A PSG Dataset Details

A.1 More comparisons between VG and PSG

More comparisons between VG-150 and PSG examples are shown in Fig. A1. In particular, we would like to highlight some specific advances in the PSG dataset from sub-figure (a), which readers can confirm from other sub-figures.

In (a), VG-150 does not contain key information of 'woman flying kite', and also has ambiguous relations like 'at'. Fortunately, PSG addresses the key information but with a more general predicate 'playing'. It is because in PSG, the predicate definition follows the rule of 'being representative with proper granularity, not too specific'. Refer to Sec. A.4 for more information. Also, PSG gathers far more comprehensive and accurate triplets.

For object groundings, it is noticeable that in VG, the grounding of 'beach' is inaccurate, which only covers half of the actual beach. It can be problematic since a successful recall of a triplet requires a correct matching (big IOU) between predicted groundings and ground truth, in addition to a correct classification of triplets. Therefore, an incorrect annotation on object grounding can cause an inaccurate evaluation on scene graph generation too. Apparently, object grounding of PSG is far more accurate than VG.

A.2 PSG Dataset Statistics

The PSG dataset has a total of 48,749 annotated images with 56 predicate classes, and 80 thing and 53 stuff classes (same as the COCO dataset [5]).

Here is a list of average statistics for each image:

- 11.0 instances per image
- 5.6 relations per image
- 1.9 (34%) thing-thing relations per image
- 1.2 (21%) stuff-stuff relations per image
- 2.5 (45%) thing-stuff relations per image

A.3 PSG Dataset Construction Details

Built on COCO and Visual Genome In order to create a dataset with both panoptic segmentation *and* relationship annotations, we took advantage of the overlap between the COCO [5] and Visual Genome datasets [3], where they share 48,749 images. Namely, for a given image, it has panoptic segmentation annotations from COCO, as well as scene graph annotations from Visual Genome. However, we cannot merge the two datasets directly as not only do they have different object annotations, they also define different object categories. Therefore, we attempted to conduct the following dataset merging process.

Fig. A1: More comparisons between VG-150 and PSG examples. Every subfigure has VG triplets and their groundings on the left and PSG on the right. Apart from precise pixel-wise grounding, PSG dataset gets rid of trivial relations (*e.g.*, `head-on-dog` in (d), `person-has-hair` in (e)), and keep salient ones (*e.g.*, `person-holding-remote` in (e)). Relations with background are also included (*e.g.*, `elephant-walking-on-grass`).

Merging COCO and Visual Genome Annotations Dataset merging requires solving two intermediate tasks, saying 1) *Object Category Matching:* to figure out a mapping between the object categories in COCO to the object categories in Visual Genome, and 2) *Object Instance Matching:* for each image, to find out which object annotations in COCO correspond to which object annotations in Visual Genome.

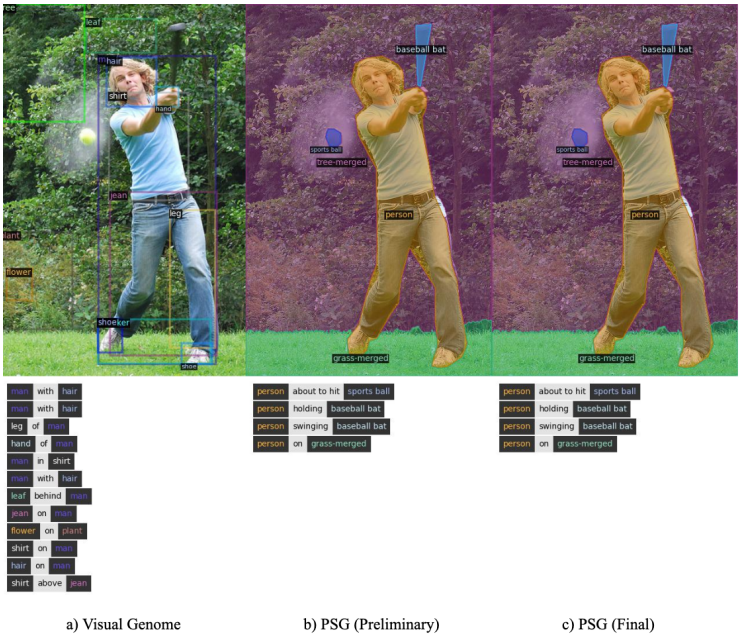

Fig. A2: PSG Dataset Constrution Process. The object categories and annotations in (a) VG-150 are different from that in COCO (image in (b) shows the COCO panoptic segmentation). After matching COCO objects to VG objects using the object matching process, any corresponding relationship annotations can also be transferred over from (a) to (b). These preliminary automatic but noisy relationship annotations are then sent to be processed by a final round of cleaning and annotation to produce the final PSG dataset in (c).

The goal of dataset merging is that, once the matching is achieved, we can transfer over the relationship annotations directly. However, since the matching process will not be completely perfect, we planned to bring these preliminary annotations to experienced and trained annotators for a final round of cleaning and annotation.

With a clear goal of dataset matching, we introduce the details of two intermediate tasks of object category matching and object instance matching. After the matching process, we can get a noisy preliminary PSG dataset (ref. Fig.A5(b)) that awaits the final cleaning.

Object Category Matching: For a given COCO object category, what object categories in Visual Genome does it correspond to? For example, "car" in COCO may be matched to both "car" and "vehicle" in Visual Genome.

We encode the text of each object category into a feature vector using fastText [1] word embeddings, and compute a similarity score for each (COCO, Visual Genome) object category pair using their cosine similarity.

This similarity score will be useful for matching object instances in COCO to that in Visual Genome, as described in the next section.

Object Instance Matching: The relationship annotations for each image in Visual Genome are tied to Visual Genome object annotations (object categories + bounding boxes), which are different from its corresponding COCO object annotations (object categories + panoptic segmentations). In order to transfer over the relationships to COCO object annotations, we can attempt to match each object instance as annotated in COCO, to an object instance as annotated in Visual Genome. Intuitively, if an object as annotated in COCO has a high overlap with an object as annotated Visual Genome, and their object categories are similar, they are likely to be referring to the same object. A sketch of the algorithm used to perform the matching is as follows. For each image, we:

1. compute the bounding box IoU of each object in COCO to each object in Visual Genome (using the tightest bounding box of the segmentation).
2. we then perform a greedy approach by always considering the instance pair with the highest IoU:
   (a) If their categories match, i.e. if the similarity score between the word embeddings of their category names are above a certain threshold, we'll **match** the pair together and remove them from the candidate pool.
   (b) If the categories don't match, we **don't match** them and regard this pair as invalid.
   (c) Move on to the next object pair with the highest IoU (start from 2. again). Repeat until there are no object pair candidates left, or if the remaining pairs have an IoU of 0.

After the matching, the relationship annotations in Visual Genome can be transferred over to the COCO object annotations. This process is repeated for all the variants VG-150 [6], GQA [2] and VrR-VG [4]. This helps to maximize the recall of potentially correct scene graph relationships and alleviates the difficulty of the final annotation task for the annotators.

Annotation Process Building upon the preliminary (noisy) PSG dataset (shown in Fig. A5), we patiently trained our annotators to 1) filter out incorrect triplets, and 2) supplement more relations between not only object-object, but also object-background and background-background pairs, using the predefined 56 predicates. The definition of 56 predicates will be explained in the next section. The noisy triplets (for later filtering) are shown to be a good practice for annotators, prompting them providing both salient and detailed information.

A.4 Predicate Dictionary

The design of predicate dictionary is inspired by COCO's practice on object categories selection. According to COCO, the selected categories must be representative, be relevant to practical applications, and be common with high occurrence. Also, a proper level of granularity should also be considered. With these principles in mind, we refer to all the predicates left in the preliminary PSG dataset, sorting them according to their occurrence, and carefully select the predicates.

Practice for the Principles To meet the principles mentioned above, several processes are designed:

- **For Representative:** we deduplicate the predicates and try to make the remaining predicates orthogonal. For example, we shrink a list of similar predicates of 'parked along', 'parked alongside', 'parked at', 'parked behind', 'parked beside', 'parked by', 'parked in', 'parked in front of', 'parked near', 'parked next to', 'parked on' (existed in VG and GQA) to only keep one predicate as '`parked on`'. Also, for the bidirected relation pairs such as 'in front of' and 'behind', we only keep one direction, *i.e.*, '`in-front-of`'. Similarly, only '`over`' is included while 'beneath' is excluded. With this process, we make our vocabulary very concise and thus representative.
- **For Practicality:** Since the goal of the PSG dataset is to facilitate the development of scene understanding tasks, inspired by VrR-VG [4], we get rid of many positional relations that fill the GQA dataset [2], such as 'on the left of' and 'on the right of', and especially focus on the visual-related predicates during our dictionary building.
- **For Coverage:** After several iterations, we finally decided to include 56 predicates with property and can well cover almost all the existing critical relations in the PSG dataset.

56 Predicates in the Dictionary

- **Positional Relations (6):** over, in front of, beside, on, in, attached to.
- **Common Object-Object Relations (5):** hanging from, on the back of, falling off, going down, painted on.
- **Common Actions (31):** walking on, running on, crossing, standing on, lying on, sitting on, leaning on, flying over, jumping over, jumping from, wearing, holding, carrying, looking at, guiding, kissing, eating, drinking, feeding, biting, catching, picking (grabbing), playing with, chasing, climbing, cleaning (washing, brushing), playing, touching, pushing, pulling, opening.
- **Human Actions (4):** cooking, talking to, throwing (tossing), slicing.
- **Actions in Traffic Scene (4):** driving, riding, parked on, driving on.
- **Actions in Sports Scene (3):** About to hit, kicking, swinging.
- **Interaction between Background (3):** entering, exiting, enclosing (surrounding, warping in).

Detailed Predicate Definitions We provided a detailed explanation on each predicate with image examples to ensure the consistent performance from annotators. We will provide the handbook in our PSG website.

B Implementation Details

All experiments are performed in a single unified codebase using the `MMDetection` framework to facilitate reproducibility.

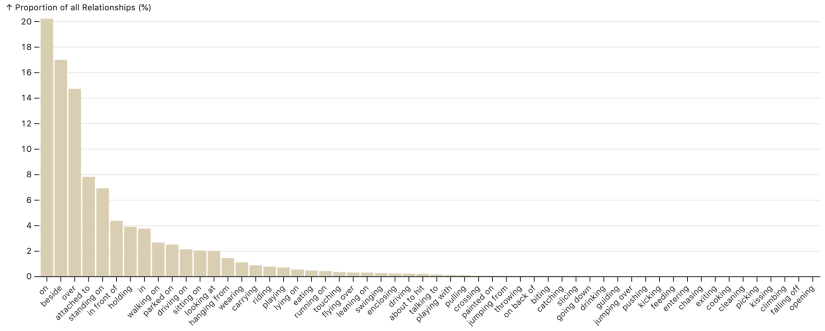

Fig. A3: Proportion of Predicate Classes (Sum=100%).

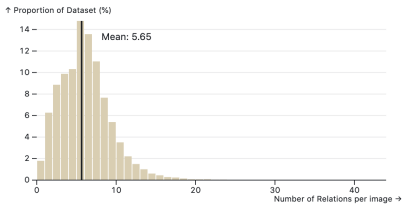

Fig. A4: Distribution of the Number of Relations per image in the PSG Dataset. The bulk of the images have around 5 - 10 relationship annotations, and ranges from 1 - 43 annotations.

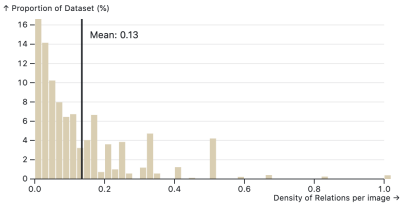

Fig. A5: Distribution of the Density of Relations per image in the PSG Dataset. The density of relations is defined as the number of annotated relations divided by the total number of possible relations in an image.

B.1 Two-Stage PSG Baseline

Fine-tuning the Base Model We first fine-tune the Panoptic FPN base model on the panoptic segmentation annotations from our PSG dataset. The model is initialized from the best performing pretrained weights provided by `MMDetection`, and then trained using a batch size of 8. The SGD optimizer is used with a learning rate of 0.02, momentum of 0.9, weight decay of 0.0001, and gradient clipping with a max L2 norm of 35. Training runs for 12 epochs, with a learning rate schedule that linearly warms up from 0.02 / 3 to 0.02 over 500 iterations, and decays by a factor of 0.1 at the 8th and 11th epochs.

Training the Scene Graph Prediction Head Using the fine-tuned Panoptic FPN above, we freeze its weights and only train the scene graph prediction head. This essentially treats the Panoptic FPN as a black-box feature extractor and panoptic segmentation predictor. For each predicted object, we extract a feature vector using RoIAlign (like in MaskRCNN), making use of the tightest bounding box around its segmentation mask. With grid features, and class predictions and bounding box localizations for each object at hand, we can feed

these into any scene graph prediction head for training and prediction. We use a batch size of 16, and the SGD optimizer with a learning rate of 0.03, momentum of 0.9, and weight decay of 0.0001, and gradient clipping with a max L2 norm of 35. Training runs for 12 epochs, with a learning rate schedule that linearly warms up from 0.03 / 3 to 0.03 over 500 iterations, and decays by a factor of 0.1 at the 7th and 10th epochs. The hyperparameters for the MOTIFS, VCTree and GPSNet models all follow the same settings in their respective papers.

B.2 PSGTR

As it is described in Section 4.2, our PSGTR model extends DETR to PSG task with new heads and a triplet Hungarian matcher. In detail, we implement each of those Feed Forward Networks (FFNs) by a 3-layer MLP, and each panoptic head, following DETR segmentation, consists of a multi-head attention layer and a 6-layer FPN-like CNN. Besides, the number of queries is set as 100 which indicates that 100 possible relations are predicted.

Training settings In general, we follow most of the training strategies of DETR. We adopt the same AdamW optimizer with 10^{-4} learning rate and 10^{-4} weight decay for PSGTR except for the backbone which is trained with learning rate of 10^{-5}. For initialization, we directly use COCO pretrained DETR to initialize the weights of our backbone and transformer. Besides, we also generally follow DETR's data augmentation which does cropping and resizing operations with settings such that the shortest side is at least 480 and at most 800 pixels while the longest at most 1333. However, it should be noted that when cropping images, we also filter the ground truth of bounding boxes and relations pairs that might be cropped. We train our model for 60 epochs with a step scheduler at epoch 40, and it finally takes us around 2 days to train on eight V100 GPUs with batch size 1.

B.3 PSGFormer

PSGFormer is built on the baseline of PSGTR so that most of the training details are shared. In detail, PSGFormer also implements each FFN by a 3-layer MLP, and each panoptic head by a multi-head attention layer and a 6-layer FPN-like CNN as DETR does. Besides, the number of object queries and relation queries are set as 100. We follow the training hyperparameters of PSGTR including optimizer, learning rate, data augmentation, *etc*. Notice that PSGFormer also has an auxiliary task of pure panoptic segmentation with object decoder, the ratio between the main task on triplet supervision and the auxiliary panoptic segmentation supervision is 5 to 1. We train our model for 60 epochs, taking around 2.5 days to train on eight V100 GPUs with batch size 1.

C Visualization of PSGTR Result Triplet-by-Triplet

Fig. A6 shows PSGTR's predict results in a triplet-by-triplet fashion, as a complementary to the lower example in Fig. 6. Notice that panoptic segmentation

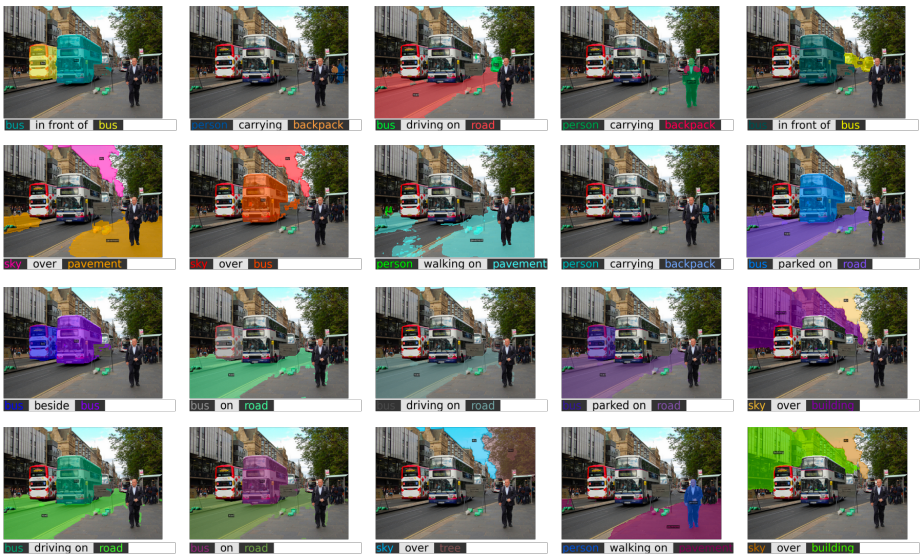

Fig. A6: Visualization of PSGTR Result Triplet-by-Triplet. PSGTR uses triplet queries to directly predict subject / object masks in the triplets, and the subject / object across triplets are not dependent, so the panoptic segmentation visualization is chaos in Fig. 6. However, if we visualize PSGTR result triplet-by-triplet, the result looks good.

visualization is chaos in Fig. 6. It is because PSGTR uses triplet queries to directly predict subject / object masks in the triplets, and the subject / object across triplets are not dependent, and the re-identification of each subject / object is no-trivial, and we use a simple post-processing method of pixel-wise argmax function to merge the segments, but it will still split one object into parts. However, it does not mean that PSGTR cannot segment objects well when predicting triplets. As we visualize the PSGTR result triplet-by-triplet in Fig. A6, the result looks good.

References

1. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
2. Hudson, D.A., Manning, C.D.: Gqa: A new dataset for real-world visual reasoning and compositional question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
3. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., Bernstein, M., Fei-Fei, L.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision (IJCV) (2017)

4. Liang, Y., Bai, Y., Zhang, W., Qian, X., Zhu, L., Mei, T.: Vrr-vg: Refocusing visually-relevant relationships. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
5. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014)
6. Xu, D., Zhu, Y., Choy, C.B., Fei-Fei, L.: Scene graph generation by iterative message passing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2017)