and 10.9 on SGDet, respectively, which still outperforms all other SGG approaches without *any form of unbiasing*.

Computational Complexity. The average number of objects detected in each image in the test split of the VG dataset is 31 and the number of possible edges is 930. We consider at most 250 edges from the sampled graph of which an average of 234 are considered plausible edges. This is less than 20% of possible combinations that current SGG models consider. Figure 3 shows the impact of varying these top-k edges, with the orange showing the lack of GGT with just an edge prior. The number of parameters in the framework is Faster-RCNN: 41.8M, graph sampling:

39.2M, and relation modeling: 18.8M for a total of 99.8M parameters. For comparison, MOTIFS has 240.7M parameters, KERN: 405.2M, BGNN: 341.9, and IMP: 203.8, while the single-stage FCSGG has 87.1M parameters. The total fps is approximately 9, which is better than MOTIFS (6.6), FCSGG (8.4), and BGNN (2.3) to name a few. Leveraging advances in linear attention will help speed up computation.

References

[1] Yuren Cong, Wentong Liao, Hanno Ackermann, Bodo Rosenhahn, and Michael Ying Yang. Spatial-temporal transformer for dynamic scene graph generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16372–16382, 2021. 1

[2] Siddhesh Khandelwal and Leonid Sigal. Iterative scene graph generation. In *Advances in Neural Information Processing Systems*, 2022. 1