Supplementary Material for paper ID #1987 (Grounded Situation Recognition)

Here we provide a more detailed explanation of methods introduced in this work and provide additional qualitative results demonstrating the efficacy of our proposed model. In Section A we discuss the details of Semantic Image Retrieval as mentioned in Section 6. In Section B we provide the implementation details of our baseline model (ISL) and proposed model (JSL). In Section C we discuss the model changes we make to JSL in order to create the Conditional Situation Localizer as discussed in Section 6. Finally, in Section E we provide qualitative results comparing the localization of ISL and JSL as well as qualitative results visualizing the situations generated for the top-5 verbs predicted by JSL.

A Semantic Image Retrieval

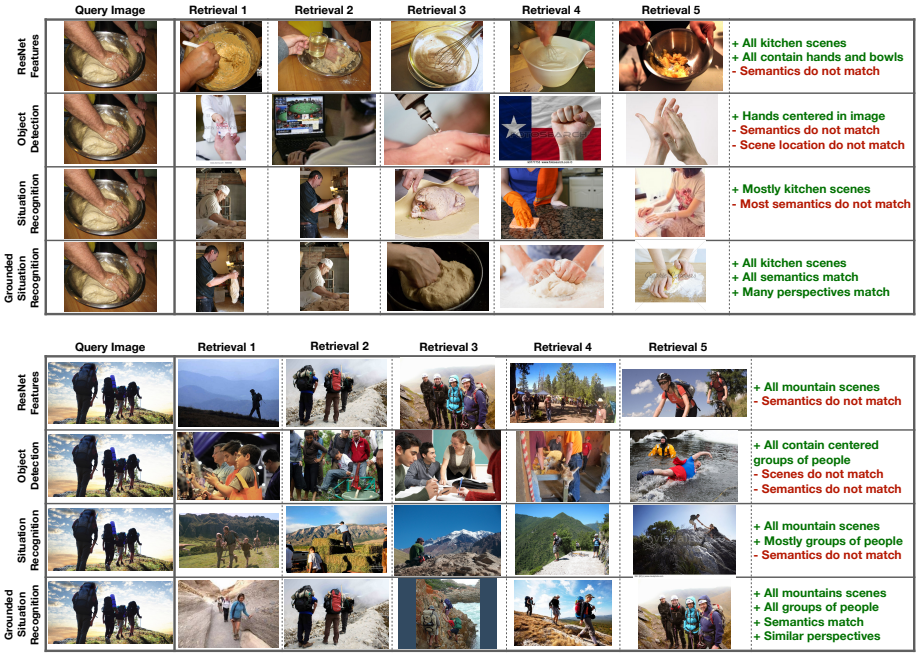

Fig. 10. Additional qualitative results for semantic image retrieval. For the query figure of a baker kneading dough or multiple hikers walking, ResNet and Object Detection based methods struggle to match the semantics of the image. Grounded situation based retrieval leads to the correct semantics with matching viewpoints

In Fig. 6 and Fig. 10 we show qualitative examples of semantic image retrieval implemented with nearest neighbor computations and a collection of different

similarity functions. In particular, we divide our validation set into a query set (1008 images, 2 images per verb) and search set (24192 images, 48 images per verb). For each of the images in our query set, we compute the similarity of the query image with all images in search set and save the top-5 most similar images to the query. We now describe how we compute image similarity using ResNet-50 features, bag-of-words object detections, situation predictions, and grounded situation predictions.

A.1 ResNet-50

We compute a featurization of each image using a ResNet-50 model pretrained on the ImageNet dataset. Similarity between images is then computed as the negative of the L2 distance between these featurizations (so that images with nearer featurizations are more similar).

A.2 Object Detections

For each image I we compute object detections using the modified RetinaNet described in Section 4. We find these detections by computing the maximum likelihood category for each box. If the logits corresponding to probability of the maximum category is greater than -1 we consider it a valid detection. To prevent multiple detections of the same object we use NMS to remove any overlapping boxes of the same object category. We save the predicted class labels $\{c_1^I, ..., c_{N_I}^I\}$ and bounding-boxes $\{b_1^I, ..., b_{N_I}^I\}$. Similarity between two images I, J is then computed as

$$\text{ObjSim}(I, J) = \frac{1}{N} \sum_{i=1}^{N} \max\{1_{[c_i^I = c_j^J]} \cdot (1 + \text{IoU}(b_i^I, b_j^J)) \mid 1 \leq j \leq M\} \quad (1)$$

so that $\text{ObjSim}(I, J)$ will be maximal when the objects detected in I have the same classes and bounding-boxes as those in J.

A.3 Situation Recognition

For each image I in our validation set, we compute $v_1^I, ..., v_5^I$ the top-5 predicted activities (verbs) associated with I. For each of these verbs v_i^I, we additionally predict the entities associated with the roles of that verb, $e_{i,1}^I, ..., e_{i,N_{v_i^I}}^I$. We then compute the situation similarity between two images I, J as

$$\text{SitSim}(I, J) = \max\{\frac{1_{[v_i^I = v_j^J]}}{i \cdot j \cdot N_v} \sum_{k=1}^{N_{v_i^I}} 1_{[e_{i,k}^I = e_{j,k}^J]} \mid 1 \leq i, j \leq 5\}. \quad (2)$$

Notice that $\text{SitSim}(I, J)$ is only non-zero if there is at least one verb shared in the top-5 verb predictions of I and J. Moreover, the similarity will be at its maximum value of 1 if any only if both I and J have the same top-1 verb and, for that verb, all predicted entities (conditioned on that top-1 verb) for both images are the same.

A.4 Grounded Situation Recognition

As above, we have, for an image I top-5 verb predictions $v_1^I, ..., v_5^I$ and entity predictions $\{e^I_{i,k_i} \mid 1 \leq i \leq 5,\ 1 \leq k_i \leq N_{v_i^I}\}$. For grounded situation predictions we also have, for each entity $e^I_{i,k}$ a bounding-box prediction $b^I_{i,k}$. We then compute similarity between two images I, J as

$$\begin{aligned}&\text{GrSitSim}(I, J)\\ &= \max\{\frac{1_{[v_i^I = v_j^J]}}{i \cdot j \cdot N_v} \sum_{k=1}^{N_{v_i^I}} 1_{[e^I_{i,k} = e^J_{j,k}]} \cdot (1 + \text{IoU}(b^I_{i,k}, b^J_{j,k})) \mid 1 \leq i, j \leq 5\}. \quad (3)\end{aligned}$$

Notice that GrSitSim is nearly identical to SitSim except that GrSitSim will be larger when predicted entities have similar bounding boxes, as measured by their intersection over union.

B Implementation Details

B.1 RNN

Architecture We use a ResNet-50 backbone pretrained on ImageNet. The embedding size for nouns is 512 and the embedding size for verbs is 256. We use a single layer LSTM as the the RNN with a hidden size of 1024 and an input size of 2816 (2048 image features, 512-dimensional embedding of the previous noun, 256-dimensional embedding of the verb). The LSTM is initialized with orthogonal weights. The 512-dimensional noun vector is initialized with zeros for the first noun prediction. The LSTM predicts a sequence length of 6 as this is the maximum length frame. Frames with less than this length are padded to length 6. The ground truth verb embedding is used as input to the LSTM for all of training, as incorrect verb predictions are always marked as having incorrect noun predictions, so there is no benefit to training with incorrect verb predictions.

Training We train the RNN using the Adam Optimizer [27] with $\beta = (0.9, 0.999)$. The initial learning rate is set to 1e-4 which is decreased by a factor of 10 at epoch 12 and 24. Additionally, we begin training by freezing the ResNet weights and only begin to propagate the gradients through ResNet at epoch 14. We train with a batch size of 32 for 100 epochs, which takes 40 hours on one 12GB TITAN V GPU and use the weights from the best performing epoch.

B.2 Object Detector

Architecture The majority of this architecture is unchanged from the original RetinaNet architecture. We ResNet-50 backbone pretrained on ImageNet. The majority of the differences from the original RetinaNet take place in the

adjustments to the network which allow for detection of 10,000 categories. As mentioned in in Section 4, we adjust the network by predicting the likelihood that each anchor box contains an object, rather than predicting a distribution over all object categories for each anchor box. We then perform NMS to remove low scoring boxes which have a high overlap with other boxes. We take the top 100 boxes most likely to contain an object and obtain the features corresponding to these boxes in the final spatial layer of ResNet using RoI align. We then linearly transform the feature vectors into a vector the size of the noun vocabulary to obtain a predicted distribution. For training, if these boxes overlap with a ground truth annotation with an IoU of at least 0.5, they are labeled will all the categories attributed to the ground truth box. A ground truth box may have multiple categories as there are multiple annotators. If it does not overlap with any ground truth box it is not labeled with any category. If it overlaps with multiple ground truth boxes, we duplicate the predicted box and each one is considered to overlap with one ground truth box. We then use binary cross entropy on these labels and the predicted distribution. When combining the RNN output and RetinaNet outputs, a box is assigned to a noun category if it has the highest predicted value for that noun category out of all 100 boxes. If none of the boxes reach a certain threshold for that noun category, then the noun is label as ungrounded in the image. We tune this threshold to be -4 for our model, so if none of the logits are above this value for the desired category, it is ungrounded.

Training We train with a batch size of 64 using the Adam Optimizer [27] with $\beta = (0.9, 0.999)$. We use a learning rate of 1e-4 for all of training. We train until convergence and then use the weights from the epoch (26) which achieved the highest accuracy on the dev set. We train the network for $\sim$72 hours on eight 12GB TITAN V GPUs. Despite the modifications we made to the RetinaNet model, training is still relatively slow as we must still perform 100 classifications for every image.

B.3 JSL

Architecture As with the RNN, we use a single layer LSTM with hidden size 1024, noun embeddings of size 512 and verb embeddings of size 256. We initialize the ResNet-50 backbone with imagenet weights and initialize the LSTM with orthogonal weights. Additionally we pad shorter frames to be of length 6 and label all of the pad symbols to be ungrounded. Like the RNN model, we always use the embedding of the ground truth verb during training, as the nouns are always considered incorrect if the verb prediction is incorrect so there is no benefit to training with the incorrect verb prediction. Additionally, for the first 5 epochs of training, we use the ground truth bounding boxes when obtaining the local features for noun classification and we use the previous ground truth noun when embedding the previous noun for the LSTM. When combining the output of the LSTM with the features from the FPN before the classification and regression branches (see Figure 5) we concatenate the FPN features with a

linear projection of size 256 of the hidden state of the LSTM. Additionally we concatenate an element wise product of these two vectors, resulting in a final input vector with a channel dimension of 768.

Training We train with a batch size of 64 using the Adam Optimizer [27] with $\beta = (0.9, 0.999)$. We use an initial learning rate of 6e-4 which we decrease by a factor of 10 at epochs 10 and 20. Like with the RNN, we begin by freezing the ResNet weights and only begin to propagate the weights to the ResNet backbone at epoch 12. We train until convergence and use the weights which have the highest performance on the validation set (epoch 27). Training takes ∼20 hours on four 24GB TITAN RTX GPUs.

Verb Prediction Network As mentioned in Section 4, we find using a separate network to predict the verb increases the accuracy of verb prediction, while keeping the total number of parameters equal to that of the independent model. To train the verb classifier, we use a ResNet backbone with a linear layer on top of the final feature vector of size 2048, just after the final average pooling. We use the Adam Optimizer with an initial learning rate of 1e-4 which we decrease by a factor of 10 at epoch 18. We train just the final linear layer for the first 5 epochs, then just the linear layer and final block for the next 5 epochs. We continue this pattern, unfreezing one additional ResNet block every 5 epochs until epoch 15. We never propagate through the first block as we find this decreases the overall accuracy, likely due to overfitting. We use standard cross entropy loss and a batch size of 256. Training takes ∼1 hour on eight 12GB TITAN V GPUs.

C Conditional Situation Recognition

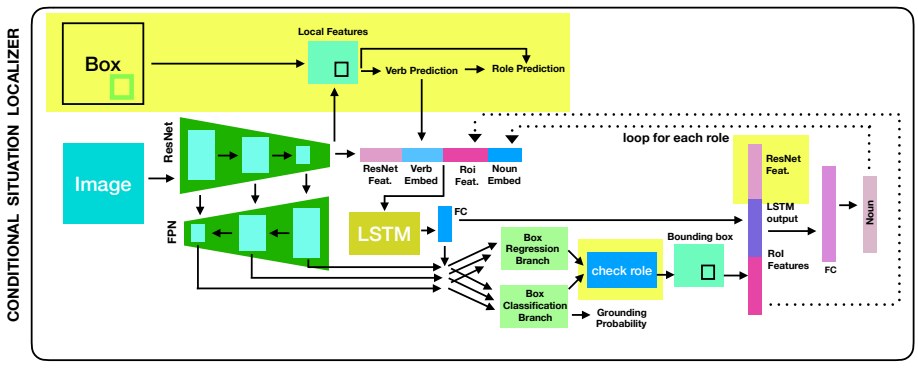

Fig. 11. Model schematics for the CSL model. Differences between JSL and CSL are highlighted in yellow.

The Conditional Situation Localizer (CSL) is a modification of JSL which conditions its output on a specific bounding-box, as illustrated by Figure 7. The network architecture of CSL is illustrated by Figure 11, with differences from JSL highlighted in yellow. Rather than predicting the verb via a separate network, the verb prediction is done from the local features inside the bounding-box. As in JSL, these local features are obtained by performing RoI Align on the last spatial features of ResNet. Then the verb prediction and these local features are used to predict the role that the object within the box plays with respect to the verb. For example in Figure 7A1, the local features surrounding the query were first used to predict the action as 'Calling' and then this verb prediction and those local features where used to predict that the object in the bounding-box fills the second role for this verb, which corresponds to 'Tool' in this case.

CSL then works exactly as JSL except the input bounding-box is used for the predicted role. So if the model predicts that the bounding-box corresponds to the second role for the predicted verb, then on the second pass of the LSTM, the bounding-box prediction made by the classification and regression branches are overwritten by the position of the input bounding-box. This is demonstrated by the "check role" portion of Figure 11. At each pass, the network checks if the current iteration is equal to the role predicted by the input bounding-box. If it is, then that bounding-box is used, otherwise the predicted bounding-box is used.

D Language Only Baselines

D.1 Language Baseline

We conduct several experiments to examine if language-only baseline models are able to exploit some inherent bias in grounded situation recognition to solve the task. Mallya et al. [36] test for bias in the imSitu dataset (which SWiG builds upon) by running their model without access to image pixels. They obtain a value accuracy of 52.12 under the ground truth verb setting, substantially lower than 70.48 for models with access to pixels.

We have similar results with JSL: a value score of 53.44 when run without access to pixels compared to 73.53 otherwise. Additionally, when the most common answer per role is predicted this yields a value of 53.67. Predicting the most common answer per role and average of all locations per role yields a grounded value of 16.25. Predicting the most common answer per role and that they are ungrounded yields a grounded value of 22.44. Both of these value are slightly below the grounded value score of 28.38 for JSL without pixels and well below the grounded value score of 57.50 for JSL with pixels. This difference reveals that the task cannot be solved merely by exploiting biases in the dataset.

D.2 Object Baseline

The models may additionally be solving the task by exploiting the bias of simply identifying what objects are in the image without needing to understand their

role or interactions. The ideal study to test if models are relying on this bias would be to predict the situations given ground truth bounding boxes of all objects in the image for all SWiG categories, however this is prohibitively expensive to collect, so we consider several evocative variants. For all our experiments, we input object categories and detections of the form $[x_1, y_1, x_2, y_2]$ into an LSTM and use the final hidden state to predict the verb. We run this experiment using (1) ground truth SWiG boxes (in a random order to prevent the model from taking advantage of the role order), (2) detections from the ISL object detector, and (3) detections from RetinaNet pretrained on COCO. We get the below verb accuracy scores:

(1) SWiG ground truth: 63.94
(2) SWiG object detector: 13.22
(3) COCO object detector: 7.34

Not surprisingly, (1) has a very high accuracy as all objects involve the verb and the number of objects is fixed for a verb. This is likely a large over-estimation of our ideal experiment. However, (2) and (3) are well below JSL's verb accuracy of 39.60. While these are perhaps an underestimation of the ideal input due to detection errors, they indicate that situation prediction is difficult given only the label and location of objects.

E Qualitative

We present additional qualitative results further demonstrating the efficacy of JSL. Figure 12 shows a comparison between groundings generated by JSL and ISL for the same image. We illustrate these differences on a sample of images where both ISL and JSL are able to classify the nouns correctly, but ISL fails to correctly locate the entities in the frame. Here we show two common reasons that ISL fails to locate the correct object. The first 2 rows of Figure 12 demonstrate the case where there are multiple people in the scene and ISL is unable to pick the correct one for a given role. Because ISL cannot condition its detection on situation, it is often unable to select the correct object when there are multiple objects of the same category present in an image. The bottom 2 rows of Figure 12 show cases where ISL correctly locates the object, but fails to create an accurate bounding box around that object. This demonstrates a potential advantage of JSL as predicting the objects in sequence may allow for more accurate localization.

Additionally, Figure 13 shows the generated situations for different verbs given the same image. For each image we obtain the top 5 most probable verbs and then generate the grounded situations for each of these verbs. The top two rows of Figure 13 are examples where the top verb guess is correct. The first row demonstrates the model's ability to describe the scene in terms of the interaction between two participants as well as what actions they are doing together. In this case, it is clear that both girls are studying, but one is explaining something to

the other. Looking at multiple possible verbs captures these complexities. The following two rows are examples where the correct verb is in the top 5 and the bottom two rows show examples where the correct verb is not in the top 5. This tends to happen when the action is very unusual or occurring in a strange context, such as purposefully spilling a cup of water on a keyboard.

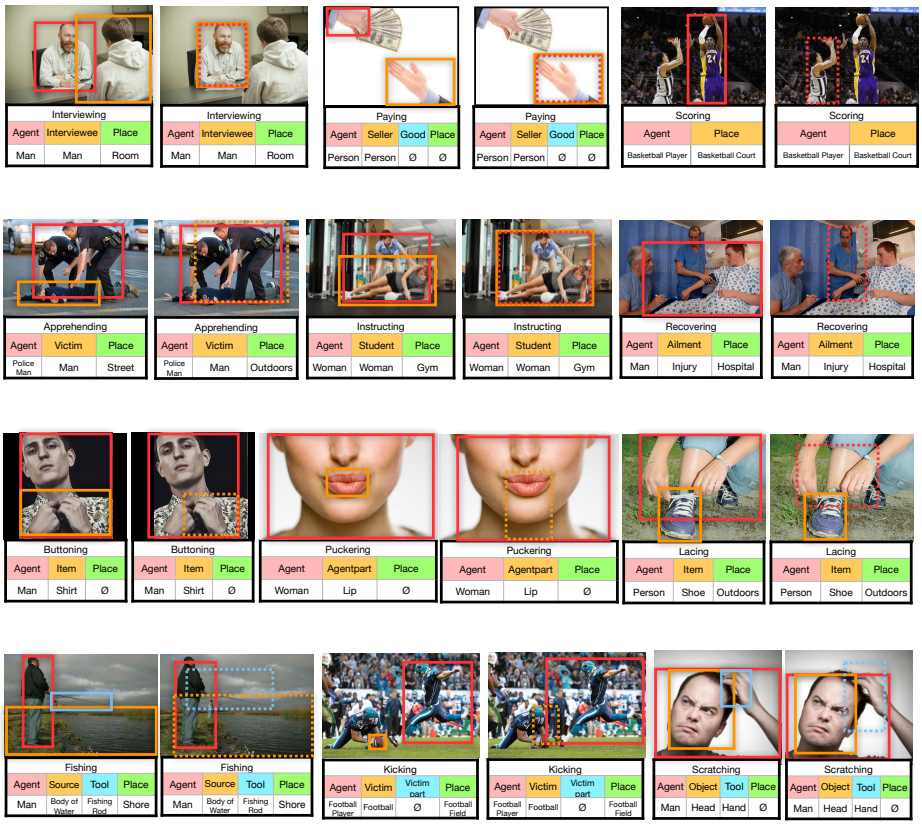

Fig. 12. For all images, the detections generated by JSL are shown first followed by the detections generated by ISL. Incorrect detections are shown with dotted lines and boxes are colored to correspond with roles.

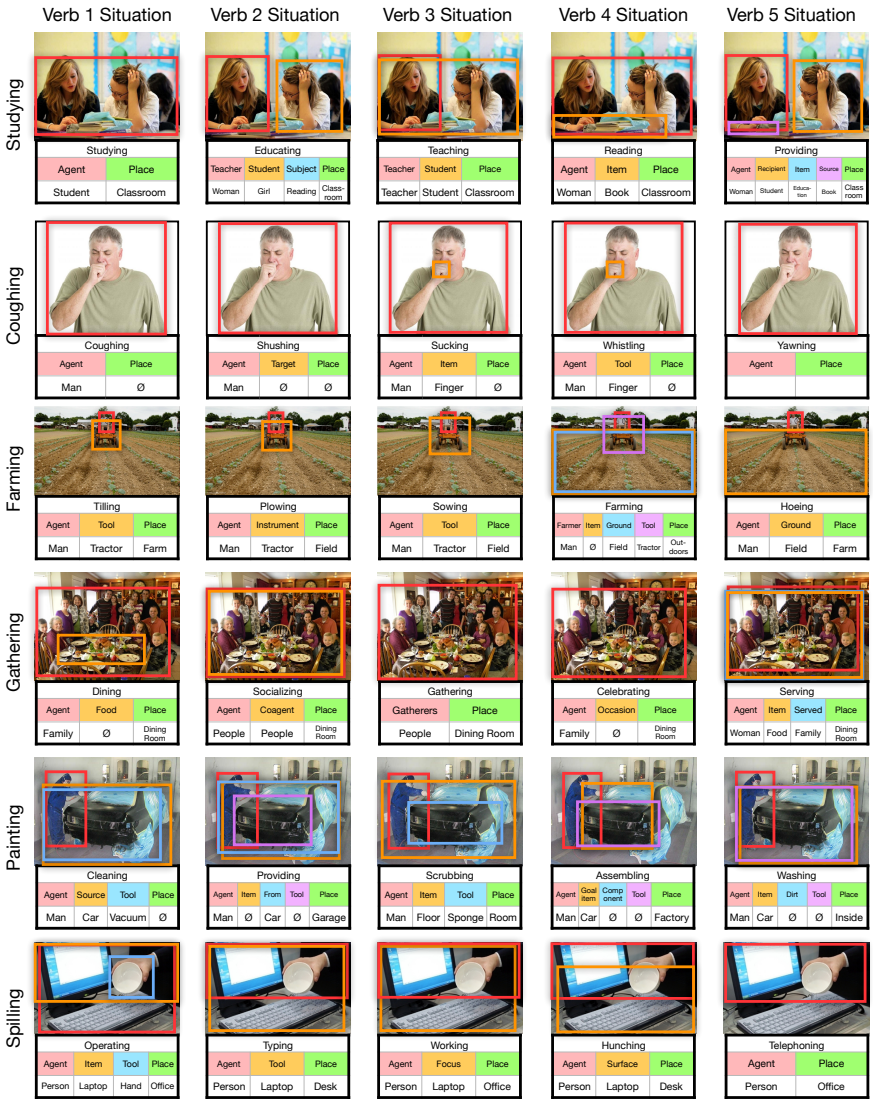

Fig. 13. Top-5 predictions for a sample of images in the SWiG dev set. The ground truth verb is indicated on the left of each row.