A Reproducibility

In this section we disclose the used parameters of some of the mentioned models for reproducibility purposes. The only models that are not mentioned here are RNM and RS as they just apply a random strategy.

A.1 OSM

We use a SSD-512 model to generate the region proposals. The model was initialized with weights from a model pretrained on ImageNet. We used stochastic gradient descent with initial learning rate 1e-3, momentum 0.9 and weight decay 5e-4 to optimize the loss. The learning rate was degraded by a factor 10 after 80,000 and 100,000 iterations. Batches of size 32 were used during training. Additionally, we found it important to use a warm-up scheme at the start of the training, where we gradually increased the learning rate from 1e-6 to 1e-3, after which we resumed the normal learning rate schedule. We included standard data augmentations during training, i.e., color jitter, random object crops, rescalings and horizontal flips.

The OSM model uses 64 region proposals extracted by the single-shot detection model. We used a ResNet-18 model, pretrained on ImageNet to encode the local regions. A bidirectional GRU model with one hidden layer of size 512 was used to encode the sentence. We optimized the loss with stochastic gradient descent with initial learning rate 1e-3, momentum 0.9 and weight decay 1e-4. We ignored the loss term when there were no region proposals with mean intersection over union larger than 0.5. The learning rate was reduced by a factor 10 when the validation performance became stagnant. We used batches of size 8.

A.2 SCRC

We reused the single shot detection model from before (see sec. A.1) to generate 64 region proposals per image. The local and global features were generated by two separate ResNet-18 models, both pretrained on ImageNet. We used a bidirectional GRU with 512 hidden units for the language model. The local and global recurrent context models use a uni-directional GRU with 512 hidden units. In the original paper they also pretrain the GRUs on the captioning task. We decided not to do this however as there was no captioning dataset that was close to the dataset described in this paper. These GRUs were thus initialised randomly. We reused the optimization scheme from section A.1 to train the model.

A.3 STACK-NMN

The images were first resized from 1600×900 to 512×512 before extracting the feature maps with ResNet-101. To extract the feature maps from this model we cut it off at the fourth channel. The output of the used ResNet model is a tensor of size $32 \times 32 \times 1024$ per image of size 512×512. The STACK-NMN model makes use of $Feat_H$ and $Feat_W$ parameters internally representing the amount of feature channels for the height and width. These were both set to 32. These parameters are important as they allow the network to transform the center of a ground truth bounding box to a cell in the 32×32 grid or vice versa. An other crucial parameter to the STACK-NMN model is the amount of reasoning steps. This influences both inference speed as well as accuracy. We tested the following values: [1, 2, 4, 6, 8, 9, 16] and found that 4 reasoning steps gave us the best results. The model was trained until the validation accuracy didn't increase over 10 epochs. The best model on the validation set is saved and used in the experiments. We also used a batch size of 64. The rest of the parameters were not changed and were left as the original parameters in the implementation.

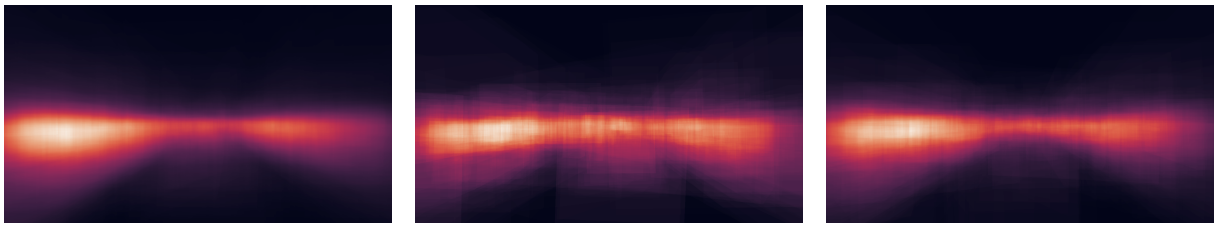
A.4 MAC

To extract the visual feature maps from the images in the `Talk2Car` dataset we reuse the method mentioned in STACK-NMN. We used the following parameters for the MAC model; We added L2-regularization to the model and used gradient clipping at 5. Next, Exponential Moving Average was also used for the weights of the model with a weight decay of 0.999. We experimented with different learning rates but found that 0.0001 gave us the best results when using the Adam optimizer. When the loss between two epochs didn't decrease more than 0.2 we multiplied the learning rate by 0.5 as is the default value in MAC. The following changes were made to transform MAC to the object referral task based on the implementation of STACK-NMN; We added a cos/sin based positional encoding to the feature maps by concatenation inspired by (Vaswani et al., 2017). We also use 32 for $Feat_H$ and $Feat_W$ to calculate the corresponding cell of the center of a bounding box. Instead of using the question and memory as the

input to the output unit, we used the pre-Softmax attention map from the last read unit of the reasoning process. This attention map is then passed to a fully connected layer that predicts the cell in which the center of the bounding box lies. With a convolutional layer we pass over the image to predict the offsets of the bounding box relative to the center of the bounding box. We also experimented with different amounts of reasoning steps ([1,2,3,4,8,10,12]) and found that with our modified version of MAC 10 reasoning steps worked the best for our task. The batch size was set to 32. The rest of the parameters remained unchanged to the original paper.

A.5 BOBB

The algorithm that is used for this model is described in Algorithm 1 in section A and has been used on the bounding boxes of the training set. A heatmap of the location of the objects in the training set can be seen in Fig. 4(a). The resulting bounding box that was found is: [0, 435, 445, 325] with format $[x_1, y_1, w, h]$. x_1 and y_1 represent the lower left corner of the bounding box. This found bounding box corresponds with the bias seen on the map.



(a) Training set (b) Validation set (c) Test set

Figure 4: The heatmaps of the locations of all objects in the training set (a), validation set (b) and test set (c) respectively.

Algorithm 1 BOBB Algorithm

```
procedure FindBestBBox(train_gt_bboxes, imgWidth=1600, imgHeight=900, threshold=0.5)
    bestAm ← 0
    bestBox ← None
    X ← linspace(0, ImgWidth, step = 5)
    Y ← linspace(0, ImgHeight, step = 5)
    for x1 in X do
        for x2 in X do
            if x2 ≤ x1 then continue
            for y1 in Y do
                for y2 in Y do
                    if y2 ≤ y1 then continue
                    box ← [x1, y1, x2 − x1, y2 − y1]
                    am ← getAmountOfIoUAboveThresh(box, train_gt_bboxes, threshold)
                    if am > bestAm then
                        bestAm ← am
                        bestBox ← box
    return bestBox
```