[Supplementary Material]
Better to Follow, Follow to Be Better: Towards Precise Supervision of Feature Super-Resolution for Small Object Detection

Junhyug Noh[1] Wonho Bae[2] Wonhee Lee[1] Jinhwan Seo[1] Gunhee Kim[1]
[1]Seoul National University [2]University of Massachusetts Amherst
[1]{jh.noh, wonhee, jinhwanseo}@vision.snu.ac.kr, gunhee@snu.ac.kr [2]wbae@umass.edu

In this supplementary material, we describe the details of our proposed feature super-resolution model for small object detection and further experiments as follows.

- Section 1 explains about the discrepancy in RRF more in detail.
- Section 2 explains about the losses focusing on thresholds used to filter out irrelevant proposals while computing losses.
- Section 3 gives a detailed account on the configurations of the architecture and training process.
- Section 4 provides the further experiments on the architecture of our model.
- Section 5 demonstrates the consistent effectiveness of our model when using RoI align instead of RoI pooling.
- Section 6 presents the results of additional experiments on MS COCO [9].
- Section 7–8 present the performance of our model by class on PASCAL VOC [2] and Tsinghua-Tencent 100K [12].
- Section 9 compares the feature maps from the existing feature extractor and our SR feature extractor. Moreover, the comparison on the super-resolved features from the variants of super-resolution model is provided as well.
- Section 10 provides some failure cases of our model. Furthermore, some selected detection results are also provided to demonstrate the superiority of our proposed model.

1. Discrepancy in Relative Receptive Field (DRRF)

In this section, we discuss the discrepancy in RRF (DRRF) focusing on how it varies as the size of a bounding box differs. We first derive the Eq.(3) from the main paper in detail as below.

$$\begin{aligned}
DRRF_{1/2}(w, I_W) &= \frac{RRF(w/2, I_W/2)}{RRF(w, I_W)} \\
&= \frac{\left(R_W + (w/2-1)\times D\right)/(I_W/2)}{\left(R_W + (w-1)\times D\right)/I_W} \\
&= \frac{2R_W + wD - 2D}{R_W + wD - D} \\
&= \frac{2R_W + 2wD - 2D}{R_W + wD - D} - \frac{2wD - 2D - wD + 2D}{R_W + wD - D} \\
&= 2 - \frac{wD}{R_W + wD - D} \\
&= 2 - \frac{w}{\left(\frac{R_W}{D} - 1\right) + w} \\
&= 2 - \frac{w}{c + w}
\end{aligned} \tag{7}$$

where $c = \frac{R_W}{D} - 1$. As R_W and D are the constants determined by a backbone structure, c is also a constant. For instance, $R_W = 291, D = 16$ and $c \approx 17.19$ for Faster R-CNN with ResNet-50.

Figure 8 shows how $DRRF_{1/2}$ changes as the size of a bounding box w increases for three different backbones. The plots are different by backbones since R_W's are different as 291, 835 and 219 for ResNet-50, ResNet-101 and MobileNet, respectively, although D's are the same for 16.[1] Notice that while $DRRF_{1/2}$ is not significantly large when $w \geq 100$, it dramatically increases as w decreases. In particular, for the range of w where we treat bounding boxes as small ($\leq 96 \times 96$), this discrepancy is severely large. Therefore, it is not reasonable to use the features from the existing feature extractor as targets to train super-resolution model.

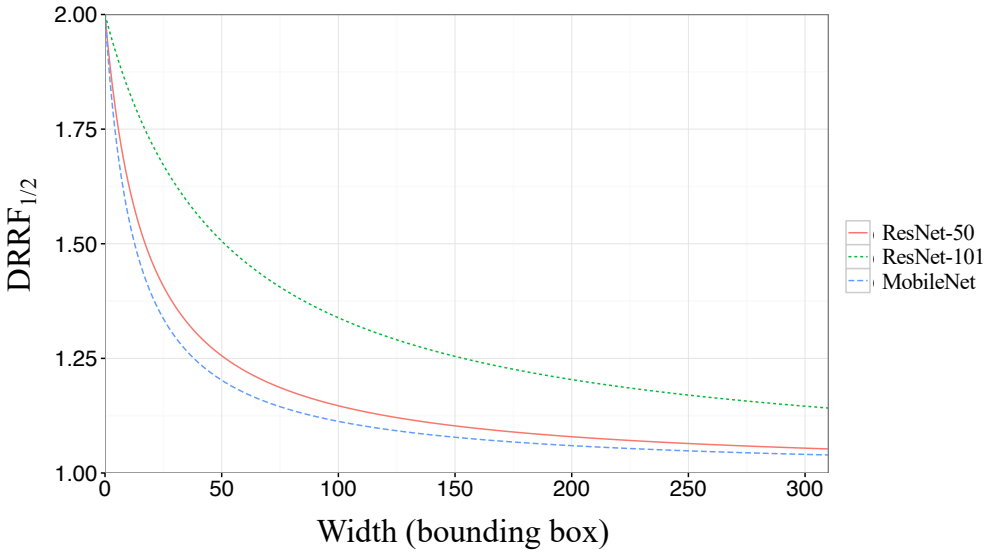


Figure 8: As the size of the bounding box decreases, DRRF, as defined in equation 7, increases. It implies that if the size of a proposal is large, the discrepancy in RRF is not significant. However, it can be significantly large when the size of a proposal is small.

In comparison with the DRRF from the existing feature extractor, let's take a closer look at the DRRF when super-resolution target extractor is used. We denote it as $DRRF^{SR}_{1/2}$ and it is computed as Eq. (8). For the super-resolution target extractor, the size of the receptive field corresponding to a single feature cell is approximately two times larger than that of the backbone feature extractor. Thus, R_W in Eq.(2) is replaced by $2R_W$ to compute RRF of the super-resolution target extractor which is denoted as RRF^{SR}.

$$
\begin{aligned}
DRRF^{SR}_{1/2}(w, I_W) &= \frac{RRF(w/2, I_W/2)}{RRF^{SR}(w, I_W)} \\
&\approx \frac{\left(R_W + (w/2 - 1) \times D\right) / (I_W/2)}{\left(2R_W + (w - 1) \times D\right) / I_W} \\
&= \frac{2R_W + wD - 2D}{2R_W + wD - D} \\
&= 1 - \frac{1}{\left(\frac{2R_W}{D} - 1\right) + w} \\
&= 1 - \frac{1}{c' + w}
\end{aligned} \tag{8}
$$

For ResNet-50 with $w = 1$, $DRRF^{SR}_{1/2} = 0.97$, which implies the RRFs are almost identical whereas $DRRF_{1/2} = 1.95$.

[1] We calculate ARF referring to TensorFlow library.

2. Losses

In the main paper, we define the losses with abuse of notation regarding thresholds due to the limited space. Instead, we briefly explain how thresholds are applied to compute losses. In this section, we rigorously explain how the losses are defined without abuse of notation in the order of content, adversarial, classification and localization losses.

Let $area_i^{1.0}$ and $area_i^{0.5}$ be the area of i-th RoI on the original input image $I^{1.0}$ and downsampled image $I^{0.5}$, respectively. First, content loss ($\mathcal{L}_{cont}$) are computed as follows.

$$\mathbb{I}_{cont,i} = \begin{cases} \mathbf{1}, & \text{if } area_i^{1.0} > l_{cont} \wedge area_i^{0.5} \leq u_{cont} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{9}$$

$$\mathcal{L}_{cont} = \sum_{i=1}^{N} \mathbb{I}_{cont,i} \|\mathbf{T}_i^{1.0} - \mathbf{S}_i^{0.5}\|_2^2. \tag{10}$$

where l_{cont} and u_{cont} denote the lower and upper bounds of $area_i^{1.0}$ and $area_i^{0.5}$ regarding $\mathcal{L}_{cont}$. Note that only the RoIs that satisfy the condition of the indicator are used to compute $\mathcal{L}_{cont}$. If $area_i^{1.0}$ is too small, $\mathbf{T}_i^{1.0}$ is not a desirable target for $\mathbf{S}_i^{0.5}$ to follow due to the low-resolution of $\mathbf{T}_i^{1.0}$. On the other hand, if $area_i^{0.5}$ is too large, $\mathbf{F}_i^{0.5}$ is detail enough to not need further enhancement through super-resolution.

Super-resolution discriminator takes a role of distinguishing between $\mathbf{T}_i^{1.0}$ and $\mathbf{S}_i^{0.5}$. In other words, it should be able to discriminate high-resolution target features and super-resolved features. To this end, adversarial losses ($\mathcal{L}_{gen}$, $\mathcal{L}_{dis}$) are defined as follows.

$$\mathbb{I}_{adv,i}^{+} = \begin{cases} \mathbf{1}, & \text{if } area_i^{1.0} > t_{adv} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{11}$$

$$\mathbb{I}_{adv,i}^{-} = \begin{cases} \mathbf{1}, & \text{if } area_i^{0.5} \leq t_{adv} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{12}$$

$$\mathcal{L}_{gen} = -\sum_{i=1}^{N} \mathbb{I}_{adv,i}^{-} \log D(\mathbf{S}_i^{0.5}) \tag{13}$$

$$\mathcal{L}_{dis} = -\sum_{i=1}^{N} \left(\mathbb{I}_{adv,i}^{+} \log D(\mathbf{T}_i^{1.0}) + \mathbb{I}_{adv,i}^{-} \log \left(1 - D(\mathbf{S}_i^{0.5})\right) \right) \tag{14}$$

where t_{adv} denotes a threshold for both $area_i^{1.0}$ and $area_i^{0.5}$ regarding $\mathcal{L}_{gen}$ and $\mathcal{L}_{dis}$. For instance, only high-resolution features corresponding to the large enough RoIs whose area ($area_i^{1.0}$) is larger than t_{adv} are involved in computing $\mathcal{L}_{dis}$. On the other hand, super-resolution features corresponding to the small enough RoIs whose area ($area_i^{0.5}$) is smaller than t_{adv} are used to compute both $\mathcal{L}_{gen}$ and $\mathcal{L}_{dis}$.

Lastly, for $\mathcal{L}_{cls}$ and $\mathcal{L}_{loc}$, t_{main} is used to determine whether i-th RoI is treated as small or large. If $area_i^{1.0}$ is larger than t_{main}, $\mathbf{F}_i^{1.0}$ is passed into the large predictor. Otherwise, $\mathbf{F}_i^{1.0}$ is first super-resolved into $\mathbf{S}_i^{1.0}$ through the super-resolution feature generator and then passed to the small predictor. The values used for thresholds on different datasets are provided in Table 5.

The final loss for the SR feature generator is computed as the weighted sum of content, generator, classification and localization losses. The weight of each loss is the multiplication of the reciprocal of normalizer which is the same as the number of proposals that satisfy the aforementioned conditions, and a coefficient as a hyperparameter. For instance, since the normalizer for the generator loss is $\sum_{i=1}^{N} \mathbb{I}_{adv,i}^{-}$ and the coefficient is set to 1, the weight of the generator loss is $1/(\sum_{i=1}^{N} \mathbb{I}_{adv,i}^{-})$. The normalizers for the other losses can be simply calculated by summing the indicators as with the generator loss except for the content loss. We further reduce the normalizer of the content loss by the number of features (Height $\times$ Width $\times$ Channel) of $T_i^{1.0}$ to prevent the content loss from being dominant. On the other hand, the coefficients for content, generator, classification and localization losses are set to $5, 1, 1$ and 2, respectively.

3. Configuration Details

In this section, we clarify the configurations of both our model architecture and training in details. We generally follow the configurations used in Faster RCNN [4, 10] for the base model. More specifically, for a **sub** and **base** layer in Figure 3,

Dataset	l_{cont}	u_{cont}	t_{adv}	t_{main}
Tsinghua-Tencent 100K [12]	16×16	32×32	32×32	32×32
PASCAL VOC [2], MS COCO [9]	16×16	128×128	96×96	96×96

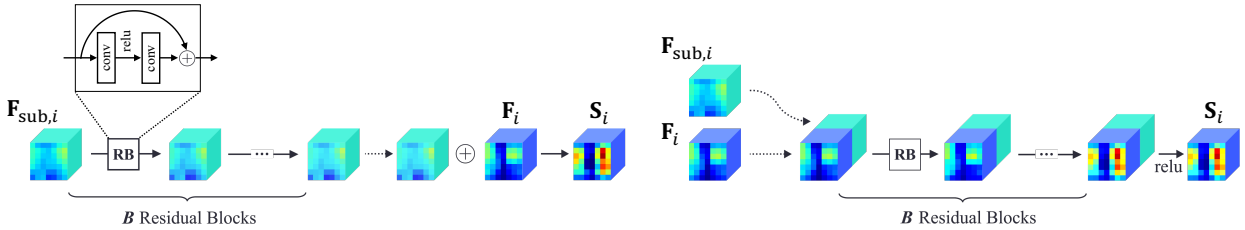
Table 5: The lower/upper bounds (l_{cont}, u_{cont}) and thresholds (t_{adv}, t_{main}) used to filter out the invalid features of proposals for different losses on different datasets.

we use *conv1* block and *conv4* block for ResNet, and *conv4dw* and *conv11* for MobileNet. For the super-resolution part, the output channels of the first convolution layer before $\mathbf{F}_{\text{sub}}$ for ResNet and MobileNet are set to 512 and 256, respectively. Also, we set the number of residual blocks (B) in the super-resolution feature generator to 3. In terms of training, we use stochastic gradient descent with momentum of 0.9, and train the generator twice for every training of the discriminator. Lastly, we implement all of our algorithms using TensorFlow [1, 6] and employ the implementation of third-party for RoI pooling[2] as well as RoI align [11].

4. Ablation Experiments of Model Architecture

In this section, we present more experimental results for our model architecture. More specifically, we measure the contribution of each key component on the performance of our model. The following results are based on Tsinghua-Tencent 100K [12] with ResNet-50 [4] as a backbone unless otherwise stated.

The first experiment is on the structure of the super-resolution target generator. Figure 9 shows two different structures of the generator. Figure 9a is the structure of the generator proposed in Perceptual GAN [7] whereas Figure 9b describes the generator used in our model. In Figure 9a, the feature ($\mathbf{F}_{\text{sub},i}$) extracted from the **sub** layer is enhanced through $B(=6)$ residual blocks and combined with the feature ($\mathbf{F}_i$) from the **base** layer at the end. Each residual block consists of two 3×3 convolution layers followed by batch normalization. On the other hand, our proposed generator consists of $B(=3)$ residual blocks which take the concatenated feature as an input. After iteratively refined through the residual blocks, the first half of the feature is sliced out. Table 6 shows the meaningful increases in metrics on both small and medium objects using our generator compared with the generator proposed in Perceptual GAN.



(a) Perceptual GAN [7] (b) Ours

Figure 9: The structures of Perceptual GAN and our super-resolution feature generator.

Generator type	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
Perceptual GAN [7]	76.1	86.3	80.9	92.1	**94.3**	93.2	93.4	**93.0**	93.2	87.3	90.4	88.8
Ours	**78.2**	**86.5**	**82.2**	**94.7**	93.8	**94.3**	**93.6**	**93.0**	**93.3**	**88.4**	**91.1**	**89.7**

Table 6: Comparison on the different architectures of the super-resolution feature generator with ResNet-50 on Tsinghua-Tencent 100K.

For the next ablation study, we compare the performance by varying the **sub** layer from *conv1* to *conv3*. As stated in the main paper, we extract sub-features from the earlier layer to secure the fine and high-frequency information. The results provided in Table 7 align with our assumption that the features from the earlier layer contain more detailed information so that they help to identify small objects better.

Lastly, we compare the model architectures with single unified predictor and two separated predictors: small and large predictors. In fact, we previously designed our model to have one shared predictor, but we changed our model to have two

[2]https://github.com/endernewton/tensorflow

Layer name	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
conv1	**78.2**	**86.5**	**82.2**	**94.7**	**93.8**	**94.3**	**93.6**	93.0	**93.3**	**88.4**	**91.1**	**89.7**
conv2	76.7	86.0	81.1	93.2	**93.8**	93.5	93.5	**93.1**	**93.3**	87.6	90.2	88.9
conv3	77.2	75.2	76.2	92.6	92.7	92.7	93.4	91.2	92.3	86.9	85.8	86.3

Table 7: Comparison on the super-resolution feature generators using different sub layers with ResNet-50 on Tsinghua-Tencent 100K.

separate predictors because the super-resolved features cannot perfectly imitate the high-resolution target features. According to Table 8, adding a small predictor (Separated) gives slight improvement over the model with only the large predictor (Unified). If one considers time/memory complexity more important, only one shared predictor can be employed.

Predictor	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
Unified	77.7	86.4	81.8	94.6	**94.0**	**94.3**	93.3	92.9	93.1	88.0	**91.1**	89.5
Separated	**78.2**	**86.5**	**82.2**	**94.7**	93.8	**94.3**	**93.6**	**93.0**	**93.3**	**88.4**	**91.1**	**89.7**

Table 8: Comparison on the number of predictors used with ResNet-50 on Tsinghua-Tencent 100K.

5. Super-Resolution of RoI Aligned Features

In this section, we demonstrate our super-resolution method consistently improves the performance even when RoI align operation is applied as a substitute for RoI pooling. As stated in the main paper, RoI align operation alleviates the distortion issue of RoI pooling operation. Since small RoIs do not contain enough information in the first place, however, we expect our super-resolution method can still help improve the features extracted using RoI align method.

Model	Small			Medium			Large			Overall		
	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1	Rec.	Acc.	F1
MobileNet [5]	56.1	72.9	63.4	**85.1**	**84.3**	**84.7**	**90.9**	83.6	**87.1**	**74.7**	80.7	**77.5**
+ RoI align [3]	**56.8**	**76.9**	**65.3**	83.4	83.8	83.6	86.7	**84.2**	85.4	73.7	**81.6**	77.4
ResNet-50 [4]	**68.8**	**81.9**	**74.9**	**90.8**	**93.1**	**91.9**	**91.6**	92.3	**91.9**	**82.5**	**89.2**	**85.7**
+ RoI align [3]	66.1	81.1	72.9	90.2	**93.1**	91.6	90.1	**93.1**	91.6	81.1	89.0	84.9
ResNet-101 [4]	69.8	81.5	75.2	90.9	93.5	92.2	**92.4**	**92.0**	**92.2**	83.1	89.2	86.0
+ RoI align [3]	**70.4**	**82.3**	**75.9**	**91.8**	**93.9**	**92.8**	91.5	91.9	91.7	**83.6**	**89.6**	**86.5**

Table 9: Comparison between RoI pooling and align on Tsinghua-Tencent 100K *test* set. There is no significant difference between RoI pooling and align methods for the base models.

According to the results of the base models on Tsinghua-Tencent 100K dataset as shown in Table 9, there is not much difference between RoI pooling and align methods. It may be because small objects are dominant in Tsinghua-Tecent 100K dataset. Hence, we performance the experiments only on PASCAL VOC dataset where we confirm the improvement of using RoI align is not marginal. Table 10 shows the performance of the base models and ours when RoI align is applied. Although, as expected, the improvement is not as large as the case of RoI pooling, we still achieve the significant and consistent improvement with different backbone structures.

Model	mAP	AP-S	AP-M	AP-L	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MobileNet [5]	71.5	9.1	40.2	72.8	74.3	**81.3**	69.7	58.2	49.2	79.6	85.8	81.9	49.2	75.5	68.2	79.1	83.0	79.3	81.6	41.3	73.7	**71.4**	78.2	70.1
+ Ours	**73.4**	**13.0**	**42.6**	**73.2**	**81.1**	**81.3**	**71.8**	**60.1**	**58.0**	**80.9**	**88.5**	**82.1**	**50.7**	**77.6**	**68.3**	**79.2**	**83.4**	**80.4**	**83.3**	**44.6**	**74.2**	**71.4**	**78.4**	**72.9**
ResNet-50 [4]	77.6	11.5	45.0	81.2	82.4	81.7	79.8	**71.0**	63.7	**89.4**	88.3	88.8	56.0	**85.4**	**65.1**	90.3	88.6	82.8	83.1	44.6	82.1	**73.7**	83.7	72.2
+ Ours	**79.4**	**15.4**	**47.8**	**82.0**	**87.3**	**84.7**	**83.0**	67.8	**70.1**	86.9	**90.3**	**90.1**	**58.9**	84.5	64.0	**91.1**	**91.3**	**83.8**	**85.6**	**50.2**	**83.3**	73.1	**85.1**	**77.1**
ResNet-101 [4]	79.0	9.9	48.1	82.1	81.5	83.7	81.7	70.7	66.8	88.2	90.1	89.7	**59.8**	**88.9**	63.9	91.8	**87.2**	82.5	83.9	46.2	**82.5**	79.0	84.9	77.6
+ Ours	**80.7**	**16.5**	**50.3**	**82.8**	**87.7**	**86.9**	**83.6**	**71.6**	**69.5**	**90.4**	**91.5**	**90.6**	58.8	86.6	**66.7**	**92.0**	**87.2**	**86.1**	**87.1**	**50.5**	82.2	**80.3**	**86.6**	**78.5**

Table 10: Detailed performance on VOC 2007 *test* set. For both base models and ours, RoI align is applied instead of RoI pooling. Ours still achieve significant improvement regardless of CNN backbones, especially for small subset (AP-S).

6. Experiments on MS COCO dataset

We present the additional experimental results on MS COCO 2017 *val* set to further verify the generality of our model along with the detailed performance results on *test-dev* set. Table 11 shows the detection performance on *val* set of COCO 2017. As with the results on the other datasets, we can verify the similar tendency on the *val* set. First, all performance measures increase as our super-resolution model is added. Second, the performance gains are more significant for small and medium subsets in both AP and AR metrics. Lastly, Table 12 shows more detailed results on *test-dev* set where we have already provided the brief results in the main paper.

Model	AP-.5:.95	AP-.5	AP-.75	AP-S	AP-M	AP-L	AR-1	AR-10	AR-100	AR-S	AR-M	AR-L
MobileNet [5]	19.4	38.7	17.1	3.5	16.6	30.6	20.4	33.5	35.8	11.7	34.1	51.8
+ Ours	**21.6**	**40.7**	**20.6**	**7.1**	**20.9**	**30.7**	**22.4**	**37.3**	**39.7**	**18.7**	**40.1**	**52.5**
ResNet-50 [4]	29.5	51.6	29.8	6.4	26.0	**45.3**	26.7	42.0	44.5	18.2	42.7	**61.8**
+ Ours	**31.0**	**53.7**	**32.0**	**10.0**	**28.6**	45.1	**27.7**	**44.3**	**46.8**	**23.7**	**46.6**	61.5
ResNet-101 [4]	31.9	54.5	32.6	7.6	27.9	48.9	28.4	43.8	46.5	19.7	44.4	63.8
+ Ours	**34.0**	**56.6**	**35.7**	**11.6**	**31.5**	**49.0**	**29.5**	**46.7**	**49.3**	**26.5**	**49.2**	**63.9**

Table 11: Detailed performance on COCO 2017 *val* set. AP and AR denote the average precision and average recall. Also, S, M and L denote the subset of small ($area \leq 32 \times 32$), medium ($32 \times 32 < area \leq 96 \times 96$) and large ($area > 96 \times 96$) objects, respectively. AR-$\{1, 10, 100\}$ means the average recall given $\{1, 10, 100\}$ detections per image.

Model	AP-.5:.95	AP-.5	AP-.75	AP-S	AP-M	AP-L	AR-1	AR-10	AR-100	AR-S	AR-M	AR-L
MobileNet [5]	19.3	38.7	16.9	5.4	20.6	**29.2**	20.3	32.3	34.0	12.2	36.5	**53.2**
+ Ours	**21.9**	**41.0**	**21.0**	**10.9**	**23.8**	29.0	**22.4**	**36.4**	**38.1**	**19.3**	**41.3**	53.1
ResNet-50 [4]	29.5	52.0	29.8	10.2	31.5	**44.7**	27.0	41.7	43.6	20.1	46.8	64.7
+ Ours	**31.2**	**54.2**	**32.4**	**14.3**	**32.4**	**44.7**	**28.2**	**44.2**	**46.0**	**25.0**	**49.3**	**64.8**
ResNet-101 [4]	32.0	54.7	32.8	11.3	34.3	**48.1**	28.5	43.6	45.7	21.5	48.9	**67.3**
+ Ours	**34.2**	**57.2**	**36.1**	**16.2**	**35.7**	**48.1**	**29.9**	**46.7**	**48.8**	**28.1**	**51.8**	67.2

Table 12: Detailed performance on COCO 2017 *test-dev* set. The notations are consistent with Table 11.

7. Detailed Performance on PASCAL VOC

We provide the performance of our proposed model by class on PASCAL VOC dataset as shown in Table 13. Notice that we obtain the substantial improvement on the categories where the objects tend to be small, such as *bottle* and *plant*.

Model	mAP	AP-S	AP-M	AP-L	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MobileNet [5]	73.2	5.1	39.3	75.9	77.5	81.7	70.5	61.0	52.3	81.0	86.4	85.7	51.7	79.0	66.6	80.2	85.7	80.4	81.3	43.9	76.6	71.6	79.6	71.7
+ Ours	**77.0**	**10.1**	**47.2**	**76.9**	**82.2**	**84.6**	**76.0**	**63.3**	**61.4**	**82.8**	**89.9**	**87.6**	**56.3**	**82.7**	**69.3**	**83.3**	**87.7**	**81.7**	**84.4**	**53.9**	**81.0**	**72.9**	**80.4**	**77.8**
ResNet-50 [4]	77.1	6.8	42.9	81.1	80.7	**84.7**	78.9	68.2	62.3	85.7	87.4	89.9	55.9	84.5	63.1	88.4	88.5	80.3	83.3	47.8	80.6	72.9	**83.5**	74.5
+ Ours	**79.1**	**10.5**	**47.9**	**81.4**	**82.5**	84.6	**81.3**	**73.0**	**65.1**	**87.2**	**89.5**	**91.6**	**58.1**	**86.4**	**63.4**	**91.6**	**90.9**	**82.4**	**84.6**	**52.9**	**82.7**	**74.9**	82.6	**77.6**
ResNet-101 [4]	78.8	5.9	46.2	82.3	80.8	84.6	82.0	68.4	64.8	86.9	90.1	**91.6**	58.7	84.1	60.8	**90.2**	88.9	83.4	84.2	50.1	83.2	76.4	**87.9**	77.3
+ Ours	**80.6**	**11.1**	**48.9**	**82.7**	**84.3**	**85.3**	**83.5**	**70.6**	**68.8**	**87.6**	**90.4**	91.3	**59.4**	**88.2**	**62.4**	89.9	**90.9**	**84.7**	**86.4**	**54.8**	**86.3**	**79.8**	87.4	**79.9**

Table 13: Detailed performance on VOC 2007 *test* set.

8. Detailed Performance on Tsinghua-Tencent 100K

In this section, we provide the performance of our proposed model by class compared to the state-of-the-art models. In term of F1 score which takes both accuracy and recall into account, our model achieves the highest F1 scores for about two thirds of 45 classes as shown in Table 14.

9. Comparison on Features from Different Extractors and Super-Resolution Variants

To compare the target features from the existing feature extractor and our proposed SR feature extractor, we provide the feature maps as shown in Figure 10. We can easily tell the feature maps from the existing feature extractor $\mathbf{F}^{1.0}$ are

Class		i2	i4	i5	il100	il60	il80	io	ip	p10	p11	p12	p19	p23	p26	p27
Zhu *et al*. [12]	R	82	94	95	97	91	94	89	92	**95**	91	89	94	94	93	96
	A	72	83	92	**100**	91	93	76	87	78	89	88	53	87	82	78
	F	76	88	93	**98**	91	93	81	89	85	89	88	67	90	87	86
Perceptual GAN [7]	R	84	95	95	95	92	95	92	91	89	**96**	**97**	**97**	95	94	**98**
	A	85	**92**	**94**	97	95	83	79	**90**	84	85	88	84	92	83	**98**
	F	84	93	94	95	93	88	85	90	86	90	92	90	**93**	88	**98**
Liang *et al*. [8]	R	87	**97**	**96**	97	**98**	**100**	**94**	88	92	95	95	91	94	**95**	**98**
	A	**90**	**92**	**94**	93	**98**	94	**86**	**90**	**89**	**90**	**94**	75	**93**	**89**	**98**
	F	88	**94**	94	94	**98**	96	89	88	90	**92**	**94**	82	**93**	91	**98**
Ours	R	**94**	96	**96**	**100**	97	97	**94**	**94**	**95**	**96**	95	93	**97**	**95**	97
	A	87	91	**94**	97	97	**96**	**86**	88	87	87	88	**96**	87	**89**	93
	F	**90**	**94**	**95**	**98**	97	**97**	**90**	**91**	**91**	**92**	91	**95**	92	**92**	95

Class		p3	p5	p6	pg	ph4	ph4.5	ph5	pl100	pl120	pl20	pl30	pl40	pl5	pl50	pl60
Zhu *et al*. [12]	R	91	95	87	91	82	88	82	98	**98**	96	94	96	94	94	93
	A	80	89	87	**93**	94	**88**	**89**	97	**100**	90	90	89	84	87	93
	F	85	91	87	91	87	88	85	97	**98**	92	91	92	88	90	93
Perceptual GAN [7]	R	93	96	**100**	93	78	88	85	96	**98**	96	93	96	92	96	91
	A	92	90	83	**93**	**97**	68	69	97	98	**92**	91	90	86	87	92
	F	92	92	90	93	86	76	76	96	**98**	**93**	91	92	88	91	91
Liang *et al*. [8]	R	**96**	98	97	98	86	90	90	**100**	97	**98**	**97**	**97**	94	**97**	**98**
	A	81	91	90	**93**	94	80	78	**98**	99	90	**92**	91	**92**	90	**95**
	F	87	94	93	**95**	89	84	83	**98**	97	**93**	**94**	93	**92**	**93**	**96**
Ours	R	94	**99**	92	**100**	**94**	**94**	**92**	98	96	89	96	96	**95**	94	94
	A	**93**	**93**	**97**	91	94	87	85	97	97	87	85	**93**	89	**93**	93
	F	**94**	**95**	**94**	**95**	**94**	**91**	**88**	97	97	88	90	**94**	**92**	**93**	94

Class		pl70	pl80	pm20	pm30	pm55	pn	pne	po	pr40	w13	w32	w55	w57	w59	wo
Zhu *et al*. [12]	R	**93**	95	88	91	95	91	93	67	98	65	71	72	79	82	45
	A	95	**94**	91	81	60	**92**	93	84	76	65	89	86	**95**	**75**	52
	F	93	94	89	85	73	91	93	74	85	65	78	78	86	78	48
Perceptual GAN [7]	R	91	**99**	88	94	**100**	**96**	**97**	83	97	**94**	85	**96**	94	**95**	53
	A	97	86	90	77	81	89	93	78	92	66	83	88	93	71	54
	F	93	92	88	84	89	**92**	94	80	94	77	83	91	**93**	**81**	53
Liang *et al*. [8]	R	**93**	**99**	**94**	**96**	97	**96**	96	82	**100**	90	91	95	94	93	42
	A	98	92	**98**	**97**	86	90	**97**	81	97	**90**	**95**	**95**	90	68	50
	F	95	**95**	**95**	**96**	91	**92**	**96**	81	98	**90**	**92**	**95**	91	78	45
Ours	R	**93**	96	87	90	94	**96**	96	**91**	**100**	83	**94**	93	**95**	91	**71**
	A	**100**	89	93	85	**92**	88	93	**88**	**98**	64	88	94	92	73	**65**
	F	**96**	93	90	87	**93**	**92**	94	**89**	**99**	73	91	94	**93**	**81**	**68**

Table 14: Detailed performance on Tsinghua-Tencent 100K *test* set. R, A and F refer to recall, accuracy and F1 score, respectively.

significantly different from the low-resolution feature maps $\mathbf{F}^{0.5}$. However, the feature maps from the SR feature extractor $\mathbf{T}^{1.0}$ are fairly close to the low-resolution feature maps $\mathbf{F}^{0.5}$, which demonstrates the validity of using our proposed SR feature extractor.

As an extension of Figure 6 in the main paper, we further provide examples that demonstrate the effectiveness of our model compared to the other super-resolution methods: SR without supervision and SR with naïve supervision. In Figure 11, SR without supervision does not improve the features much compared to the low-resolution features and the similar pattern appears for SR with naïve supervision although there are the target features (Naïve). On the other hand, the low-resolution input features (LR) are reasonably well super-resolved into their target features (Ours) through our super-resolution model.

10. Comparison on Failure and Success Cases

We have examined the prediction examples to see if our approach has particular weaknesses. Figure 12 shows some of the failure cases of our model. For each row, we present the test result of the base model on the left, our model in the middle and groundtruth on the right. As illustrated in Figure 12, the most common failure cases of our model are due to false positives. For the objects that the base model detects as the background (false negatives), our model recognizes them as objects but for wrong classes (false positive). For instance, in the first row, our model recognizes the *p130* sign as *pm30* which the base model recognizes as the background. In fact, false positives tend to lower the detection metrics such as F1 score and mAP than false negatives. However, given the fact that the performance of our model is significantly higher than the base model, we can infer our model makes correct predictions (true positives or true negatives) more than the base model in general.

Figure 13–15 demonstrate the superiority of our model with some selected examples from Tsinghua-Tencent 100K, PASCAL VOC and COCO datasets. For each pair, we show the test results of the base model on the left and our model on the right. Compared to the base model, our model often detects small objects better with higher confidence.

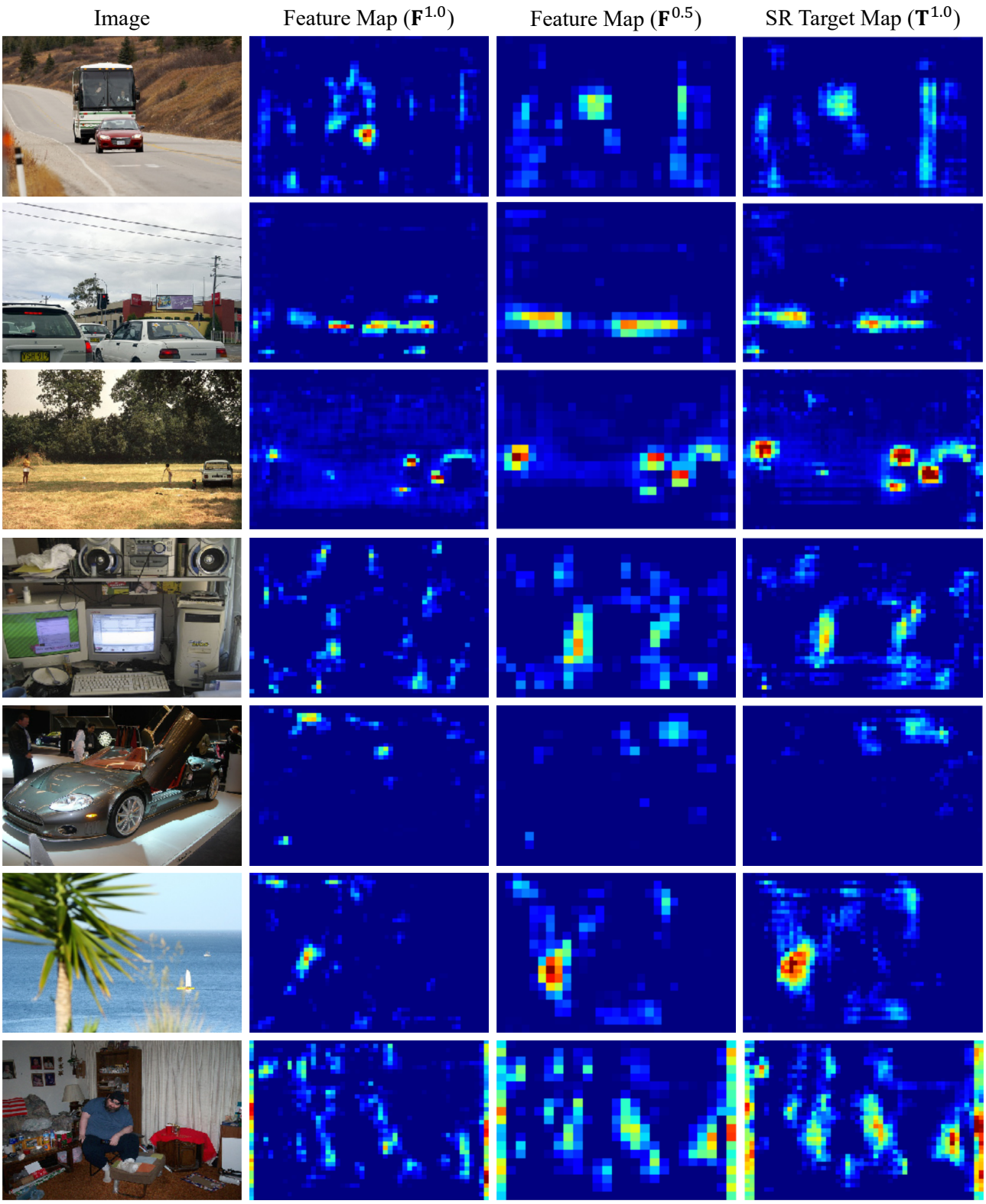


Figure 10: Comparison on the feature maps from different feature extractors. Both high-resolution ($F^{1.0}$) and low-resolution ($F^{0.5}$) feature maps are extracted from the existing feature extractor using high and low-resolution images, respectively, whereas the high-resolution target feature maps ($T^{1.0}$) are extracted from our proposed SR feature extractor.

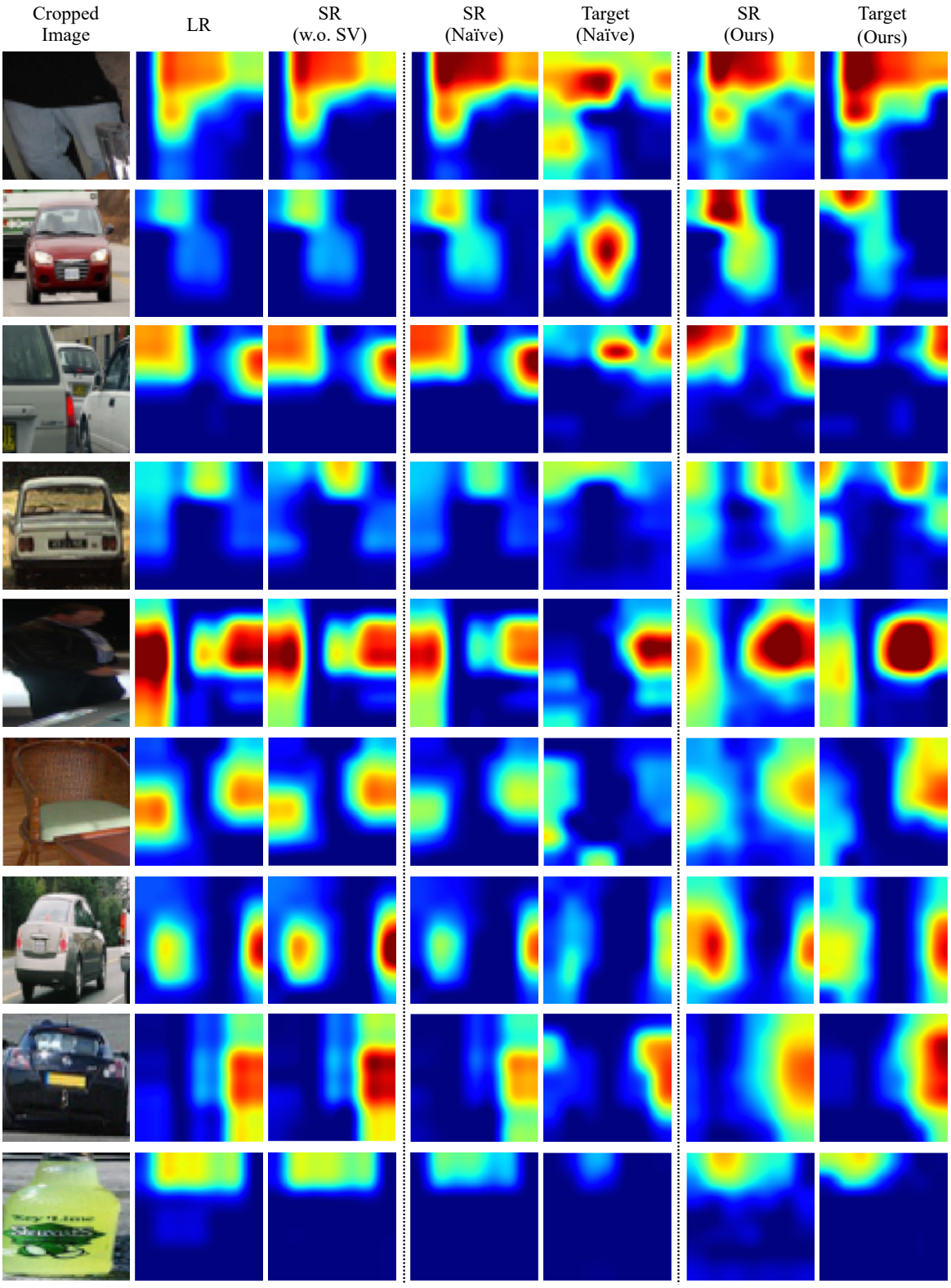


Figure 11: The qualitative results for how RoI features differ by different super-resolution methods on PASCAL VOC dataset. SR without supervision does not make much improvement compared to the input feature. Such tendency remains unchanged for the SR with naïve supervision method. On the other hand, ours look very close to its target feature.

Figure 12: Failure cases on Tsinghua-Tencent 100K (row1 and 2), PASCAL VOC 2007 (row3 and 4) and MS COCO 2017 (row5 and 6) datasets. For Tsinghua-Tencent 100K, green, red and blue rectangles represent true positives, false positives and false negatives, respectively. For each row, we show the test result of the base model (left), our model (middle) and groundtruth (right). The background is cropped out of some images for better visualization.

Figure 13: Detection examples on Tsinghua-Tencent 100K *test* dataset. Green, red and blue rectangles represent true positives, false positives and false negatives, respectively. Each pair indicates the results from the base model (left) and our model (right)

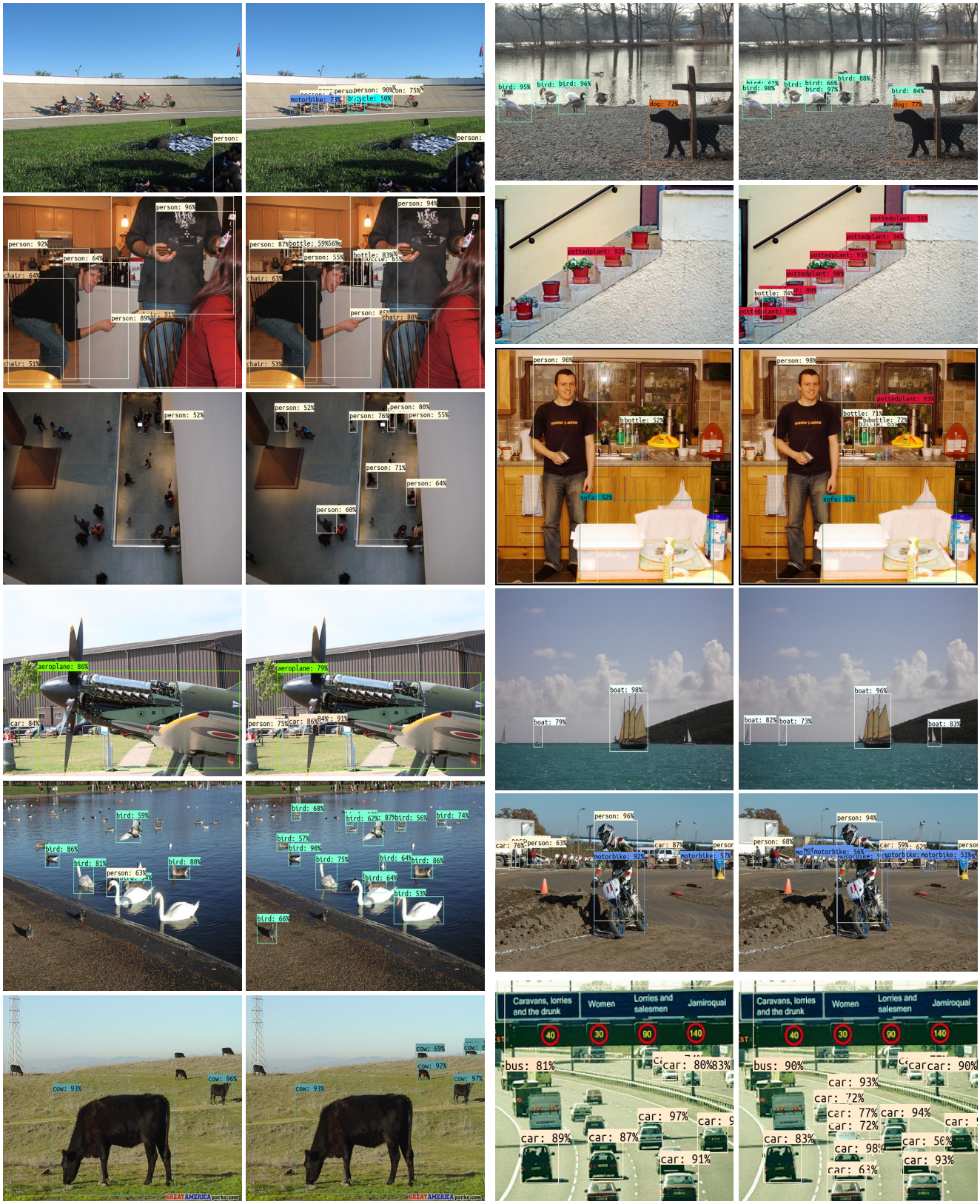


Figure 14: Detection examples on PASCAL VOC 2007 *test* dataset. For each pair, we show the test results of the base model (left) and our model (right). The background is cropped out of some images for better visualization.

Figure 15: Detection examples on MS COCO 2017 *val* dataset. For each pair, we show the test results of the base model (left) and our model (right). The background is cropped out of some images for better visualization.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 2010.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861*, 2017.

[6] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors. In *CVPR*, 2017.

[7] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual Generative Adversarial Networks for Small Object Detection. In *CVPR*, 2017.

[8] Zhenwen Liang, Jie Shao, Dongyang Zhang, and Lianli Gao. Small Object Detection Using Deep Feature Pyramid Networks. In *Pacific Rim Conference on Multimedia*, 2018.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015.

[11] Yuxin Wu et al. Tensorpack. `https://github.com/tensorpack/`, 2016.

[12] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-Sign Detection and Classification in the Wild. In *CVPR*, 2016.