SDDGR: Stable Diffusion-based Deep Generative Replay for Class Incremental Object Detection -Supplementary-

Junsu Kim[1] Hoseong Cho [1,2†] Jihyeon Kim[1,3†] Yihalem Yimolal Tiruneh[1] Seungryul Baek[1]

[1]UNIST [2]LG Electronics [3]KETI

In the supplementary material, we provide additional implementation details in Sec. 1, extended ablation study in Sec. 2, and additional experiments in Sec. 3. The overall procedure is outlined in Sec. 4. For a more comprehensive understanding of our method, refer to the additional visualizations in Sec. 5. Lastly, we talk about the effect of synthetic images containing multi-object to the incremental detector and future work in Sec. 6

Summary of the main paper. In the main paper, we introduced the stable diffusion (SD)-based replay method for class incremental object detection (CIOD). Despite the SD's advanced capabilities in image generation, its direct application to CIOD presents non-trivial challenges that we address through our contributions: (1) We first enhanced the SD's grounding capabilities following GLIGEN [2]. (2) SD often generates low-quality images and we propose an 'iterative class-wise refiner' that selects N quality samples for each class, by filtering out low quality images. (3) Generated images are not always well-aligned with the bounding boxes that were inputted as the grounding inputs. If non-aligned bounding boxes are used as ground-truths in the supervised setting, it can spoil overall CIOD accuracy. We bypass the issue by involving the 'L2 distillation loss': applying pre-trained network to obtain responses from generated images and enforcing the responses to a new model, without use of input bounding boxes. (4) We further optimized SD by adjusting text prompts and grounding signals, aiming for higher fidelity in generated images. This refinement process underscores our efforts to decrease dependency on SD's inherent capabilities.

1. Additional implementation details

Hyper-parameters. In our implementation, we utilized distributed data parallel in PyTorch [6] using 4 GPUs. The total batch size is 32 in the training process. During the generation process, each GPU utilizes 4 random noises (i.e. batch size of 4 on each GPU), allowing for the simultaneous generation of 16 images. For the stable diffusion sampler, we employ the PLMS [4] method. Additionally, we allocated both positive and negative prompts to control generated outputs. The *scene environments* of the custom positive prompt in Sec. 4.1 (main paper) used {*"demonstrating ultra high detail, 4K, 8K, ultra-realistic, crisp edges, smooth, hyper-detailed textures"*}. The negative prompt used {*"blurry, overlapping objects, distorted proportions, monochrome, grayscale, bad hands, deformed, lowres, error, normal quality, watermark, duplicate, worst quality, obscured faces, low visibility, unnatural colors, long body, bad anatomy, missing fingers, extra digit, fewer digits, cropped"*}. All other hyper-parameters related to the generation carefully followed the settings of [2].

This research was conducted when Hoseong Cho and Jihyeon Kim were graduate students (Master candidates) at UNIST†.

Table 1. Ablation study of generation-refinement on COCO 2017 (two-phase setting, 70+10). The experiment was conducted excluding L2 knowledge distillation to solely evaluate the effect of synthetic data. The best result is highlighted in **bold**. A red upward arrow ↑ indicates the performance improvement.

Setting	AP	$AP_{.5}$	$AP_{.75}$
$\mathcal{N}=\infty$	33.4	50.5	36.4
$\mathcal{N}=50$	**39.8** 6.4↑	**57.7** 7.2↑	**43.4** 7.0↑

Table 2. Approximate time computation based on the use of 4 A100 GPUs.

Class-wise counts	Generation time	Training time	Total time
$\mathcal{N}=\infty$	∼84 hours	∼42 hours	∼126 hours
$\mathcal{N}=50$	∼15 hours	∼21 hours	∼36 hours

2. Additional ablations

Class-wise $\mathcal{N}$ limitation. We further evaluate the impact of the hyper-parameter $\mathcal{N}$ that limits the number of synthesized images per each class. From Tab. 1, we can observe that when synthetic data is generated using the entire set of previous annotations $\mathcal{Y}_{m-1}$ ($\mathcal{N}=\infty$), the new model $\mathcal{M}_m$

Algorithm 1 Training with generated images and real data.

Input: Generated (synthetic) dataset $\mathcal{D}_{\text{gen}}$, Real new dataset $\mathcal{D}_m$ for new task $\mathcal{T}_m$, Old model $\mathcal{M}_{m-1}$, New model $\mathcal{M}_m$, Old tasks $\mathcal{T}_{m-1}$.
Define: m: task index.
Define: $\mathbf{y}^h$: Annotation for the h-th image, $\mathbf{x}^h$: h-th image.
Define: H: The total number of data in $\mathcal{D}_{\text{total}}$.

$\mathcal{D}_{\text{gen}} \leftarrow$ generation process of old tasks $\mathcal{T}_{m-1}$ **if** $m > 1$ // Create generated dataset. Refer to main paper Sec. 4.1 and 4.2.
$\mathcal{D}_{\text{total}} \leftarrow \mathcal{D}_{\text{gen}} \cup \mathcal{D}_m$ **if** $m > 1$ **else** $\mathcal{D}_m$ // Total dataset of $\mathcal{T}_m$.
for $h = 1, \ldots, H$ **do**
  Extract h-th annotation $\mathbf{y}^h_{\text{total}}$ and image $\mathbf{x}^h_{\text{total}}$ from $\mathcal{D}_{\text{total}}$.
  if $\mathbf{x}^h_{total} \in \mathcal{D}_{gen}$ **then**
    Apply L2 knowledge distillation from $\mathcal{M}_{m-1}$ to $\mathcal{M}_m$. // Training for synthetic dataset $\mathcal{D}_{\text{gen}}$. Refer to main paper Sec. 4.4.
  else
    Apply pseudo-labeling to $\mathbf{x}^h_{\text{total}}$ using $\mathcal{M}_{m-1}$. // pseudo-labeling to new task images ($\mathbf{x}^h_{\text{total}} \in \mathcal{D}_{\text{m}}$). Refer to main paper Sec. 4.3.
    Training $\mathcal{M}_m$ with $\mathbf{x}^h_{\text{total}}$. // Training with pseudo labeled annotations and now annotations.
  end
end

Table 3. Additional comparison of different methods in $40 + 40$ setting on D-DETR baseline. The best result is highlighted in **bold**.

Method	AP	$AP_{.5}$	$AP_{.75}$
LWF [3] applied on D-DETR	18.2	26.5	19.8
ICaRL [7] applied on D-DETR	29.9	42.7	32.7
Incremental-DETR [1]	37.3	56.6	-
CL-DETR [5]	42.0	60.1	45.9
SDDGR(Ours)	**43.0**	**62.1**	**47.1**

Table 4. Additional comparison of different methods in $70 + 10$ setting on D-DETR baseline. The "-" symbol indicates a missing value, as reported in paper [1]. The best result is highlighted in **bold**.

Method	AP	$AP_{.5}$	$AP_{.75}$
LWF [3] applied on D-DETR	4.3	6.2	4.8
ICaRL [7] applied on D-DETR	18.3	28.1	19.6
Incremental-DETR [1]	-	-	-
CL-DETR [5]	40.4	58.0	43.9
SDDGR(Ours)	**40.9**	**59.5**	**44.8**

tends to overfit to the synthetic data, and thereby exhibits a notable accuracy drop. Furthermore, as shown in Tab. 2, it takes approximately 126 hours for overall training process. In contrast, when $\mathcal{N} = 50$ is used, the time is significantly reduced to 36 hours, while achieving the best results. This implies the importance of choosing the proper number of synthetic samples during training.

3. Additional experiements

We provide an additional comparison of our SDDGR model with several prominent methods. These include LWF, ICARL, Incremental-DETR, and the previous state-of-the-art, CL-DETR, all of which are based on the Deformable-DETR framework. Tables 3 and 4 illustrate the superior performance of our SDDGR approach in both the $40 + 40$ and $70 + 10$ settings. These results highlight the effectiveness of the SDDGR method, which outperforms other methodologies in the same baseline.

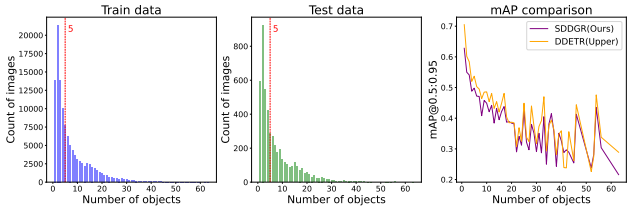

Figure 1. (Col. 1-2) dist. of object # in train/test sets, (Col. 3) mAP gap between SDDGR(Ours) vs. DDETR according to # of objects.

4. Overall procedure

The overall procedure of the SDDGR framework is summarized in Alg. 1 to describe the sequence of synthetic image generation and training.

5. Additional visualization

We experimented with the effect of the different inputs on the final image quality in Fig. 2. We also visualized more examples for generated images according to their classes in Fig. 3. Additionally, Fig. 4 presents the detection results using three of our baselines and Fig. 5 presents the more detection results of ours.

6. Discussion and future work

This section explores the challenges of generating synthetic images, especially those depicting scenes with multi-objects. It is commonly believed that the performance of generative models, such as Stable Diffusion (SD), declines as the complexity of the scene increases. To directly analyze the challenges' impact on SDDGR, we have examined

the distribution of images by the number of objects alongside the model's performance, and we present this analysis in Fig. 1.

Essentially, there is a sharp decrease in the frequency of images containing more than five objects (denoted by the red line) on COCO dataset distribution, as depicted on the leftmost and middle of Fig. 1. Furthermore, the rightmost of Fig. 1, contrasts the mean Average Precision (mAP) of our SDDGR model in a 70+10 setting with that of the upper-bounded Deformable-DETR (D-DETR), which trained with integrated both new and older classes. From the graph, we found that the accuracy disparity between SDDGR and the upper-bounded D-DETR is minimal across different object counts. This observation suggests that the inherent challenges of complex scene generation do not significantly affect SDDGR's performance. However, we think explicitly tackling the aspect could be a future research direction.

References

[1] Na Dong, Yongqiang Zhang, Mingli Ding, and Gim Hee Lee. Incremental-detr: Incremental few-shot object detection via self-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 543–551, 2023. 2

[2] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *CVPR*, 2023. 1

[3] Zhizhong Li and Derek Hoiem. Learning without forgetting. *ECCV*, 2016. 2

[4] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *ICLR*, 2022. 1

[5] Yaoyao Liu, Bernt Schiele, Andrea Vedaldi, and Christian Rupprecht. Continual detection transformer for incremental object detection. In *CVPR*, 2023. 2

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NIPS*, 2019. 1

[7] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 2

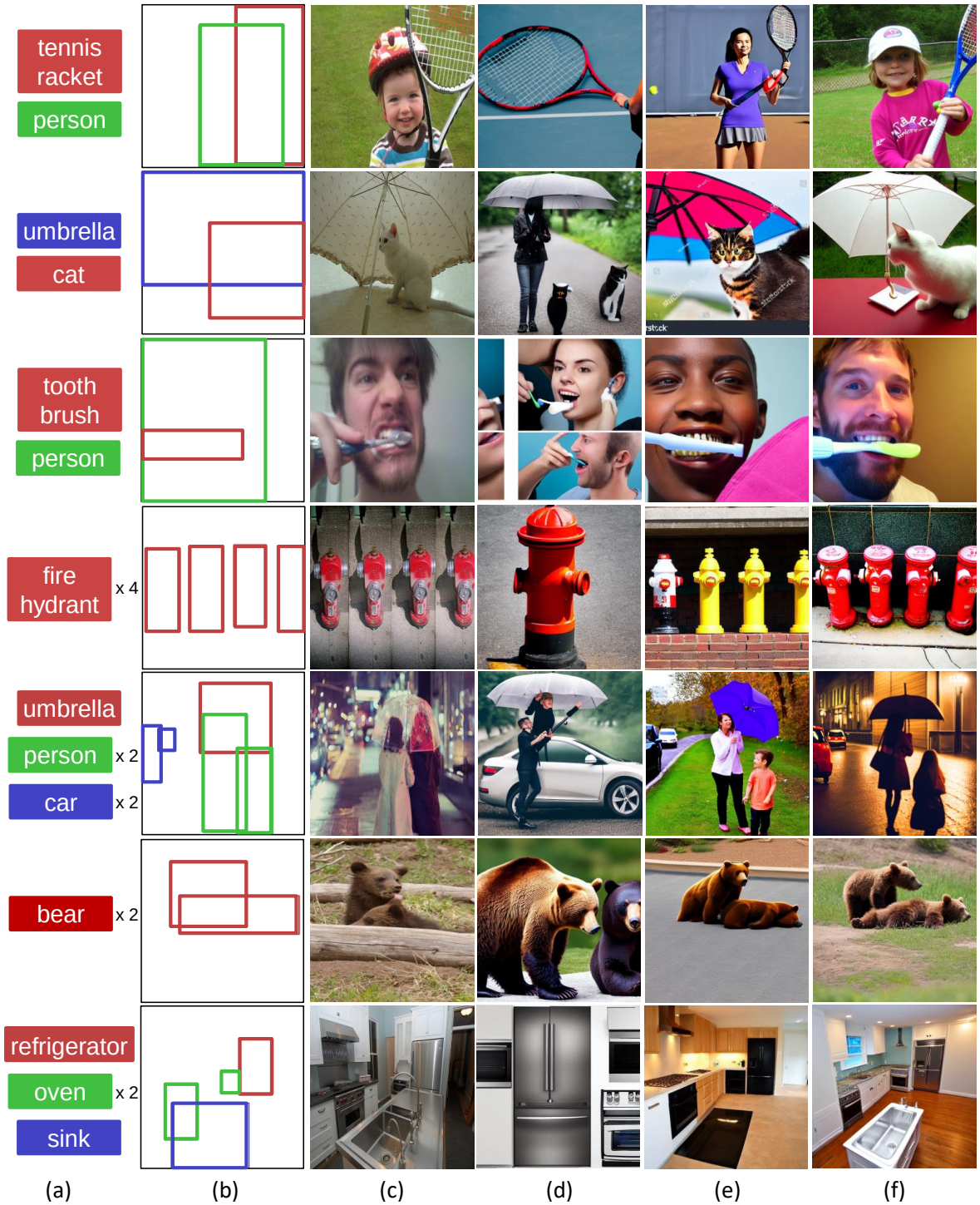

Figure 2. Differences in image generation based on input types. Each row in this figure represents different examples used for image synthesis, with varying prompts. The first row uses prompts 'A photo of tennis racket and person, realistic, ... details.' The second row uses prompts 'A photo of umbrella and cat, ...'. The third row uses prompts 'A photo of toothbrush and person, ...'. The fourth row uses prompts 'A photo of four fire hydrants, ...'. The fifth row uses prompts 'A photo of umbrella, two persons, and two cars, ...'. The sixth row uses prompts 'A photo of two bears, ...'. Finally, the last row uses prompts 'A photo of a refrigerator, two ovens, and a sink, ...'. For each set of images: (a) and (b) show the grounding inputs (object classes and bounding boxes), (c) displays COCO real images, (d) depicts images generated by inputting only the text prompt, (e) shows images generated by combining grounding inputs and the text prompt, and (f) illustrates images generated by incorporating the combination of text prompt, grounding inputs, and CLIP's image embedding.

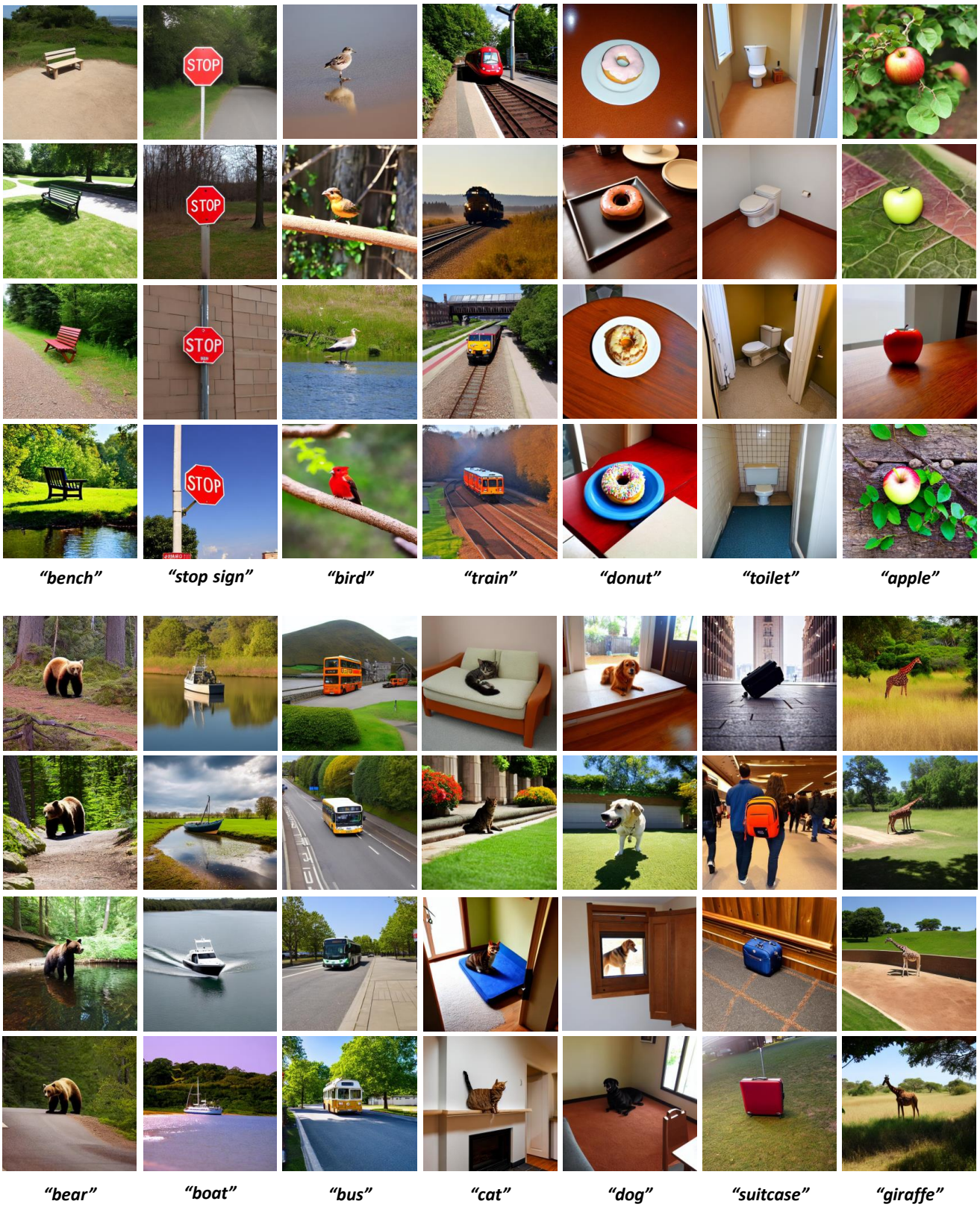

Figure 3. Example generated images for the class-specific generation process.

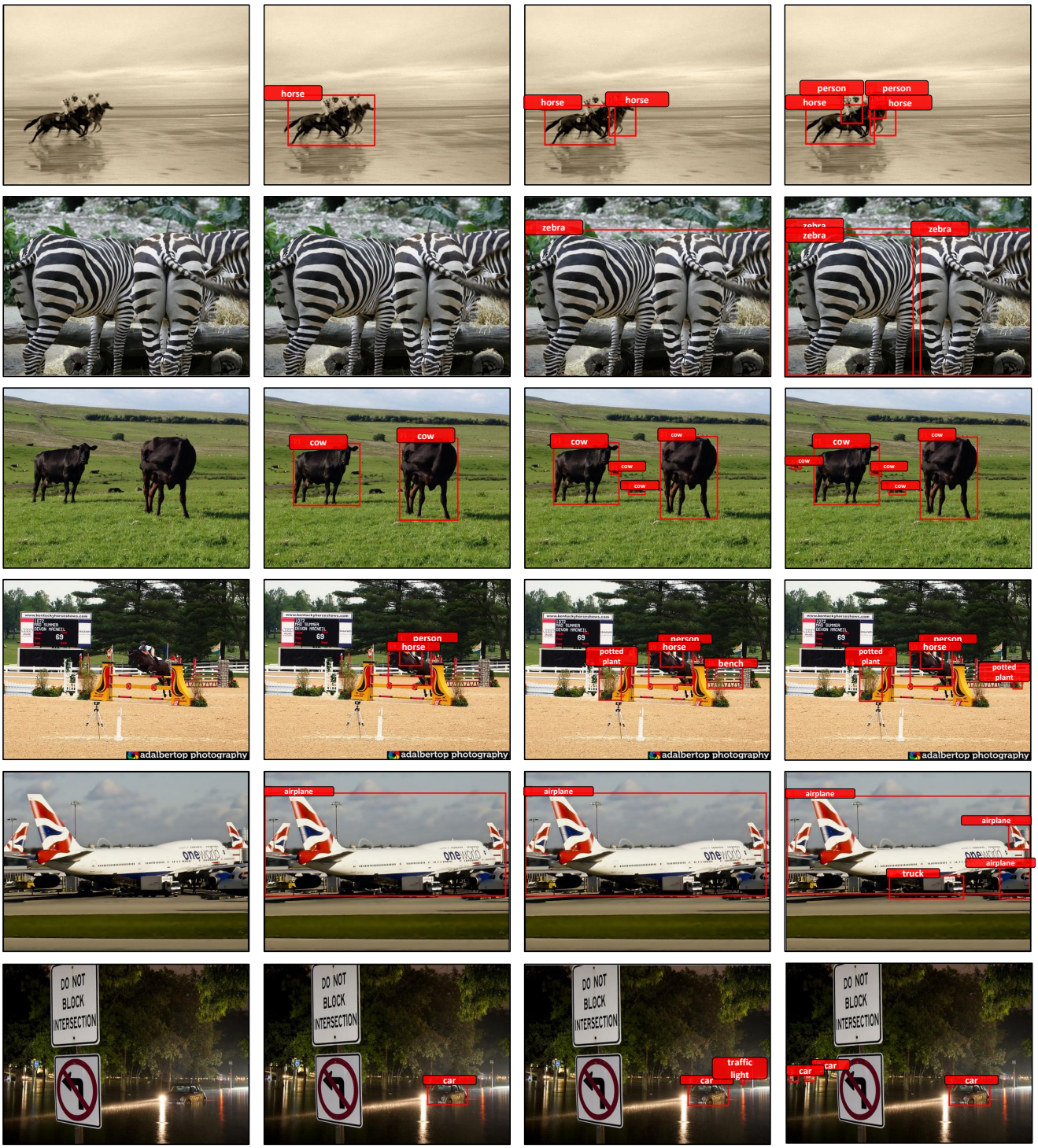

Figure 4. Our detection results on COCO 2017 (two-phase setting, 70+10).

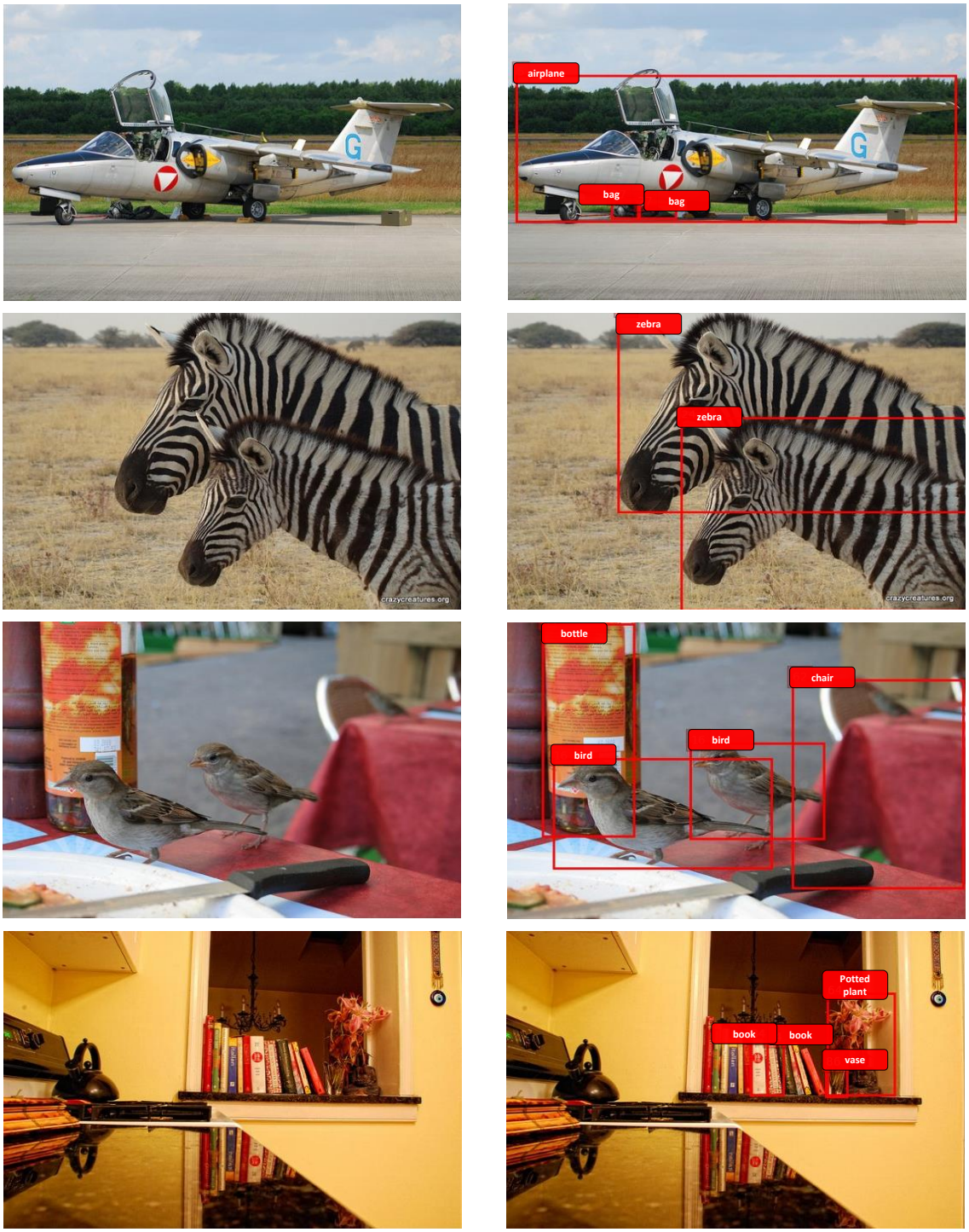

Figure 5. Our detection results on COCO 2017 (two-phase setting, 70+10).