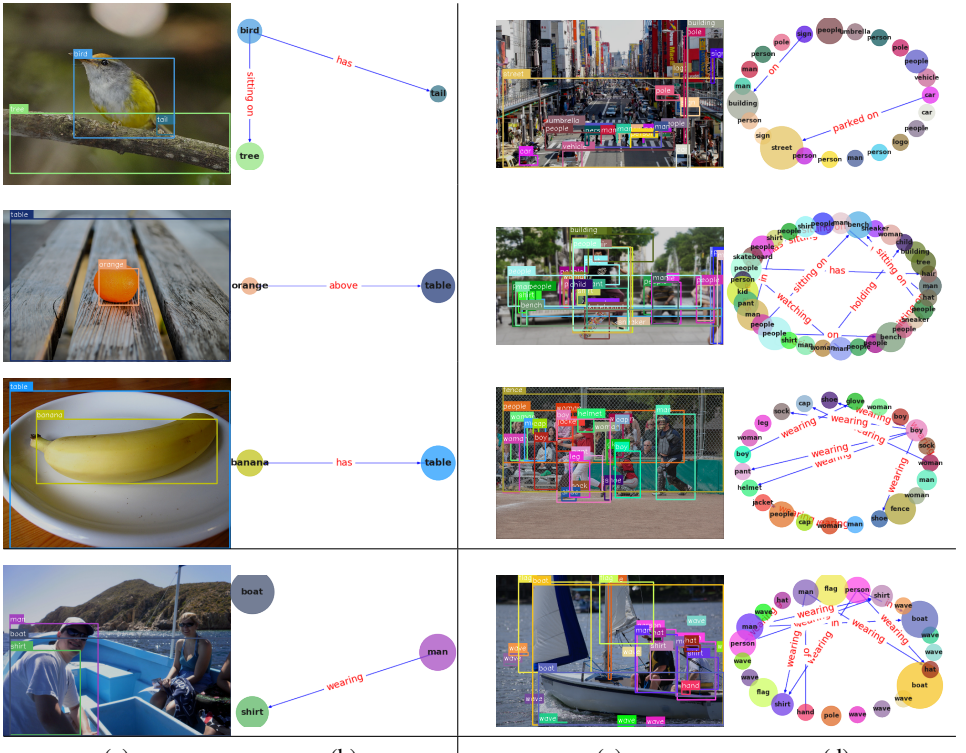
Supplementary Material to "Graph Density-Aware Losses for Novel Compositions in Scene Graph Generation"

Boris Knyazev [1,2]
bknyazev@uoguelph.ca
Harm de Vries[3]
mail@harmdevries.com
Cătălina Cangea[4]
Catalina.Cangea@cst.cam.ac.uk
Graham W. Taylor[1,6,2]
gwtaylor@uoguelph.ca
Aaron Courville[6,7,5]
aaron.courville@umontreal.ca
Eugene Belilovsky[5]
belilove@mila.quebec

[1] University of Guelph
[2] Vector Institute for Artificial Intelligence
[3] Element AI
[4] University of Cambridge
[5] Mila, Université de Montréal
[6] Canada CIFAR AI Chair
[7] CIFAR LMB Fellow

1.1 Additional Results and Analysis



(a) (b) (c) (d)

Figure 1: Examples of small (b) and large (d) training scene graphs and corresponding images ((a) and (c) respectively) from Visual Genome (split [17]). Two related factors mainly affect the size of the graphs: 1) the complexity of a scene (compare left and right images in the top three rows); and 2) the amount of annotations (*e.g.* in the bottom row two images have similar complexity, but in one case the annotations are much more sparse). On average, for more complex scenes, the graphs tend to be larger and more sparse ($d \leq 2\%$ on the depicted large graphs, (d)), because for many reasons it is challenging to annotate all edges. In contrast, simple scenes can be described well by a few nodes, making it easier to label most of the edges, which makes graphs more dense ($d > 15\%$ on the depicted small graphs, (b)).

(a) (b) (c) (d)

Figure 2: For comparison, we illustrate scene graphs of the same images, but annotated in GQA. As opposed to VG, scene graphs in GQA are generally larger and more dense, primarily due to "left of" and "right of" edge annotations (see detailed dataset statistics in Table 1).

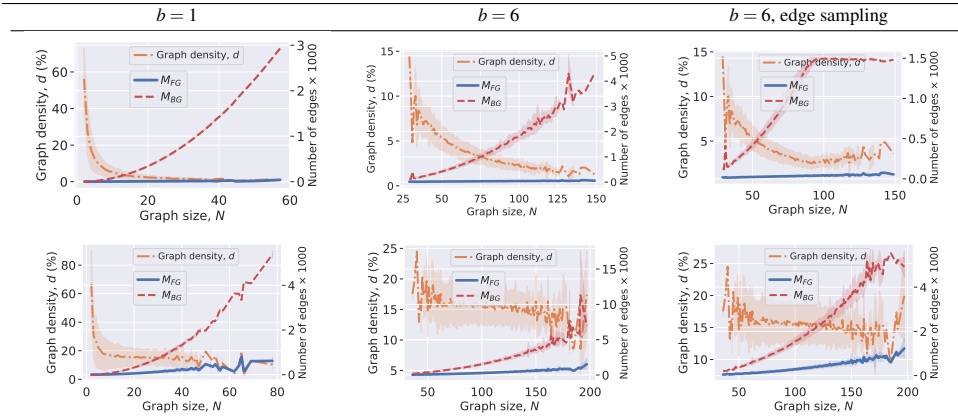


Figure 3: Plots additional to Figure 4 in the main paper. Graph density d of a batch of scene graphs for different batch sizes b for VG (top row) and GQA (bottom row). For $b = 1$: $M_{FG} \approx 0.5N$ for VG and $M_{FG} \approx 8N$. Larger batches make the density of a batch less variable and, on average, larger. This is because by stacking scene graphs in a batch, there are no BG edges between different graphs, so M_{BG} grows slower making the density larger. Thus, increasing the batch size partially fixes the discrepancy of edge losses between small and large scene graphs. Subsampling edges during training also helps to stabilize and increase graph density. We explore this effect in more detail on Figure 4.

	VG [VG]	**VTE [VG]**	**GQA [GQA]**
#obj classes	150	200	1,703
#rel classes	50	100	310
# train images	57,723	68,786	66,078
# train triplets (unique)	29,283	19,811	470,129
# val images	5,000	4,990	4,903
# test images	26,446	25,851	10,055
# test-ZS images	4,519	653	6,418
# test-ZS triplets (unique/total)	5,278/7,601	601/2,414	37,116/45,135

	min-max	avg±std	min-max	avg±std	min-max	avg±std	min-max
N train	2-62	12±6	2-98	13±9	2-126	17±8	2-126
d (%) train	0.04-100	7±8	0.7-100	12±13	0.5-100	17±10	0.03-100
N test	2-58	12±7	2-110	13±9	2-97	17±8	2-67
d (%) test	0.12-100	6±8	0.6-100	11±12	0.6-100	17±10	0.05-100
N test-ZS	2-55	14±7	2-78	8±11	2-97	19±8	2-65
d (%) test-ZS	0.05-50	2±4	0.05-100	3±7	0.03-50	3±3.6	0.02-50

Table 1: Statistics of Visual Genome [VG] variants used in this work. GQA-nLR is our modification of GQA [GQA] with predicates 'to the left of' and 'to the right of' excluded from all training, validation and test scene graphs. Graph density d decreases dramatically in this case, since those two predicates are the majority of all predicate labels.

1.2 Evaluation

Evaluation of zero/few shot cases. To evaluate n-shots using image-level recall, we need to keep in the test images only those triplets that have occurred no more than n times and remove images without such triplets. This results in computing recall for very sparse annotations, so the image-level metric can be noisy and create discrepancies between simple images with a few triplets and complex images with hundreds of triplets. For example, for an image with only two ground truth triplets, R@100 of 50% can be a quite bad result, while for an image with hundreds of triplets, this can be an excellent result. Our Weighted Triplet Recall is

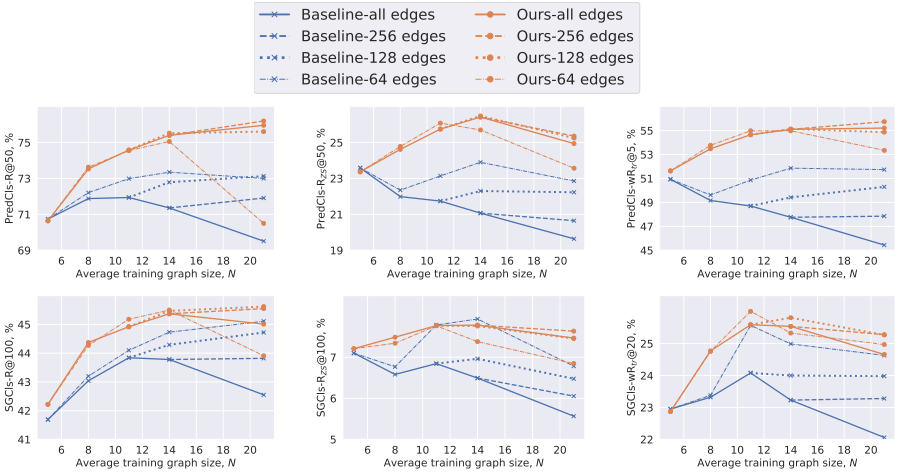


Figure 4: In these experiments, we split the training set of Visual Genome (split [12]) into five subsets with different graph sizes in each subset. This allows us to study how well the models learn from graphs of different sizes. **Edge sampling.** One way to stabilize and increase graph density is to sample a fixed number of FG and BG edges (on the legend, we denote the total possible number of edges per image, M). So we compare results with and without this sampling. Sampling improves baseline results of models trained on larger graphs, however, our best results are still consistently higher than the best baseline results. Even though sampling partially solves the problem of varying density, it does not solve it in a principle way as we do. Sampling only sets the upper bound on the number of edges, so some batches have fewer edges, creating a discrepancy between the losses of edges. Setting a lower bound on the number of edges is challenging, because some graphs are very sparse, meaning that frequently many more graphs need to be sampled to get enough edges, which requires much more computational resources. **Potential oversmoothing.** In case of our loss the results for larger graphs do not improve in some cases or even slightly get worse. We believe that besides density normalization, another factor making it challenging to learn from large graphs, can be related to the "oversmoothing" effect [9]. Oversmoothing occurs when all nodes after the final graph convolution start to have very similar features. This typically happens in deep graph convolutional networks. But oversmoothing can also occur in *complete* graphs, which are used in the SGG pipeline as the input to message passing (see Figure 1 in the main paper). Complete graphs lead to node features being pooled (averaged) over a very large neighborhood (i.e. all other nodes) and averaging over too many node features is detrimental to their discriminative content. A direction for resolving this issue can be using some form of edge proposals and attention over edges [11, 13].

computed for all test triplets joined into a single set, so it resolves this discrepancy.

Constrained vs unconstrained metrics. In the graph constrained case [12], only the top-1 predicted predicate is considered when triplets are ranked, and follow-up works [7, 14] improved results by removing this constraint. This unconstrained metric more reliably evaluates models, since it does not require a perfect triplet match to be the top-1 prediction, which is an unreasonable expectation given plenty of synonyms and mislabeled annotations in scene graph datasets. For example, 'man *wearing* shirt' and 'man *in* shirt' are similar predictions, however, only the unconstrained metric allows for both to be included in ranking. The SGDET+ metric [13] has a similar motivation as removing the graph constraint, but it does not address the other issues of image-level metrics.

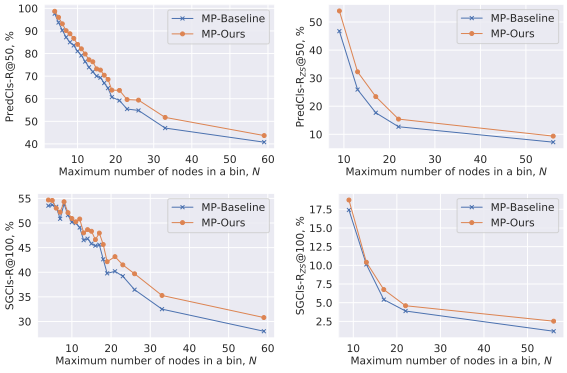


Figure 5: In these experiments, we take a Message Passing model, trained on all scene graphs of Visual Genome, and test it on graphs of different sizes. To plot the curves, we sort the original test set by the number of nodes in scene graphs and then the split the sorted set into several bins with an equal number of scene graphs in each bin (1000 per bin). Each point denotes an average recall in a bin. As was shown in Figure 4, our loss makes learning from larger graphs more effective. Here, we also show that our loss makes the models perform better on larger graphs at test time. We believe that since our loss penalizes larger graphs more during training, the model better learns how to process large graphs. However, we observe the performance drops for larger graphs both for the baseline and our loss. One of the reasons for that might be related to oversmoothing, which can be more pronounced in large graphs (also discussed in Figure 4). Another reason can be a lack of large graphs (*e.g.* > 40 nodes) in the training set. This would align with prior work [6], showing that generalization to larger graphs is challenging and proposed attention as a way for addressing it. This problem, in the context of scene graphs, can be addressed in future work. For example, one interesting study may include training on so called "region graphs", describing small image regions, available in Visual Genome [6] and attempting to predict full scene graphs.

	VG [32]	**VTE [33]**	**GQA [8]**	**GQA-nLR**
Object detector	Faster R-CNN [8]	Mask R-CNN [8], chosen in lieu of Faster R-CNN, since it achieves better performance due to multitask training on COCO. In SGGen, we extract up to 50 bounding boxes with a confidence threshold of 0.2 as in [8].		
Detector's backbone	VGG16	ResNet-50-FPN		
Detector pretrained on	VG [32]	COCO (followed by fine-tuning on GQA in case of SGGen)		
Learning rate	$0.001 \times b$	$0.001 \times b$	$0.002 \times b$ (increased due to larger graphs in a batch)	$0.001 \times b$
Batch size (# scene graphs), b	6			
# epochs	MP: 20, lr decay by 0.1 after 15 epochs; NM: 12, lr decay by 0.1 after 10 epochs			

Table 2: Architecture details. In NM's implementation, the number of epochs is determined automatically based on the validation results. We found it challenging to choose a single metric to determine the number of epochs, so we fix the number of epochs based on manual inspection of different validation metrics.

Comparison on SGGen, $P(\mathcal{G}|I)$. In SGCls and PredCls, we relied on ground truth bounding boxes B_{gt}, while in SGGen the bounding boxes B_{pred} predicted by a detector should be used to enable a complete image-to-scene graph pipeline. Here, even small differences

Model	Backbone	Loss	SGGen-GC			SGGen		
			R@100	R_{ZS}@100	mR@100	R@100	R_{ZS}@100	mR@100
FREQ [14]	VGG16		27.6	0.02	5.6	30.9	0.1	8.9
MP [11, 6]	VGG16	BASELINE (3)	24.3	0.8	4.5	27.2	0.9	7.1
	VGG16	OURS (6)	25.2	0.9	5.8	28.2	1.2	9.5
NM [14]	VGG16	BASELINE (3)	29.8	0.3	5.9	35.0	0.8	12.4
	VGG16	OURS (6)	29.4	1.0	**8.1**	35.0	1.8	15.4
	VGG16	OURS (6), NO FREQ	30.4	**1.7**	7.8	35.9	**2.4**	15.3
KERN [1]	VGG16	BASELINE (3)	29.8	0.04	7.3	35.8	0.02	**16.0**
RelDN [16]	VGG16	BASELINE (3)	**32.7**	–	–	**36.7**	–	–
RelDN [16]	ResNeXt-101	BASELINE (3)	36.7	–	–	40.0	–	–
NM [10]	ResNeXt-101	BASELINE (3)	**36.9**	0.2	6.8	–	–	–
NM+TDE [10]	ResNeXt-101	BASELINE (3)	20.3	2.9	9.8	–	–	–
VCTree+TDE [10]	ResNeXt-101	BASELINE (3)	23.2	**3.2**	**11.1**	–	–	–

Table 3: Comparison of our SGG methods to the state-of-the-art on Visual Genome (split [17]). We report results with and without the graph constraint, **SGGen-GC** and **SGGen** respectively. All models use Faster R-CNN [8] as a detector, but the models we evaluate use a weaker backbone compared to [10]. Interestingly, TDE's improvement on zero-shots (R_{ZS}) and mean recall (mR) comes at a significant drop in R@100, which means that frequent triplets are recognized less accurately. Our loss does not suffer from this. Table cells are colored the same way as in Table 1 in the main paper.

between B_{gt} and B_{pred} can create large distribution shifts between corresponding extracted features (V,E) (see Section 3.1), on which SGCls models are trained. Therefore, it is important to *refine* the SGCls model on (V,E) extracted based on predicted B_{pred}, according to previous work [1, 14]. In our experience, this refinement can boost the R@100 result by around 3% for Message Passing and up to 8% for Neural Motifs (in absolute terms). In Table 3, we report results after the refinement completed both for the baseline loss and our loss in the same way. Similarly to the SGCls and PredCls results, our loss consistently improves baseline results in SGGen. It also allows Neural Motifs (NM) to significantly outperform KERN [1] on zero-shots (R_{ZS}@100), while being only slightly worse in one of the mR@100 results. The main drawback of KERN is its slow training, which prevented us to explore this model together with our loss. Following our experiments in Table 1 and Figure 6 (see the main paper), we also confirm the positive effect of removing FREQ from NM. A more recent work of Tang et al. [10] shows better results on zero-shots and mean recall, however, we note their more advanced feature extractor, therefore it is difficult to compare our results to theirs in a fair fashion. But, since they also use the baseline loss (3), our loss (6) can potentially improve their model, which we leave for future work.

Finally, we evaluate SGGen on GQA using Message Passing (Table 4), where we also obtain improvements with our loss. GQA has 1703 object classes compared to 150 in VG making object detection harder. When evaluating SGGen, the predicted triplet is matched to ground truth (GT) if predicted and GT bounding boxes have an intersection over union (IoU) of $\geq$50%, so more misdetections lead to a larger gap between SGCls and SGGen results.

Loss	R@300	R_{ZS}@300	mR_{tr}@300
BASELINE (3)	6.2	0.5	1.3
OURS (6)	**6.3**	**0.7**	**2.4**

Table 4: **SGGen** results on GQA [4] using MP. Mask R-CNN [2] fine-tuned on GQA is used in this task.

References

[1] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6163–6171, 2019.

[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.

[3] Drew Hudson and Christopher D Manning. Learning by abstraction: The neural state machine. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5903–5916. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8825-learning-by-abstraction-the-neural-state-machine.pdf.

[4] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019.

[5] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems*, pages 4204–4214, 2019.

[6] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, and et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, Feb 2017. ISSN 1573-1405. doi: 10.1007/s11263-016-0981-7. URL http://dx.doi.org/10.1007/s11263-016-0981-7.

[7] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *Advances in neural information processing systems*, pages 2171–2180, 2017.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[9] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. The truly deep graph convolutional networks for node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[10] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[12] 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*, pages 5410–5419, 2017.

[13] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

[14] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5831–5840, 2018.

[15] Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. Visual translation embedding network for visual relation detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5532–5540, 2017.

[16] Ji Zhang, Kevin J Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. Graphical contrastive losses for scene graph parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11535–11543, 2019.