PhraseCut: Language-based Image Segmentation in the Wild Supplemental Material

The supplementary material provides details on the data collection pipeline and the long-tail distribution of concepts in the dataset. We also visualize more results from our proposed HULANet, including predictions from individual modules, failure cases, and comparison against baselines.

1. VGPhraseCut Dataset

1.1. Further details on data collection

As described in the main paper, we start our data collection by mining Visual Genome (VG) boxes and phrases that are discriminative. We then collect region annotations for each phrase from various human annotators. Human annotators are verified by comparing their annotations against VG boxes. Finally, we merge and split collected regions to produce instance-level segmentation masks for each phrase. The steps are described in more details below.

Step 1: Box sampling Each image in VG dataset contains an average of 35 boxes, many of which are redundant. We sample a set of non-overlapping boxes across categories, also removing ones that are too small or large.

Define r as the box size proportional to the image size. We ignore boxes with $r < 0.02$ or $r > 0.9$. For each image, we add the VG boxes to a sample pool one by one. The current box is ignored if it overlaps a box already in the sample pool by $\texttt{IoU} > 0.2$. When sampling from the pool, the weight of each box being sampled is $w = \sqrt{\min(0.1, r)}$ so that we are less likely to get boxes with $r < 0.1$. Every time a new box is sampled, we divide the weights by 5 for all boxes from the same category as the newly sampled box, so that category diversity is encouraged. Since the VG boxes are noisy, we only use annotations on these boxes to generate phrases, but not use the boxes as the ground-truth of the corresponding regions.

Step 2: Phrase generation Figure 1 shows an example of how we generate query phrases when collecting the VGPhraseCut dataset. The goal is to construct phrases that are concise, yet sufficiently discriminative of the target.

For VG boxes that have no other box from the same category in the image, the "basket ball on floor" box for example, we randomly add an additional attribute / relationship annotation (if there is one) to the generated phrase. This avoids ambiguity caused by the missing VG box annotations, and makes it easier to find the corresponding regions, without making the phrase very long. Phrases generated this way are recognized as the "cat+" subset in evaluation.

For VG boxes with unique attribute / relationship annotations within the same category, we generate the phrase by combining its attribute / relationship annotation with the category name. In the "wizard bear" and "bear holding paper" examples, we obtain the phrase to refer to a single VG box, and avoided adding less helpful information ("on wall" or "on floor") to the generated phrase. They are recognized as the "att+" and "rel+" subsets in evaluation.

For the rest, we include all annotations we have on the sampled box into the generated phrase, like what we did in the "small white bear on wall" example. In these cases, the sampled VG boxes are usually more difficult to distinguish from other boxes, so we add all annotations to make the descriptions as precise as possible. This is one of the sources of multi-region descriptions in our dataset.

Step 3: Referred region annotation We present the images and phrases from the previous step to human annotators on Amazon Mechanical Turk, and ask them to draw polygons around the regions that correspond to those phrases. In total we collected $383,798$ phrase-region pairs from 869 different workers, which will be filtered in the next step. In addition, we have $42,587$ phrases skipped by workers, 50.0% with reason "difficult to select", 24.8% for "wrong or ambiguous description", 23.7% for "described target not in image", and 1.5% for "other reasons".

Step 4: Automatic worker verification We designed an automatic worker verification metric in the spirit that statistically, annotations from better workers overlap more with corresponding VG boxes.

We rate each worker based on their overall agreement with VG boxes and the number of annotations they have done to identify a set of trusted workers.

We label a small set of annotations as "good", "so-so", or "bad" ones, and notice that the quality of annotations are strongly correlated with $\texttt{IoP} = s_{\texttt{intersection}}/s_{\texttt{polygon}}$ and $\texttt{IoU} = s_{\texttt{intersection}}/s_{\texttt{union}}$, where $s_{\texttt{polygon}}$ is the area of worker labeled polygons, $s_{\texttt{intersection}}$ and $s_{\texttt{union}}$

Figure 1. **An illustrative example of phrase generation.** We show how we generate phrases from sampled Visual Genome (VG) boxes in four steps. The raw image and additional VG boxes are displayed on top. We do not generate phrases on these additional VG boxes, but they are used to decide the uniqueness of sampled boxes. VG annotations of categories, attributes and relationships on each box are highlighted in blue, yellow and green respectively.

are the intersection and union between worker labeled polygons and VG boxes. On the labeled set of annotations, we learn a linear combination of `IoP` and `IoU` that best separates "good" and "bad / so-so" annotations, defined as $\texttt{agreement} = \texttt{IoP} + 0.8 \times \texttt{IoU}$.

We calculate the average `agreement` score of all annotations from each worker, and set a score threshold based on the total number of annotations from this worker: $\texttt{thresh} = \max(0.7, 0.95 - 0.05 \times \#\text{annotations})$. A worker is trusted if the average `agreement` score is above the `thresh`. Workers with fewer than 10 annotations are ignored. Only annotations from trusted workers are included in our dataset.

In this step, 371 out of 869 workers are verified as trusted. 9.27% (35,565 out of 383,798) phrase-region pairs are removed from our dataset. In rare cases we have multiple annotations on the same input. We randomly select one of them, resulting in 345,486 phrase-region pairs.

Step 5: Automatic instance labeling from polygons Non-overlapping polygons are generally considered as separate instances with a few exceptions. As a final step, we heuristically refine these instance annotations.

First, one instance can be split into multiple polygons when there are occlusions. If the category for a box is not plural and the bounding box of two polygons has a high overlap with it, then they are merged into a single instance. Second, workers sometimes accidentally end a polygon and then create a new one to cover the remaining part of the same instance. We merge two polygons into one instance if they overlap with each other, especially when one is much

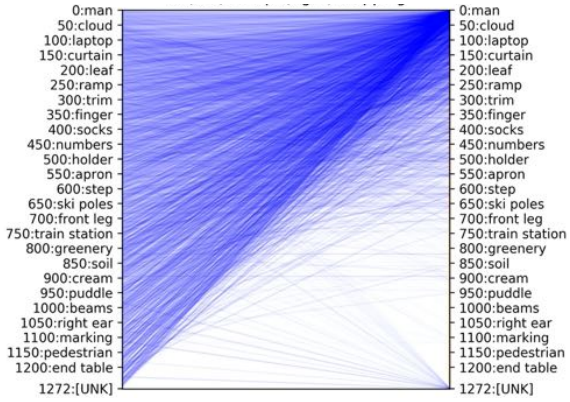

Figure 2. **Category matching in Mask-RCNN top.** Each input category (left) are matched to its best substitute (right) measured by performance on training set. Categories are ordered from top to bottom by frequency.

larger than the other. Third, people tend to cover multiple instances next to each other with a single polygon. If a single polygon matches well with a set of VG boxes, these VG boxes are of similar sizes, and the referring phrase indicate plural instances, we split the polygon into multiple instances according to the VG boxes.

1.2. Additional dataset visualizations

Figure 3 shows the frequency histograms for categories, attributes and relationship descriptions. Compared with tag clouds, the histograms better reveal the long-tail distribution of our dataset.

Figure 4 provides detailed visualizations for three typical categories: "man", "car" and "tree". The attribute and relationship description distributions vary a lot for different categories. Attributes and relationships mainly describe clothing, states and actions for "man". In the "car" category, attributes are focused on colors, while relationships are about locations and whether the car is parked or driving.

We can see several sets of opposite concepts in "tree" attributes, such as "large - small", "green - brown", "bare/leafless - leafy", "growing - dead", etc. Relationships for "tree" mainly describe the locations.

2. Category Matching in Mask-RCNN top

In *Mask-RCNN top* we map each input category to its substitute category. Given an input category, we consider every referring phrase in the training set containing this category, and pick the best category with which the detected mask yields highest `mean-IoU` on each referring phrase. The final substitute category is the one that most frequently picked as the best. The mappings are shown in Figure 2. Using detections of a related and frequent category is often better. Detections from categories with frequency ranked beyond 600 are rarely used.

3. Additional Results from HULANet

3.1. Modular heatmaps

In Figure 5 and Figure 6, we show HULANet predictions and modular heatmaps.

Figure 5 demonstrates that our attribute module is able to capture color ("black", "brown"), state ("closed"), material ("metal") and long and rare attributes ("pink and white"). In the first ("black jacket") example, the category module detects two jackets, while the attribute module is able to select out the "black" one against the white one.

Figure 6 shows how our relationship module modifies the heatmaps of supporting objects depending on different relationship predicates. With the predicate "wearing", the relationship module predicts expanded regions of the detected "jacket" especially vertically. The relational prediction of "parked on" includes regions of the "street" itself as well as regions directly above the "street", while the predicate "on" leads to the identical region prediction as the supporting object. In the last example of "sitting at", a broader region around the detected "table" is predicted, covering almost the whole image area.

3.2. Failure case analysis

Figure 7 displays typical failure cases from our proposed HULANet. Heatmaps from internal modules provide more insights where and why the model fails.

In the first example, our backbone Mask-RCNN fails to detect the ground-truth "traffic cones", which are extremely small and from rare categories. Similarly, in the second "dark grey pants" example, the "pants" is not detected as a separate instance in the backbone Mask-RCNN, therefore the category module can only predict the whole mask of the skateboarder.

The third "window" example shows when the category module (and the backbone Mask-RCNN) fails to distinguish mirrors from windows. In the fourth example, our attribute module fails to recognize which cat is "darker" than the other.

We then display two failure cases for the relationship module. It fails on the first one because the supporting object ("suitcase") is not detected by the category module, and fails on the second one for unable to accurately model the relation predicate "on side of".

In the last example, although our attribute module figures out which sofa is "plaid", the final prediction is dominated by the category module and fails to exclude non-plaid sofas.

3.3. More comparisons against baseline methods

As an extension to Figure 7 in the main paper, Figure 8 here shows more examples of prediction results compared against baseline methods. The "white building", "chair" and "large window" examples demonstrate that our HULANet is better at handling occlusions.

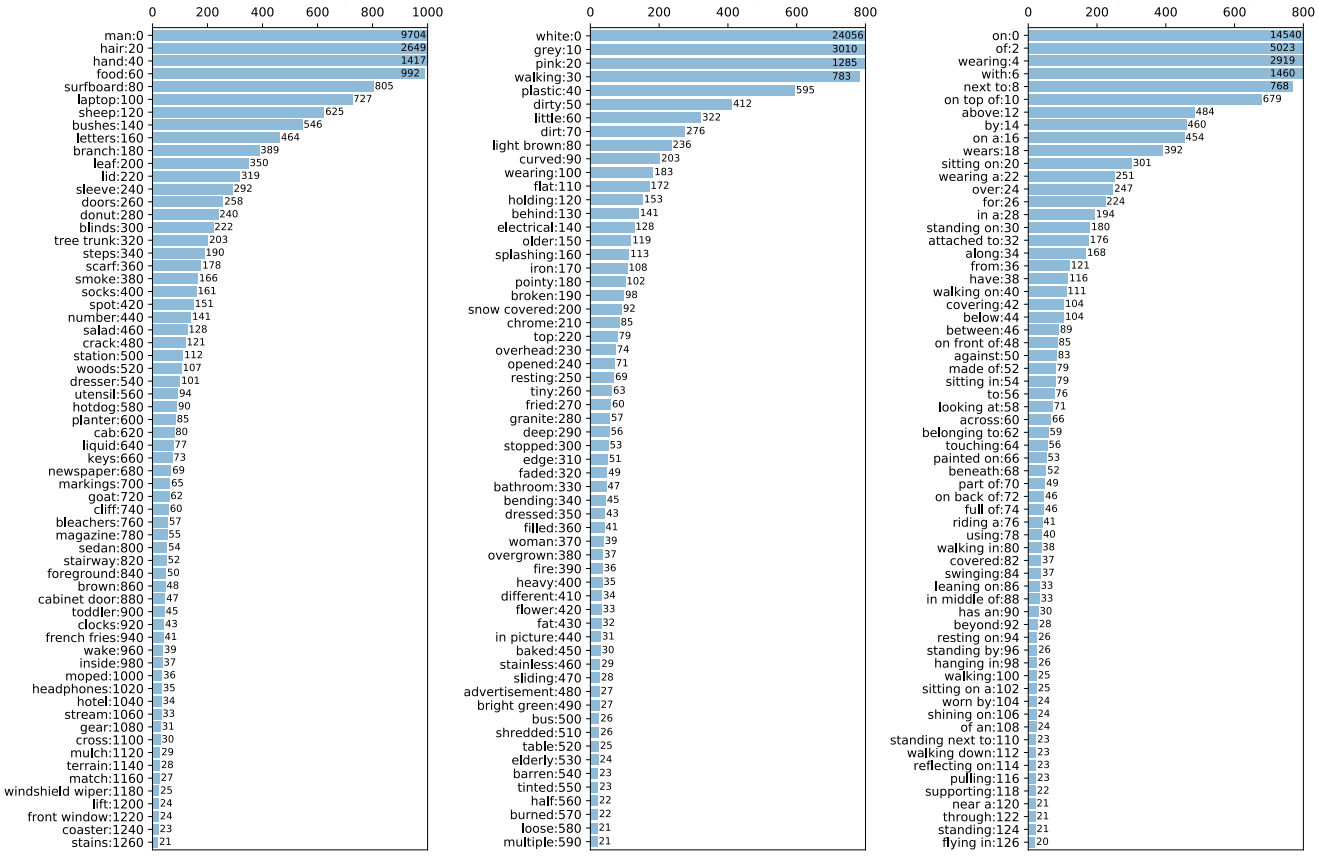

Figure 3. **Frequency histograms of categories (left), attributes (middle) and relationship descriptions (right).** Y-axis shows each entry and its frequency ranking; X-axis shows their frequency in the whole dataset.

Figure 4. **Visualizations of "man", "car" and "tree" categories.** From left two right, we display two examples, attribute cloud and relationship cloud within the given category. The size of each phrase is proportional to the square root of its frequency in the given category.

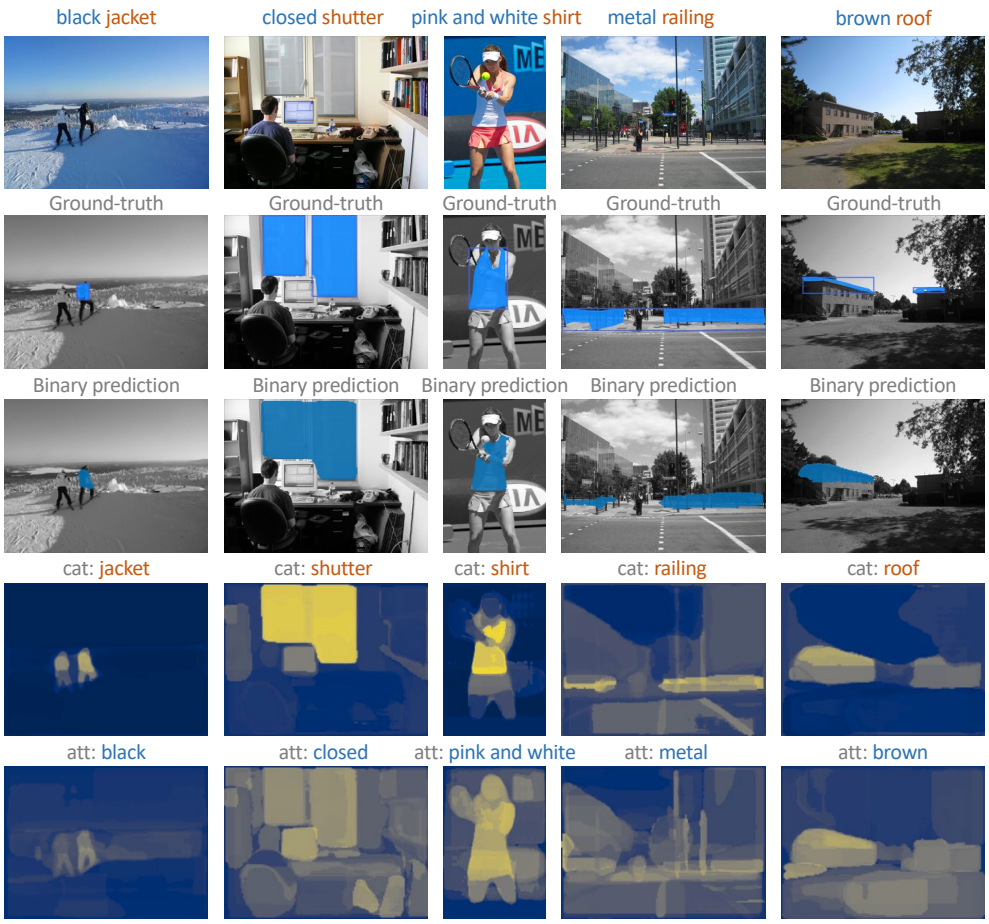

Figure 5. **HULANet prediction results and heatmaps on phrases with attributes.** Rows from top to down are: (1) input image; (2) ground-truth segmentation and instance boxes; (3) predicted binary mask from HULANet (cat+att+rel); (4) heatmap prediction from the category module; (5) heatmap prediction from the attribute module.

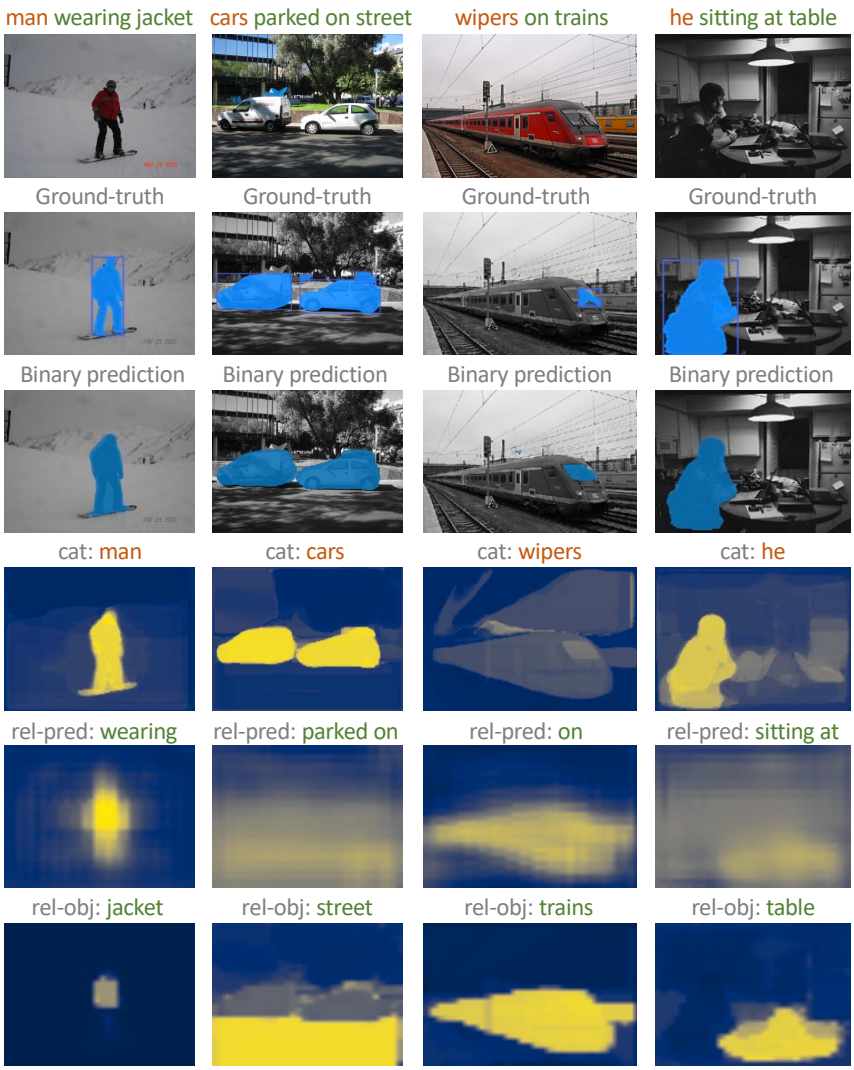

Figure 6. **HULANet prediction results and heatmaps on phrases with relationships.** Rows from top to down are: (1) input image; (2) ground-truth segmentation and instance boxes; (3) predicted binary mask from HULANet (cat+att+rel); (4) heatmap prediction from the category module; (5) heatmap prediction from the relationship module; (6) heatmap prediction of the supporting object (in the relationship description) from the category module.

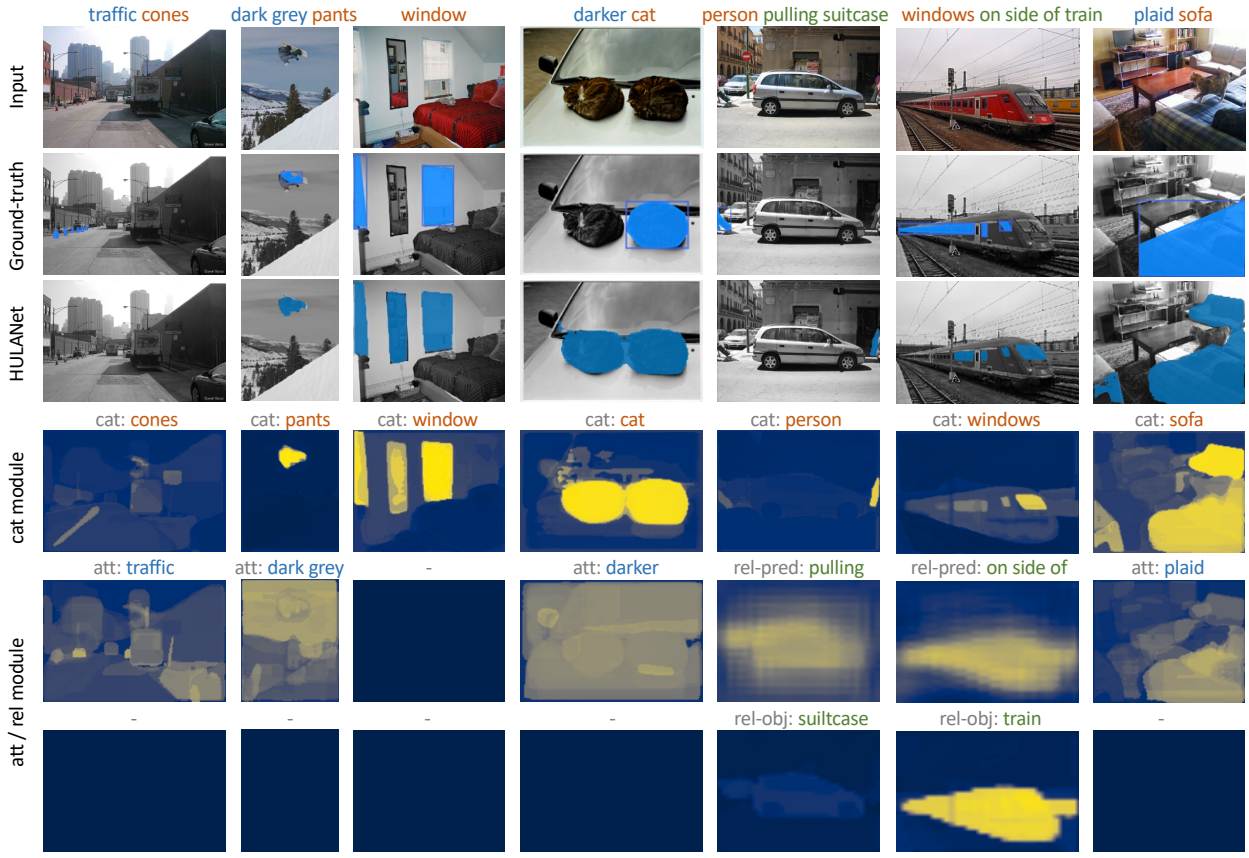

Figure 7. **Negative results from HULANet on VGPhraseCut test set.** Rows from top to down are: (1) input image; (2) ground-truth segmentation and instance boxes; (3) predicted binary mask from HULANet (cat+att+rel); (4) heatmap prediction from the category module; (5-6) heatmap predictions from additional (attribute or relationship) modules.

Figure 8. **More prediction results comparing against baseline models on VGPhraseCut test set.** Rows from top to down are: (1) input image; (2) ground-truth segmentation and instance boxes; (3) MattNet baseline; (4) RMI baseline; (5) HULANet (cat + att + rel).