Supplementary File

1. Training Details

We optimize our model using Adam [5] with an initial learning rate of 0.0004 and with a batch size of 15 images (and all their expressions). The learning rate is halved every 8,000 iterations after the first 8,000-iteration warm-up. The word embedding size and hidden state size of the LSTM are set to 512. We also set the output of all MLPs and FCs within our model to be 512-dimensional. To avoid overfitting, we regularize the word-embedding and output layers of the LSTM in the language attention network using dropout with ratio of 0.5. We also regularize the two inputs (visual and language) of matching function using a dropout with a ratio of 0.2. For the constrastive pairs, we set $\lambda_1 = 1.0$ and $\lambda_2 = 1.0$ in the ranking loss L_{rank}. Besides, we set $\lambda_{attr} = 1.0$ for multi-label attribute cross-entropy loss L_{subj}^{attr}.

2. Computational Efficiency

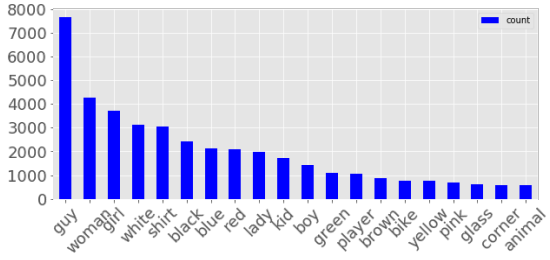
During training, the full model of MatNet converges at around 30,000 iterations, which takes around half day using single Titan-X(Pascal). At inference time, our fully automatic system goes through both Mask R-CNN and MatNet, which takes on average 0.33 seconds for a forward, where 0.31 seconds are spent on Mask R-CNN and 0.02 seconds on MatNet.

3. Attribute Prediction

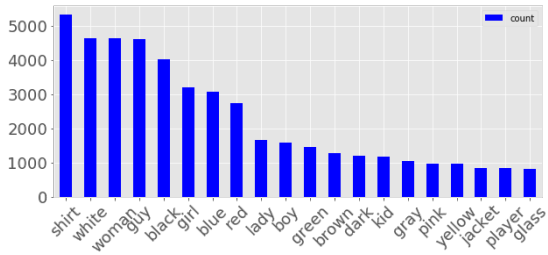
Our full model is also able to predict attributes during testing. Our attribute labels are extracted using the template parser [4]. We fetch the object name, color and generic attribute words from each expression, with low-frequency words removed. We use 50 most frequently used attribute words for training. The histograms for top-20 attribute words are shown in Fig. 1, and the quantitative analysis of our multi-attribute prediction results is shown in Table. 1.

	Split	Precision	Recall	F1
RefCOCO	val	63.48	29.91	40.66
RefCOCO+	val	61.78	20.11	30.14
RefCOCOg	val	68.18	34.79	46.07

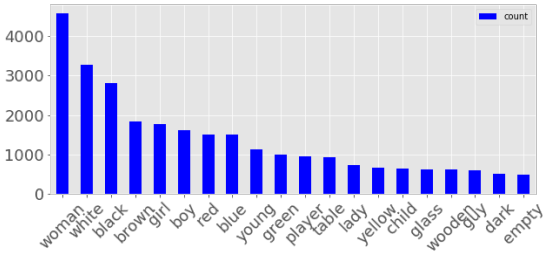
Table 1: Multi-attribute prediction on the validation split of each dataset.



(a) RefCOCO



(b) RefCOCO+



(c) RefCOCOg

Figure 1: Attribute histogram for three datasets.

4. MAttNet + Grabcut

In Section 4.3 (Segmentation from Referring Expression), we show our MAttNet could be extended to referential segmentation by using Mask R-CNN as our backbone net. The experiments show that our model achieves superior performance over previous state-of-art approaches [3, 6]. Actually, the mask branch of our model could be any foreground-background decomposition method. The simplest replacement might be GrabCut. We show the results of MatNet+GrabCut in Table 2 Note even though GrabCut is an inferior segmentation method, it still far outperforms the best numbers in [6]. Thus, we believe decoupling box lo-

RefCOCO							
Model	Split	Pr@0.5	Pr@0.6	Pr@0.7	Pr@0.8	Pr@0.9	IoU
D+RMI+DCRF [6]	val	42.99	33.24	22.75	12.11	2.23	**45.18**
MAttNet+**GrabCut**	val	**51.25**	**41.89**	**29.77**	**17.13**	**5.38**	42.86
D+RMI+DCRF [6]	testA	42.99	33.59	23.69	12.94	2.44	**45.69**
MAttNet+**GrabCut**	testA	**52.94**	**42.60**	**27.68**	**13.29**	**2.92**	44.37
D+RMI+DCRF [6]	testB	44.99	32.21	22.69	11.84	2.65	**45.57**
MAttNet+**GrabCut**	testB	**47.18**	**38.27**	**29.97**	**20.35**	**7.85**	40.71
RefCOCO+							
Model	Split	Pr@0.5	Pr@0.6	Pr@0.7	Pr@0.8	Pr@0.9	IoU
D+RMI+DCRF [6]	val	20.52	14.02	8.46	3.77	0.62	29.86
MAttNet+**GrabCut**	val	**45.24**	**37.09**	**26.51**	**14.95**	**4.34**	**37.18**
D+RMI+DCRF [6]	testA	21.22	14.43	8.99	3.91	0.49	30.48
MAttNet+**GrabCut**	testA	**47.10**	**37.86**	**24.66**	**11.67**	**2.27**	**38.32**
D+RMI+DCRF [6]	testB	20.78	14.56	8.80	4.58	0.80	29.50
MAttNet+**GrabCut**	testB	**38.52**	**31.13**	**24.44**	**16.71**	**6.20**	**33.30**

Table 2: Comparison of [14] and MatNet+GrabCut.

calization (comprehension) and segmentation brings a large gain over FCN-style approaches for this instance-level segmentation problem.

5. Mask R-CNN Implementation

Our implementation of Mask R-CNN is based on the single-GPU Faster R-CNN implementation [1][1]. For the mask branch, we strictly follow the structure in the original paper [2]. The only differences are: 1) We sample $R = 256$ regions from $N = 1$ image during each forward-backward propagation due to the constraint of single GPU, while [2] samples $R = 128$ regions from $N = 16$ images using 8 GPUs. 2) During training, the shorter edge of our resized image is 600 pixels instead of 800 pixels, for saving memory. 3) Our model is trained on a union of COCO's 80k train and 35k subset of val (trainval35k) images, but excluding the val/test (valtest4k) images in RefCOCO, RefCOCO+ and RefCOCOg.

We firstly show the comparison between Faster R-CNN and Mask R-CNN on object detection in Table. 3. Both models are based on ResNet101 and were trained using same setting. In the main paper, we denote them as **res101-frcn** and **res101-mrcn** respectively. It is shown that Mask R-CNN has higher AP than Faster R-CNN due to the multi-task (mask branch) training.

We then compare our Mask R-CNN implementation with the original one [2] in Table 4. Note this is not a strictly fair comparison as our model was trained with fewer images, but reflects some difference. Overall, the AP of our implementation is $\sim$2 points lower than the original paper.

[1]Our implementation: https://github.com/lichengunc/mask-faster-rcnn.

net	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
res101-frcn	34.1	53.7	36.8
res101-mrcn	35.8	55.3	38.6

Table 3: Object detection results.

net	AP	AP_{50}	AP_{75}
res101-mrcn (ours)	30.7	52.3	32.4
res101-mrcn [2]	32.7	54.2	34.0

Table 4: Instance segmentation results.

The main reason may due to the shorter 600-pixel edge setting and smaller training batch size. Even though, our pixel-wise comprehension results already outperform the state-of-art ones with a huge margin (see Table 4 in the main paper), and we believe there exists space for further improvements.

6. More Examples

We show more examples of comprehension using our full model in Fig. 2 (RefCOCO), Fig. 3 (RefCOCO+) and Fig. 4 (RefCOCOg). For each example, we show the input image (1st column), the input expression with our predicted module weights and word attention (2nd column), the subject attention (3rd column) and top-5 attributes (4th column), box-level comprehension (5th column), and pixel-wise segmentation (6th column). As comparison, we also show some incorrect comprehension in Fig. 5.

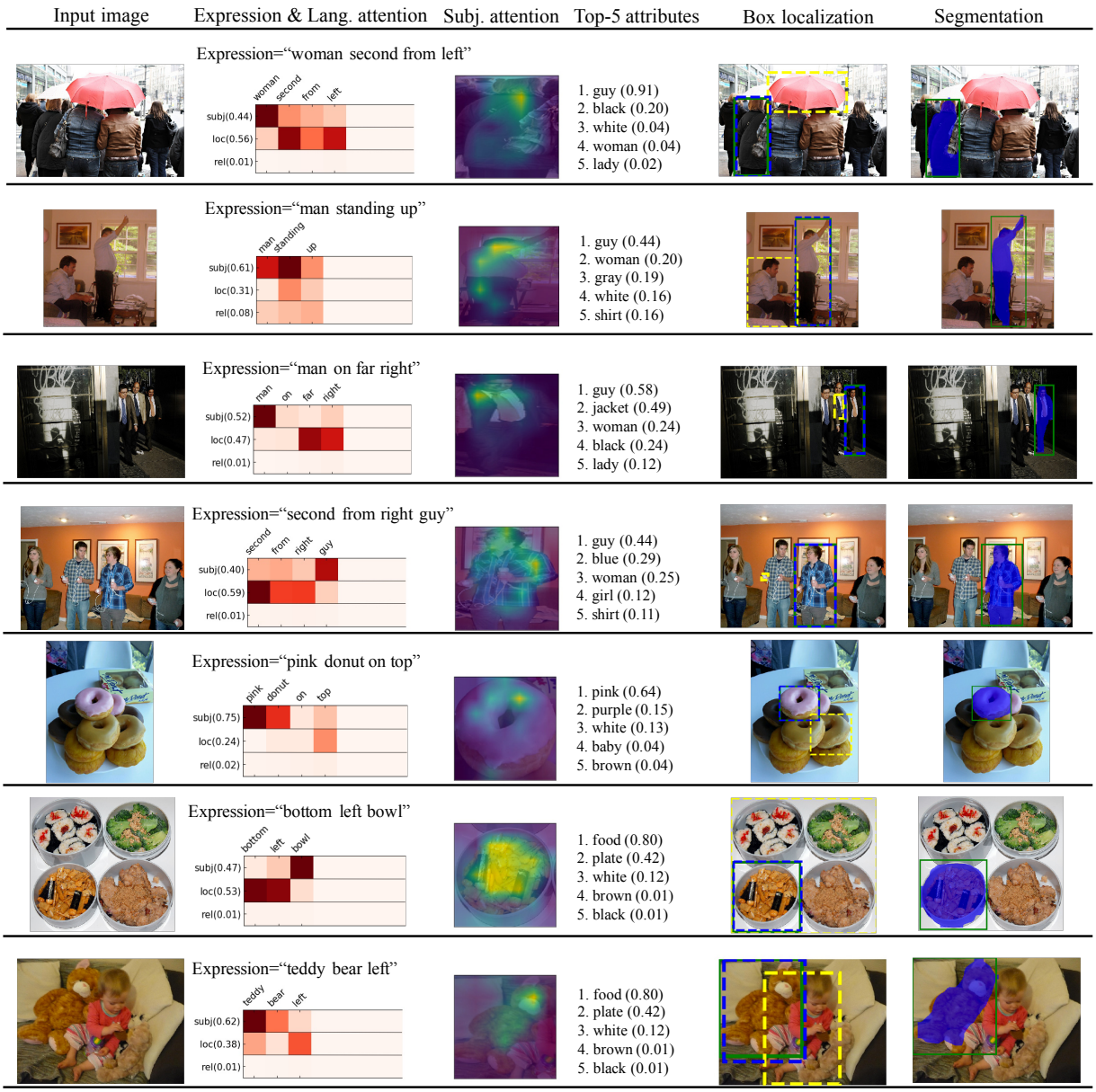


Figure 2: Examples of fully automatic comprehension on **RefCOCO**. The 1st column shows the input image. The 2nd column shows the expression, word attention and module weights. The 3rd column shows our predicted subject attention, and the 4th column shows its top-5 attributes. The 5th column shows box-level comprehension where the red dotted boxes show our prediction and yellow dotted boxes shows the relative object, and the green boxes are the ground-truth. The 6th column shows the segmentation.

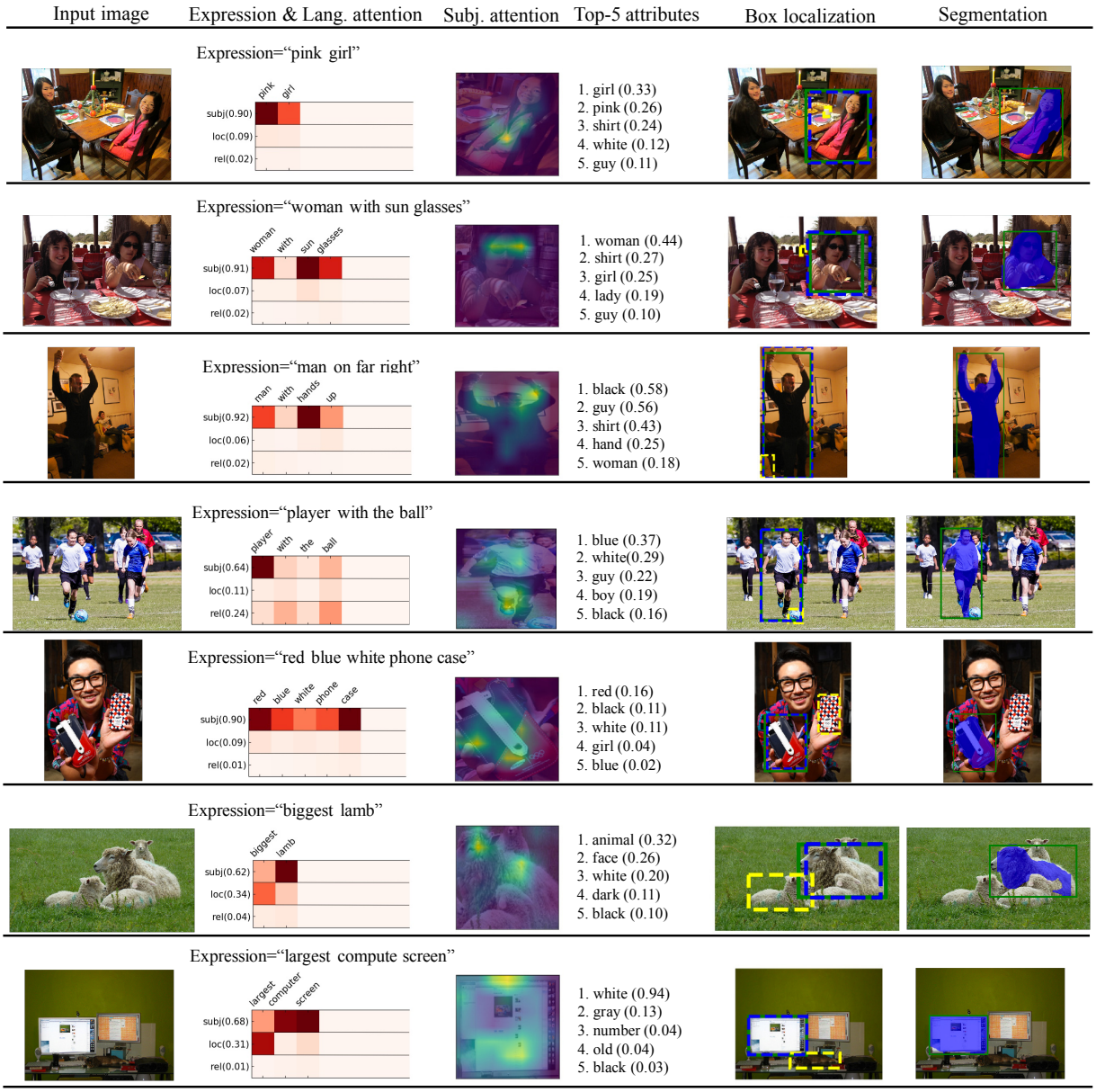


Figure 3: Examples of fully automatic comprehension on **RefCOCO+**.The 1st column shows the input image. The 2nd column shows the expression, word attention and module weights. The 3rd column shows our predicted subject attention, and the 4th column shows its top-5 attributes. The 5th column shows box-level comprehension where the red dotted boxes show our prediction and yellow dotted boxes shows the relative object, and the green boxes are the ground-truth. The 6th column shows the segmentation.

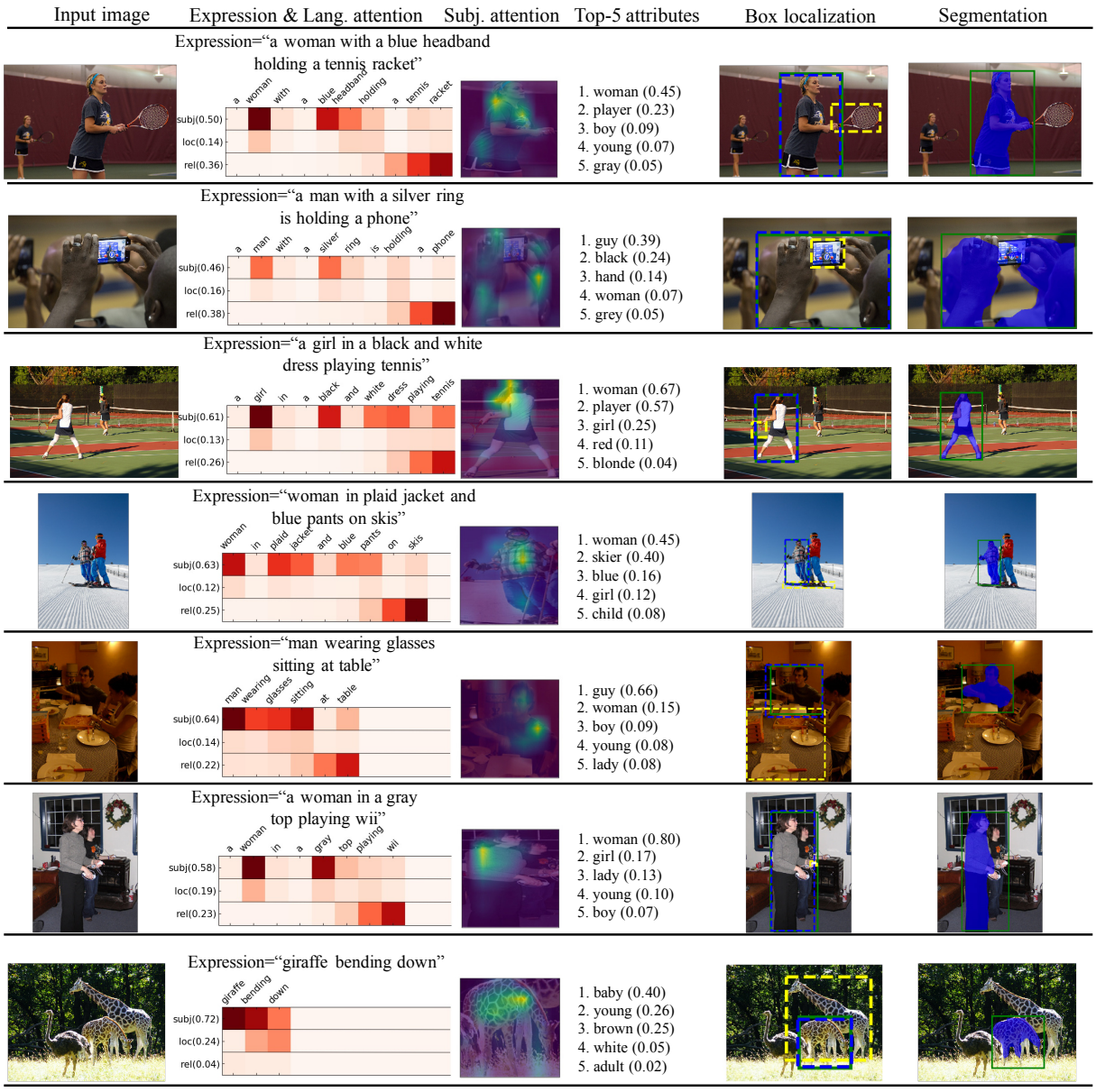


Figure 4: Examples of fully automatic comprehension on **RefCOCOg**. The 1st column shows the input image. The 2nd column shows the expression, word attention and module weights. The 3rd column shows our predicted subject attention, and the 4th column shows its top-5 attributes. The 5th column shows box-level comprehension where the red dotted boxes show our prediction and yellow dotted boxes shows the relative object, and the green boxes are the ground-truth. The 6th column shows the segmentation.

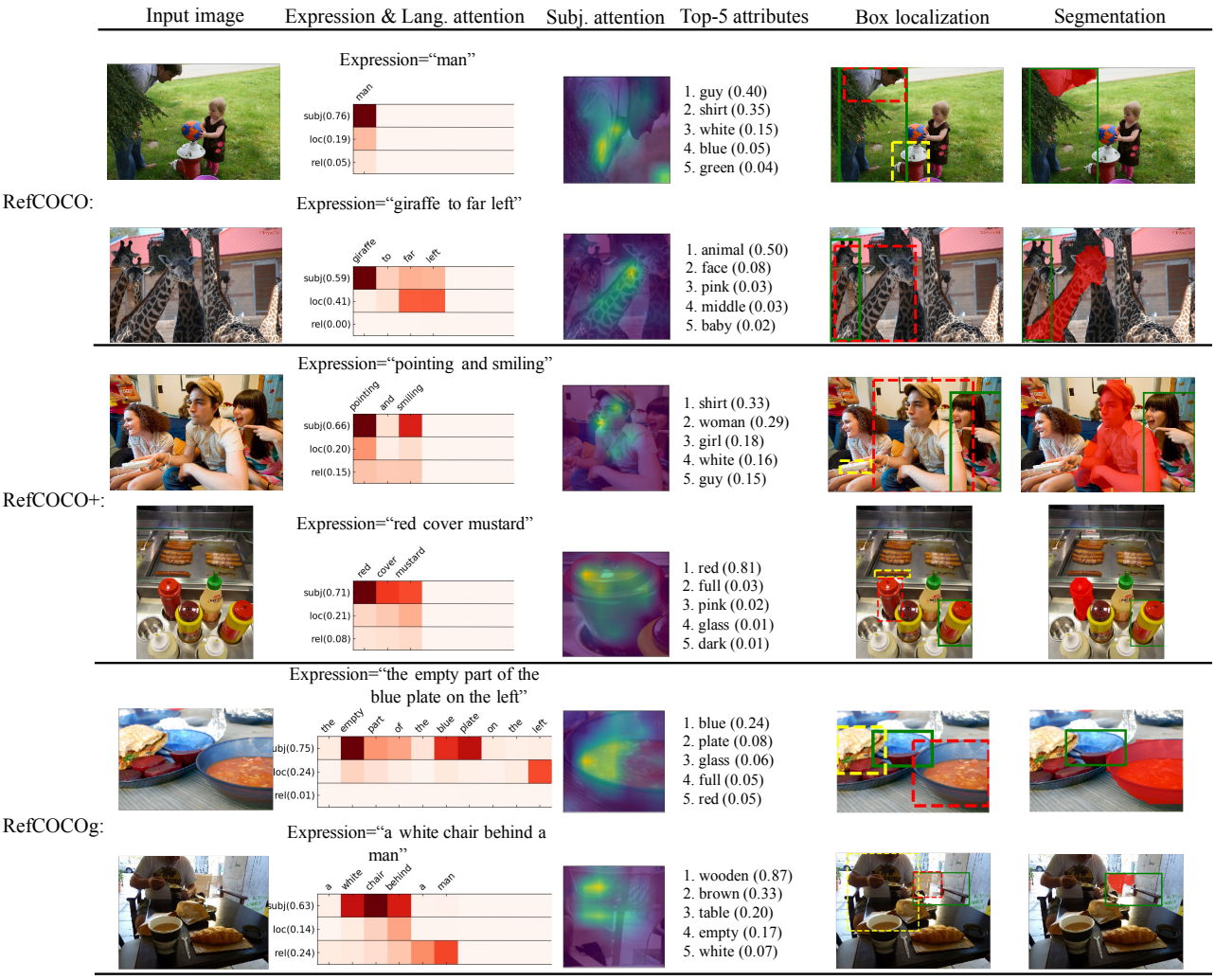


Figure 5: Examples of incorrect comprehension on three datasets. The 1st column shows the input image. The 2nd column shows the expression, word attention and module weights. The 3rd column shows our predicted subject attention, and the 4th column shows its top-5 attributes. The 5th column shows box-level comprehension where the red dotted boxes show our prediction and yellow dotted boxes shows the relative object, and the green boxes are the ground-truth. The 6th column shows the segmentation.

References

[1] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*, 2017.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[3] R. Hu, M. Rohrbach, and T. Darrell. Segmentation from natural language expressions. In *ECCV*, 2016.

[4] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014.

[5] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] C. Liu, Z. Lin, X. Shen, J. Yang, X. Lu, and A. Yuille. Recurrent multimodal interaction for referring image segmentation. In *ICCV*, 2017.