AISE: Adaptive Input Sampling for Explanation of Black-box Models

Supplementary Material

Tsykunov Evgeny, Lee Wonju, and Minje Park

6 Visualization of the estimated saliency maps

Figure 5 shows estimated saliency maps per kernel width. Preserve paradigm is used. For the visualization, we used an image from the VOC dataset. The first column visualizes a perturbed image with the corresponding kernel width. To vary the Gaussian kernel width we changed the standard deviation (sigma). The kernel is manually placed near the right-most person just for visualization purposes. The third column displays samples that were obtained with a LIPO optimizer which has the objective to maximize the saliency score. We used 70 samples for visualization purposes. The fourth column shows the result of KDE using all 70 samples weighted with corresponding saliency scores. The right-most column shows the final saliency map obtained per sigma.

It is possible to see the local optima regions are more narrow for the small kernel width (e.g. see sigma 0.05 from the first row in Fig. 5). The search algorithm might have a harder time finding narrow important areas. For example, KDE output from the first row shows that LIPO was not able to find salient features of the right-most person (number of samples need to be increased for that). On the other side, more wide optimal regions, e.g. generated with sigma 0.2, are more easily discovered by search algorithm. It is also fair to claim that wide optimal regions with high probability contain narrow regions obtained with small sigma values. All these observations potentially might be used to even more speed up the adaptive search.

7 Global optimizer

7.1 LIPO: Global optimization of Lipschitz functions.

One of recent approaches that meets our requuirements is the global optimization of Lipschitz functions proposed by [12]. Thanks to the Lipschitz property of the saliency map (maximum rate of growth is limited) and the robustness of the optimizer, we can use this lightweight and parameter-free optimization method for adaptive sampling purposes.

The main idea of [12] is that, having evaluations at pixels $p_1, p_2, \cdots, p_t$, it's possible to approximate an objective function $S(p)$ with an upper bound function $U(p)$ (such that $U(p) > S(p)$ for p $\in \Omega$):

$$U(p) = \min_{i=1\ldots t} (S(p_i) + l \cdot ||p - p_i||^2), \tag{6}$$

where l is a Lipschitz constant for S. The authors proposed a LIPO algorithm that randomly selects a point (pixel p_j) to evaluate and see if the upper bound $U(p_j)$ is better than the current best point. If the selected point is better than the current best one, it is used for evaluation. This helps to make the upper bound to be tight, minimizing $U(p) - S(p)$. Lipschitz constant l can be estimated for every optimization step to be equal to the biggest slope between already evaluated points.

7.2 DIRECT: Lipschitzian optimization without the Lipschitz constant.

Another attractive optimization technique for our purpose is the DIRECT algorithm initially proposed by [8] and it has been modified in many different ways by researchers [7]. DIRECT is popular since it eliminates the need for 'tuning parameter' to set the balance between local and global search. One significant difference compared to LIPO is that DIRECT is deterministic and initially has one parameter for the desired accuracy.

DIRECT algorithm is called after its main idea of *di*viding *rect*angles. The search space is normalized to form a unit hypercube (in general multi-dimensional case), and then the hypercube is divided into sub-cubes and black-box objective function is evaluated at the center of each sub-cube. The method operates iteratively by solving two problems: what to divide next (the most promising rectangles based on goal precision) and how to divide it. The principle of the algorithm applied to our problem is visualized in Figure 6. From Figure 6 is it possible to see that at each step there are multiple samples to be evaluated. Therefore, samples to infer might be combined in batch for batch inference to speed up the process.

8 Mask modeling

2-dimensional Gaussian mask $G(p,\sigma)$ is defined as follows

$$G(p,\sigma)_{raw} = \frac{1}{2\pi\sigma^2} exp(-\frac{p[0]^2 + p[1]^2}{2\sigma^2}) \tag{7}$$

$$G(p,\sigma) = \frac{G(p,\sigma)_{raw}}{max(G(p,\sigma)_{raw})} \tag{8}$$

Where p is the pixel that defines the location of the kernel, and σ is the standard deviation. Normalization to [0, 1] range is applied.

9 Implementation details

All classification results were obtained using the whole test set of PASCAL VOC 2007 dataset (4952 images) and 5000 first images from validation set of COCO dataset. For YOLO, first 1000 images from PASCAL VOC 2007 were used for evaluation.

Explanations for non-confident predictions are meaningless in most cases and thus can spoil the analysis. To make our experiments less prone to the model's prediction capability, we used explanations on confident predictions only. Predictions with confidence less than 0.5 are omitted in the experiment (0.5 is a common threshold choice for making a decision).

We used dlib library [10] for the LIPO implementation because dlib provides several improvements to the original LIPO. One of such improvements is to make local convergence fit and optimize (up to a certain precision) a quadratic surface around the best point seen so far. Exploration and exploitation steps follow each other, one by one, while generating samples. For DIRECT implementation we used scipy implementation [27]. Scipy has the option to use the locally biased form of DIRECT algorithm [5] but we found it less efficient and used the original unbiased DIRECT.

We used Gaussian kernel to mask the input image (we did not observe benefit from other kernel types), and the standard deviation is used for effective kernel width (see appendix

for more details). Saliency map generation does not require the optimizer to converge with the floating point precision, and thus we set the search precision limit to 0.1 for classification and 0.05 for detection (*solver_epsilon* parameter of dlib implementation for LIPO, *eps* parameter of scipy implementation of DIRECT), which slightly improves the computation efficiency.

In classification we used fixed linearly-spaced sigma values {0.1, ..., 0.25} in case of using three kernels and {0.075, ..., 0.25} for the four kernel case (having increased inference budget it is possible to afford more than three or four kernels, the same applies for detection). For example, having 150 inferences as a computing budget, we used 25 samples per each of three kernels (50 actual inferences per-kernel) with sigmas {0.1, 0.175, 0.25}.

For detection, we have prior information that we can leverage. Kernel widths are derived from min_box_size = min(box_height, box_width). In particular, sigmas are obtained via division of the min_box_size by {7, ..., 1} for three kernels and by {8, ..., 1} for more than three kernels. Thus, for 99 allowed inferences, for the min_box_size = 0.4 (from unit interval) and three kernels, we have 33 samples/inferences per each kernel with sigmas {0.06, 0.1, 0.4}. We also limit the search space to the box area plus the corresponding half-box size in each dimension (it is an extremely rare case when responsible pixels are located far away from the corresponding bounding box).

9.1 Implementation details (quantitative comparison)

Pointing game implementation is based on TorchRay package [3]. Max-Sensitivity implementation is based on Captum with 10 perturbed samples and 0.02 perturbation radius.

For AISE-300 we used 50 samples (100 inferences) per each of {0.1, 0.175, 0.25} kernels. For AISE-700 we used 87 samples (174 inferences) per each of {0.075, 0.13, 0.19, 0.25} kernels. For detection AISE-150 case, we used 50 samples per each of three kernels of adaptive size.

10 Qualitative comparison

For AISE-LIPO, LIPO [12] is used as an optimizer, while for AISE-DIRECT, DIRECT [8] is used as an optimizer.

10.1 Classification

Figures 7 and 8 show the qualitative comparison between AISE, RISE [14], and Extremal Perturbations [3] methods for classification task. Figure 9 demonstrates comparison for false positive examples. We use Resnet-50 trained on the VOC dataset.

10.2 Object detection

Figures 10 and 11 show the qualitative comparison between AISE and D-RISE [15] methods for object detection task. We use YOLOX trained on the VOC dataset.

10.3 Number of samples/iterations

For classification AISE is using 150 masks, while for detection 99 masks are uses in total. Masks were evenly distributed per each of three kernels. For classification, we used {0.1,

0.175, 0.25} kernel sigmas. For detection, kernel widths are obtained via division of the min_box_size by {7, 4, 1}, where min_box_size by = min(box_height, box_width).

RISE is using 8000 randomly sampled masks. Extremal Perturbations is using 800 forward and backward passes. D-RISE is using 5000 randomly sampled masks.

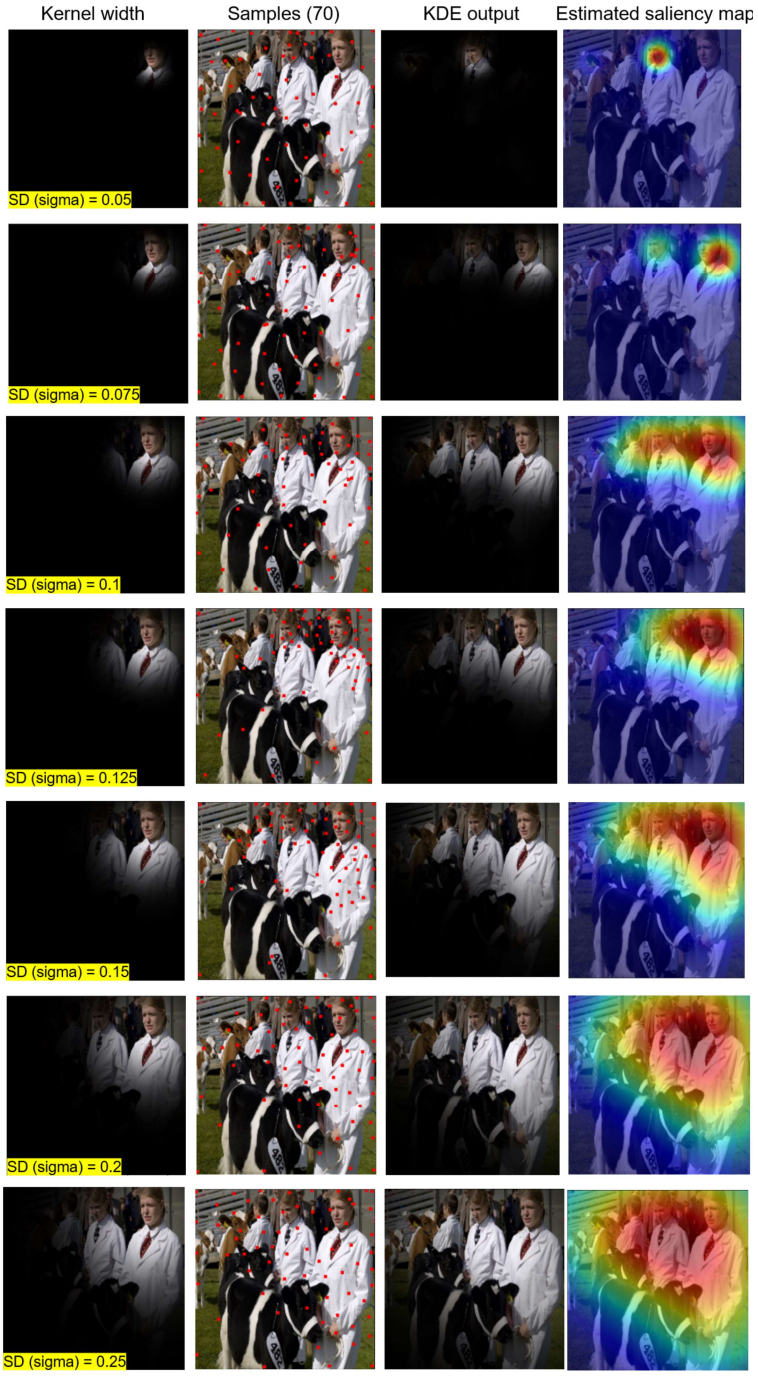

Figure 5: Estimated saliency maps for different kernel widths. The image is from the PASCAL VOC dataset with 99% confidence of a person class. The model is ResNet-50 trained on the VOC dataset. LIPO is an adaptive sampler. Preserve paradigm is used.

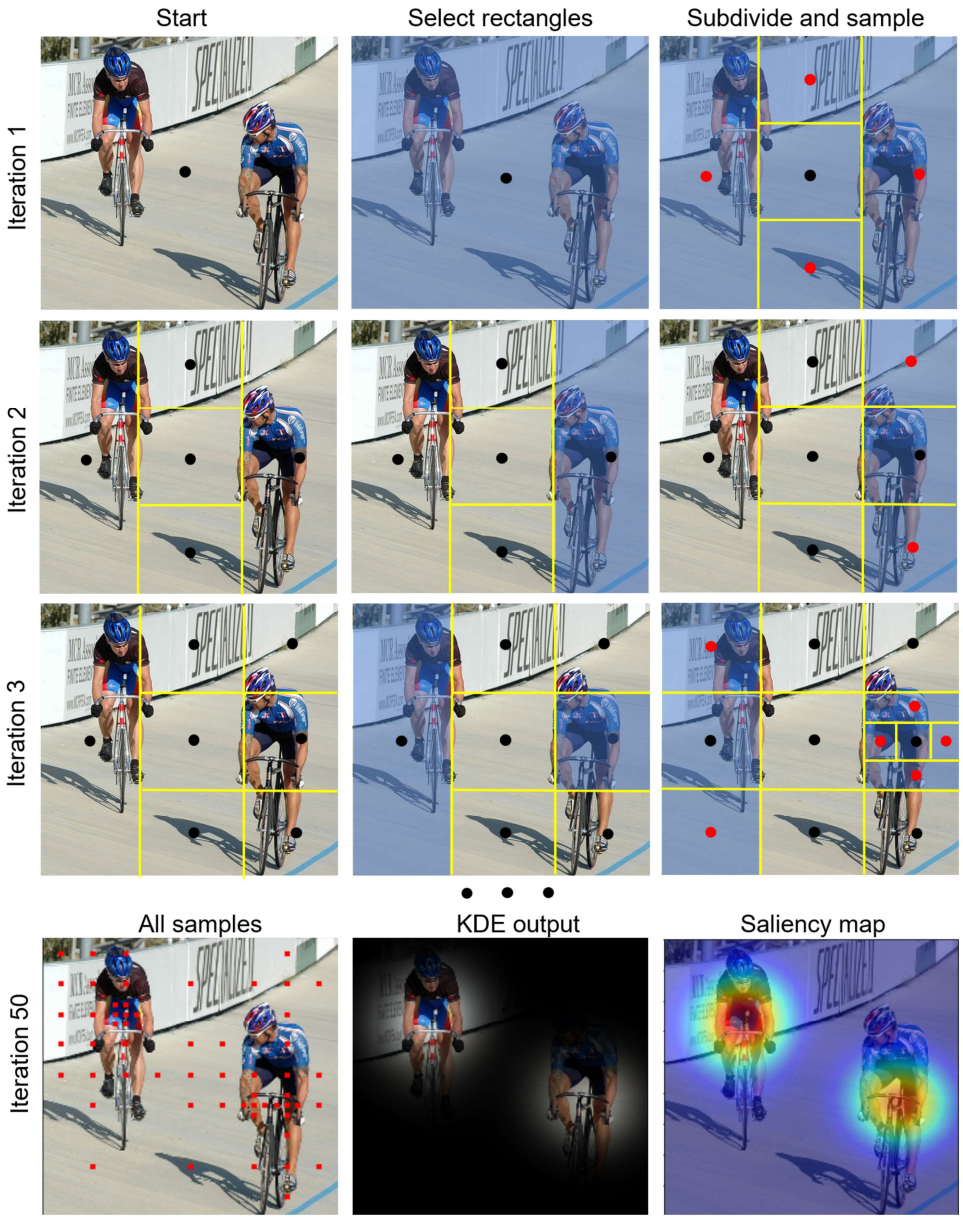

Figure 6: DIRECT first iterations and the result of 50 iterations with the target class of 'bicycle'.

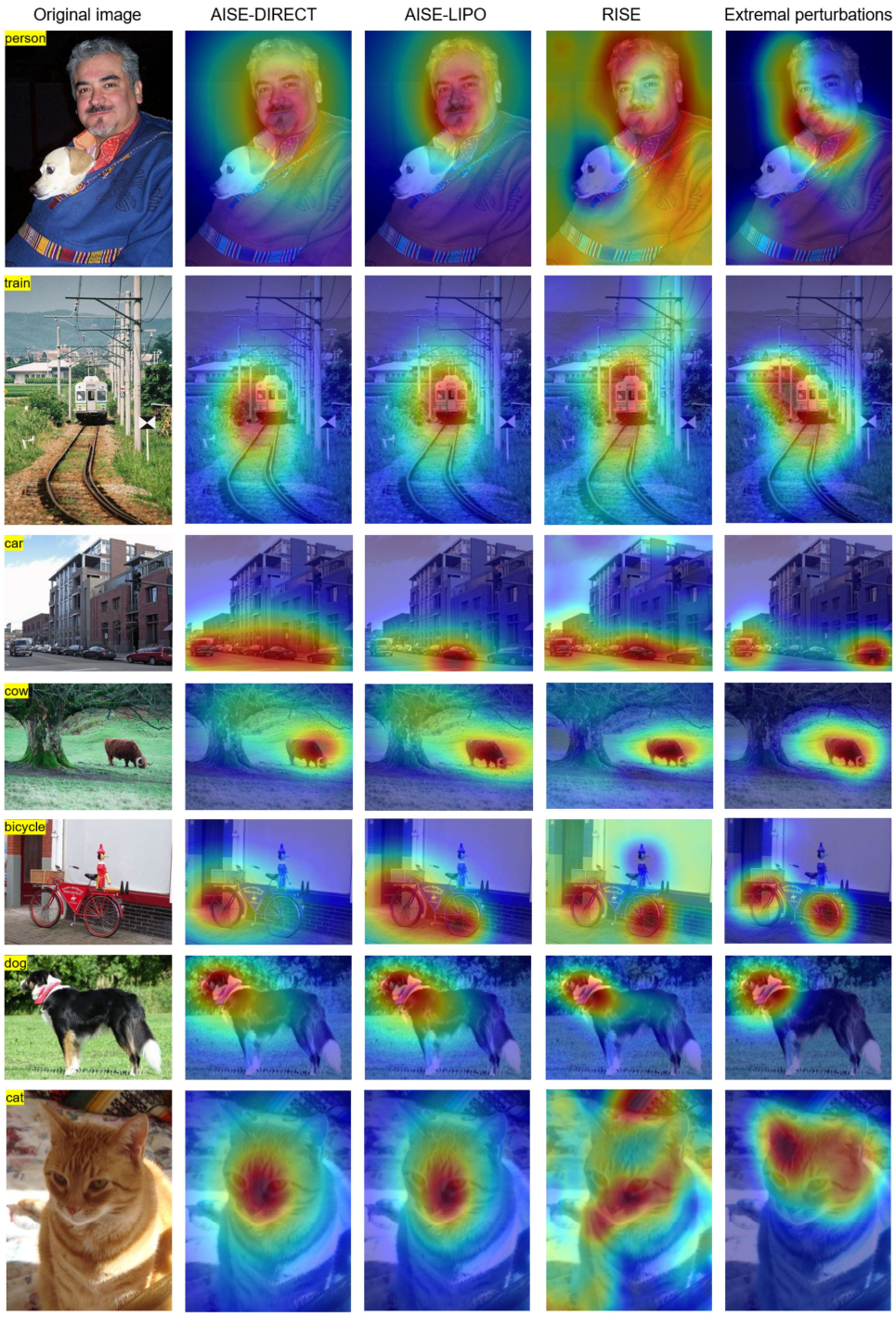

Figure 7: [Classification] Comparison with other methods.

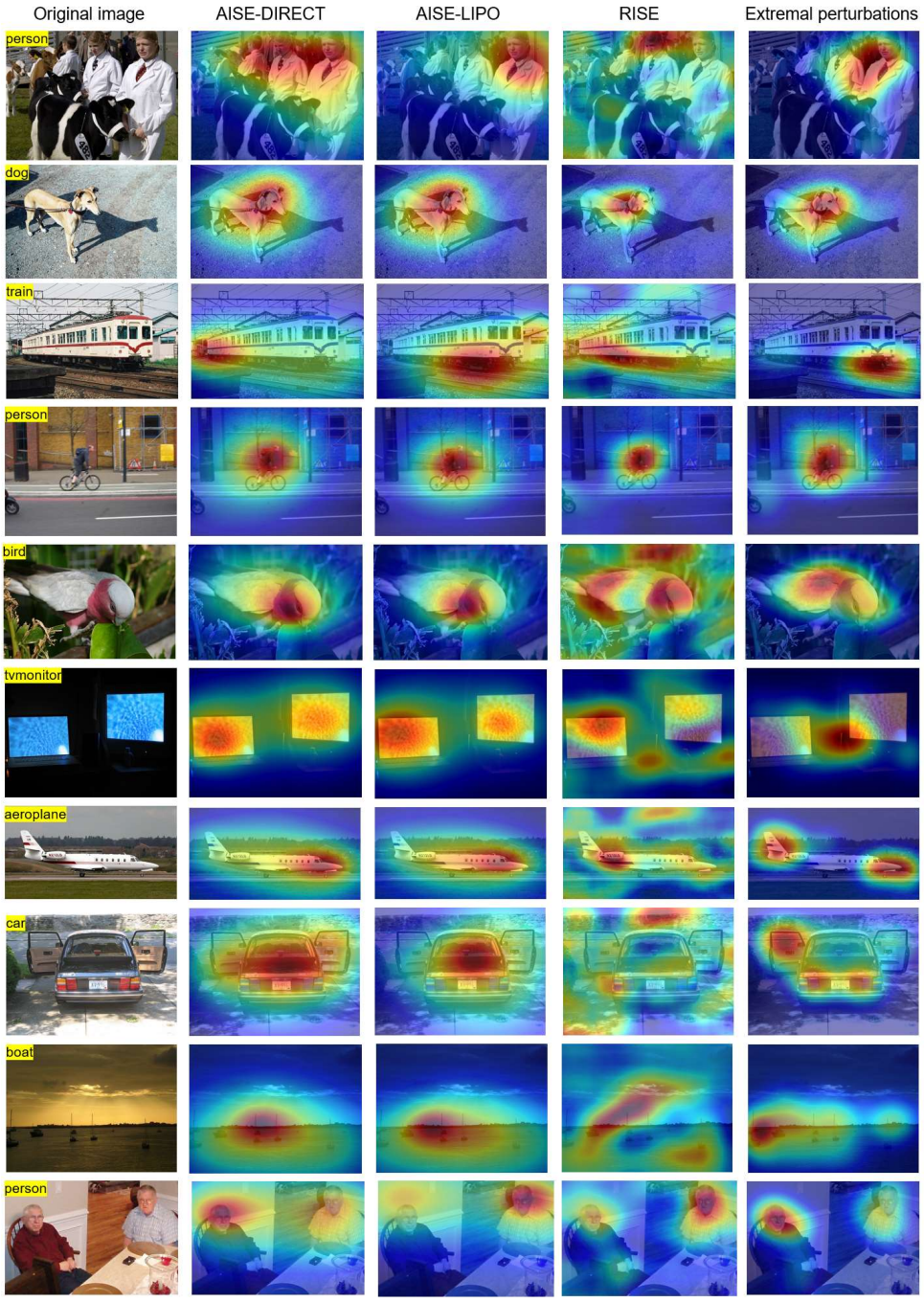

Figure 8: [Classification] Comparison with other methods, more examples.

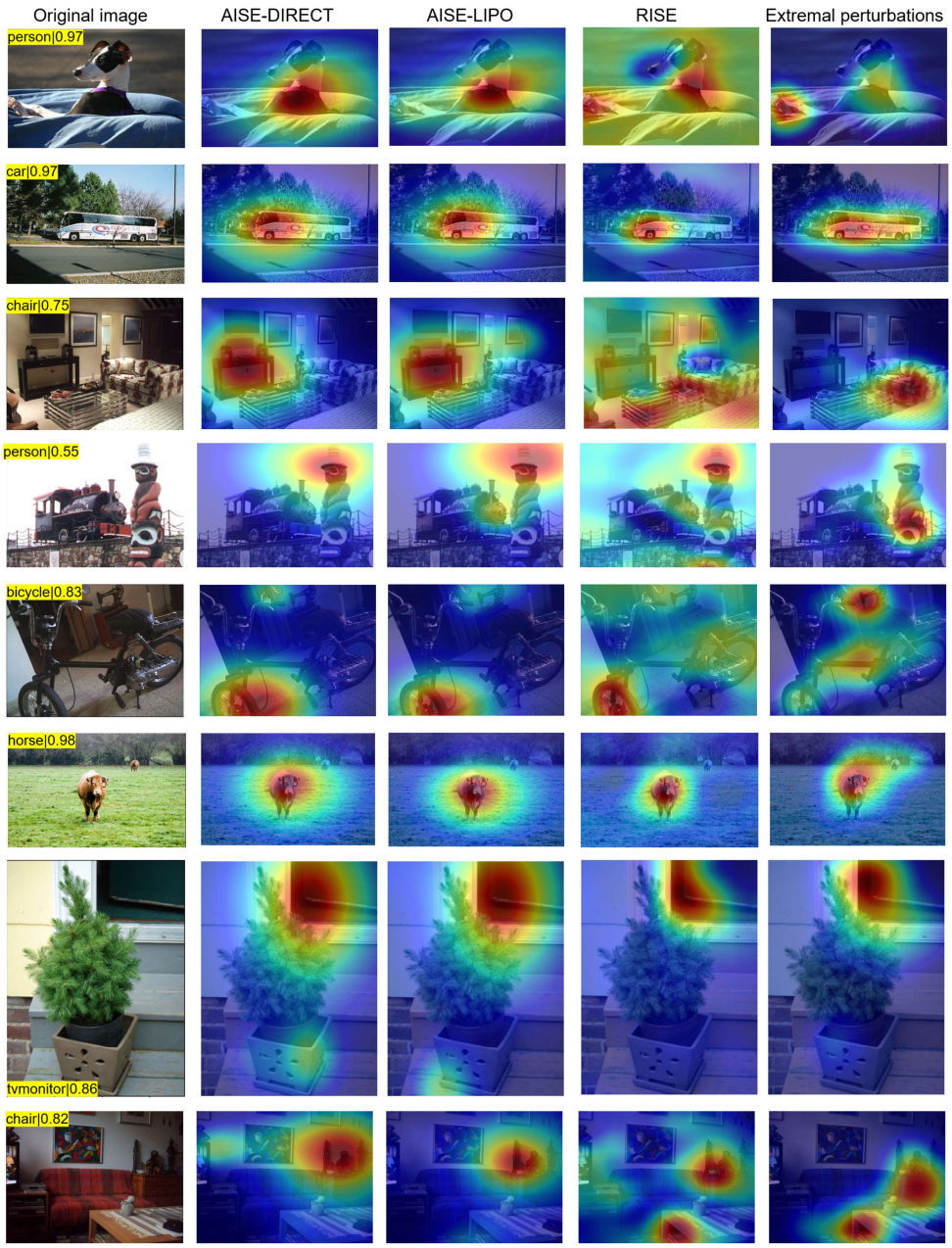

Figure 9: [Classification] Comparison with other methods for false positive examples.

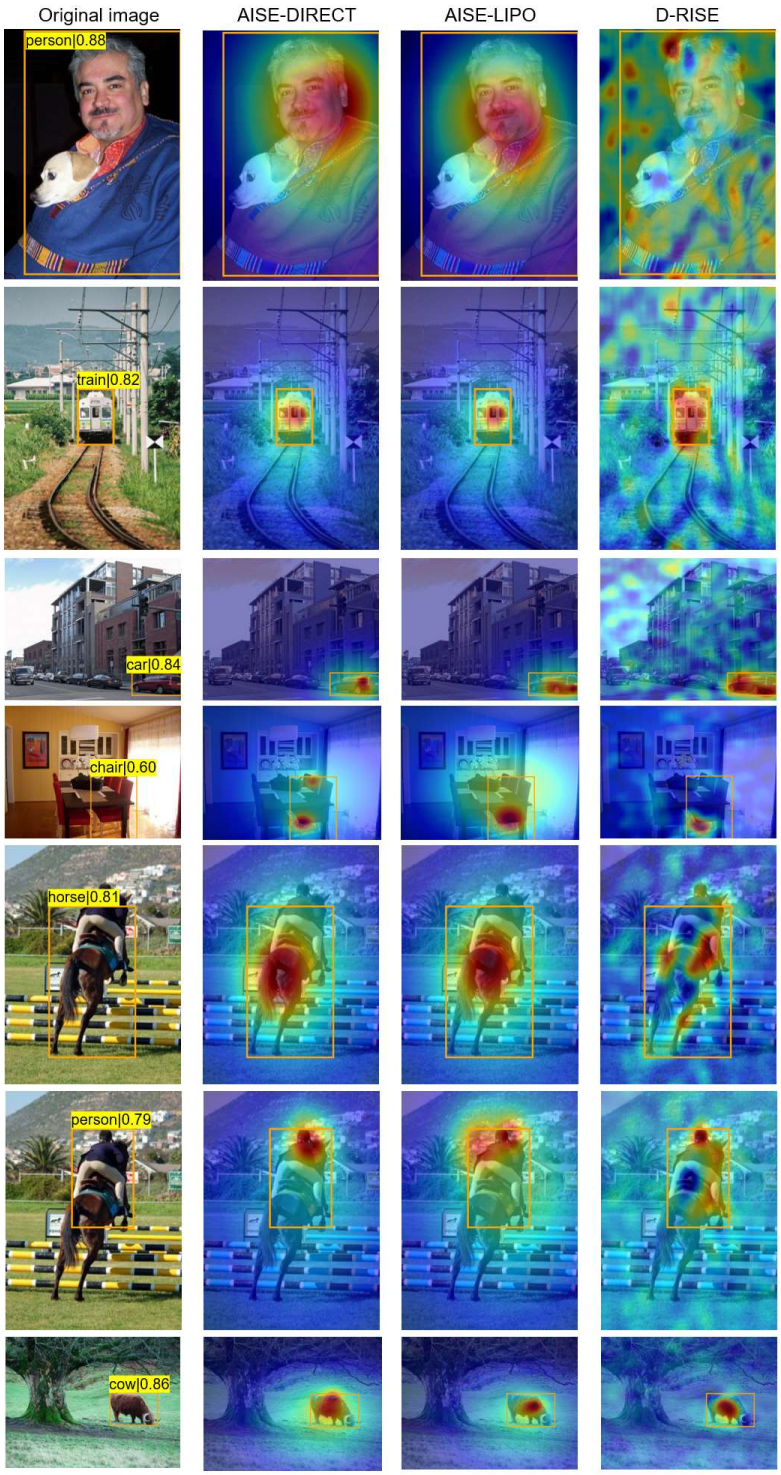

Figure 10: [Detection] Comparison with other methods.

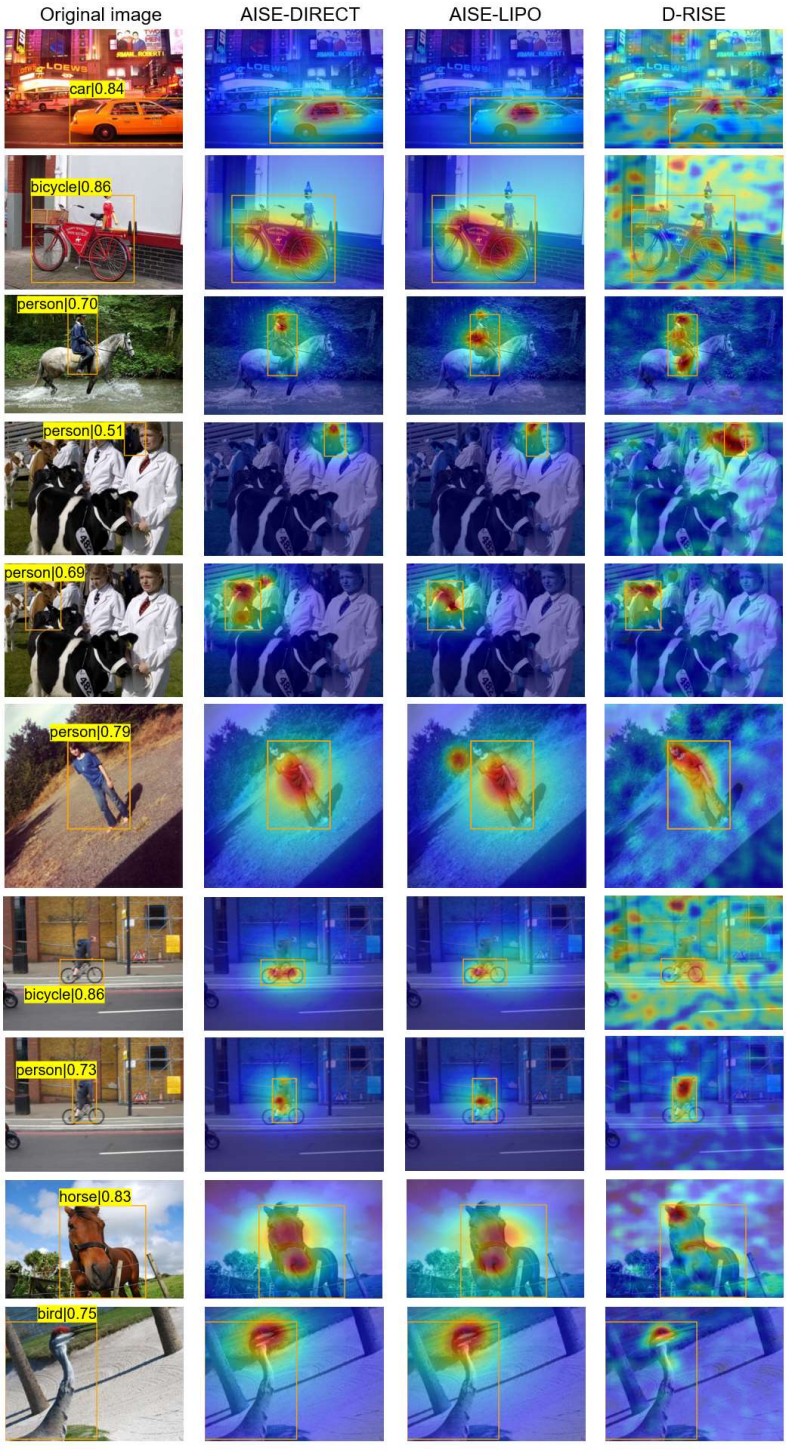

Figure 11: [Detection] Comparison with other methods, more examples.