Bridging Knowledge Graphs to Generate Scene Graphs: Supplementary Material

Alireza Zareian, Svebor Karaman, and Shih-Fu Chang

Columbia University, New York NY 10027, USA
{az2407,sk4089,sc250}@columbia.edu

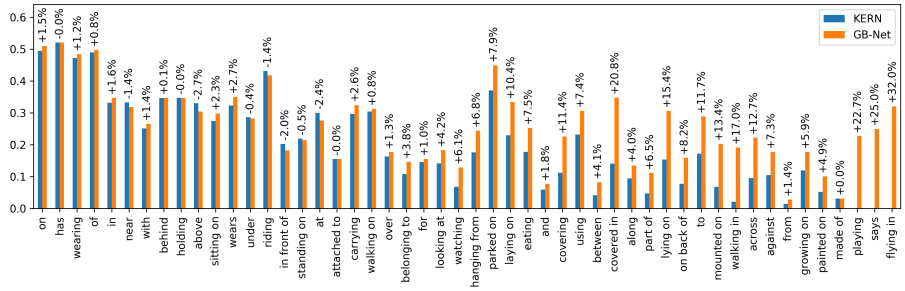

Fig. 1. Comparison of our method GB-Net with KERN [1] in terms of recall at 50 per predicate class, without graph constraint. The horizontal axis was ordered decreasingly based on frequency in VG.

In this document, we provide additional details that were omitted from the main manuscript due to the space constraint. We start by an analysis of our quantitative results, were we show our method addresses the shortcoming of the state of the art in modeling the tail of the predicate distribution. We then report an empirical analysis of the computational cost of our method, showing it is significantly faster than the state of the art, while also more accurate. After that, we clearly describe the process of creating our commonsense graph, and we provide a set of qualitative examples to illustrate how our method exploits such commonsense knowledge. We conclude by describing the code that we will provide to reproduce the results.

1 Per-class performance

Figure 1 illustrates the recall of our method for each predicate class separately, where predicates are ordered decreasingly according to their frequency in the training set. While state-of-the-art methods such as KERN [1] obtain much lower performance on the tail of the distribution, our method significantly improves

Table 1. Time and memory cost of our method compared to the state of the art

Method	Test time (sec/image)	Train time (min/epoch)	# parameters (million)
KERN [1]	0.79	401.2	405.2
GB-Net	0.52	191.6	444.6

the performance of the tail without losing on the frequent predicates, resulting in a more reliable and consistent performance overall.

2 Computational cost

We compute the training and test speed of our method and compare to KERN [1] using identical hardware, with one GPU of type NVIDIA GeForce GTX 1080 Ti with 11 gigabytes of memory, and summarize the results in Table 1. Perhaps the most important aspect of computation is the run time when deploying the model on new images. To this end, we run each trained model on the entire test set of Visual Genome (VG), *i.e.* 26446 images, and get the average run time over all images in terms of seconds. Our method is 34% faster than the state of the art, while being significantly more accurate as demonstrated in Table 1 of the main paper.

Another important factor is the duration of training. We record the time it takes to train each model on one epoch of the VG training set, *i.e.* 56224 images, and get the average over 10 training epochs. As Table 1 shows, our method is more than twice faster than the state of the art. One of the reasons is that KERN has two stages of message passing, each with three steps, first to infer entities, and then to infer predicates, while our method infers both entities and predicates jointly, through 3 steps of global message passing.

Finally, we compare the number of trainable parameters each method has. Our method has 10% more parameters than KERN, while it is 52% and 34% faster than KERN during training and test respectively. Note that in all methods, 139.8 millions of the parameters belong to the Faster R-CNN detector, which we fix while training for scene graph generation.

3 Commonsense graph construction

Our method utilizes a background knowledge graph which encodes commonsense knowledge about the target entity and predicate classes. Such information have been proved to be effective for scene graph generation [1, 3]. Intuitively, this is because they can help the model disambiguate between possible visual classes, using higher-level semantic meanings and relationships of classes. For instance, object affordance is a type of commonsense, because a typical human knows a `Bike` can be used for `riding`. This fact can be used by the model when it

perceives a bounding box of `Person` above a bounding box of `Bike`, and it is not sure whether to classify their relationship as `riding` or `onTopOf` or `mountedOn`.

Commonsense does not have to be homogeneous. Besides affordance, ontological hierarchy is another type of useful information for scene graph generation. A typical human knows `Man` and `Woman` are both subtypes of `Person`, and hence can generalize the properties of `Man` to `Woman`. Another example is statistical commonsense, where an average human knows a `Vase` is more likely to be `on` a `Table` than anywhere else, not because of its semantic characteristics, but because it is usually that way.

Our method is independent of the content of the commonsense graph, as long as its nodes are exactly the set of classes required by the target task, and every fact is encoded as a directed, typed, weighted edge. Accordingly, our commonsense graph has 150 entities and 50 predicates as conventionally used in the SGG task, and we compile the edges (*i.e.* facts) from a variety of sources, to make it as rich and comprehensive as possible. To this end, we first manually find for each of the 200 VG classes, the WordNet Synset [6] that represents its meaning. This step is required to fix a deterministic meaning for each class, such that for instance, `Cabinet` always represents the furniture piece and is not confused with its meaning in politics.

Given the 200 nodes that are grounded on WordNet, we start collecting commonsense facts to encode as edges. The most prominent repository of commonsense knowledge is ConceptNet [5], which is a large-scale graph where nodes are *concepts*, including but not limited to entity and predicate classes, and edges encode relational facts about pairs of concepts. There are over 21 million edges of 34 types connecting over 8 million nodes in ConceptNet. We query each of our 200 nodes to find its semantically closest node in ConceptNet, by using the WordNet Synset title and the ConceptNet search API. Then we query all ConceptNet edges between each pair of those 200 nodes, and further manually clean and prune those edges, leading to 104 edges of 5 types. Those 5 edge types are: `partOf` as in `Hand-partOf-Person`, `relatedTo` as in `Lamp-relatedTo-Light`, `isA` as in `Dog-isA-Animal`, `mannerOf` as in `MountedOn-mannerOf-AttachedTo`, and `usedFor` as in `Bike-usedFor-Riding`. Further, for asymmetric relationships (all except `relatedTo`), we create a reverse edge with a distinct type. For instance, because we have `Hand-partOf-Person`, we also create the fact `Person-hasPart-Hand`. Our message passing framework, unlike conventional ones, only propagates messages in the direction of each edge, and not backwards. Hence, the way the `Person` node is affected by `Hand` would be different from how `Hand` is affected by `Person`, because we do not share parameters across edge types. Since there is no edge confidence in ConceptNet, these edges all have a binary weight (1.0 if exists and 0.0 if not).

Besides ConceptNet, we also use WordNet to get ontological similarity of words. Although WordNet is not originally identified as a commonsense graph, the knowledge we extract from it is trivial, generally known information, and hence can be considered commonsense. Inspired in part by [4], we use three similarity metrics of the WordNet API (namely path similarity, Leacock Chordorow

(LCH) similarity, and Wu-Palmer (WUP) similarity [6]), and a manually tuned threshold for each, to determine whether two entity classes are relevant or not. This is encoded in the edge type `WordNetSimilarTo` with binary weights. This strategy does not work well for predicate classes, so this edge is only between pairs of entities.

Finally, we use the VG training set to get co-occurrence statistics between categories, inspired by [7] but in a more comprehensive manner. We estimate conditional probabilities of subject given predicate, object given predicate, predicate given subject, predicate given object, subject given object, and object given subject, as well as the correlation of entity classes as they connect to the same predicate, and the correlation of predicate classes as they connect to the same entity. These edge types capture a variety of statistical interactions between classes. Each of those statistics result in a pairwise matrix, which we then sparsify by keeping the top 5 element in each row and setting the rest to zero. For instance, `riding` is connected to its top 5 most likely objects, namely `Horse`, `Bike`, `Skateboard`, `Motorcycle` and `Wave`. Note that although `riding` is not connected to `Elephant`, its 6th most likely object, `Elephant` and `Horse` are both connected to `Animal` through `isA` edges, and they are connected to each other through `WordNetSimilarTo`, allowing our message passing framework to exploit this rich structure and infer `Elephant` can be ridden too.

Overall, these three sources lead to 19 edge types (including backward edge types for asymmetric relationships). Carefully compiled from multiple sources, our commonsense graph is more sophisticated and complete, compared to those made for recent knowledge-aware computer vision systems such as [2] and [4]. Note that none of those papers publicized their knowledge graph, so we were unable to compare. The process of graph generation involves manual effort, thus we have made our commonsense graph publicly available as a part of our code.

4 Qualitative results

To demonstrate the performance of our method qualitatively, Figures 2-14 show examples of scene graphs generated by our method, compared to the ones generated by KERN. These examples illustrate how our method predicts more commonsensical graphs despite visual ambiguities in the scene. We observed several patterns in which our method outperforms KERN. Since KERN (and most other SGG methods such as [7]) first classify each entity and then classify predicates, they are unable to utilize predicate semantics to enhance entity classification. Thus in many cases, KERN misclassifies an entity, due to visual ambiguity and clutter, while our method makes the correct prediction that might be less apparent visually, but lead to a more consistent scene graph semantically. In some other cases, KERN misclassifies an entity not because it is visually cluttered, but because the bounding box is too loose and covers a big portion of background. Our method is more robust to such bounding box inaccuracies, resulting a higher overall performance. Finally, in some cases entities are classified correctly by KERN, but the choice of predicates is inappropriate. Our method usually picks

the correct predicate, in accordance to commonsense knowledge, such as object affordances. More detailed discussion can be found in each figure's caption.

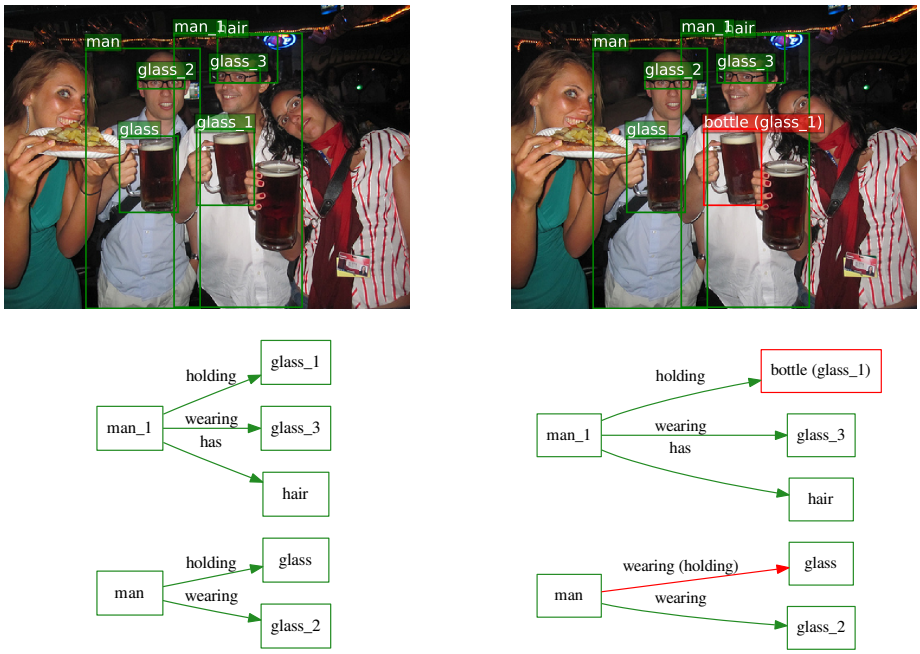

Fig. 2. Example comparison of our method GB-Net (left) with KERN [1] (right). Misclassified entities and predicates are colored red, and the correct class is included in parentheses. This is a challenging image with 4 occurrences of "glass" with two different meanings (eyeglasses and beer glass). Our method is able to choose the appropriate relation (wearing or holding) for each instance. KERN mistakes a glass for a bottle and predicts a "wearing" relation between a man and his drink.

5 Software package

We will provide a software package that reproduces every single reported number, *i.e.*, all numbers in Table 1 and 2 of the main paper. To make it easy to reproduce, we provide an IPython Notebook for each experiment. We also provide a README file with a step by step guide, as well as a mapping between the notebook files and table cells in the paper. To reproduce the results from scratch, one would run the training notebook of each experiment followed by the evaluation notebook. To bypass training, we provide all the parameter checkpoints through a link in the README. This way the readers only need to run

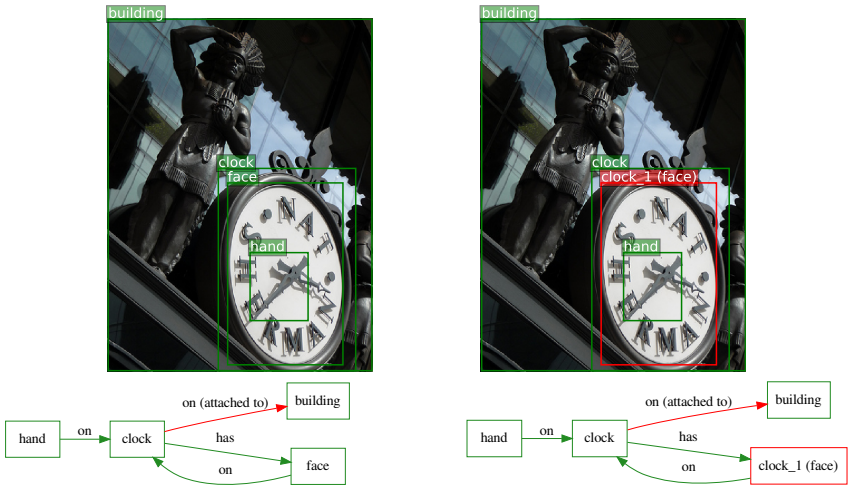

Fig. 3. Example comparison of our method GB-Net (left) with KERN [1] (right). The concept of a clock face is challenging for KERN but our method can produce such output, by exploiting the prior knowledge and statistics that clocks can have faces and the face would be on the clock. KERN predicts the triplet clock has clock, which does not make sense.

the evaluation notebook. In case a GPU is not available for deploying the model, we also provide a link to the pre-computed model outputs in the README. Finally, the notebooks already contain the saved evaluation results within them, which can be checked without running evaluation at all.

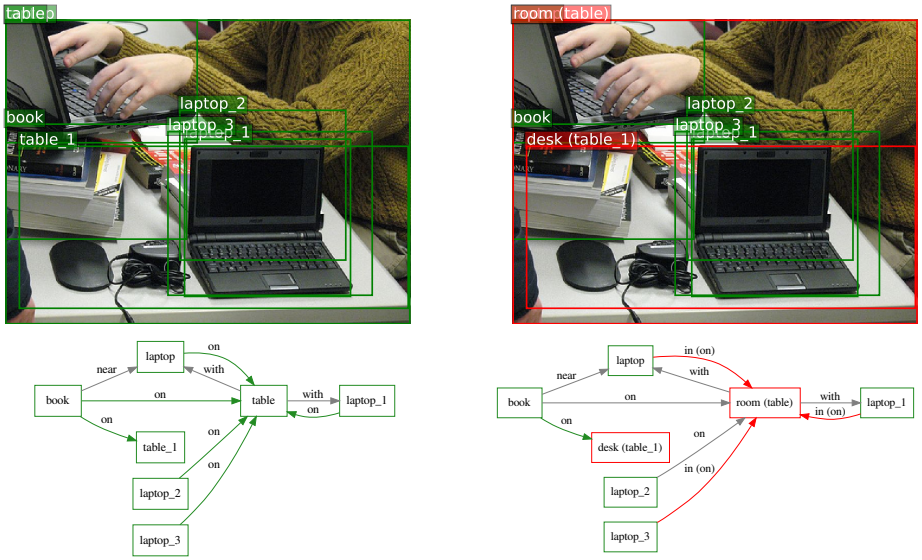

Fig. 4. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the table as a room, possibly because the bounding box contains the entire scene, but this leads to incorrect triplets such as laptop on room. Our method predicts the more appropriate class table, that makes every triplet more commonsensical.

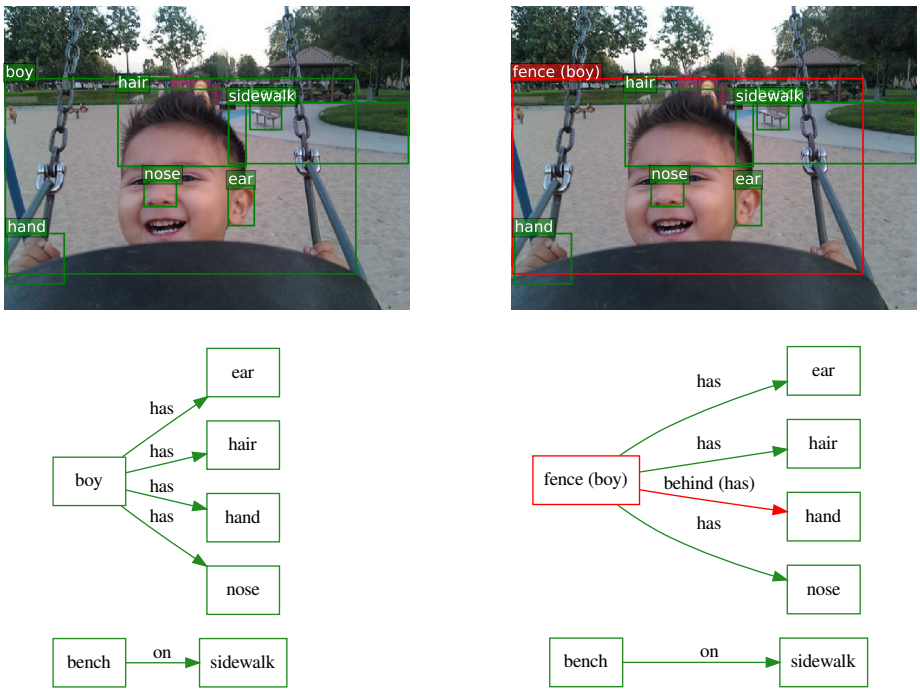

Fig. 5. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the boy as fence, which leads to the nonsensical triplets fence has ear, fence has nose, etc. Our method is less likely to make such meaningless predictions.

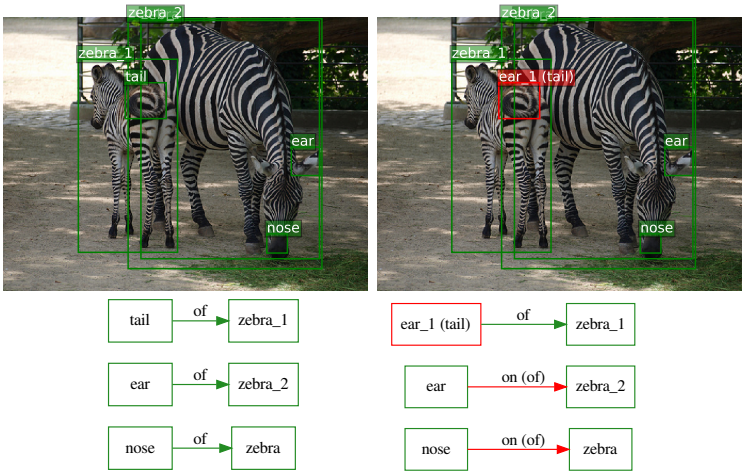

Fig. 6. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN predicts triplets such as ear on zebra and nose on zebra, etc., while our method predicts more semantically sound triplets ear of zebra and nose of zebra, reflecting the ownership relationship between the zebra and its body parts.

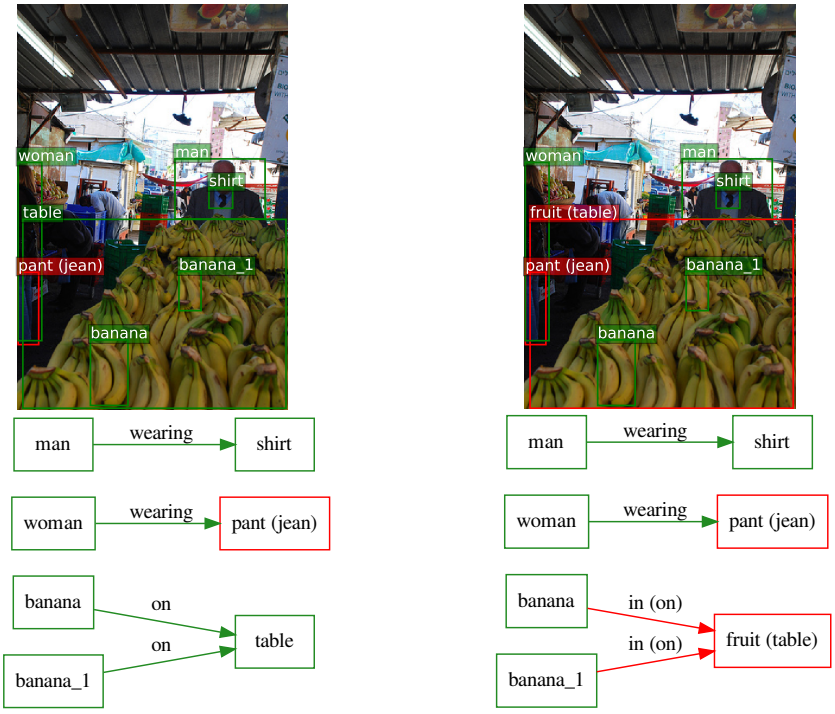

Fig. 7. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the table as fruit, possibly because it is entirely covered by fruites. But this leads to nonsensical triplet banana in fruit. Our method correctly classifies the table, which leads to a more commonsensical scene graph.

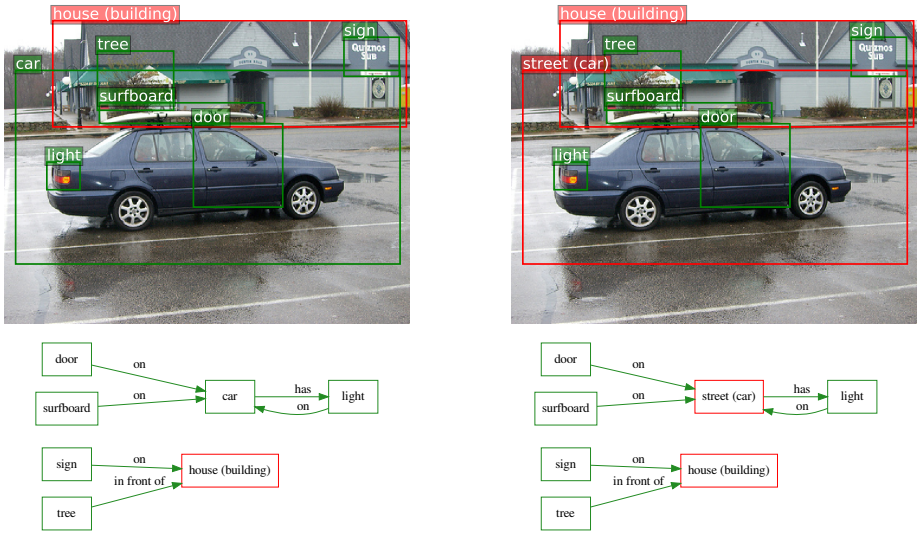

Fig. 8. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies car as street, possibly because the bounding box is too loose and contains a large portion of the street. Our method is aware that door on street is not commonsensical, and hence predicts the more appropriate choice, *i.e.* car.

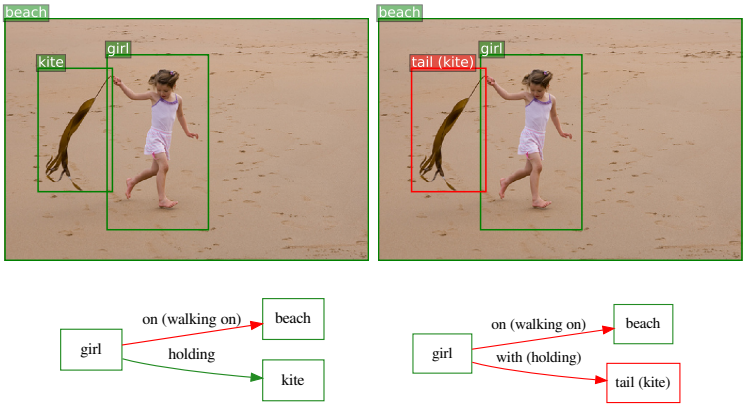

Fig. 9. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the kite as a tail, because it actually looks more like a tail. Our method predicts kite that is visually less clear, but leads to a more commonsensical graph overall.

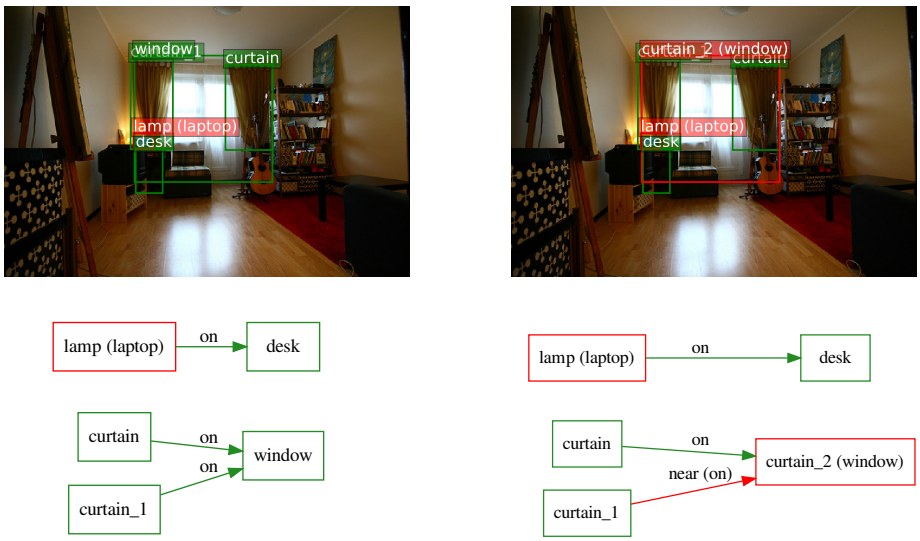

Fig. 10. Example comparison of our method GB-Net (left) with KERN [1] (right). Our method correctly detects the two pieces of curtain on window, while KERN predicts the less appropriate triplet curtain on curtain, possibly because the bounding box of the window contains the curtain as well.

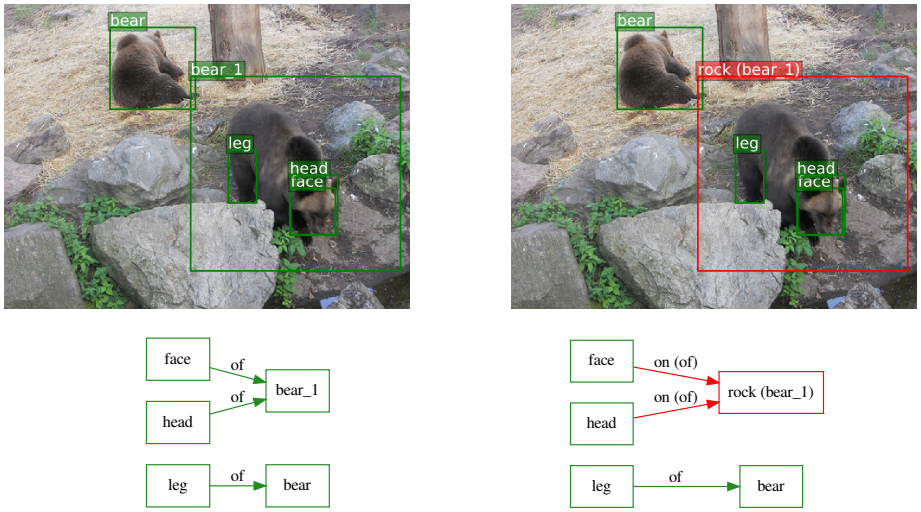

Fig. 11. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the bear as rock, possibly due to the too loose bounding box that includes rocks as well. This leads to nonsensical triplets such as face on rock and head on rock, while our method produces more likely and accurate triplets face of bear and head of bear.

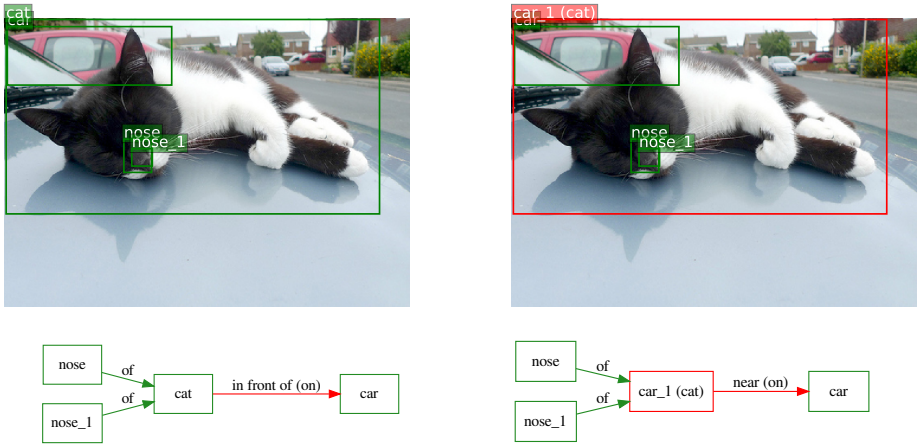

Fig. 12. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the cat as a car, possibly because the bounding box is too loose and covers a large area of both cars. Our method exploits the fact that cars are unlikely to have noses.

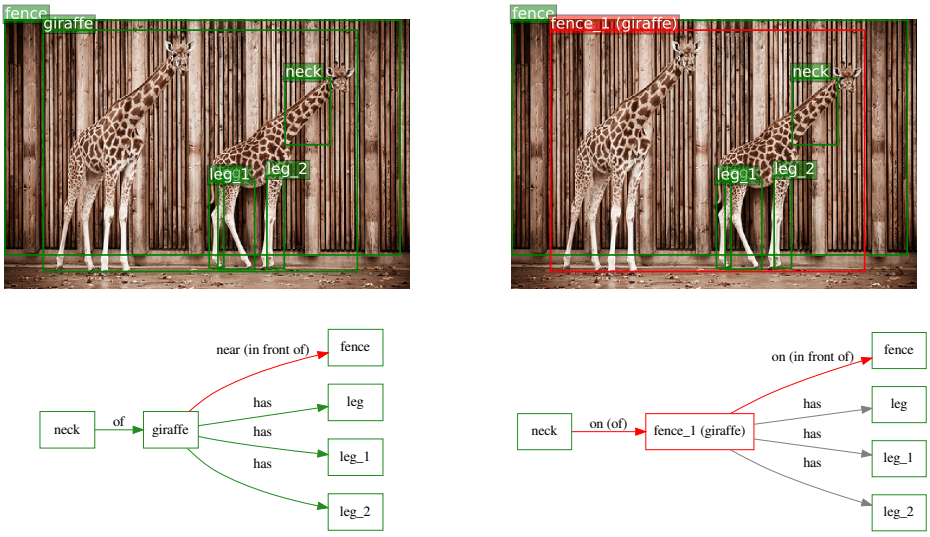

Fig. 13. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies the giraffe as a fence, leading to nonsensical triplets such as fence on fence, fence has leg, etc. Our method avoids such inappropriates compositions.

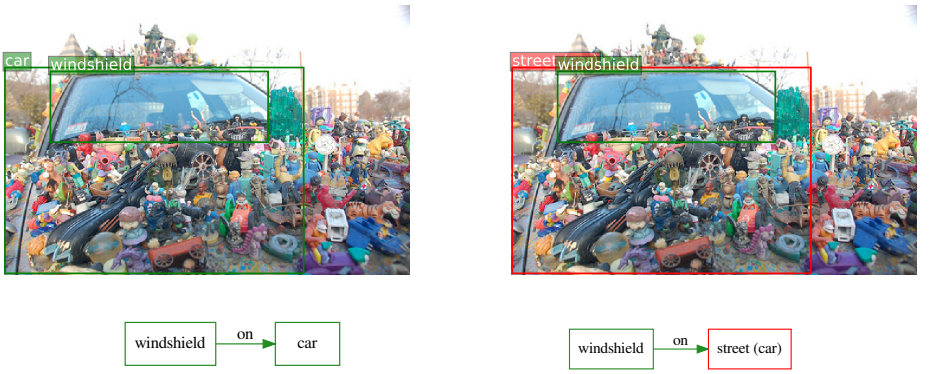

Fig. 14. Example comparison of our method GB-Net (left) with KERN [1] (right). KERN misclassifies car as street due to the extreme occlusion, while our method exploits the fact that cars are more likely to have windshields than streets.

References

1. Chen, T., Yu, W., Chen, R., Lin, L.: Knowledge-embedded routing network for scene graph generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6163–6171 (2019)
2. Chen, X., Li, L.J., Fei-Fei, L., Gupta, A.: Iterative visual reasoning beyond convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7239–7248 (2018)
3. Gu, J., Zhao, H., Lin, Z., Li, S., Cai, J., Ling, M.: Scene graph generation with external knowledge and image reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1969–1978 (2019)
4. Kato, K., Li, Y., Gupta, A.: Compositional learning for human object interaction. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 234–251 (2018)
5. Liu, H., Singh, P.: Conceptnet—a practical commonsense reasoning tool-kit. BT technology journal **22**(4), 211–226 (2004)
6. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
7. Zellers, R., Yatskar, M., Thomson, S., Choi, Y.: Neural motifs: Scene graph parsing with global context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5831–5840 (2018)