Unsupervised Learning of Discriminative Attributes and Visual Representations — Supplementary Material

Chen Huang[1,2] Chen Change Loy[1,3] Xiaoou Tang[1,3]
[1]Department of Information Engineering, The Chinese University of Hong Kong
[2]SenseTime Group Limited
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
{chuang,ccloy,xtang}@ie.cuhk.edu.hk

1. Details of the Optimization Function

In this section, we will give the details of the proposed weakly-supervised hash learning algorithm. The triplet-based network architecture with a ranking loss function is illustrated in Figure 2 in the main paper.

The triplet ranking layer is on the top of the network, which takes as input the binary hash codes $\{\boldsymbol{b}_i, \boldsymbol{b}_i^+, \boldsymbol{b}_i^-\}$ of three samples in a triplet and computes the hinge loss:

$$l(\boldsymbol{b}_i, \boldsymbol{b}_i^+, \boldsymbol{b}_i^-) = \max\left(0, \rho + H(\boldsymbol{b}_i, \boldsymbol{b}_i^+)) - H(\boldsymbol{b}_i, \boldsymbol{b}_i^-))\right), \tag{1}$$

where $H(\cdot,\cdot)$ is the Hamming distance between hash codes, ρ is a margin between the Hamming distances of within-cluster code pair $\{\boldsymbol{b}_i, \boldsymbol{b}_i^+\}$ and between-cluster code pair $\{\boldsymbol{b}_i, \boldsymbol{b}_i^-\}$. The hinge loss is a convex approximation to the 0-1 ranking error, which measures the network's violation of the ranking order specified in the triplet.

For continous optimization we relax our hashing function to:

$$\boldsymbol{b} = h(\boldsymbol{x}; \boldsymbol{W}) = 2\sigma(\boldsymbol{W}^T f(\boldsymbol{x})) - 1, \tag{2}$$

where $\boldsymbol{W}$ denotes the hashing weights, and $\sigma(x) = 1/(1 + exp(-x))$ is the logistic function. The Hamming distance thus becomes:

$$H(\boldsymbol{b}_i, \boldsymbol{b}_j) = (K - \boldsymbol{b}_i^T \boldsymbol{b}_j)/2, \tag{3}$$

where K is the number of hash bits; and the hinge loss in Eq. 1 can be simplified to

$$l(\boldsymbol{b}_i, \boldsymbol{b}_i^+, \boldsymbol{b}_i^-) = \max\left(0, \rho + \frac{1}{2}(\boldsymbol{b}_i\boldsymbol{b}_i^- - \boldsymbol{b}_i\boldsymbol{b}_i^+)\right). \tag{4}$$

The final objective function for hash learning is:

$$\begin{aligned} \min \quad & \sum_i \varepsilon_i + \alpha \mathrm{tr}[\boldsymbol{W}^T f(\boldsymbol{X}) f(\boldsymbol{X})^T \boldsymbol{W}] \\ & + \beta \|\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}\|_2^2 + \gamma \|\boldsymbol{W}\|_2^2, \\ s.t.: \quad & \max\left(0, \rho + (\boldsymbol{b}_i\boldsymbol{b}_i^- - \boldsymbol{b}_i\boldsymbol{b}_i^+)/2\right) \le \varepsilon_i, \\ \forall i, \quad & \boldsymbol{b}_i = h(\boldsymbol{x}_i; \boldsymbol{W}), \quad and\ \varepsilon_i \ge 0, \end{aligned} \tag{5}$$

where ε_i is a slack variable, ρ is set to $K/2$, and α, β, γ are the regularization parameters. By replacing $\varepsilon_i = l_i = \max\left(0, \rho + (\boldsymbol{b}_i\boldsymbol{b}_i^- - \boldsymbol{b}_i\boldsymbol{b}_i^+)/2\right)$, our objective function can be converted to unconstrained optimization:

$$\begin{aligned} \min \quad & \sum_i l_i + \alpha \mathrm{tr}[\boldsymbol{W}^T f(\boldsymbol{X}) f(\boldsymbol{X})^T \boldsymbol{W}] \\ & + \beta \|\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}\|_2^2 + \gamma \|\boldsymbol{W}\|_2^2, \\ s.t.: \quad \forall i, \quad & l_i = \max\left(0, \rho + (\boldsymbol{b}_i\boldsymbol{b}_i^- - \boldsymbol{b}_i\boldsymbol{b}_i^+)/2\right), \\ & \boldsymbol{b}_i = h(\boldsymbol{x}_i; \boldsymbol{W}). \end{aligned} \tag{6}$$

Stochastic gradient descent is used to solve Eq. 6. During CNN training, we evaluate the ranking loss and back-propagate the gradients to the lower layers so that they can adjust their parameters to minimize the loss. For any triplet $\{\boldsymbol{b}_i, \boldsymbol{b}_i^+, \boldsymbol{b}_i^-\}$, if the ranking loss $l_i > 0$, its gradients with respect to the hash codes are:

$$\frac{\partial l_i}{\partial \boldsymbol{b}_i} = \frac{1}{2}(\boldsymbol{b}_i^- - \boldsymbol{b}_i^+), \tag{7}$$

$$\frac{\partial l_i}{\partial \boldsymbol{b}_i^+} = -\frac{1}{2}\boldsymbol{b}_i, \tag{8}$$

$$\frac{\partial l_i}{\partial \boldsymbol{b}_i^-} = \frac{1}{2}\boldsymbol{b}_i. \tag{9}$$

This back-propagation algorithm actually adjusts the feature mapping function $f(\cdot)$ and hashing function $h(\cdot)$, so that the Hamming distance between the generated hash codes $\{\boldsymbol{b}_i, \boldsymbol{b}_i^+\}$ by Eq. 2 is small, and the distance between $\{\boldsymbol{b}_i, \boldsymbol{b}_i^-\}$ is large.

2. More Examples of the Discovered Attributes

Figures 1 and 2 show more example attributes discovered from natural images on the CIFAR-10 dataset and binary contour patches on the BSDS500 dataset, respectively.

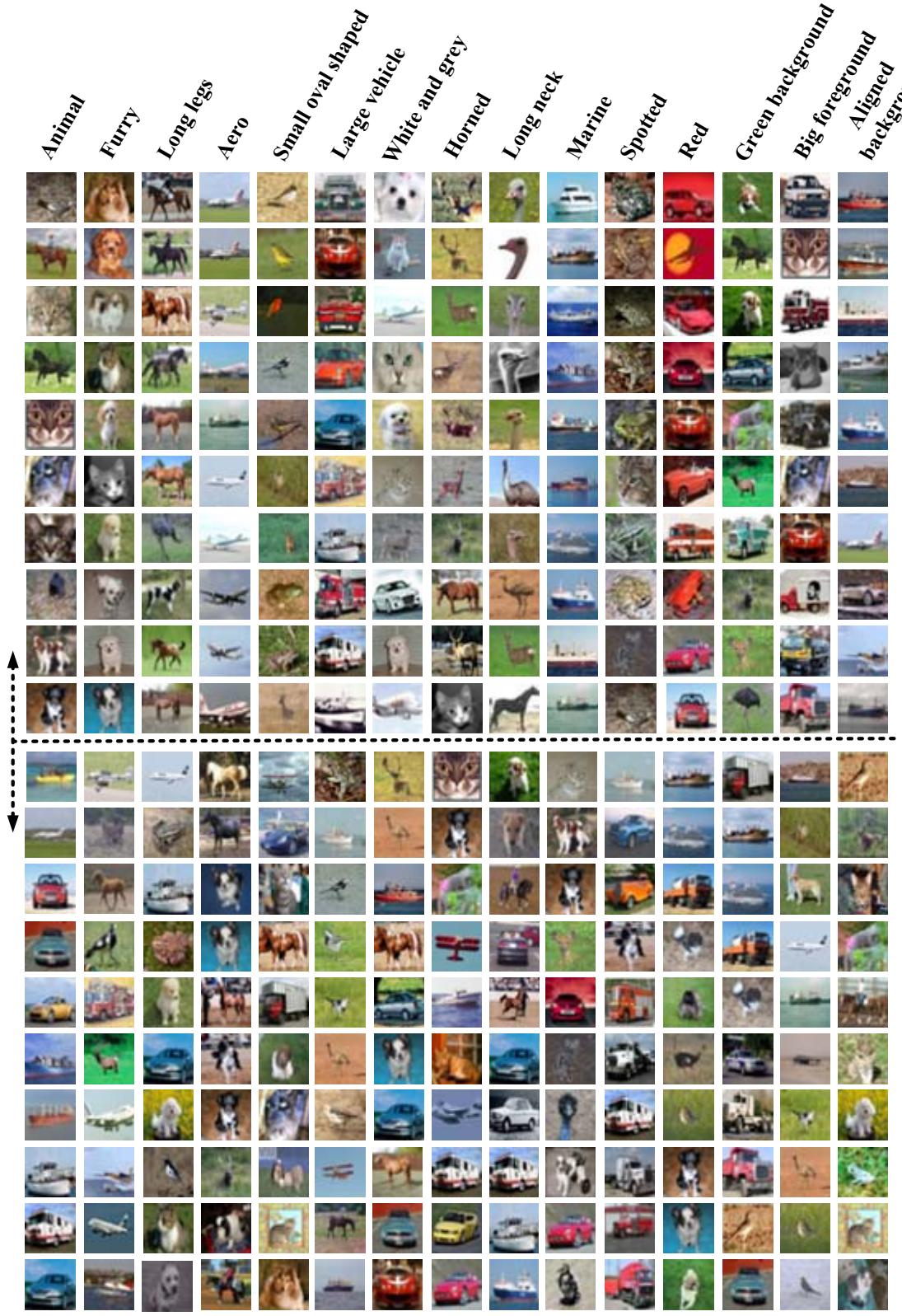


Figure 1. Unsupervised discovery of attribute codes from natural images on the CIFAR-10 dataset. Each code bit corresponds to a hyperplane (dashed line) of one attribute predictor where samples transition from one side to the other. We show on each side the 10 most confident samples for 15 bits (columns). We find most attributes semantically meaningful and easily human-nameable.

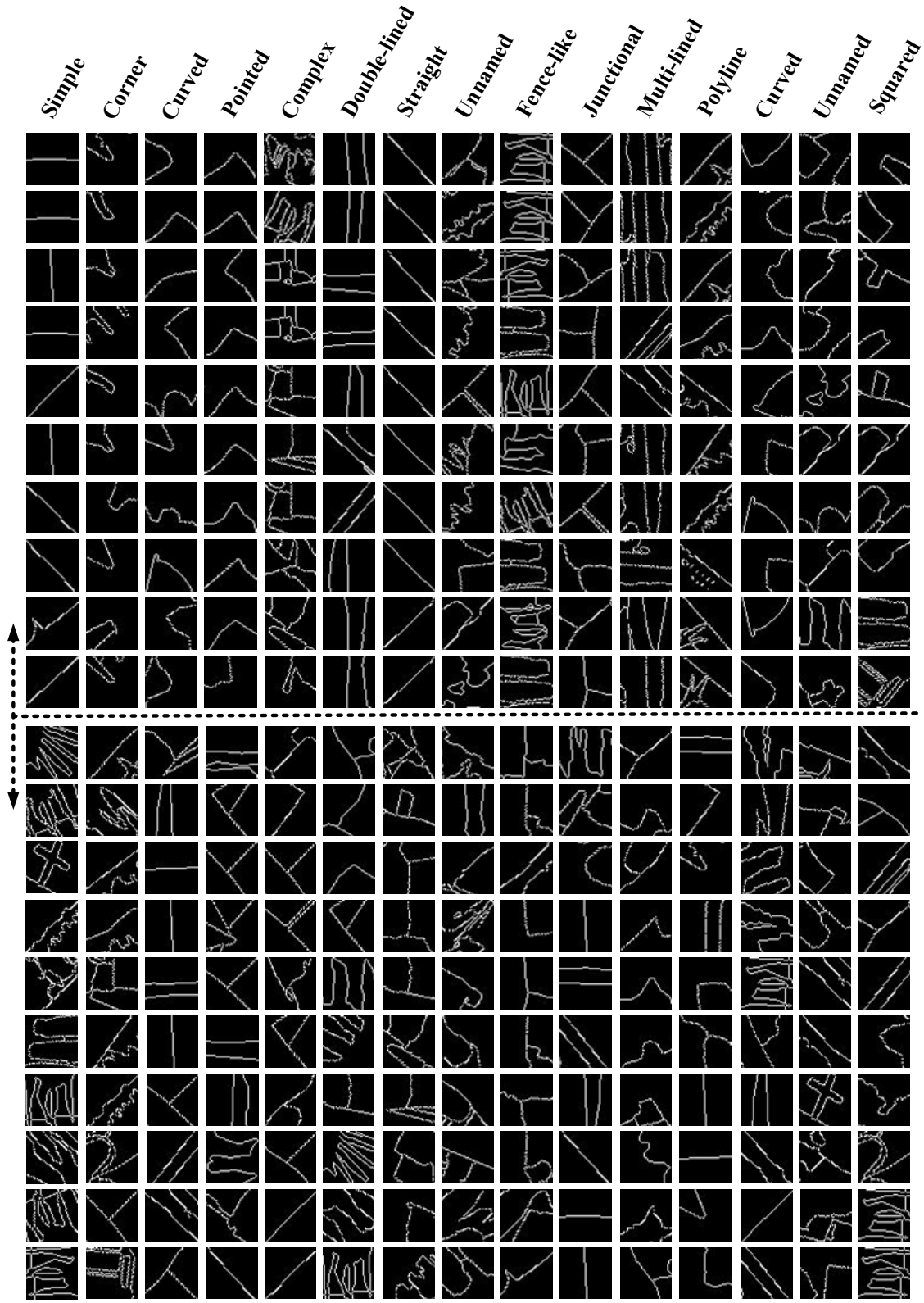
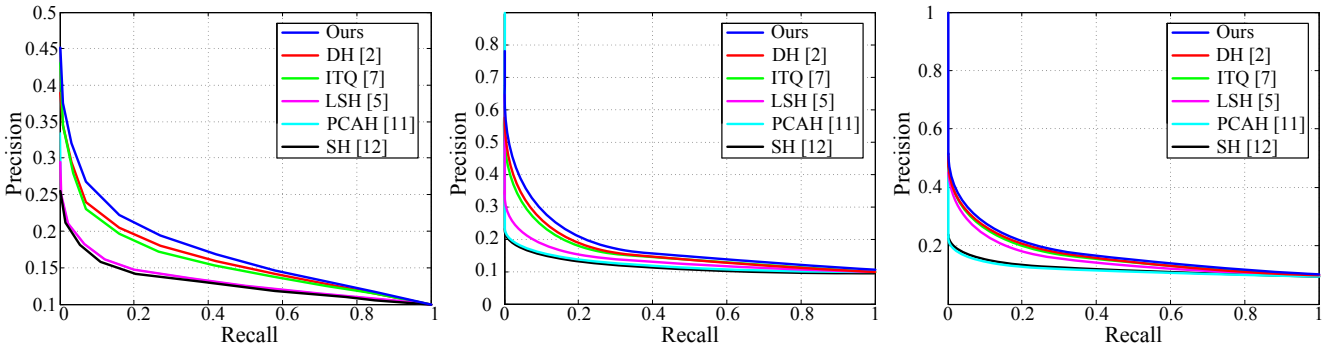


Figure 2. Unsupervised discovery of attribute codes from binary contour patches on the BSDS500 dataset. Each code bit corresponds to a hyperplane (dashed line) of one attribute predictor where samples transition from one side to the other. We show on each side the 10 most confident samples for 15 bits (columns). We find most attributes semantically meaningful and easily human-nameable.

Table 1. Retrieval results on the CIFAR-10 dataset: K-bit (16-64) Hamming ranking accuracy by mAP, precision @ $N = 1000$, precision @ Hamming radius $r = 2$. Hamming look-up with $r = 2$ for $K = 64$ is not evaluated because it is prohibitively expensive for such long codes.

Methed	mAP (%)			precision (%) @ N			precision (%) @ $r = 2$	
	16	32	64	16	32	64	16	32
SH [12]	12.55	12.42	12.56	18.83	19.72	20.16	18.52	20.60
PCAH [11]	12.91	12.60	12.10	18.89	19.35	18.73	21.29	2.68
LSH [5]	12.55	13.76	15.07	16.21	19.10	22.25	16.73	7.07
ITQ [7]	15.67	16.20	16.64	22.46	25.30	27.09	22.60	14.99
DH [2]	16.17	16.62	16.96	23.79	26.00	27.70	23.33	15.77
Ours	**16.82**	**17.01**	**17.21**	**24.54**	**26.62**	**28.06**	**23.61**	**26.24**



(a) K=16 bits (b) K=32 bits (c) K=64 bits

Figure 3. Recall vs. precision curves at K =16, 32 and 64 code bits on the CIFAR-10 dataset.

Figure 4. Top 8 images retrieved by different hashing methods in comparison to ours ($K = 64$) on the CIFAR-10 dataset.

3. More Results of Image Retrieval

Table 1 shows more quantitative retrieval results on the CIFAR-10 dataset, including the precision when Hamming radius r is set to 2. Figure 3 shows the Recall vs. precision curves for different methods with K =16, 32 and 64 code bits, where our approach consistently performs best. Figure 4 presents more visual examples when $K = 64$.

4. More Results of Contour Detection

Figure 8 shows more of our contour detection results compared with those of Sketch Token [9] and DeepContour [10] on the BSDS500 dataset.

5. Experiments on Object Proposal

Zitnick and Dollár [13] point out that the enclosed contours by a bounding box are indicative of the presence of objects. So using our contour detection results, we further experiment with proposing candidate objects in an image, which is key to many state-of-the-art object detection systems such as Fast R-CNN [6]. To this end, we directly apply EdgeBox 70 [13] to our predicted contour attribute maps in addition to the contour map. The underlying hypothesis is that the contour attributes capture a richer class of edge properties, beyond edge intensity. For example, if we recognize some "curved" edges from an image, it informs that we are more likely to detect, say, an apple.

But proposing objects at every edge attribute map and then ensembling such proposals is very inefficient. We therefore choose instead to ensemble the K attribute maps into one map, and run EdgeBox 70 only on this map and the edge map. All the proposed boxes on the two maps are kept, except for those having large Intersection over Union (IoU) ($>$0.7) with others but lower-scores by EdgeBox 70. As for the ensembling scheme, we simply apply the max-pooling across K maps so as to retain the most salient contour attribute per location. Figure 5 illustrates this process.

To evaluate object proposals, we use the PASCAL VOC 2007 [3] test set, which consists of 4952 images with 20 object categories. The metrics are Area Under the Curve (AUC) and the number of proposals N vs. recall for IoU threshold of 0.7.

Table 2 compares our object proposal method with the state-of-the-arts. We observe significant improvements over the EdgeBox 70 baseline due to the exploitation of edge attributes. Our edge-based version already performs better, which benefits from the shared learning of both edge and edge attributes. This version also has a fast speed. Our full method performs on par with state-of-the-arts [4, 8] and achieves a higher recall of 91% from 5000 proposals, with less running time. The computational costs of [4, 8] respectively stem from refining proposals in a coarse-to-fine inverse cascading and re-ranking EdgeBox's proposals. In contrast, we only need to feed the ensembled attribute map and edge map to EdgeBox 70. This validates the effectiveness of our learned contour attributes. Figure 6 shows the recall with changing number of object proposals N or IoU threshold for different methods. Figure 7 shows the visual difference between EdgeBox 70 [13] and our method. We believe the combination with more advanced post-processing schemes (*e.g.*, using re-ranking in [8]) is highly promising for further improvements.

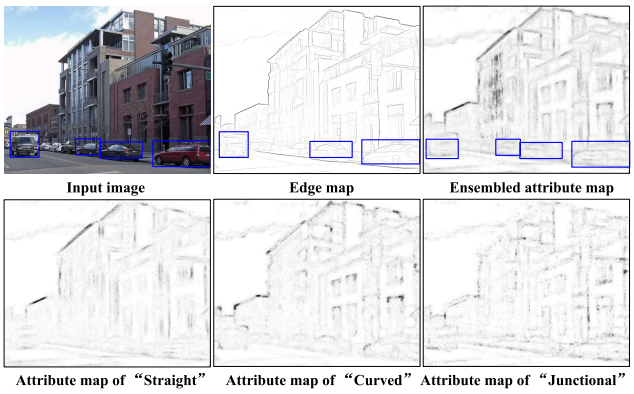


Figure 5. Object proposal from the edge map and ensembled attribute map. Also shown are 3 example attribute maps out of 16 for ensembling. Notice the richness of contour attributes and their supplementation to the edge map for object proposal.

Table 2. Object proposal results on PASCAL VOC 2007, where both of our versions, using edge map only and using edge map+ensembled attribute map, work on $K = 16$. Under the IoU threshold 0.7, the evaluation metrics are AUC, the number of proposals N needed to achieve 25%, 50% and 75% recall, the maximum recall using 5000 boxes, and the running time.

Method	AUC	N@25%	N@50%	N@75%	Recall	Time
MCG [1]	.42	9	81	1363	83%	30s
EdgeBox 70 [13]	.46	12	108	800	87%	**.25s**
DeepProposal70 [4]	.49	5	50	540	-	.75s
DeepBox [8]	**.60**	**3**	**20**	**183**	87%	2.5s
Ours (edge)	.48	9	82	574	90%	**.25s**
Ours (edge+attribute)	.52	6	39	495	**91%**	.4s

References

[1] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[2] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, 2015.

[3] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[4] A. Ghodrati, M. Pedersoli, T. Tuytelaars, A. Diba, and L. V. Gool. DeepProposal: Hunting objects by cascading deep convolutional layers. In *ICCV*, 2015.

[5] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[6] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[7] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[8] W. Kuo, B. Hariharan, and J. Malik. Deepbox:learning objectness with convolutional networks. In *ICCV*, 2015.

[9] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013.

[10] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015.

[11] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[12] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2009.

[13] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.

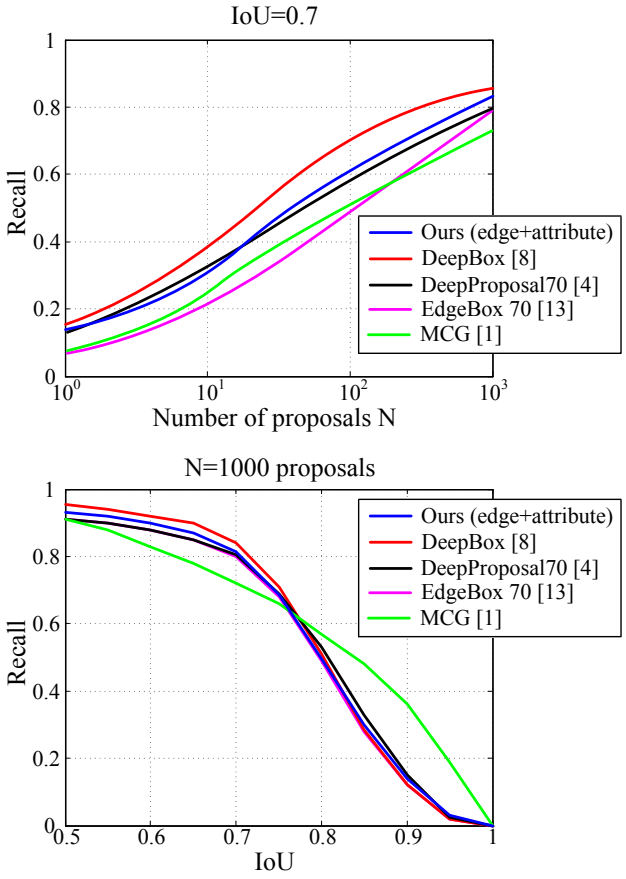


Figure 6. Recall vs. number of proposals N for IoU threshold 0.7 (top) and Recall vs. IoU threshold for $N = 1000$ proposals (bottom) on the PASCAL VOC 2007 test set. Our method ensembles $K = 16$ edge attribute maps.

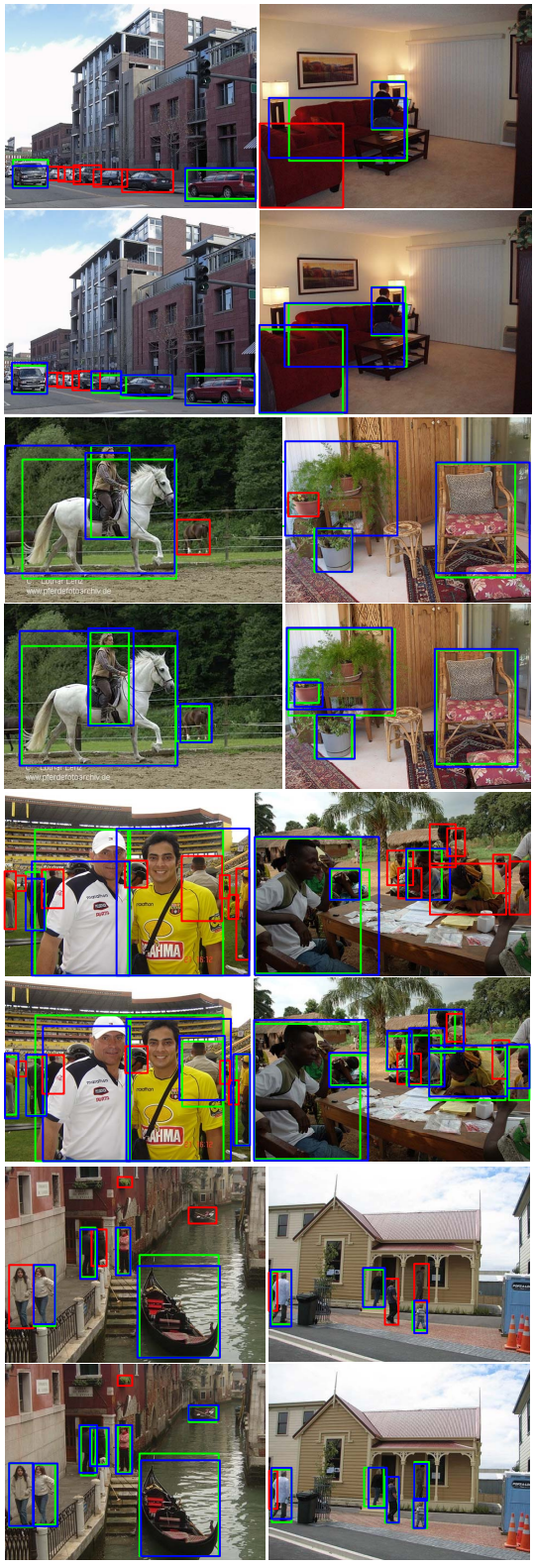


Figure 7. Qualitative comparison of EdgeBox 70 [13] (top row) and our approach (bottom row: edge+attribute, $K = 16$), with IoU threshold 0.7 at 1000 object proposals. Blue bounding boxes are the closest object proposals to the ground truth shown in green. Red boxes are ground truth boxes that are missed.

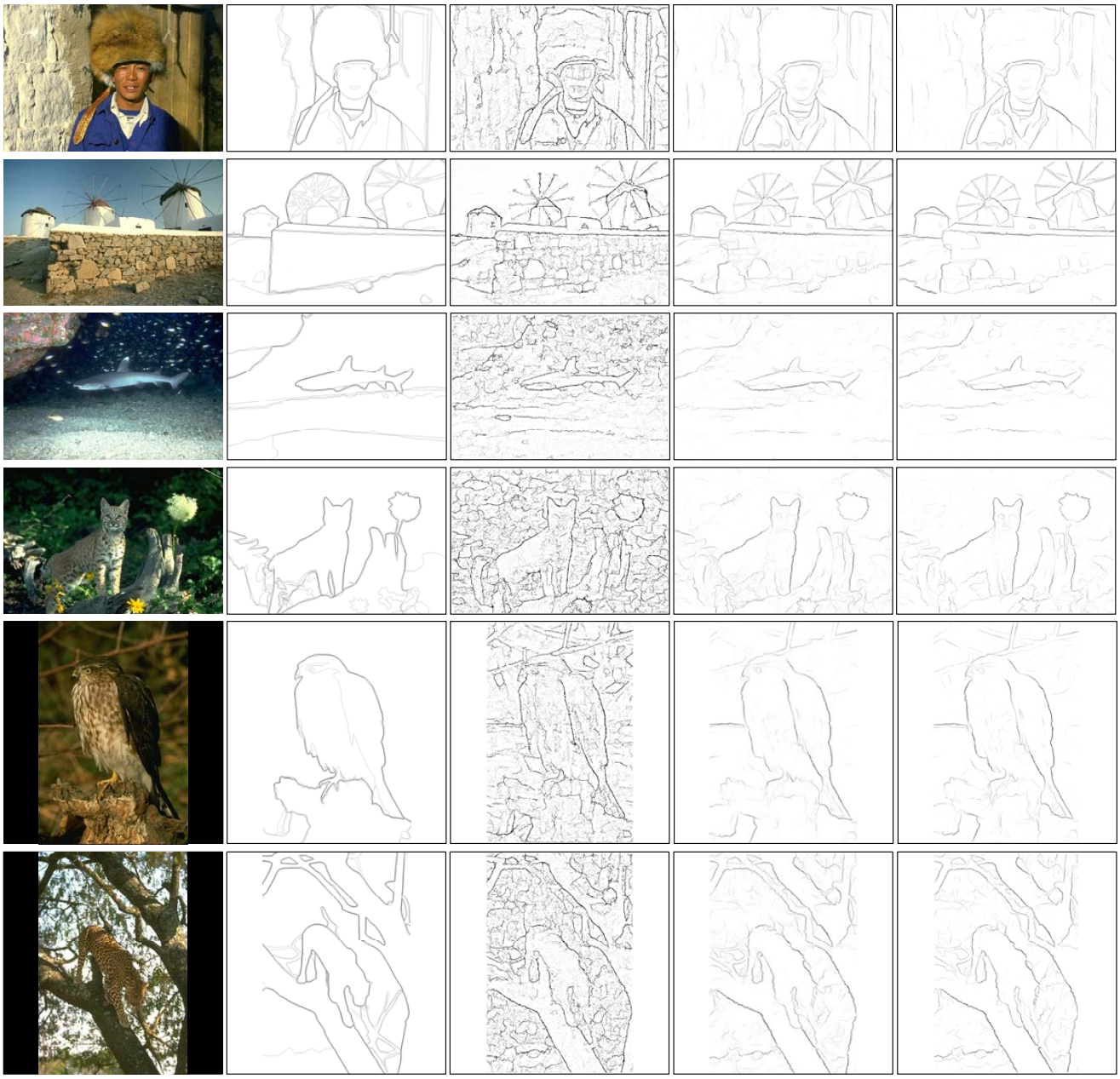


Figure 8. Edge detection results on the BSDS500 dataset. Our patch-wise method works on $K = 16$.