Supplementary material for "Improving Object Detection with Deep Convolutional Networks via Bayesian Optimization and Structured Prediction"

Yuting Zhang*†, Kihyuk Sohn†, Ruben Villegas†, Gang Pan*, Honglak Lee†

* Department of Computer Science, Zhejiang University, Hangzhou, Zhejiang, China, {zyt,gpan}@zju.edu.cn

† Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA, {yutingzh,kihyuks,rubville,honglak}@umich.edu

Contents

S1. Parameter estimation for finetuning with structured SVM objective

The model parameters are updated via gradient descent. The gradient, for example, with respect to the CNN parameters Θ ($\neq w$) for positive examples is given as follows:

$$\frac{\partial h_{\text{pos},i}}{\partial \Theta} = \begin{cases} 0 & , \ h_{\text{pos},i} = 0 \\ -w^\top \frac{\partial \phi(x_i, y_i)}{\partial \Theta} & , \ h_{\text{pos},i} = \ 1 - w^\top \phi(x_i, y_i) \\ w^\top \frac{\partial (\phi(x_i, \hat{y}_i) - \phi(x_i, y_i))}{\partial \Theta} & , \ \text{otherwise} \end{cases} \tag{S-1}$$

where $\hat{y}_i = \arg\max_{y \in \mathcal{Y}_i^{(l=1)}} \left(w^\top \phi(x_i, y) + \Delta(y, y_i) \right)$. Similarly, the gradient for negative examples can be computed as follows:

$$\frac{\partial h_{\text{neg},i}}{\partial \Theta} = \begin{cases} 0 & , \ h_{\text{neg},i} = 0 \\ w^\top \frac{\partial \phi(x_i, \hat{y}_i)}{\partial \Theta} & , \ h_{\text{neg},i} = 1 + w^\top \phi(x_i, \hat{y}_i) \end{cases} \tag{S-2}$$

where $\hat{y}_i = \arg\max_{y \in \mathcal{Y}_i^{(l=1)}} w^\top \phi(x_i, y)$. The gradient with respect to the parameters of all layers of CNN can be computed efficiently using backpropagation. When finetuning the entire network, the parameter updated in the hard mining procedure illustrated by Algorithm S-1 is done by replacing $\hat{w}$ with the CNN parameters.

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

S2. Details on hard negative data mining

The active set consisting of the hard training instances are updated in two steps during the iterative learning process. First, we include instances $y \in \mathcal{Y}_i$ to the active set when they are likely to be active, i.e., affect the gradient:

$$w^\top (\phi(x_i, y) - \phi(x_i, y_i)) + \Delta^{\text{loc}}(y, y_i) \geq \max\left\{0, 1 - w^\top \phi(x_i, y_i)\right\} - \epsilon_1, \forall i \in I_{\text{pos}} \tag{S-3}$$

$$1 + w^\top \phi(x_i, y) \geq -\epsilon_1, \forall i \in I_{\text{neg}} \tag{S-4}$$

Second, once new parameters are estimated, we exclude instances from the current active set when they are likely to be inactive, i.e., have no effect on the gradient:

$$w^\top (\phi(x_i, y) - \phi(x_i, y_i)) + \Delta^{\text{loc}}(y, y_i) \leq \min\left\{0, 1 - w^\top \phi(x_i, y_i)\right\} - \epsilon_2, \forall i \in I_{\text{pos}} \tag{S-5}$$

$$1 + w^\top \phi(x_i, y) \leq -\epsilon_2, \forall i \in I_{\text{neg}} \tag{S-6}$$

In our experiments, we used $\epsilon_1 = 0.0001$ and $\epsilon_2 = 0.2$. The values of ϵ_1, ϵ_2 are the same as those for the SVM training in R-CNN [4]. We did not observe a noticeable performance fluctuation due to different ϵ_1, ϵ_2 values. Algorithm S-1 summarizes the hard-mining procedure.

Algorithm S-1 Parameter estimation with hard mining

Require: Initial parameters w_0, maximum epoch number $\texttt{epoch}_{\text{max}}$, training images $\{x_i, y_i\}_{i=1}^M$, positive and negative index $I_{\text{pos}}, I_{\text{neg}}$

Ensure: Final parameters $\hat{w}$.

The active set $\mathcal{A}, \leftarrow \{(x_i, y_i) : i \in I_{\text{pos}}\}$
$\mathcal{A}, \mathcal{A}_{\text{inc}} \leftarrow \varnothing$, $\hat{w} \leftarrow w_0$
for $\texttt{epoch} = 1, \ldots, \texttt{epoch}_{\text{max}}$ **do**
  for $i = 1, \ldots, M$ **do**
    $\mathcal{A}_{\text{inc}} \leftarrow \mathcal{A}_{\text{inc}} \cup \{(y, x_i) : y$ s.t. (S-3), (S-4) for $x_i, \hat{w}\}$
    if $|\mathcal{A}_{\text{inc}}| \geq \texttt{update\_threshold}$ or $i == M$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_{\text{inc}}$, $\mathcal{A}_{\text{inc}} \leftarrow \varnothing$
      update the classifier/network parameters $\hat{w}$ on $\mathcal{A}$
      $\mathcal{A} \leftarrow \mathcal{A} \backslash \{(y, x) \in \mathcal{A} : y$ s.t. (S-5), (S-6) for $x, \hat{w}\}$
    end if
  end for
end for

S3. Implementation details on model parameter estimation

For our experiments on PASCAL VOC 2007 and VOC 2012, we first finetune the CNN pretrained on ImageNet by stochastic gradient descent with a 21-way softmax classification layer, where 20 ways are for the 20 object categories of interest, and the rest 1 way is for the background. In this step, the SGD learning rate starts at 0.0003, and decreases by 0.3 every 15000 iterations with a mini-batch size of 48. We set the momentum to 0.9, and the weight decay to 0.0005 for all the layers.

After that, we replace the softmax loss layer with a 20-way structured loss layer, where each way is a binary classifier, and the hinge loss for different category are simply summed up.

For classification layer only learning, L-BFGS is adopted, as batch gradient descent for a single layer. Each category has an associated active set for hard negative mining. The classifier update happens independently for each category when 5000 (`update_threshold` in Algorithm S-1) new hard examples are added to the active set. It is worth mentioning that, in the beginning of the hard negative mining, significantly more positive images are present than the negative images, resulting in serious unbalance of the active training samples. As a heuristic to avoid this problem, we limit the number of positive image to the number of the active negative images when classifier update happens in the first epoch. We run the hard negative mining for 2 epochs in total. The first epoch is for initializing the active set with the above heuristic, and the rest is for learning with the all the training data. Compared to the linear SVM training in R-CNN [4], our L-BFGS based solution to the structured objective costs $8 \sim 10$x longer time. However, it turns out to be significantly more efficient than SVM$^{\text{struct}}$ [6].

For the entire network finetuning, we initialize the structured loss layer with the weights obtained bythe classification-layer-only learning. The whole network is then finetuned by backpropagating the gradient from the top layer with a fixed SGD learning rate of 10^{-6}. For implementation simplicity, we keep updating the active sets until the end of an epoch, and update the classifiers per epoch (i.e., `update_threshold` $= +\infty$ in Algorithm S-1). Like before, each category still has one particular active set. However, the network parameters (except for the classifier) are shared across all the category so that the feature extraction time is not scaled up with the number of categories. In practice, we found one epoch was enough for both hard negative mining and SGD in the entire network finetuning case. Running more epochs did not make noticeable improvement on the final detection performance on PASCAL VOC 2007 test set, but cost a significantly larger amount of training time.

S4. Efficiency of fine-grained search (FGS)

In this section, we provide more details on the local FGS presented in Algorithm 1 of the main text.

GPR practical efficiency: For the initial proposals given by selective search, $|\mathcal{D}_{\text{local}}|$ usually turns out to be 20 to 100, and line 9,10 can be efficiently solved in around 9 and 6 L-BFGS [8] iterations for StructObj, respectively.

GPU parallelism for CNN: One image can have multiple search regions (e.g., line 6), and 20 object categories together yield more regions. FGS proposes one extra box per iteration for every search region. These boxes are fed into the CNN together to utilize GPU parallelism. For VGGNet, we use the batch size of 8 for computing the CNN features within the FGS procedure.

Time overhead: For PASCAL VOC 2007, FGS ($t_{\max} = 8$) induced only $\sim 15\%$ total overhead compared to initial time cost, which mostly consists of CNN feature extraction from bounding boxes proposed by selective search (SS). Specifically, $1/3$ of the overhead is caused by CNN feature extraction from the newly proposed boxes (line 11); the rest is caused by GPR (line9, 10), NMS (line 5), and pruning (line 12). Each GP iteration (line 2-16) counts for $\sim 2\%$ with respect to the initial time cost, and ≤ 8 GP iterations were sufficient for convergence. Figure S-1 shows the trends of the accumulated time overhead introduced by FGS per iteration. The time overhead due to FGS may vary with different datasets (e.g., VOC 2012), but in general, it is $\leq 20\%$ compared to initial time cost.

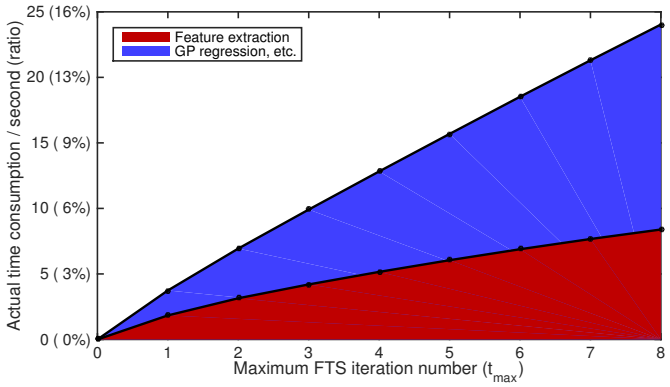


Figure S-1: Time cost with different maximum number of iterations. The "ratio" is with respect to the time cost of extracting features for the initial bounding boxes.

S5. Step-wise performance of fine-grained search (FGS)

We evaluated the mAP at each GP iteration using R-CNN(VGG)+StructObj+FGS+BBoxReg. The mAPs from 0 to 8 GP iterations are reported in Table S-1. mAP increases rapidly in the first 4 iterations, and becomes stable in the following iterations.

S6. Test set mAP on PASCAL VOC 2007 using VGGNet with different region proposal methods

In Figure 2 of the paper, we report and compare the test set mAPs on PASCAL VOC 2007 with different region proposal algorithms (e.g., selective search (SS) [9] at different modes, Objectness [1]) using an oracle detector. In this section, we

# GP iter	0	1	2	3	4	5	6	7	8
mAP	66.6	67.5	67.8	68.2	68.3	68.6	68.4	68.6	68.5

Table S-1: Test set mAPs on PASCAL VOC 2007 for "R-CNN(VGG)+StructObj+FGS+BBoxReg" with different number of GP iterations

performed similar experiments using a real detector trained with structured SVM objective based on VGGNet features. The summary results are given in Table S-2 and S-3.

Region proposal methods \ IoU threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
SS (∼2000 boxes per image)	74.3	73.8	72.5	69.6	61.2	47.4	31.2	15.4
SS + Objectness (∼3000 boxes per image)	73.4	72.9	71.7	68.9	60.6	47.0	31.1	15.3
SS extended (∼3500 boxes per image)	74.3	73.9	72.8	70.1	62.8	48.5	32.3	15.9
SS quality (∼10000 boxes per image)	74.1	73.7	72.7	70.0	63.7	51.4	35.7	**17.9**
SS + FGS (∼2150 boxes per image)	**76.6**	**76.1**	**75.0**	**72.4**	**65.8**	**54.1**	**37.2**	17.4

Table S-2: Test set mAPs on PASCAL VOC 2007 with different region proposal methods at varying IoU thresholds from 0.1 to 0.8 without bounding box regression.

Region proposal methods \ IoU threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
SS (∼2000 boxes per image)	77.4	76.9	75.9	73.4	66.5	55.8	40.9	19.4
SS + Objectness (∼3000 boxes per image)	76.9	76.5	75.5	73.0	66.0	55.2	40.4	19.3
SS extended (∼3500 boxes per image)	77.6	77.2	76.2	73.8	67.6	57.0	41.8	19.7
SS quality (∼10000 boxes per image)	77.1	76.7	75.8	73.3	67.1	57.0	42.3	19.4
SS + FGS (∼2150 boxes per image)	**78.3**	**77.8**	**76.8**	**74.4**	**68.5**	**57.9**	**43.1**	**20.4**

Table S-3: Test set mAPs on PASCAL VOC 2007 with different region proposal methods at varying IoU thresholds from 0.1 to 0.8 with bounding box regression.

For both cases (with and without bounding box regression), the FGS showed improved performance over other region proposal methods using smaller number of region proposals. In particular, the SS + FGS method (row 5 in Table S-2 and S-3) even outperformed the SS "quality" mode [9], which requires $\sim 5\times$ more computational expenses than our proposed method to compute CNN-based detection scores for bounding box proposals.

Although the current state-of-the-art CNN-based detector outperforms other object detection methods [3, 7, 2] by a large margin, there still remains a significant gap with that of the hypothetical oracle detector. This motivates us to further research on improving the quality of the CNNs for better visual object recognition performance.

S7. Precision-recall curves on PASCAL VOC 2007

In this section, we present the precision-recall curves for four different models. Specifically, we show results for VGGNet, VGGNet trained with structured SVM objective (VGGNet + StructObj), VGGNet with FGS (VGGNet + FGS), and VGGNet with both (VGGNet + StructObj + FGS) in Figure S-2. In general, the improvement from the structured SVM objective is more significant for the high recall range (i.e., recall ≥ 0.5) than the low recall range other than "sheep" class. FGS usually improves the precision for most object categories.

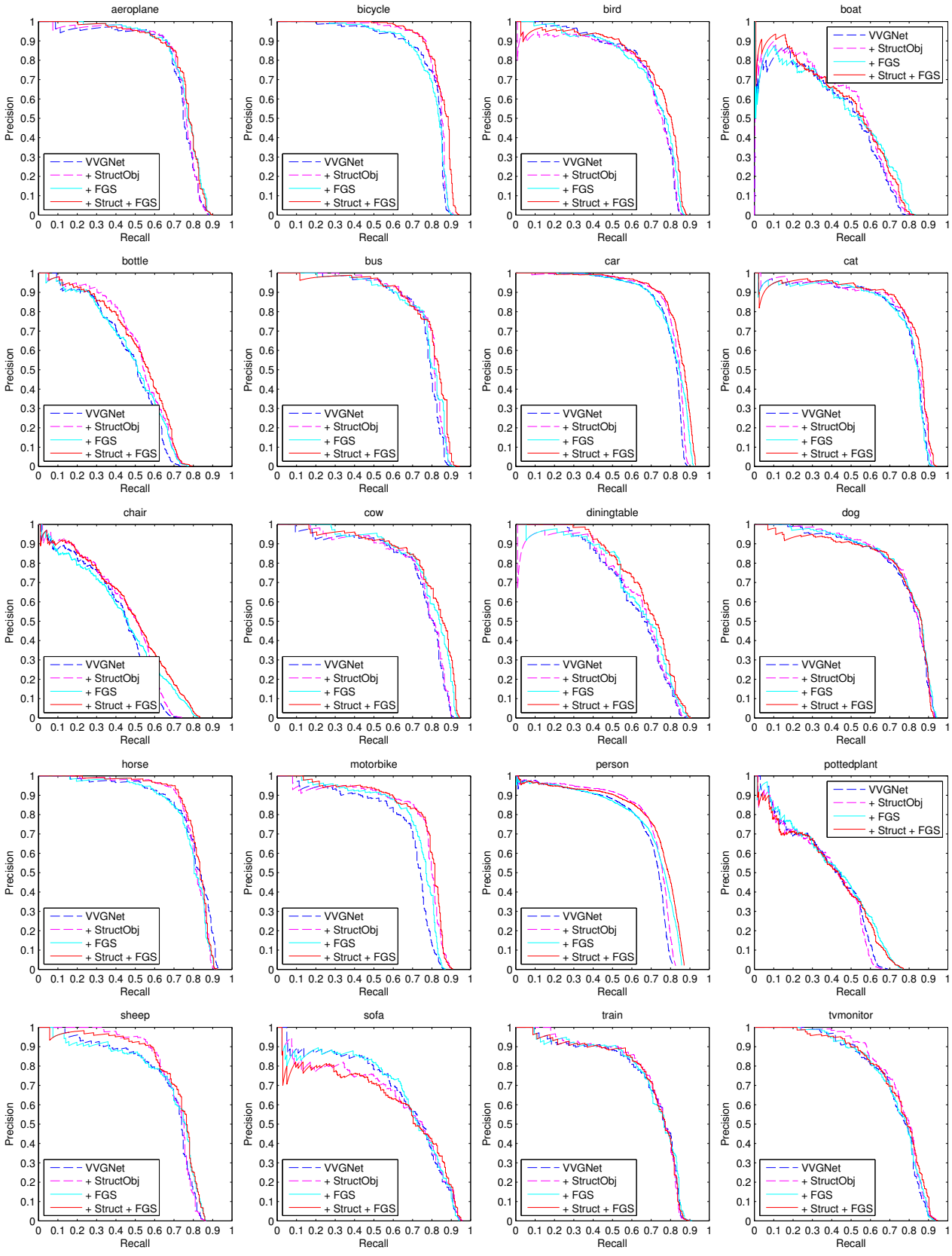


Figure S-2: Category-wise precision-recall curves for four different models (VGGNet, + StructObj, + FGS, + StructObj + FGS) on PASCAL VOC 2007 database.

S8. Localization accuracy on PASCAL VOC 2007

In this section, we analyze the localization behavior of different methods. We find the predicted bounding boxes that most accurately localize the ground truth from each method by picking the detected box with the highest overlap (i.e., IoU) with respect to each ground truth bounding box. Comparisons between the different methods are performed by estimating the distribution of the overlaps for every category. Our findings are shown in Figure S-3 (without BBoxReg) and Figure S-4 (with BBoxReg + the baseline without BBoxReg). Methods with better localization ability should have higher frequency at higher IoU (i.e., IoU between 0.6 and 0.9) and lower frequency at lower IoU (i.e., IoU between 0 and 0.4). Curve peaks leaning to the right signify better localization.

As shown in Figure S-3 and S-4, individually applying FGS and StructObj results in better localization compared to the baseline R-CNN, regardless of using or not using bounding box regression. However, the performance improvement due to FGS and StructObj independently results in different phenomena. In general, we found that FGS pushes the distribution peak to the right, and StructObj pulls the distribution peak higher while pushing the frequencies in the low IoU interval down. This indicates that FGS can propose more accurately localized boxes if the original set of bounding boxes are reasonably well localized, and StructObj can make detection scores more accurate (i.e., give low detection scores to boxes with low overlap and high detection scores to boxes with high overlap) based on the overlap of the proposed bounding boxes with respect to the ground truth. Combining FGS and StructObj together capitalizes on the advantages of both, and leads to the best localization accuracy.

S9. Examples with the largest improvement on PASCAL VOC 2007 test set

In this section, we show examples with the largest improvement in localization accuracy using our best proposed method (VGGNet + StructObj + FGS) over the baseline detection method (VGGNet) from PASCAL VOC 2007 test set. For each example in Figure S-5, we show the category of interest on the left-bottom corner of the image, and draw the detection of our best proposed method (in yellow box) that is best matched with the particular ground truth (in green box) and the best matched detection of the baseline (in red box). The number on the top right of the detected bounding box denotes the IoU with the ground truth.

S10. Top-ranked false positives on PASCAL VOC 2007 test set

In this section, we show examples of the top-ranked false positive detections[1] (in green box) of our best proposed method (VGGNet + StructObj + FGS) from PASCAL VOC 2007 test set (Figure S-6). We categorize the false positives into four categories as in [5]:

- loc: poor localization,
- sim: confusion with similar objects,
- oth: confusion with other objects,
- bg: confusion with background or unlabeled objects.

The overlap (ov) measured by the IoU between the false positive detection and its best matching ground truth bounding box is provided. For "loc" examples, the closest bounding box annotated with the same object category is provided as a ground truth (in yellow box). For "sim" or "oth" examples, the closest bounding box annotated with any object category is provided as a ground truth (in red box).

S11. Random detection examples on PASCAL VOC 2007 test set

Finally, we show randomly selected detection examples of our best proposed method (VGGNet + StructObj + FGS) from PASCAL VOC 2007 test set. In Figure S-7, we use bounding boxes with different colors for different categories. The category label with detection score is displayed on the top-left corner of each bounding box. Detections with low scores are ignored.

[1] The top-ranked false positives are selected among false positive bounding boxes with the highest detection scores.

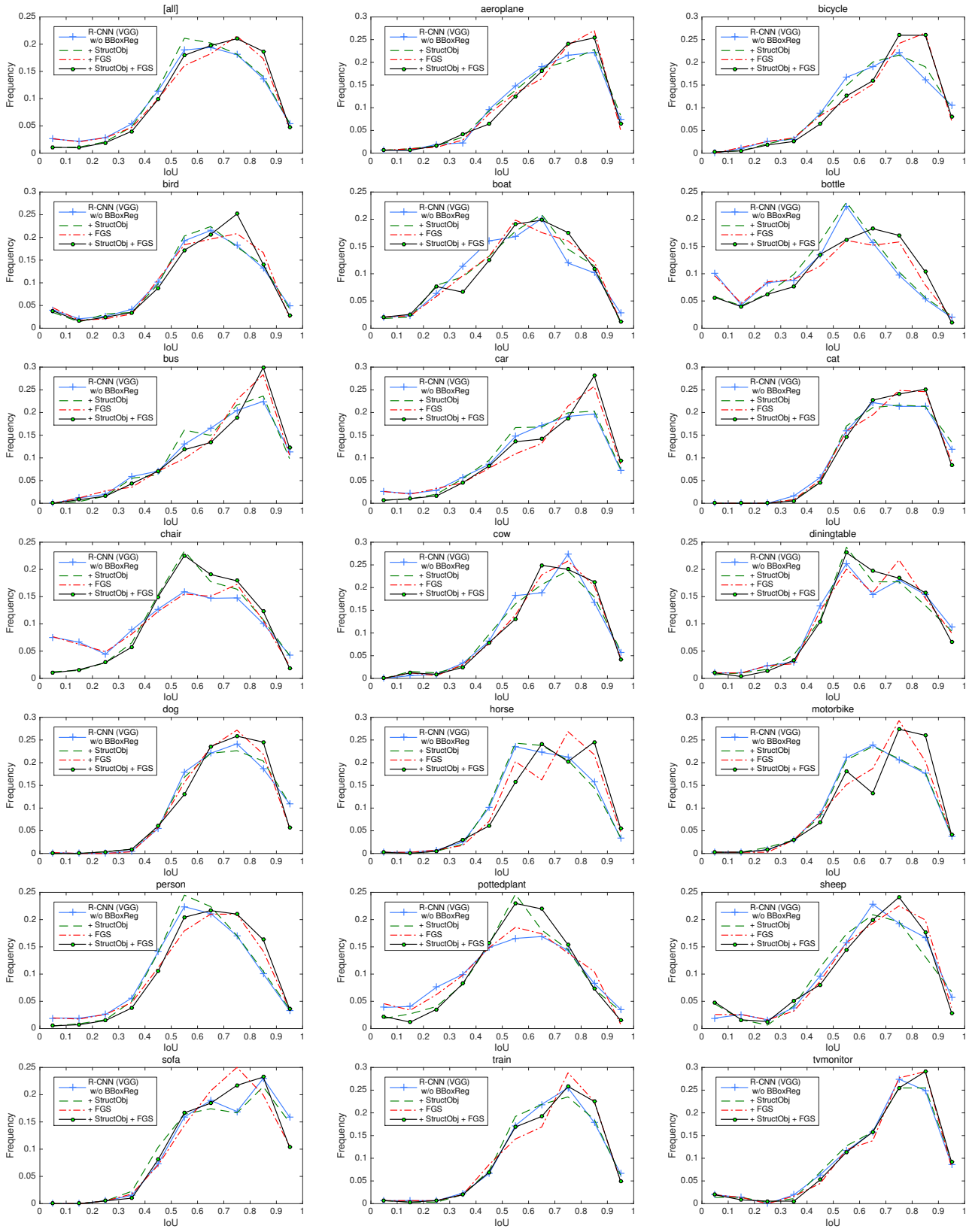


Figure S-3: Category-wise localization accuracy (in terms of the IoU between a ground truth annotation and its closest detected box) distributions for four different models (VGGNet without BBoxReg, + StructObj, + FGS, + StructObj + FGS) on PASCAL VOC 2007 datasets.

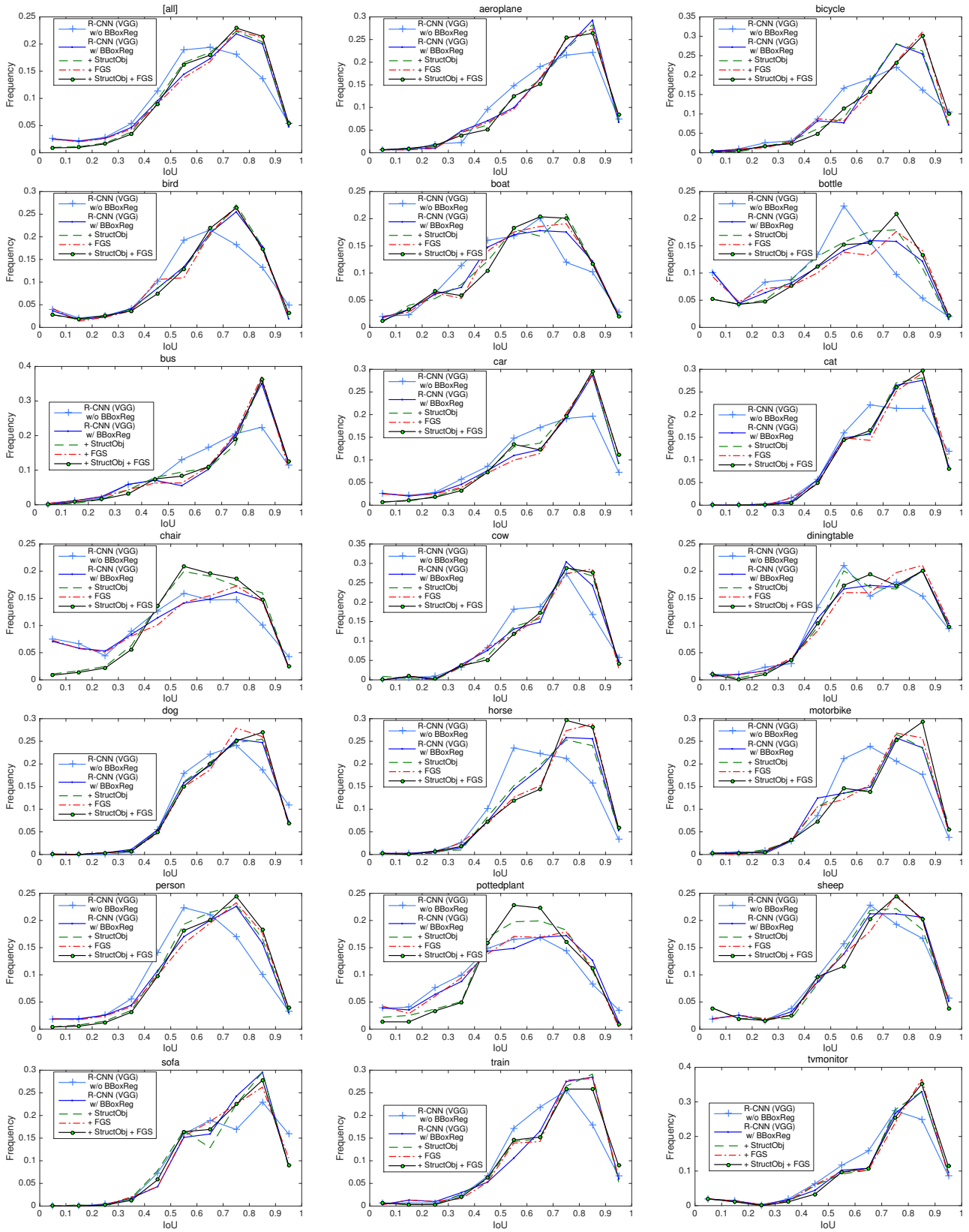
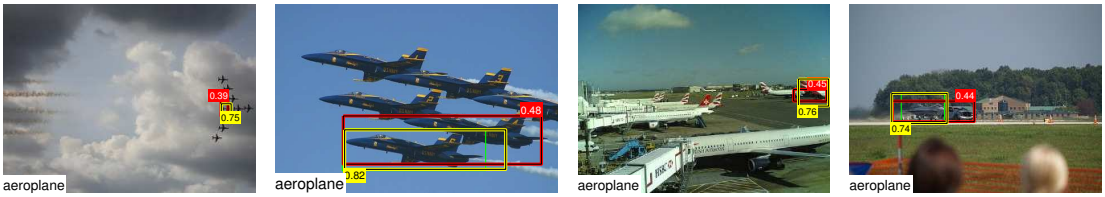
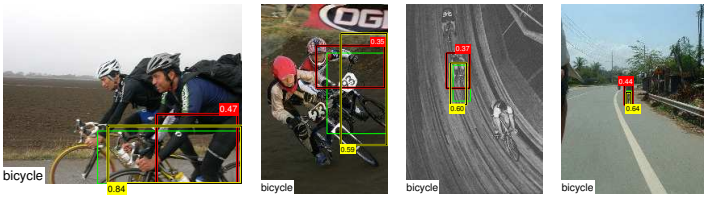


Figure S-4: Category-wise localization accuracy (in terms of the IoU between a ground truth annotation and its closest detected box) distributions for five different models (VGGNet without BBoxReg; VGGNet with BBoxReg, + StructObj, + FGS, + StructObj + FGS) on PASCAL VOC 2007 datasets.



aeroplane (examples with the largest improvement)



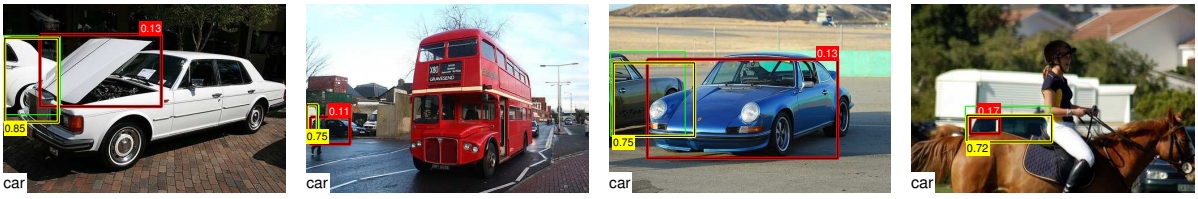
bicycle (examples with the largest improvement)

bird (examples with the largest improvement)

boat (examples with the largest improvement)

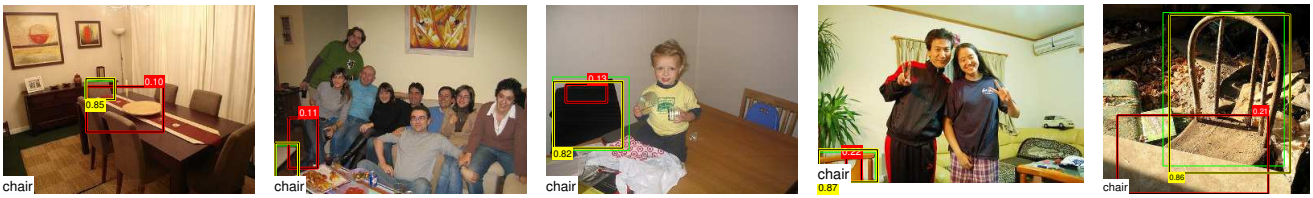
bottle (examples with the largest improvement)

bus (examples with the largest improvement)

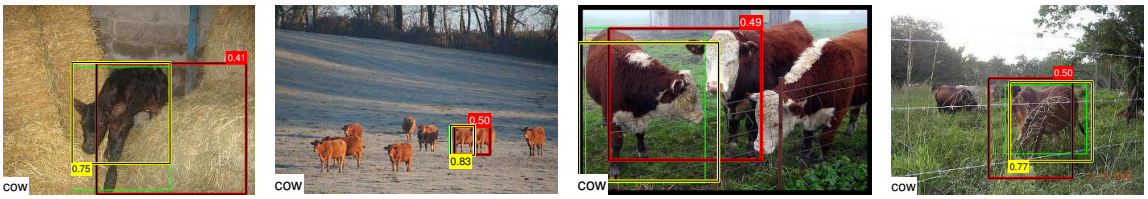


car (examples with the largest improvement)

cat (examples with the largest improvement)

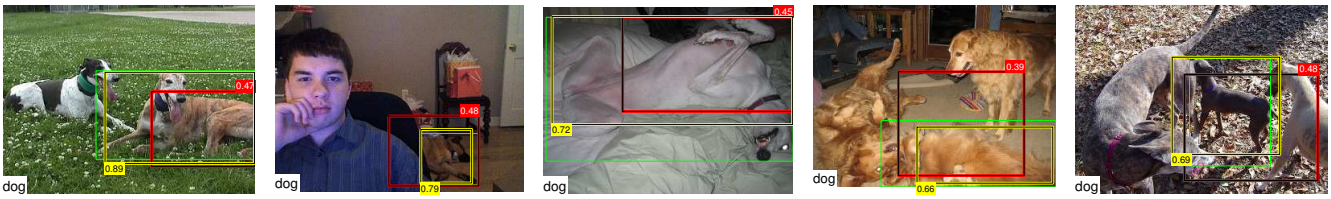


chair (examples with the largest improvement)

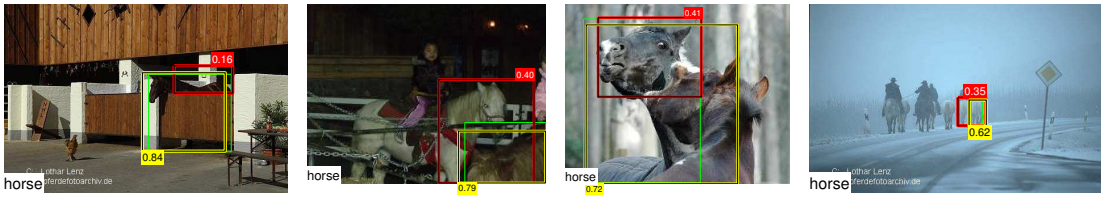


cow (examples with the largest improvement)

diningtable (examples with the largest improvement)

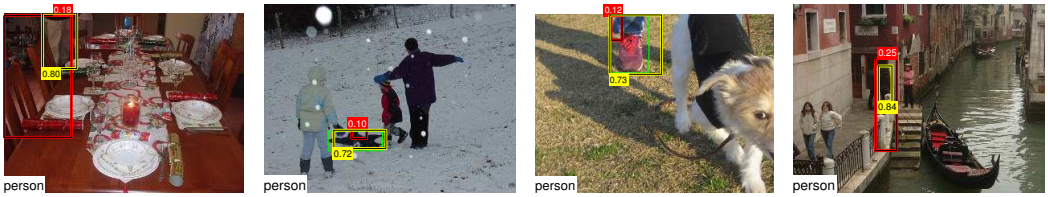


dog (examples with the largest improvement)



horse (examples with the largest improvement)

motorbike (examples with the largest improvement)



person (examples with the largest improvement)

pottedplant (examples with the largest improvement)

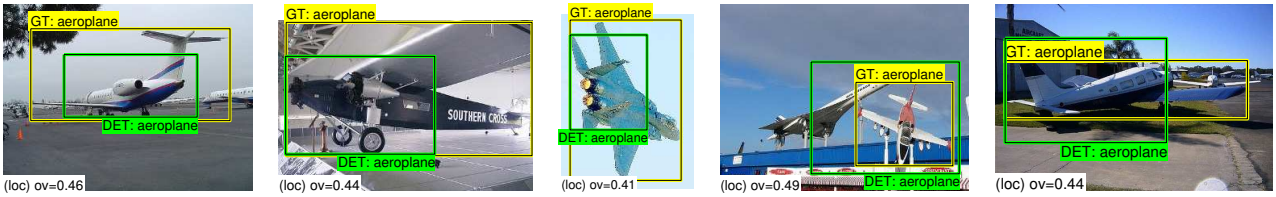
sheep (examples with the largest improvement)

sofa (examples with the largest improvement)

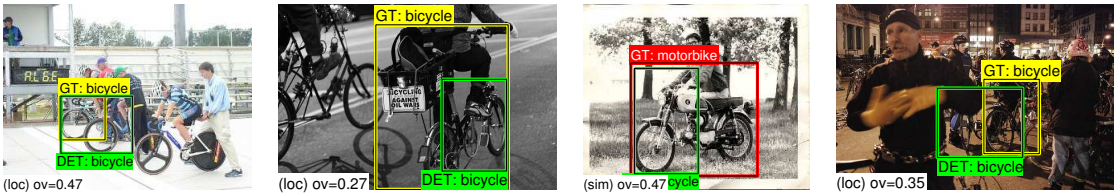
train (examples with the largest improvement)

tvmonitor (examples with the largest improvement)

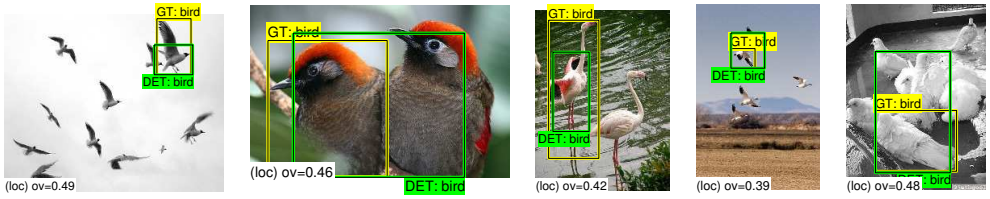
Figure S-5: Examples with the largest improvement with regards to the baseline method on PASCAL VOC 2007 test set. Refer to the text for more detail.



aeroplane (top-ranked false positive)

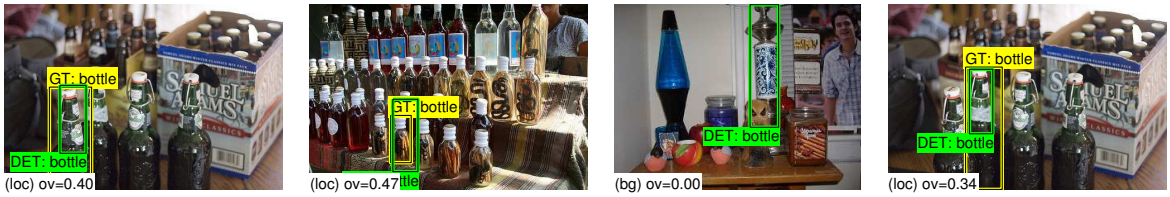


bicycle (top-ranked false positive)

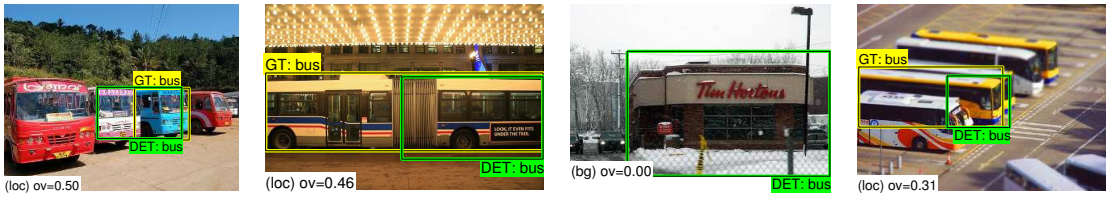


bird (top-ranked false positive)

boat (top-ranked false positive)

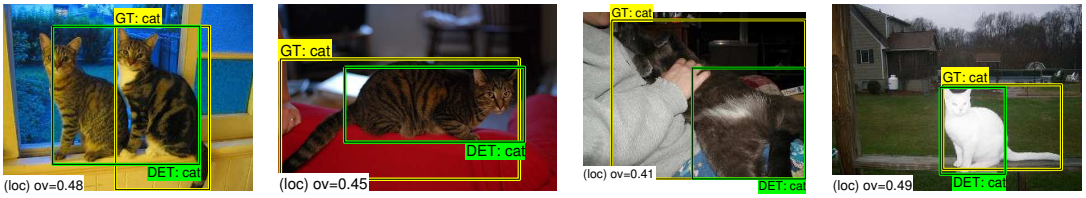


bottle (top-ranked false positive)

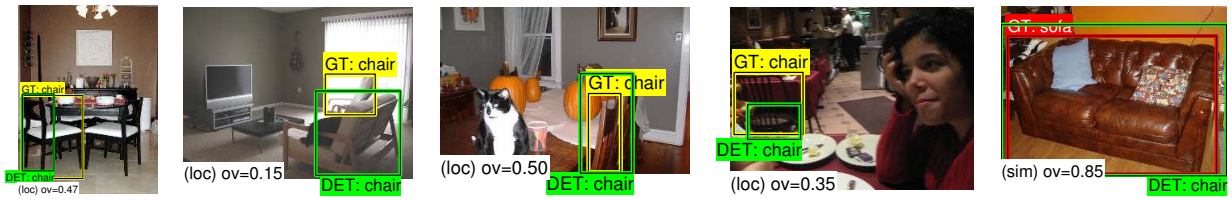


bus (top-ranked false positive)

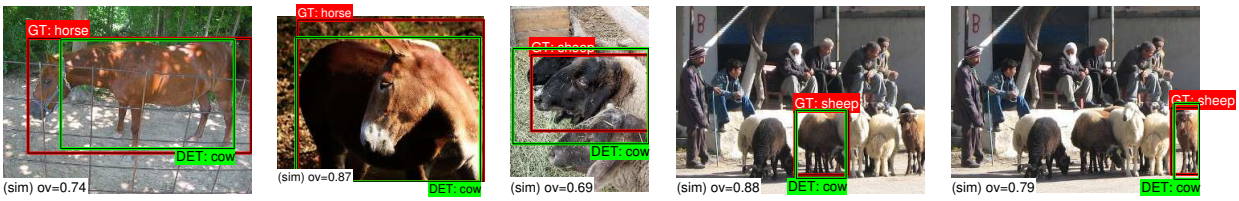
car (top-ranked false positive)



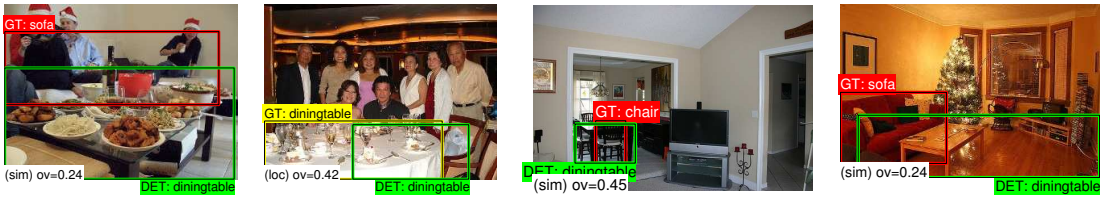
cat (top-ranked false positive)



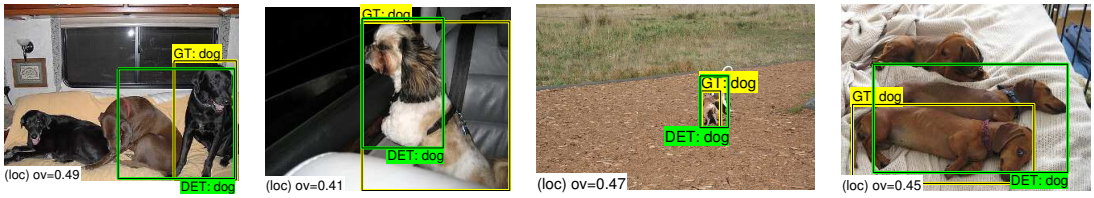
chair (top-ranked false positive)



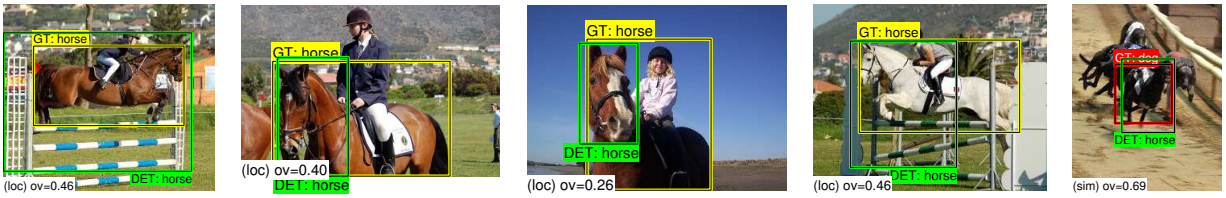
cow (top-ranked false positive)



diningtable (top-ranked false positive)

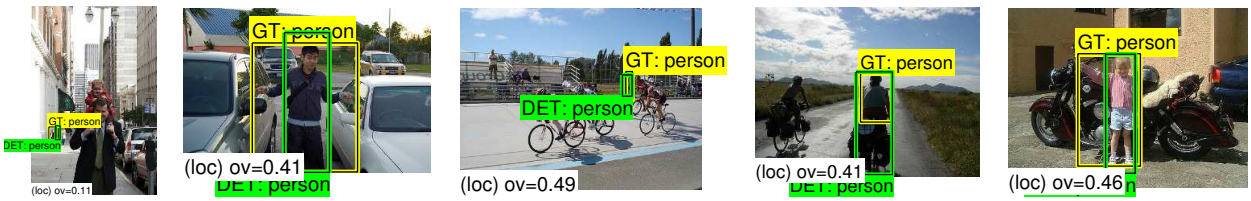


dog (top-ranked false positive)

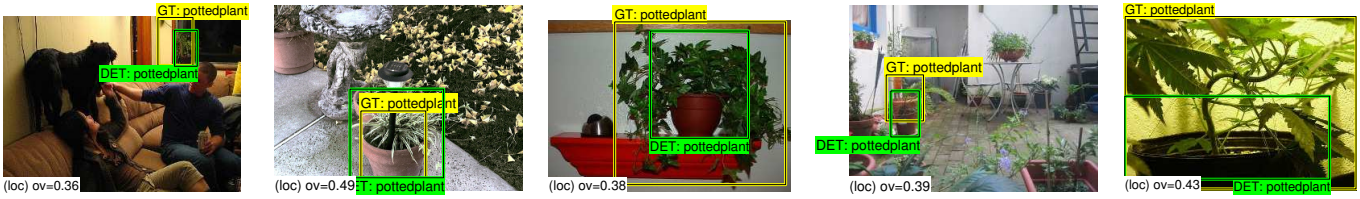


horse (top-ranked false positive)

motorbike (top-ranked false positive)

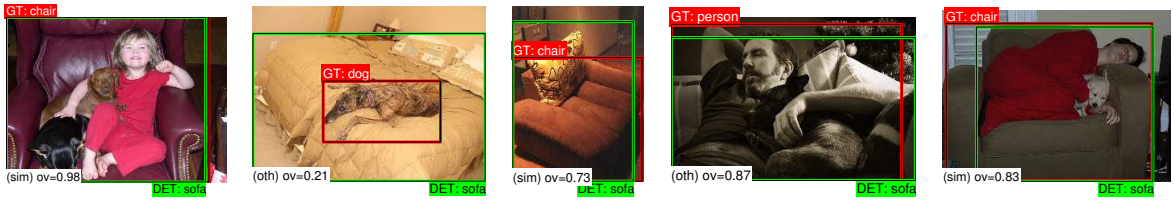


person (top-ranked false positive)

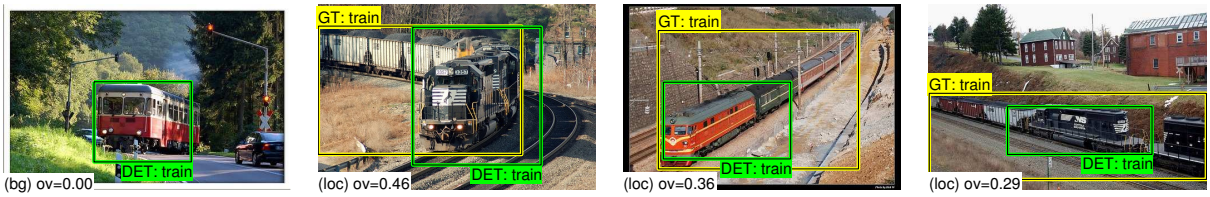


pottedplant (top-ranked false positive)

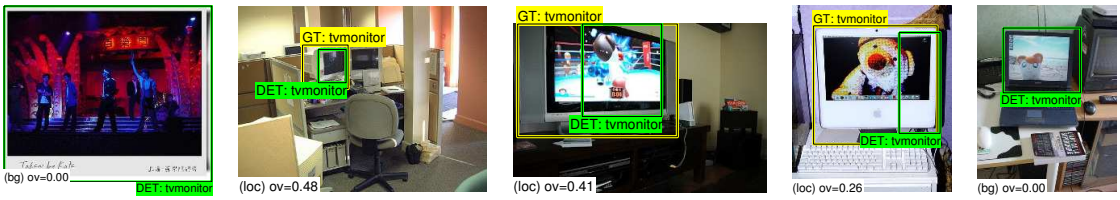
sheep (top-ranked false positive)



sofa (top-ranked false positive)

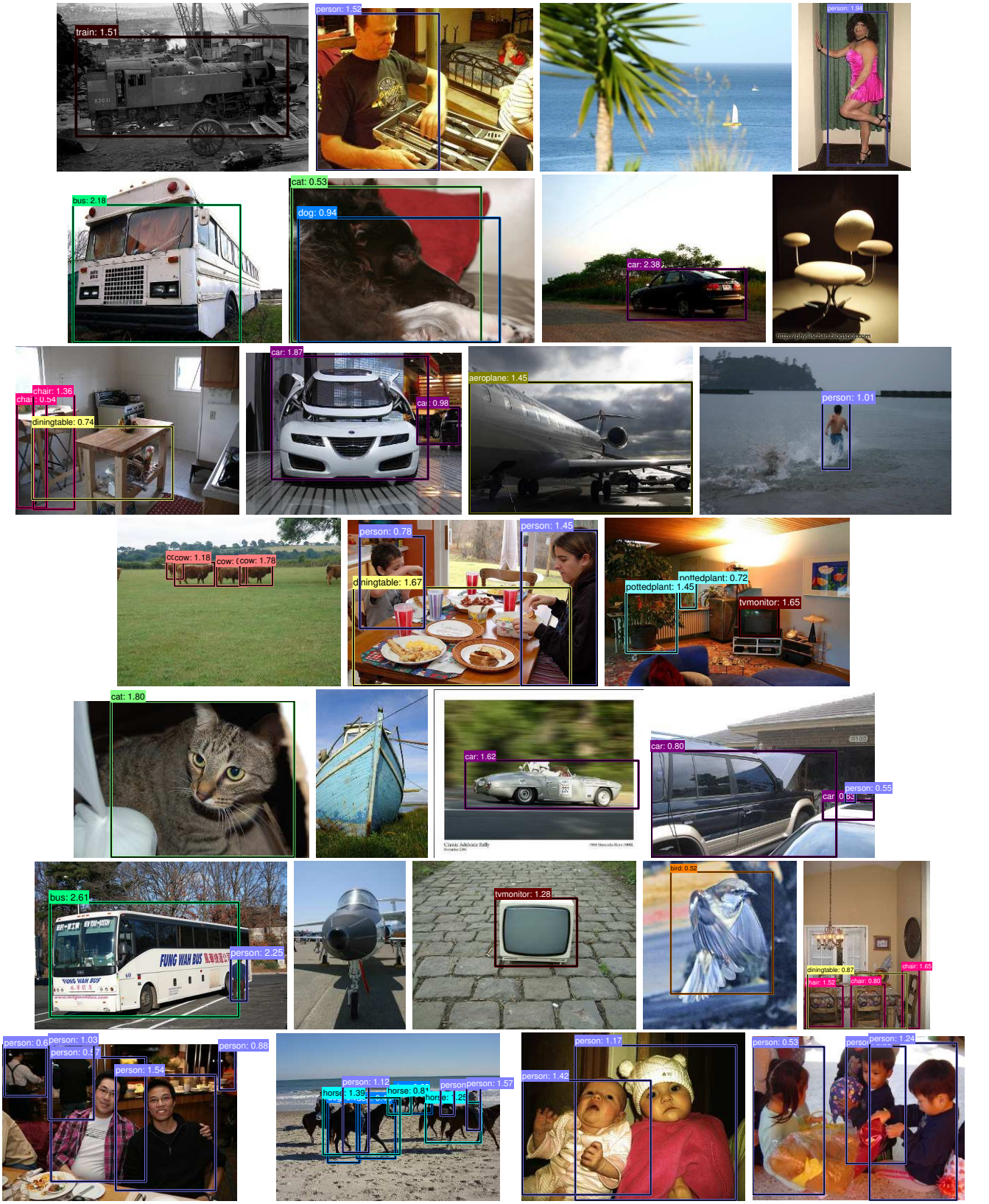


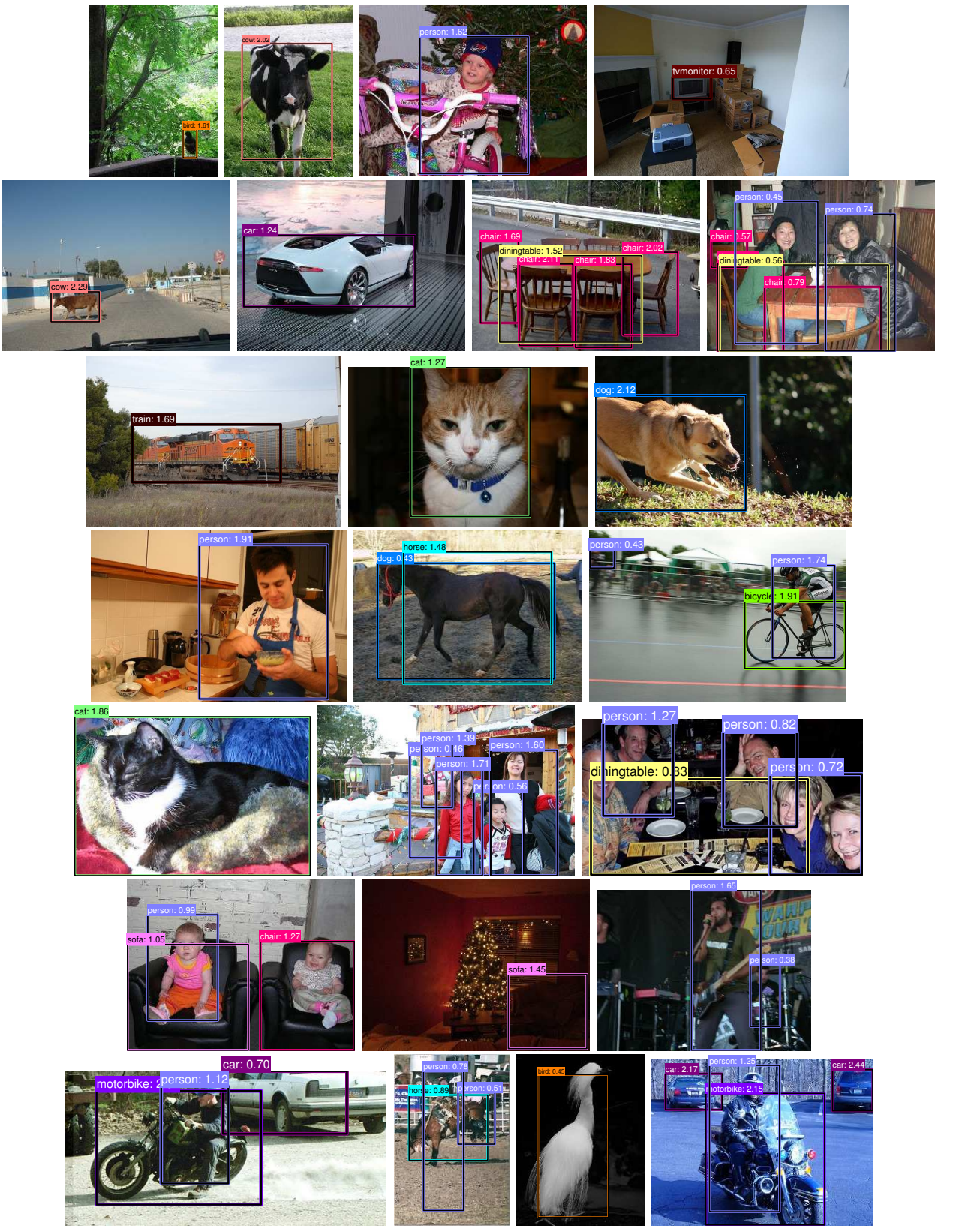
train (top-ranked false positive)

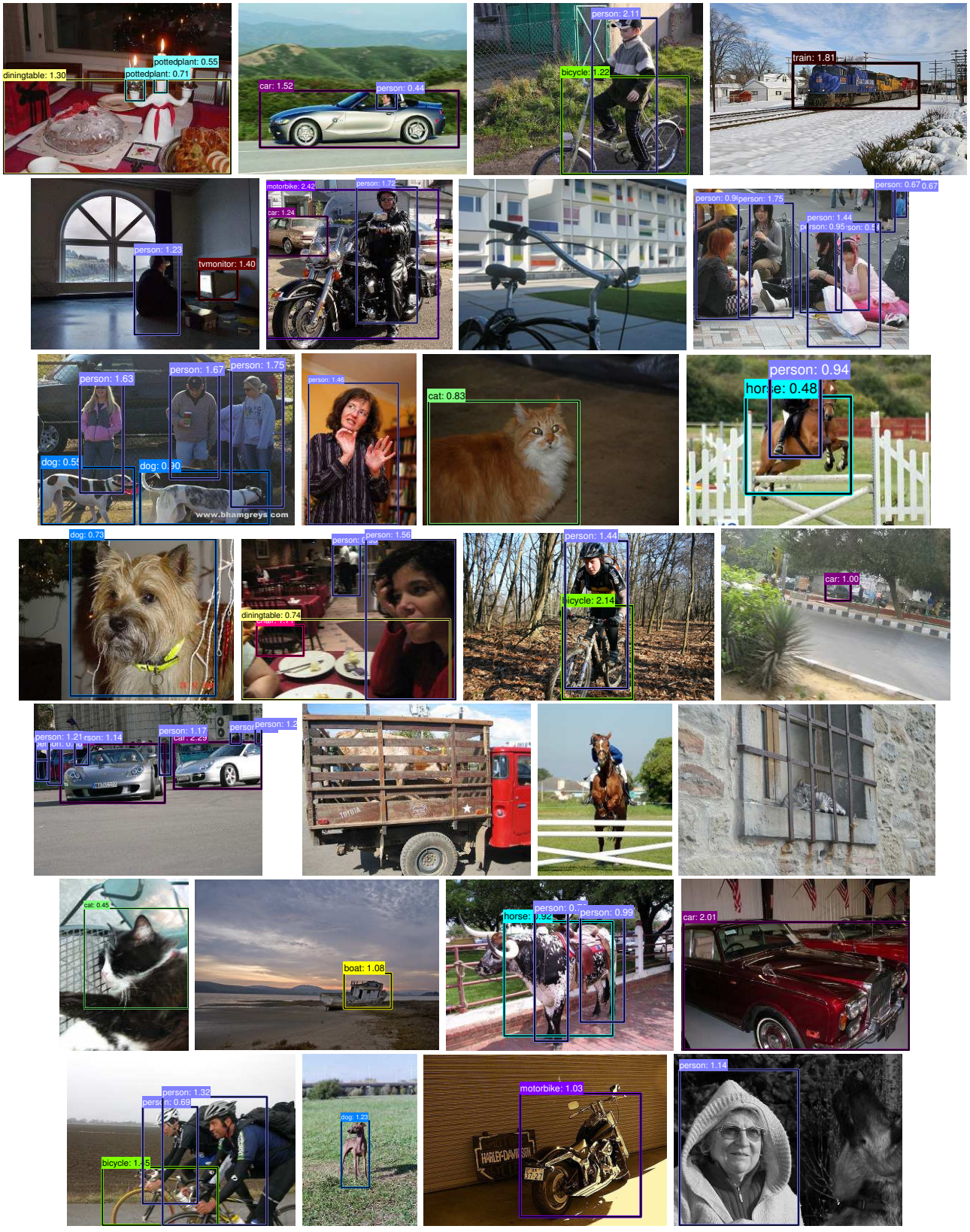


tvmonitor (top-ranked false positive)

Figure S-6: Top-ranked false positives of "VGGNet + StructObj + FGS" on PASCAL VOC 2007 test set.


train: 1.51
person: 1.52
person: 1.84
bus: 2.18
cat: 0.53
dog: 0.94
car: 2.38
chair: 1.36
diningtable: 0.74
car: 1.87
aeroplane: 1.45
person: 1.01
cow: 1.18
person: 0.78
person: 1.45
diningtable: 1.67
pottedplant: 1.45
pottedplant: 0.72
tvmonitor: 1.65
cat: 1.80
car: 1.62
car: 0.80
person: 0.55
bus: 2.61
person: 2.25
tvmonitor: 1.26
bird: 0.52
diningtable: 0.87
chair: 1.65
chair: 0.80
person: 0.88
person: 1.03
person: 1.54
person: 1.12
person: 1.57
person: 1.17
person: 1.42
person: 0.53


bird: 1.61
cow: 2.02
person: 1.62
tvmonitor: 0.65
cow: 2.29
car: 1.24
chair: 1.69
diningtable: 1.52
chair: 2.02
chair: 1.83
person: 0.45
person: 0.74
chair: 0.57
diningtable: 0.56
chair: 0.79
train: 1.69
cat: 1.27
dog: 2.12
person: 1.91
horse: 1.48
person: 0.43
person: 1.74
bicycle: 1.91
cat: 1.86
person: 1.39
person: 1.60
person: 1.27
person: 0.82
diningtable: 0.83
person: 0.72
person: 0.99
sofa: 1.05
chair: 1.27
sofa: 1.45
person: 1.65
car: 0.70
motorbike:
person: 1.12
person: 0.78
horse: 0.89
bird: 0.45
car: 2.17
person: 1.25
car: 2.44
motorbike: 2.15


diningtable: 1.30
pottedplant: 0.55
pottedplant: 0.71
car: 1.52
person: 0.44
person: 2.11
bicycle: 1.22
train: 1.81
person: 1.23
tvmonitor: 1.40
motorbike: 2.42
person: 1.72
car: 1.24
person: 1.63
person: 1.67
person: 1.75
dog: 0.5
dog: 0.90
www.bhamgreys.com
person: 1.46
cat: 0.83
person: 0.94
horse: 0.48
dog: 0.73
person: 1.56
diningtable: 0.74
person: 1.44
bicycle: 2.14
car: 1.00
person: 1.21
person: 1.14
person: 1.17
person: 1.2
cat: 0.45
boat: 1.08
horse: 0.92
person: 0.99
car: 2.01
person: 1.32
person: 0.69
bicycle: 1.5
dog: 1.23
motorbike: 1.03
person: 1.14

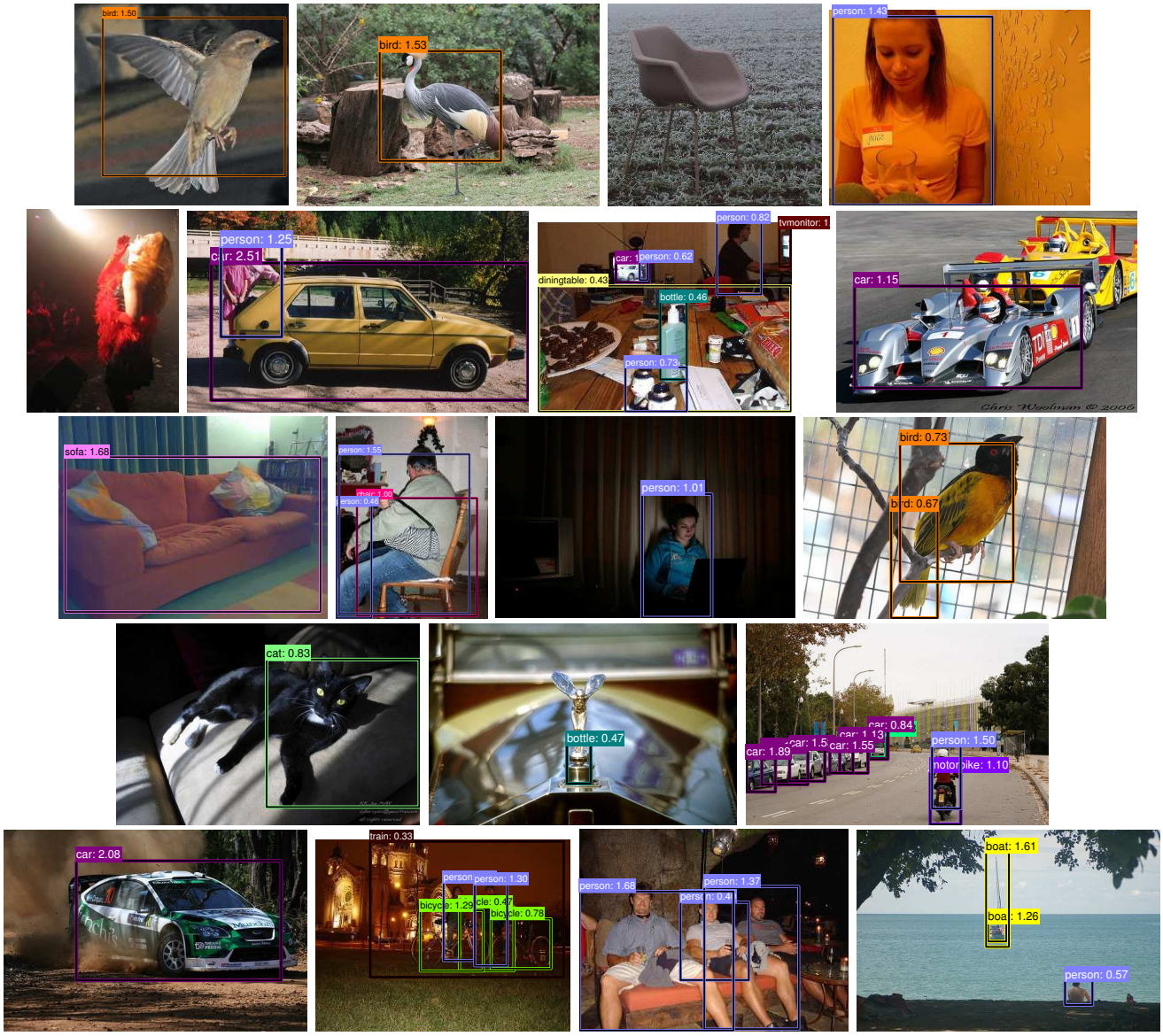


Figure S-7: Random detection examples of "VGGNet + StructObj + FGS" on PASCAL VOC 2007 test set.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012. 3

[2] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008. 4

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 4

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[5] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV*, 2012. 6

[6] T. Joachims. Svm-struct: Support vector machine for complex outputs. http://www.cs.cornell.edu/people/tj/svm_light/svm_struct.html, 2008. 2

[7] B. Pepikj, M. Stark, P. Gehler, and B. Schiele. Occlusion patterns for object class detection. In *CVPR*, 2013. 4

[8] M. Schmidt. minFunc toolbox. http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html. 3

[9] J. R. R. Uijlings, K. E. A. Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. 3, 4