Appendix

A. Experimental Details

A.1. Architectures

For our GAN model see the detailed architectures of the generator and discriminators in Tables 4 and 5 respectively.

Table 4. **Architecture of the generator** G**.** All GraphTripleConv and Upsample-Refine blocks include Batch Norm (BN) [30] followed by ReLU/LeakyReLU. $\tilde{n}, \tilde{m}$ are the number of nodes and edges in a batch; bs is batch size.

Layer	**Nodes**	**Edges**
input-1 (embedd. layer)	$\tilde{n}$x200	$\tilde{m}$x200
concat. with boxes B	$\tilde{n}$x204	$\tilde{m}$x200
GraphTripleConv-1	204x64	608x64
GraphTripleConv-2	64x64	192x64
GraphTripleConv-3	64x64	192x64
GraphTripleConv-4	64x64	192x64
GraphTripleConv-5	64x(32·7·7)	192x64
output-1	$\tilde{n}$x32x7x7	–
conv1	32x64	
conv2	64x64	
output-2	$\tilde{n}$x64x7x7	
input-2: vis. features V'	$\tilde{n}$x512x7x7	
concatenate	output-2, input-2	
conv3	(64+512)x64	
output-3	$\tilde{n}$x64x7x7	
input-3: boxes B	$\tilde{n}$x4	
Boxes2Layout	build feature maps based on output-3 and B	
output-4	bsx64x37x37	
	Global feature maps	
Upsample-Refine-1	64x128	
Upsample-Refine-2	128x256	
Upsample-Refine-3	256x512	
conv1x1	512x512	
output-5 ($\hat{H}$)	bsx512x37x37	
RoIAlign	extract node, edge feat. based on output-5 and B	
output-6 ($\hat{V}$, $\hat{E}$)	$\tilde{n}$x512x7x7	$\tilde{m}$x512x7x7

In the generator, GraphTripleConvNet[2], Boxes2Layout[3], Upsample-Refine[4] are borrowed from the sg2im implementation of [31]. For the baseline IMP++ model we use a publicly available implementation[5] with a default architecture and the graph-normalized loss [36] from another public code[6].

A.2. Hyperparameters and Tasks

SGCls. We use batch size $bs = 24$ and $lr = (1e{-}3) \times bs$. Training was done on NVIDIA V100 with 32GB of GPU memory or RTX6000 with 24GB. We train all models for a fixed number of 20 epochs according to [36]. **PredCls** results are obtained following a standard procedure [82] of reusing the SGCls model and setting object classes to ground truth. We use PyTorch 1.5+ to train all models. The difference between the graph constraint and no constraint metrics is discussed in [82, 36]. We also report the results on the **SGGen/SGDet** task in § B.6. However, we highlight that SGGen/SGDet is not the focus of our work and we do not expect large improvements, since we do not update the detector on generated features.

[2]GraphTripleConvNet: https://github.com/google/sg2im/blob/master/sg2im/graph.py

[3]Boxes2Layout: https://github.com/google/sg2im/blob/master/sg2im/layout.py

[4]Upsample-Refine: https://github.com/google/sg2im/blob/master/sg2im/crn.py

[5]IMP+/IMP++: https://github.com/rowanz/neural-motifs

[6]Graph-normalized loss: https://github.com/bknyaz/sgg

Table 5. **Architectures of the discriminators** D**.** Following [56], the discriminators are regularized by having a fully-convolutional architecture with four layers interleaved with ReLU. All convolutions are 3x3 without padding and normalized with Spectral Norm (SN) [50]. $\tilde{n}, \tilde{m}$ are the number of nodes and edges in a batch; bs is batch size; $|\mathcal{C}| = 151, |\mathcal{R}| = 51$ - number of object and predicate classes including the "background" (no object, no edge) class [82].

Layer	D_{node}	D_{edge}	D_{global}
input	$\tilde{n}$x(512+151)x7x7	$\tilde{m}$x(512+51)x7x7	bsx512x37x37
real feature/labels	V, O	E, R	H
fake feature/labels	$\hat{V}, \hat{O}$	$\hat{E}, R$	$\hat{H}$
SN-conv1	(512+151)x256	(512+51)x256	512x256
nonlinearity	ReLU	ReLU	LeakyReLU
pooling	–	–	Average-2
SN-conv2	256x128	256x128	256x256
nonlinearity	ReLU	ReLU	LeakyReLU
pooling	–	–	Average-2
SN-conv3	128x64	128x64	256x128
nonlinearity	ReLU	ReLU	LeakyReLU
pooling	–	–	Average-2
SN-conv4	64x1	64x1	128x1
output	$\tilde{n}$x1	$\tilde{m}$x1	bsx1

A.3. Implementing IMP with TDE

To apply TDE to a SGG model, the model is assumed to have a contextual layer, such as LSTM in Neural Motifs or TreeLSTM in VCTree, *conditioned* on object predictions from the object detector. This contextual layer to some extent captures the frequency bias in the dataset *during training*. So, the main goal of TDE is to debias this contextual layer *at test time*. Since, IMP does not have such a conditional contextual layer and, consequently, is less biased, applying TDE is both (1) unclear implementation-wise and (2) has questionable benefits from the conceptual point of view. It is still possible to use Total Effect (TE) according to Eq. 6 in [67], which we report in the main text. But, there is almost no gain w.r.t. IMP in this case.

B. Additional Experimental Results

All presented results in this work are on Visual Genome [38] (split of [74]).

Table 6. Results of GRAPHN with topk $= 5$ (same as reported in the main text) compared to NEIGH with different values of topk. The models are based on IMP++ [82, 36].

MODEL	ZERO-SHOT RECALL		10-SHOT RECALL		100-SHOT RECALL		ALL-SHOT RECALL		
	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls-mR
GAN+GRAPHN, $\alpha = 2$	**9.89**±0.15	28.90±0.14	21.96±0.30	**43.79**±0.27	41.22±0.33	69.17±0.24	50.06±0.29	78.98±0.09	27.79±0.48
GAN+GRAPHN, $\alpha = 5$	9.62±0.29	**29.18**±0.33	**22.24**±0.11	43.74±0.10	41.39±0.26	69.11±0.05	**50.14**±0.21	78.94±0.03	**27.98**±0.23
GAN+GRAPHN, $\alpha = 10$	9.84±0.17	28.90±0.46	22.04±0.33	43.54±0.36	**41.46**±0.15	69.13±0.24	50.10±0.23	**79.00**±0.09	27.68±0.37
GAN+GRAPHN, $\alpha = 20$	9.65±0.15	28.68±0.28	21.97±0.30	43.64±0.20	41.24±0.08	**69.31**±0.17	49.89±0.28	78.95±0.04	27.42±0.36
GAN+NEIGH, topk $= 2$	9.49±0.21	28.58±0.40	21.72±0.23	43.62±0.05	41.04±0.26	69.07±0.09	49.64±0.29	78.94±0.11	27.33±0.41
GAN+NEIGH, topk $= 5$	9.38±0.25	28.68±0.40	21.75±0.25	43.45±0.14	41.05±0.49	68.94±0.14	49.75±0.30	78.94±0.10	27.05±0.15
GAN+NEIGH, topk $= 10$	9.58±0.22	28.63±0.39	21.86±0.23	43.77±0.15	41.14±0.30	69.03±0.09	49.86±0.38	78.89±0.01	27.41±0.51
GAN+NEIGH, topk $= 20$	9.65±0.04	28.57±0.06	21.82±0.17	43.28±0.21	40.98±0.30	69.01±0.14	49.68±0.28	78.92±0.05	27.17±0.12

Table 7. Results using models based on Neural Motifs++ [82, 36].

MODEL	ZERO-SHOT RECALL		10-SHOT RECALL		100-SHOT RECALL		ALL-SHOT RECALL		
	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls-mR
Neural Motifs++	6.81±0.10	16.91±0.31	21.07±0.03	44.75±0.12	40.16±0.10	73.58±0.14	48.30±0.06	82.17±0.09	25.48±0.15
Neural Motifs++, GAN+GRAPHN	**8.06**±0.13	**23.36**±0.08	20.99±0.04	**45.98**±0.04	39.92±0.05	**73.97**±0.13	48.05±0.12	**82.89**±0.08	**25.59**±0.02

Table 8. Comparison of GAN models using GT versus predicted bounding boxes and ORACLE-ZS perturbations.

MODEL	ZERO-SHOT RECALL		10-SHOT RECALL		100-SHOT RECALL		ALL-SHOT RECALL		
	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls	SGCls-mR
GAN+ORACLE-ZS, $\hat{B} = B$	10.11±0.34	29.27±0.10	22.05±0.38	43.78±0.09	41.38±0.50	69.06±0.16	50.19±0.36	79.00±0.08	27.91±0.56
GAN+ORACLE-ZS $\hat{\mathcal{G}}$ + test $\hat{B}$	10.52±0.31	29.43±0.42	21.98±0.39	43.03±0.13	41.12±0.19	68.73±0.17	50.05±0.35	78.65±0.09	27.52±0.46
GAN+ORACLE-ZS $\hat{\mathcal{G}}$, pred. $\hat{B}$ (§ B.3)	9.92±0.13	28.93±0.63	21.67±0.36	42.65±0.47	40.96±0.48	68.23±0.28	49.80±0.50	78.55±0.14	27.22±0.39

B.1. Topk Semantic Neighbors

In the main text, we used topk $= 5$ for GRAPHN and topk $= 10$ for NEIGH to allow for the same level of diversity of perturbations in both strategies. To make sure GRAPHN outperforms not just because of a better choice of topk, we report results of NEIGH for more values in Table 6. Overall, different values of topk do not allow NEIGH to achieve the same level of performance as GRAPHN.

B.2. BERT-based Evaluation

We show an example of BERT-based evaluation in Fig. 11. In this example, we retrieve the score of 9.8 for the token 'shorts' which was masked out in the query. To estimate the overall likelihood of the scene graph we mask out only one node per graph for faster evaluation, which can explain a high variance of SG quality. Aggregating the scores for all nodes should lead to better estimates.

B.3. Learning to Predict Bounding Boxes

In the main text, in our generative pipeline for simplicity we assumed the layout (bounding boxes B) is unchanged after perturbing the associated scene graph: $\hat{B} = B$. Here, we describe and report results of the alternative version, where the boxes $\hat{B}$ are predicted by some model f given a perturbed scene graph $\hat{B} = f(\hat{\mathcal{G}})$. We borrow the same principle as we used for producing $\hat{\mathcal{G}}$: instead of producing $\hat{B}$ from scratch (without relying on B), we perturb ground truth B, so $\hat{B} = f(\hat{\mathcal{G}}, B)$. This is an easier task, since we only need to predict bounding boxes for part of the scene

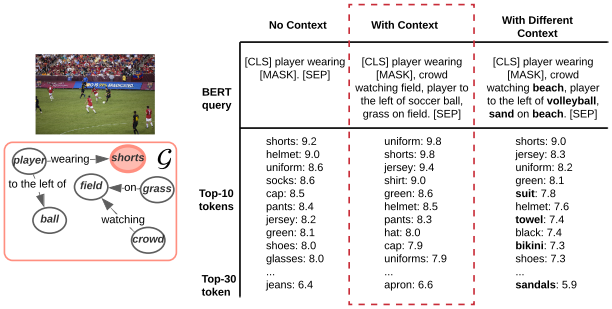


Figure 11. Example of BERT-based evaluation. The evaluation in a dashed rectangle is the one we used in the paper. The other two options are shown for comparison. 'No context' returns high scores for implausible tokens, such as 'glasses'. Using the context is important for more correct evaluation, while changing the context affects the scores accordingly. The scene graph is taken from original Visual Genome [38] for visualization purposes.

graph. To achieve that, we train a simple graph convolution network (GCN) to predict a single bounding box $\hat{b}_i$ given the rest of the layout similar to autoregressive models: $\hat{b}_i = f(\mathcal{G}, B, i)$, where the input coordinates of the i-th object in B have values $b_i = [-1, -1, -1, -1]$ (Fig. 12). Our f is similar to the GCN in our GAN pipeline and is based on GraphTripleConv, but has 3 layers with 64 hidden units and an additional MLP which predicts 4 coordinates of the bounding box. The model is trained with a margin l_1 loss: $\mathcal{L}_{\text{box}} = \min(0.5S, \max(0.05S, |\hat{b}_i - b_i|_1))$, where S is the maximum box coordinate. The idea behind this loss is to avoid having too large or too small penalties making the training more stable given that there is no single correct

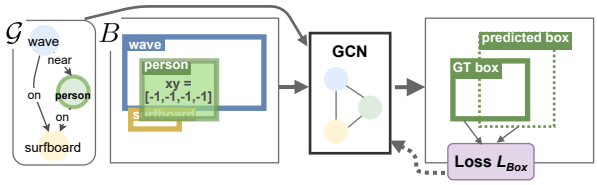


Figure 12. Overview of our model learning to predict bounding boxes, which is used only to report results in the last row of Table 8.

Table 9. Evaluation of the bounding box prediction model. FD is the Fréchet distance (lower is better). IoU denotes percentage of times the intersection over union between the predicted and GT box is $\geq 50\%$ (higher is better).

MODEL	FD test/test-zs	IoU (%) test/test-zs
No GCN, unconditional (sample GT)	0.001 / 0.001	1.8 / 2.0
GCN, predict b_i (corrupted label in $\mathcal{G}$)	0.019 / 0.034	2.8 / 2.2
GCN, predict b_i	0.018 / 0.037	12.9 / 6.3
GCN, GT b_i	$1e{-}4$ / $3e{-}4$	100 / 100

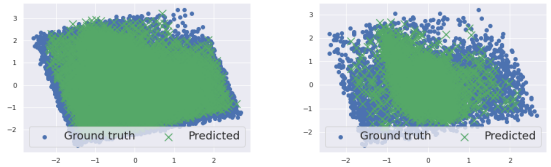


Figure 13. PCA-projected distributions of GT and predicted bounding boxes of the test **(left)** and test zero-shot **(right)** sets.

prediction in this task (i.e. besides the GT there can be many other plausible coordinates).

During training and for evaluation (Table 9), we randomly (uniformly) sample a single object in a scene graph and predict its coordinates given other bounding boxes and a SG. After the model is trained, we sequentially apply it only to perturbed nodes keeping the boxes of non-perturbed nodes unchanged. To isolate the effect of perturbation methods, we evaluate this model using ORACLE-ZS (Table 8). Surprisingly, this model performs worse on all metrics compared to both keeping the layout unchanged and using test bounding boxes.

We estimated the quality of the bounding boxes predicted by our GCN (Table 9). We found that the model respects the conditioning: results of IoU are significantly better compared to the case when we simply sample GT boxes from a distribution for a given class (ignoring SG and other boxes) or when we condition the model on a corrupted (random) label. However, we found that the model performs significantly worse when is fed with ZS compositions. In particular, the distribution appears to be more concentrated around the mean in such cases (Fig. 13). This might explain relatively poor SGG results when we attempt to use the box prediction model in combination with ORACLE-ZS perturbations. Further research is required to resolve the challenges of reliably predicting the layout for rare compositions similarly to [8].

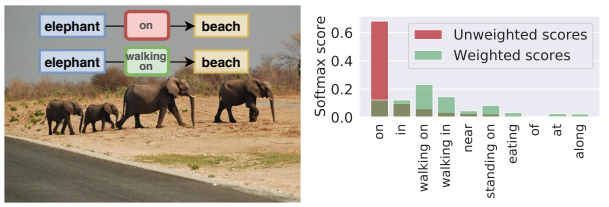


Figure 14. Example of the prediction for a zero-shot triplet. Weighting predicate scores increases the score of more descriptive predicates such as 'walking on' compared to 'on', improving mR results, which aligns with [67].

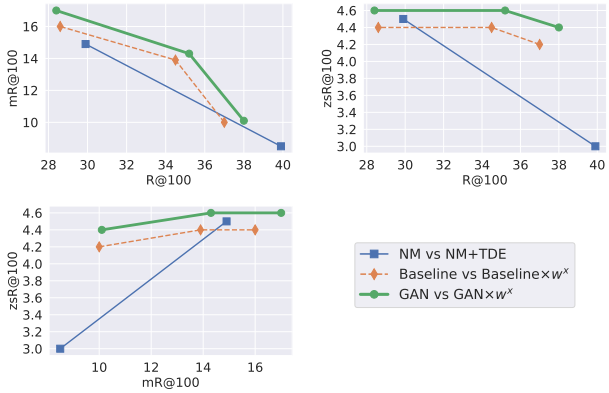


Figure 15. Trade-off between different metrics (R, mR and zsR) in the SGCls task depending on the strength ($x = [0, 1, 2]$) of weighting predicate scores ($\times w^x$) or using TDE [67].

B.4. Predicate Reweighting and Mean Recall

[67] showed that mean (over predicates) recall is relatively easy to improve by standard Reweight/Resample methods, while ZS recall is not. To further analyze this and highlight the challenging nature of compositional generalization (targeted in this work) as opposed to predicate imbalance typically targeted in some previous work, we simplify and extend the Reweight idea from [67]. In particular, we compute the frequency f_r of each predicate class r in the training set. Then, given a *trained* SGG model, we multiply softmax scores of predicates by $w^x = 1/f_r^x$, where x controls how much we need to balance predicate scores (Fig. 14). We consider $x = [0, 1, 2, 3]$, denoting models as $\times w^x$, where $x = 0$ corresponds to not changing predictions (our default setting). The advantage of our post-hoc calibration compared to Reweight [67] is that it does not require retraining the model and allows us to balance performance on frequent and infrequent predicates by carefully choosing x.

The results indicate a severe trade-off between the mean and all-shot recalls (Table 10). We can achieve very competitive mean recall results by simply using large x. However, this dramatically hurts the all-shot recall, potentially leading to invalid predictions, such as "cup walking on table" instead of "cup on table", as discussed in [76]. Overall, given the three SGG metrics (R, mR and zsR), we observe that (R, mR) and (R, zsR) are two conflicting pairs of metrics, while (mR,

Table 10. Tuning for mean (over predicates) recall versus all-shot recall using our GAN+GRAPHN model (the graph constraint evaluation). Underlined are top results among the variants of our model. †The results in [67] are obtained with a more advanced detector and, thus, are not directly comparable.

MODEL	MEAN RECALL		ALL-SHOT RECALL	
	SGCls	PredCls	SGCls	PredCls
GPS-net [44]	12.6	21.3	**40.1**	**66.9**
NM† [67]	8.5	14.6	39.9	66.0
NM+TDE† [67]	14.9	25.5	29.9	46.2
PCPL [76]	**19.6**	**35.2**	28.4	50.8
No weighting ($\times w^0$)	10.1±0.2	15.7±0.3	<u>38.1</u>±0.1	<u>62.3</u>±0.2
$\times w^1$	14.3±0.2	22.1±0.3	35.2±0.1	56.9±0.3
$\times w^2$	17.0±0.1	26.3±0.1	26.4±0.4	42.5±0.2
$\times w^3$	<u>18.3</u>±0.1	<u>28.6</u>±0.2	17.2±0.1	27.1±0.2

zsR) have some correlation (Fig. 15). Clearly, the discussed metrics measure different properties of a given SGG model. In this work, we measure compositional generalization and therefore focus on zero-shot (and closely related few-shot) recall as appropriate metrics for that particular task.

B.5. Training the Baseline Longer

In our GAN model we have two loss terms ($\mathcal{L}_{\text{CLS}}, \mathcal{L}_{\text{REC}}$) that update the model F, which can be considered as having two times more updates compared to the baseline, and so can be an implicit advantage. We investigated if the baseline model can improve itself if it is updated two times more. Our results in Table 11 suggest that the model starts to overfit leading to poor results on all metrics except mean recall. This might indicate that a stronger regularization is required for the baseline trained longer. Our GAN losses can be viewed as such a regularizer leading to better generalization results.

Table 11. Effect of training the baseline model longer. Metrics are R@100 for SGCls and R@50 for PredCls (no graph constraint).

MODEL	ZERO-SHOT		MEAN RECALL		ALL-SHOT	
	SGCls	PredCls	SGCls	PredCls	SGCls	PredCls
Baseline (IMP++)	9.3±0.1	28.1±0.1	27.8±0.1	33.7±0.3	48.7±0.1	77.5±0.1
Baseline, ×2 updates	8.2	25.8	27.8	35.1	46.6	73.5

B.6. SGGen/SGDet Results

In addition to SGCls/PredCls results presented in the main text, we follow previous work [82, 67] and provide **SGGen** (SGDet) results that rely on using the bounding boxes predicted by the detector as opposed to using GT boxes (Table 12). We follow a standard refinement procedure [82, 36] and fine-tune SGCls models with $bs = 6$ and $lr = (1e-4) \times bs$. We fine-tune for 2 epochs in all cases, which we set heuristically as our main goal in this task is not to outperform state-of-the-art, but to compare our model with the baseline. During this fine-tuning for simplicity we use only the baseline loss for all models, so that this step's main purpose is to adapt the SGCls model to predicted bounding

Table 12. SGGen results. The top-1 result in each column is **bolded**. †The results in [67] are obtained with a more advanced detector and, thus, are not directly comparable.

MODEL	ZERO-SHOT		MEAN RECALL		ALL-SHOT	
	SGGen **zsR**@100		SGGen **mR**@100		SGGen **R**@100	
Graph Constraint	✓	✗	✓	✗	✓	✗
FREQ [82, 9, 36]	0.0	0.1	5.6	8.9	27.6	30.9
IMP+ [74, 82, 36]	0.9	0.9	4.8	8.0	24.5	27.4
NM [82, 9, 36]	0.3	0.8	6.1	12.9	30.3	**35.8**
KERN [9, 36]	0.0	0.0	7.3	**16.0**	29.8	**35.8**
VCTree† [67]	0.7	–	6.9	–	36.2	–
NM† [67]	0.2	–	6.8	–	**36.9**	–
NM+TDE† [67]	**2.9**	–	9.8	–	20.3	–
Baseline (IMP++)	0.8	0.9	5.9	9.8	25.1	28.1
GAN	1.0	1.2	6.2	10.5	25.4	28.6
GAN$\times w^2$	1.1	**2.2**	**9.9**	15.9	18.5	24.0

boxes instead of ground truth. Despite these simplifications our GAN model still improves on the baseline. However, the results are worse than in previous works, perhaps because: (1) they rely on a stronger detector; (2) we do not update the detector on the generated features conditioned on rare compositions. Addressing these limitations can be the focus of future work. The models with reweighted predicates (§ B.4) significantly improve the results on mean recall (as expected), and interestingly, on one of the zero-shot metrics.

B.7. Evaluation of Generated Visual Features

In addition to the quality estimation of node features in the main text, we evaluate edge and global features qualitatively (Fig. 16) and quantitatively (Table 13). We also visualize features by averaging over the channel dimension (Fig. 17, 18). These visualizations show that the generated samples are diverse and respond to the changes in the conditioning. However, the global features generated by our GAN are noticeably more smooth, which might be an effect of the generator's architectural inductive bias or visualization strategy.

Table 13. Evaluation of generated (fake) feature using the metrics of "similarity" between two distributions X and Y [39, 51]. The same held-out set of real test features ($Y \sim V$) is used as the reference distribution in all cases. The percentage in the superscripts denotes a relative drop of the average metric when switching from test to test-zs conditioning. For all metrics, higher is better.

	DISTRIBUTION X	Fidelity (realism) PRECISION	DENSITY	Diversity RECALL	COVERAGE	AVG
NODES	Real test	0.74	1.02	0.75	0.97	0.87
	Real test-zs	0.66	0.99	0.70	0.94	$0.82^{-6\%}$
	GAN: Fake test	0.55	0.77	0.42	0.82	0.64
	GAN: Fake test-zs	0.47	0.60	0.41	0.75	$0.56^{-13\%}$
EDGES	Real test	0.73	0.97	0.72	0.97	0.85
	Real test-zs	0.53	0.99	0.59	0.87	$0.75^{-12\%}$
	GAN: Fake test	0.54	0.59	0.50	0.75	0.60
	GAN: Fake test-zs	0.38	0.36	0.50	0.58	$0.46^{-23\%}$
GLOBAL	Real test	0.30	0.96	0.31	0.99	0.64
	Real test-zs	0.20	0.91	0.35	0.73	$0.55^{-14\%}$
	GAN: Fake test	0.14	0.43	0.22	0.83	0.41
	GAN: Fake test-zs	0.11	0.23	0.29	0.68	$0.33^{-20\%}$

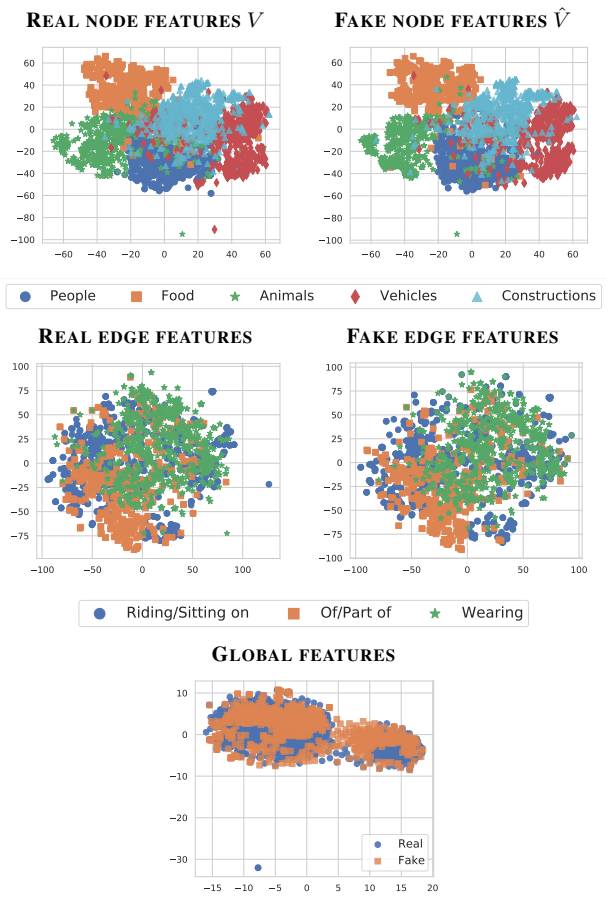


Figure 16. Visualizations of real *vs* generated node, edge and global features obtained using t-SNE.

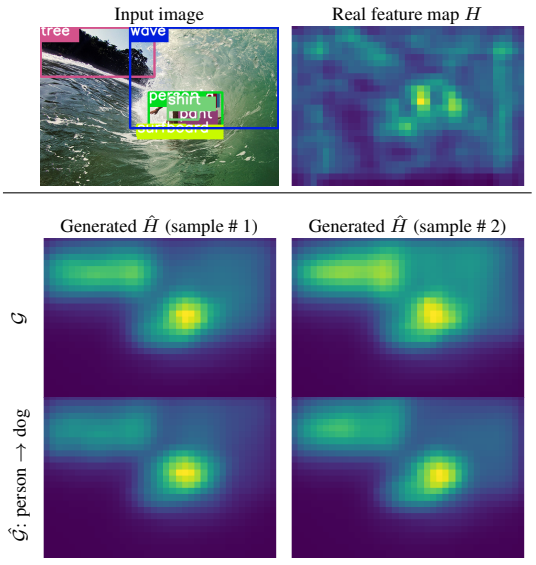


Figure 17. Real H and Fake (generated) $\hat{H}$ global feature maps (averaged over the channel dimension) with a perturbation for the triplet "person on surfboard" in $\mathcal{G}$.

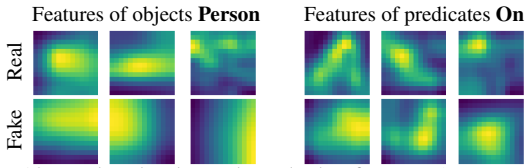


Figure 18. Real and Fake node and edge features (averaged over the channel dimension) for objects and predicates.

B.8. Limitations

Our method is limited in three main aspects. **First**, we rely on a pretrained object detector to extract visual features. Without generating augmentations all the way to the images — in order to update the detector on rare compositions — it is hard to obtain significantly stronger performance. While augmentations in the feature space can be effective [16, 71], their adoption for large-scale out-of-distribution generalization is underexplored. **Second**, by making a simplification and keeping GT bounding boxes for perturbed scene graphs, we limit (1) the amount of perturbations we can make (if we permit many nodes to be perturbed, then it is hard to expect the same layout), and (2) the diversity of spatial compositions, which might be an important aspect of compositional generalization. We attempted to verify that using ORACLE-ZS perturbations, which are created by directly using ZS triplets from the test set. Using ORACLE-ZS with GT bounding boxes (our default setting) surprisingly does not result in large improvements. However, when we replace GT boxes with the ones taken from the corresponding samples of the test set, the results improve significantly. This demonstrates that: (1) our GAN model may benefit from reliable bounding box prediction (e.g. [28]); (2) GRAPHN perturbations are already effective (close to ORACLE-ZS) and improving the results further by relying solely on perturbations is challenging. **Third**, the quality of generated features, especially, for novel and rare compositions is currently limited, which is also carefully analyzed in [8]. Addressing this challenge can further improve results both of ORACLE-ZS and non-ORACLE-ZS models.

C. Additional visualizations

Figures 19-21 show additional examples of different perturbations applied to scene graphs from Visual Genome (on top of each figure we show graph $\mathcal{G}$ along with the corresponding image). Perturbed nodes are highlighted in red. Red edges denote triplets missing both in the training and test sets. Thick red edges denote triplets only present in the test set, i.e. zero-shots. Blue edges denote triplets present in the training set, where the number indicates the total number of such triplets in the training set. These visualization show that in case of RAND, most of the created triplets are implausible as a result of random perturbations. NEIGH leads to very likely compositions, but less often provides rare plausible compositions. In contrast, GRAPHN can create plausible compositions that are rare or more frequent depending on α.

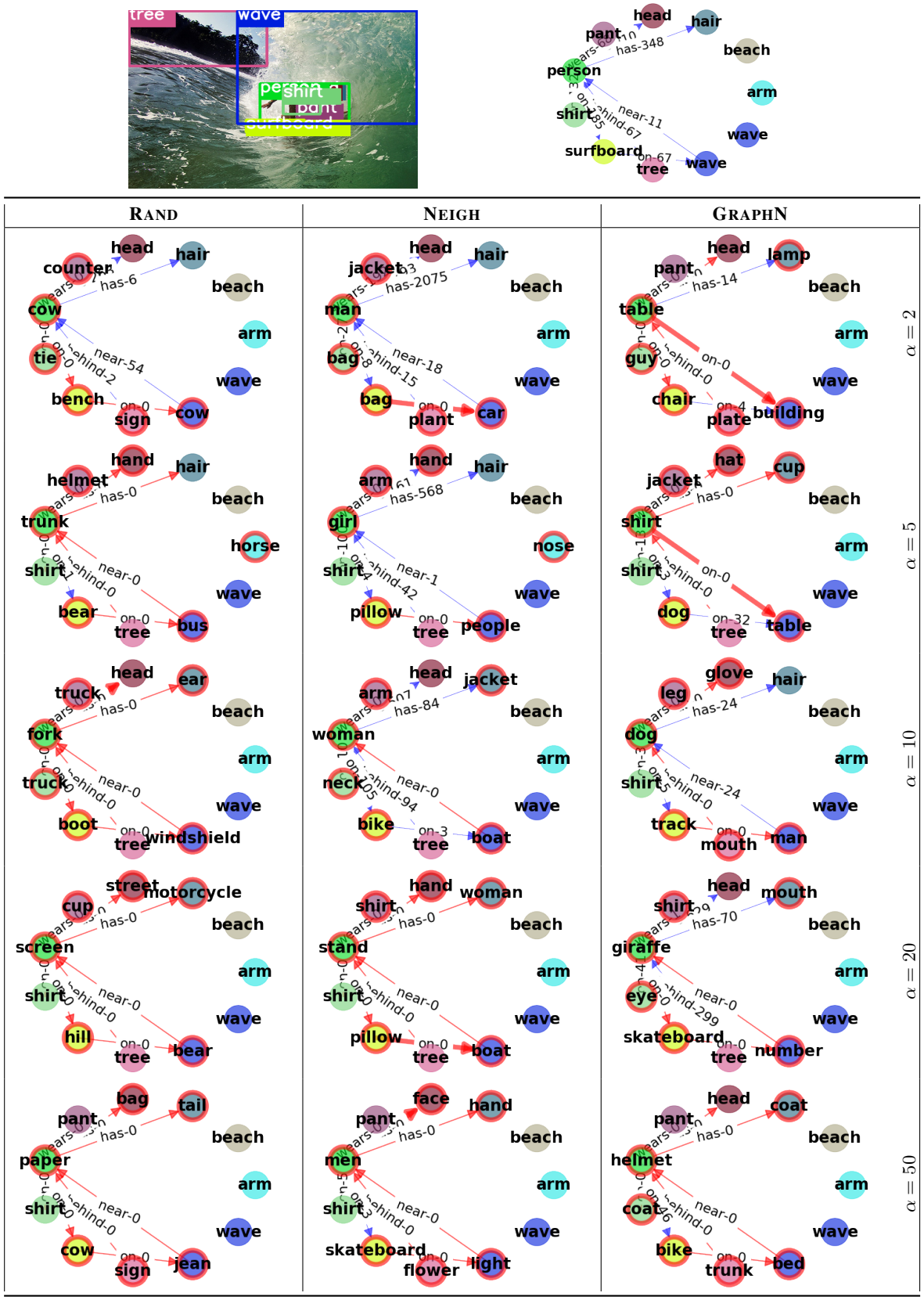


Figure 19. Visualizations of perturbations for image 2350517 from Visual Genome.

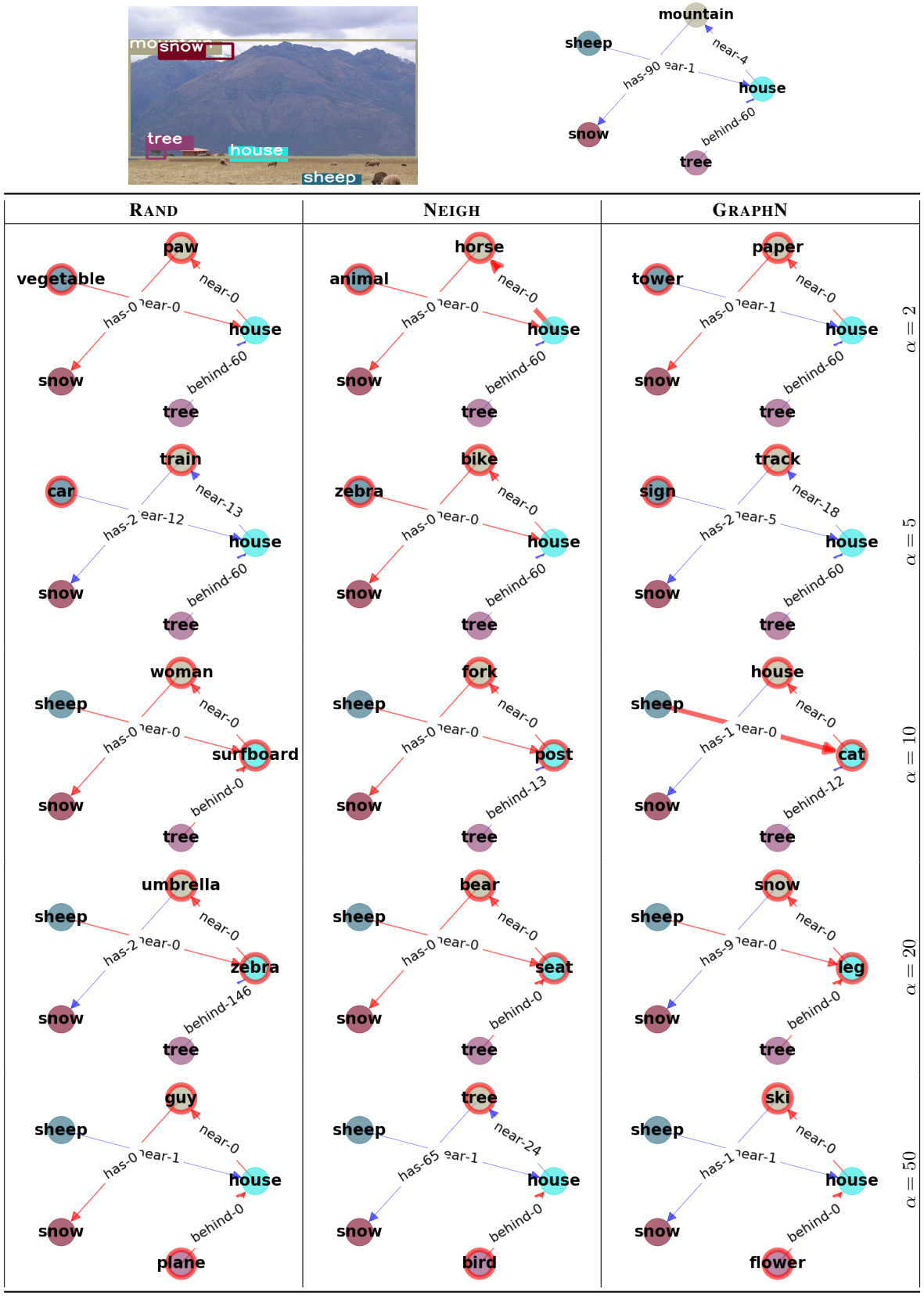
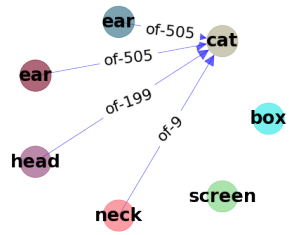


Figure 20. Visualizations of perturbations for image 2343590 from Visual Genome.



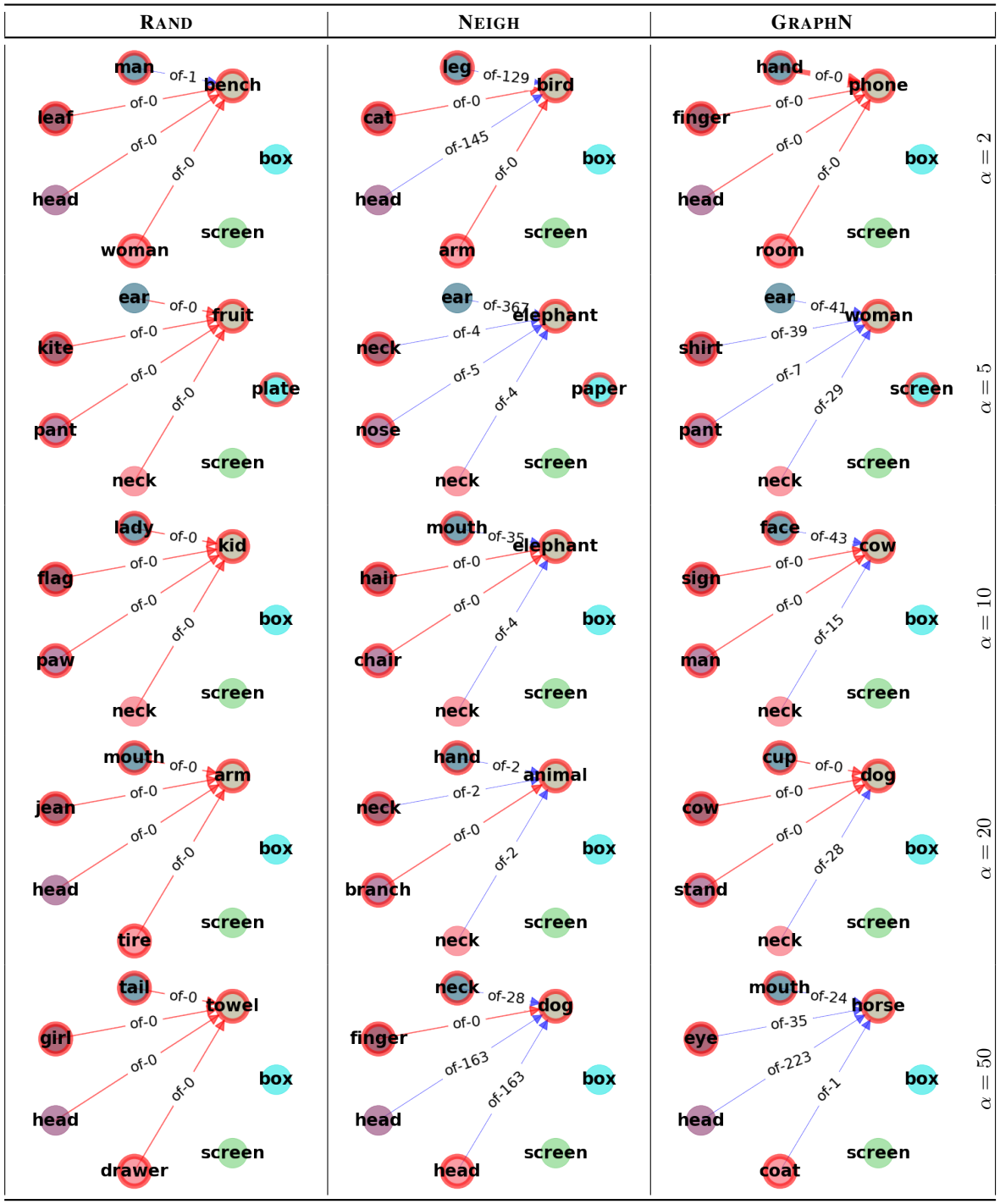


Figure 21. Visualizations of perturbations for image 1159620 from Visual Genome.