Supplementary material for "R(Det)^2: Randomized Decision Routing for Object Detection"

Abstract

In this supplementary material, we provide more experimental results about the R(Det)^2, including: ablation study on the node and tree prediction accuracy, the detailed design for generating routing probabilities and masks, and more visualization results. We also add the analysis in this supplementary material. The limitation as well as social impacts are also included in this supplementary material.

1. Supplementary experiments

We present more experiments on R(Det)^2. The implementation details are as follows:

Datasets. We evaluate R(Det)^2 on the large-scale benchmark *MS COCO* 2017 [5]. Following common practice, we train detectors on *training* split with ∼115k images and evaluate them on *val* split with 5k images. The standard mean average precision (AP) across different IoU thresholds is used as the evaluation metric.

Training details. We implement the R(Det)^2 as the plug-in head and integrate it into existing detectors. Our implementation is based on the popular mmdetection [2] platform. If not specially noted, the R(Det)^2 serves for the decision in R-CNN of two-stage detectors, as Faster R-CNN [8]. We train the models with ResNet-50 [3] backbones with 8 Nvidia TitanX GPUs. The learning rate is set to 0.02 and the weight decay is 1e-4, with momentum 0.9. The models for ablation studies are trained with the standard 1× configuration. No data augmentation is used except for standard horizontal image flipping.

Inference details. It is noteworthy that the randomized decision routing is only performed in training phase. In inference, we perform on the single image scale with the shorter side normalized to 800 pixels and longer side less than 1333 pixels.

1.1. Ablation study

Comparison of node and tree decision performance. To illustrate how R(Det)^2 works, we present the detailed performance of separate node decision and the overall tree decision in Table 1. Compared to the baseline, we find that the single-node prediction is also improved. *That is mainly because the feature representative learning is enhanced and features are learned in a more sufficient way with the proposed randomized decision routing in training phase.* Despite this, further performance gain can be achieved in the testing phase, with the help of fusing predictions from multiple nodes. When we apply the cross-entropy for classi-

task	node	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Base.		37.4	58.1	40.4	21.2	41.0	48.1
	CE loss for cls and Smooth-L_1 loss for bbox						
cls	*left*	39.1	58.4	42.9	21.6	42.8	52.8
	right	39.4	60.2	42.4	22.9	42.9	51.5
bbox	*left*	40.1	60.9	43.3	23.1	43.6	53.2
	right	40.2	60.9	43.5	23.2	43.7	53.0
	all	40.4	61.2	44.1	23.8	43.7	53.0
	CE loss for cls and IoU loss for bbox						
cls	*left*	39.3	57.9	43.2	21.5	42.7	53.3
	right	39.6	60.0	42.9	23.3	43.1	51.9
bbox	*left*	40.2	60.6	44.1	23.0	43.7	53.4
	right	40.2	60.6	43.4	23.0	43.6	53.6
	all	41.0	61.2	44.8	24.6	44.1	53.7
	Focal loss for cls and IoU loss for bbox						
cls	*left*	40.9	61.1	44.4	24.3	44.2	53.8
	right	40.9	61.0	44.4	24.2	44.3	53.6
bbox	*left*	40.5	60.9	43.8	23.9	43.9	53.1
	right	40.6	61.0	44.0	23.7	44.0	53.2
	all	41.0	61.1	44.5	24.3	44.3	53.7

Table 1. **Comparison of node decision and overall tree decision with different combination of loss functions.** The baseline model is Faster R-CNN with ResNet-50 as the backbone. CE indicates the cross-entropy loss attached with *Softmax* activation. Focal indicates the original focal loss [4]. IoU indicates the loss computed by the negative-log of intersection-over-union [9].

γ_{min}	γ_{max}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
$\mathbf{U(0.1, 0.3)}$	$\mathbf{U(0.9, 1.1)}$	40.9	61.2	44.5	23.9	44.2	53.7
$U(0.1, 0.3)$	$U(0.3, 0.7)$	40.5	61.0	44.0	23.8	44.0	52.5
$U(0.1, 0.3)$	$U(1.4, 1.6)$	40.8	61.1	44.3	24.2	44.2	53.5
$U(0.1, 0.3)$	$U(1.9, 2.1)$	40.4	60.6	44.0	23.4	43.9	53.7
$U(0, 0.1)$	$U(0.9, 1.1)$	40.4	60.2	44.2	23.1	43.7	53.3
$U(0.3, 0.5)$	$U(0.9, 1.1)$	40.9	61.0	44.2	23.6	44.2	53.7
$U(0.5, 0.7)$	$U(0.9, 1.1)$	40.4	60.9	43.7	23.2	43.7	53.5

Table 2. **Ablation study on the setting of γ_{max}, γ_{min} in *selective loss* for randomzied decision routing.** For this experiment, the R(Det)^2 is equipped into Faster R-CNN with ResNet-50 backbone.

fication and Smooth-L_1 loss for the bounding box (*bbox*) regression, the overall achieved detection AP is 40.4% and AP_{50} is 61.2%. The single-node detection with the sole *left* yields 39.1% AP, 58.4% AP_{50}, 42.9% AP_{75}. Meanwhile, the detection with the *right* node achieves 39.4% AP, 60.2% AP_{50}, 42.2% AP_{75}. By comparing the *left* and *right* nodes, we find that one node performs better on the detection accuracy with moderate IoU ($\geq$0.5) while the other node performs better on the detection with higher IoU ($\geq$0.75). This divergent decision leads to further improvement on the overall decision of the whole tree. When we use the IoU loss (native implementation in mmdetection [2], which is computed as the negative-log of intersection-over-union), the achieved overall AP is 41.0%. The *bbox* regression with *left* and *right* node yields the detection AP of 40.2%. The multinode regression brings 0.8% of AP improvement in testing phase. When Focal loss [4] is applied, the detection AP of single-node classification is 40.9%, which is slightly lower than the overall detection AP. It can be inferred from the table that the Cross-entropy loss and IoU loss are prone to generate more divergent predictions, thus the improvement on the overall detection is more significant. A possible guess is that the Cross-entropy loss (Softmax activation) and IoU loss are measured with the whole output vector. Less restrictions are imposed on the single component in the prediction vector and more divergence is encouraged for different nodes.

Setting of γ in selective loss. In Section 3 of the main paper, we introduce selective loss to reduce the relevance of different nodes. We set a higher weight γ_{max} for the selected node and a lower weight γ_{min} for the remaining node. We sample γ_{min}, γ_{max} from uniform distributions. In default, $\gamma_{min} \sim U(0.1, 0.3)$, $\gamma_{max} \sim U(0.9, 1.1)$. We further present the ablation study on the setting of γ_{min}, γ_{max} in Table 2. When γ_{min} is sampled from $U(0.1, 0.3)$, γ_{max} sampling around 1 would lead to higher detection AP. Increasing or decreasing γ_{max} would cause 0.4$\sim$0.5 AP reduction. When γ_{max} is sampled from $U(0.9, 1.1)$, γ_{min} sampling around 0.1$\sim$0.5 would yield better performance. Since the selective loss is mainly used to enlarge the decision divergence of different nodes, we need to ensure the moderate learning of separate nodes and diverse learning among multiple nodes.

Effects on trees number. In traditional machine learning, a natural way to improve the prediction accuracy of decision is to learn multiple parallel decision trees and fuse the prediction together, just as Random Forest [1]. Inspired by this, we exploit the effects on the number of parallel trees in R(Det)^2. The detection accuracy of *R(Det)^2-M* is presented in Table 3. The detection AP is highest with 1 tree and 2 nodes, yet decreases along with the number of trees. It indicates that simply applying multi-node prediction with no consideration of node difference would hamper the performance. This experiment highlights the importance of introducing divergence into node training.

#Trees	#Nodes	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	2	**40.5**	**60.9**	**43.9**	**23.9**	**43.8**	**53.1**
2	4	40.3	60.4	43.8	23.5	43.6	52.4
4	8	40.1	60.4	43.4	23.0	43.7	51.9
8	16	40.1	60.1	43.5	23.3	43.3	52.2

Table 3. **Effects on number of trees.**

1.2. Comparative study on module design

Design of routing probability branch. In Figure 3 of the main paper, we note that the implementation of R(Det)^2 requires a branch to generate the routing probabilities. In contrast to the original feature dimension (commonly 1024) for generating prediction values, the routing probability branch only needs to generate the scalar probability. Therefore, we design a much narrower branch for saving the computational burden. For this branch, we design three types of modules to generate the routing probabilities, as follows:

- *2fc:* 2 fully-connected layers with dimension d;
- *2conv:* 2 convolution layers with channels number c;

routing-prob.setting		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	*#FLOPs*	*#params*
2fc	$d = 256$	40.4	60.8	43.7	23.4	43.8	52.8	133G	20.95M
	$d = 128$	40.2	60.4	43.5	23.5	43.7	52.0	131G	19.30M
	$d = 64$	40.3	60.6	43.8	23.5	44.0	52.5	130G	18.48M
2conv	$c = 256$	40.2	60.6	43.5	23.3	43.5	52.7	187G	18.90M
	$c = 128$	40.1	60.2	43.5	23.3	43.5	52.4	151G	18.14M
	$c = 64$	40.1	60.4	43.6	23.3	43.4	52.3	138G	17.87M
	$c = 32$	40.2	60.4	43.7	23.7	43.6	52.7	133G	17.76M
1conv1fc	$c = 32, d = 256$	40.2	60.4	43.5	23.0	43.7	52.9	133G	18.15M
	$c = 64, d = 256$	40.3	60.6	43.5	24.0	43.6	53.1	137G	18.63M
	$c = 128, d = 256$	40.2	60.4	43.4	23.1	43.6	52.4	145G	19.56M

Table 4. **Comparison of different designs for routing probability generation.** The baseline model is Faster R-CNN with ResNet-50 backbone. We equip the *R(Det)*^{2}*-M* for the investigation of routing probability branch design. The number of *FLOPs* is evaluated with image size 800×1200.

routing-prob. setting	*routing-mask setting*	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	*#FLOPs*	*#params*
2fc $d = 64$	$c_m = 16$	40.3	60.6	43.8	23.5	44.0	52.5	130G	18.48M
	$c_m = 32$	40.2	60.6	43.7	23.6	43.5	52.8	130G	20.12M
	$c_m = 64$	40.4	60.6	43.8	23.4	43.7	52.5	130G	23.41M
2fc $d = 256$	$c_m = 16$	40.4	60.8	43.7	23.4	43.8	52.8	133G	20.95M
	$c_m = 32$	40.2	60.3	43.5	23.7	43.7	52.6	133G	22.59M
	$c_m = 64$	40.5	60.9	43.9	23.9	43.8	53.1	133G	25.88M

Table 5. **Comparison of different designs for routing mask generation.** The baseline model is Faster R-CNN with ResNet-50 backbone. We equip the *R(Det)*^{2}*-M* for the investigation of routing mask design. The number of *FLOPs* is evaluated with image size 800×1200.

- *1conv1fc:* 1 convolution layer with channels number c and 1 fully-connected layer with dimension d.

We evaluate and compare the detection performance of *R(Det)*^{2}*-M* of fixed mask generation with different designs of routing probability branch. The detection accuracy and model complexity of different routing probability generation is given in Table 4. Note that we only consider the FLOPs and model parameters in the R-CNN head, with 1000 region proposals. In general, the detection accuracy does not change too much with the parameter size of routing probability branch. The detection AP is highest with the *2fc* design of $d = 256$, with 40.4% of AP and 60.8% of AP_{50}. The parameter size of *2fc* design is slightly higher, while the *2conv* mainly increases the computational burden. The design of *1conv1fc* is a compromise to balance the parameter size and computational complexity. To consider the detection performance and model complexity as a whole, we commonly apply the *2fc* design with $d = 64$ or $d = 256$.

Design of routing mask branch. In order to generate the routing masks for different nodes, we construct the routing mask branch. The input for generating the route mask is a single vector for each image. Commonly we use the average features of all the involved RoIs. The output of fully-connected layer is restricted to be equal to the multiplication of nodes number and feature output for decision (typically 1024). That is, the output is also a vector with $N \times 1024$, where N is the number of nodes. Although a single fully-connected layer upon the single-vector is effective to generate the route masks, it brings too many parameters. To conquer this, we stack a convolution layer with c_m channels for dimension reduction and a fully-connected layer to balance the detection performance and computational complexity. We present the detection accuracy as well as the model complexity with changing c_m in Table 5. We evaluate the impact of c_m under 2 settings of routing probability branch. The detection accuracy is the highest when $c_m = 64$. When routing probability is generated with *2fc* $d = 256$, the detection accuracy is 40.5% of AP and 43.9% of AP_{75}.

Design of R(Det)^2-Lite. As discussed in the main paper, the model complexity of R(Det)^2 is caused by the additional branches for routing probability, routing mask and task-aware features. Since task-aware feature computation is the main cause of computational complexity, we remove

the task-aware and only consider *R(Det)*^{2}*-M* for the lite version design. Based on the experiments in Table 4 and 5, we develop *R(Det)*^{2}*-Lite* with complexity-saving design of routing probability branch (*2fc* $d = 64$) and routing mask branch (*1conv1fc* $c_m = 16$). It achieves 40.3% AP with nearly ignorable model complexity.

1.3. Performance with ViT backbones.

ViT (vision transformer) such as Swin [6] has attracted much attention recently. Therefore, other than ResNet-50 and ResNet-101, we conduct experiments with swin transformer for comparative study. Our experiment is based on the implementation of swin-transformer-detection [1] with cascade structure. Different from other detection methods, the swin transformer based detection is based on AdamW optimization [7] other than SGD. We follow the default optimization setting as swin-transformer-detection. The comparison of original decision head (*4conv1fc* with Syn Batch normalization) and our R(Det)^2 is presented in Table 6. With Swin-L as the backbone, the R(Det)^2 improves the 1$\times$ (12 epochs) detection AP of single-scale testing to 43.9%.

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Swin-L	53.0	72.6	57.9	35.8	57.0	68.8
+R(Det)^2	**53.9**	**73.4**	**58.7**	**38.0**	**58.0**	**69.9**

Table 6. **Detection accuracy of Swin transformer as the backbone.**

2. Visualization and analysis

We present more comparative visualization study in this supplementary material, as in Figure 1 and Figure 2. The detected results by ResNet-101 based Faster R-CNN are shown in the left column and those from the R(Det)^2 are shown in the right column. Unlike Faster R-CNN, the confidence output from the R(Det)^2 is lower. A large proportion of over-confident detection results are avoided. Figure 1 shows that R(Det)^2 is effective to detect objects in complex scenes, such as easily-confused background, low-resolution ones. Figure 2 shows that R(Det)^2 is effective in reducing the repeated detections, especially in cluttered scenes.

Two effects contribute to the performance improvement of R(Det)^2. First, by generating multiple node decisions, the tree-like decision structure enables us to explore diverse visual cues from different aspects. Moreover, the divergent learning with randomized routing losses of R(Det)^2 helps promote feature representation of deep neural networks. Especially, larger objects are with more visual cues and multi-node decision helps explore features from different aspects and alleviate the over-focus of single visual patterns, leading to significant AP_L improvement. It also helps reduce repeated detections inside larger objects (Figure 2 of this supplementary material). Second, the routing probabilities can be viewed as decision confidence, which is intrinsic to model the confidence/uncertainty level of classifiers/regressors. Further, smaller learning rates are assigned with the less prior nodes in R(Det)^2, which suppresses the over-optimization. Compared to existing works relying on single linear projection for decision, the extra modeling of confidence and the slow-fast learning manner help reduce the over-confident decisions. From the visualization we can see that, the detected boxes with saturated (nearly 1) confidence are largely suppressed by R(Det)^2.

[1] https://github.com/SwinTransformer/Swin-Transformer-Object-Detection

3. Limitation and Social Impacts

The proposed method combines decision trees into black-box networks, which would provides the potential for interpretable machine learning. Yet the learned models are based on the statistics of training dataset, which might be biased and bring negative social impacts.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 2

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 1, 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, pages 770–778, 2016. 1

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *ICCV*, pages 2980–2988, 2017. 1, 2

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. Microsoft coco: Common objects in context. *ECCV*, pages 740–755, 2014. 1

[6] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, pages 10012–10022, 2021. 4

[7] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ICLR*, 2019. 4

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. on Pat. Anal. and Mach. Intell.*, 39(6):1137–1149, 2017. 1

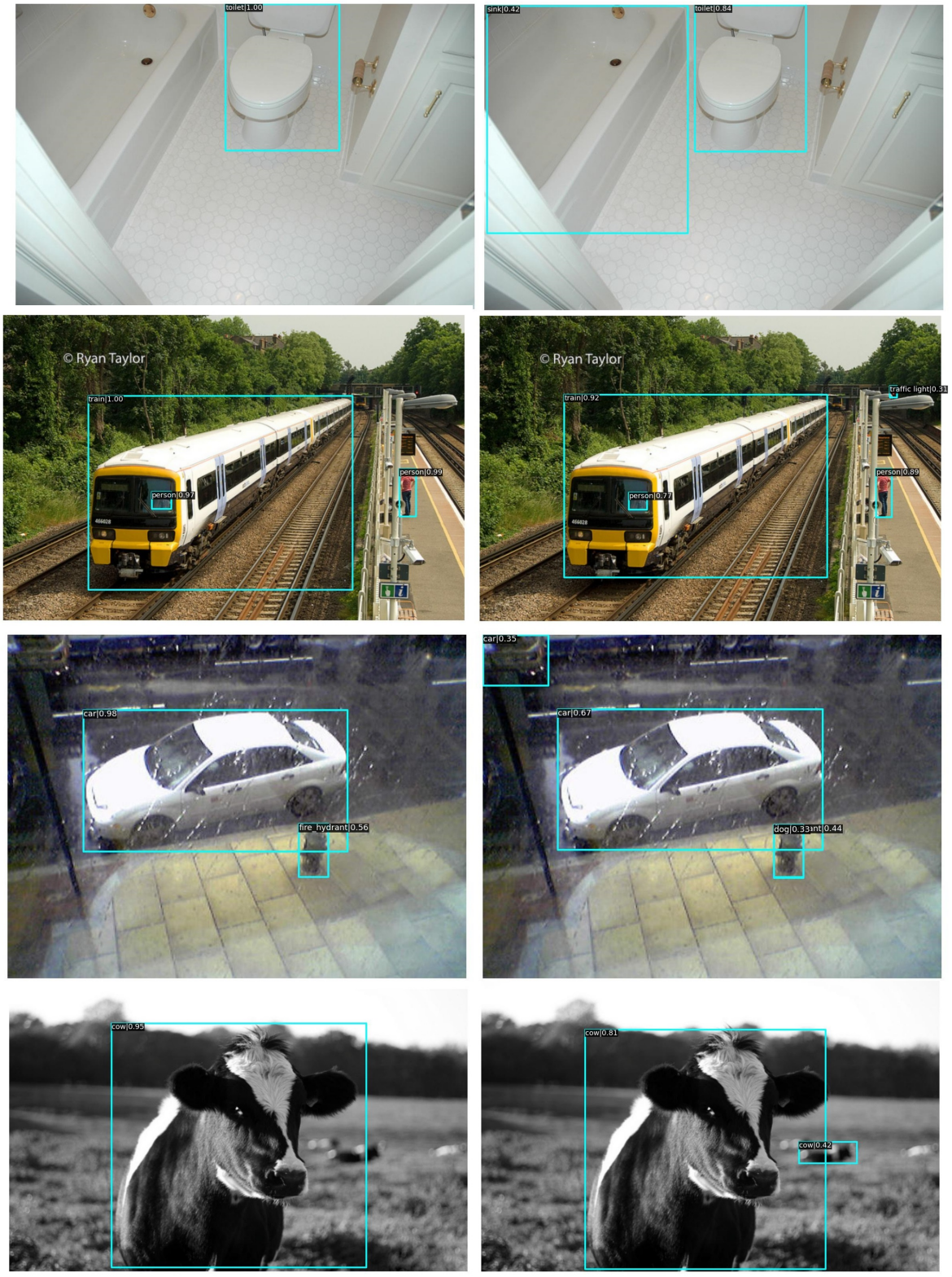

Figure 1. **Comparison of detection results for the baseline Faster R-CNN and R(Det)^2 equipped one.** The models are with ResNet-101 as the backbone and trained with *COCO* 115k-*train*. The example test images are from *COCO* 5k-*val*. The rectangles mark the detected bounding boxes with attached category labels and confidences. The detection results of baseline model are presented in the left column (39.3% AP) and those of R(Det)^2 are presented in the right column (42.5% AP).

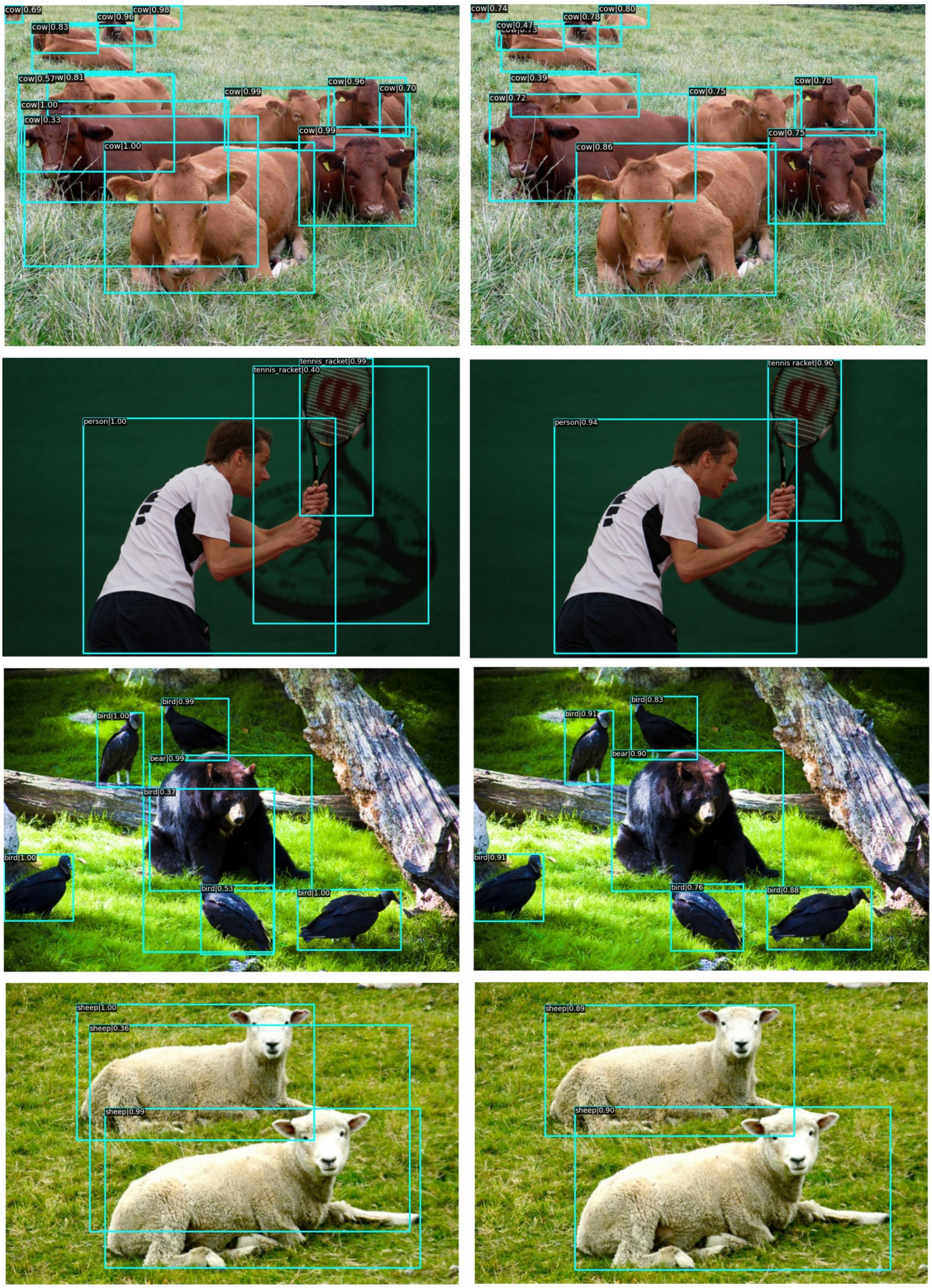

Figure 2. **Comparison of detection results for the baseline Faster R-CNN and R(Det)^2 equipped one.** The models are with ResNet-101 as the backbone and trained with *COCO* 115k-*train*. The example test images are from *COCO* 5k-*val*. The rectangles mark the detected bounding boxes with attached category labels and confidences. The detection results of baseline model are presented in the left column (39.3% AP) and those of R(Det)^2 are presented in the right column (42.5% AP).

[9] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016. 1