Supplementary Material: Continual Object Detection via Prototypical Task Correlation Guided Gating Mechanism

Binbin Yang[1*] Xinchi Deng[1*] Han Shi[2] Changlin Li[3] Gengwei Zhang[1]
Hang Xu[4] Shen Zhao[1] Liang Lin[1†] Xiaodan Liang[1]
[1]Sun Yat-sen University [2]The Hong Kong University of Science and Technology
[3]ReLER, AAII, UTS [4]Huawei Noah's Ark Lab
{yangbb3, dengxch5}@mail2.sysu.edu.cn, hshiac@cse.ust.hk, zhaosh35@mail.sysu.edu.cn,
{changlinli.ai, zgwdavid, chromexbjxh, xdliang328}@gmail.com, linliang@ieee.org

A. Further Implementation Details

Our experiments are conducted on 8 GPUs with batch size of 16 and implemented by PyTorch [5]. Following the task-based continual object detection setting in [4], we use COCO 2014 valminusminival and COCO 2014 minival datasets as the training and test set of COCO. For Pascal VOC, we train on VOC 2007 and VOC 2012 trainval set and test on VOC 2007 test set. For the class-based continual object detection setting, we use split Pascal VOC 2007 and group the classes following [2]. Without loss of generality, our proposed ROSETTA can be equipped with different object detector backbones. In our experiment, we choose two representative object detectors as our backbones: Faster R-CNN [6] and Sparse R-CNN [7]. Both these two kinds of detectors use ResNet50 [1] pre-trained on ImageNet [3].

For Faster R-CNN backbone, we use the same training scheme as [4] and generate our baseline results of joint training and fine-tuning. Following [4], the learning rate is set to 0.01 for the first 10 epochs and 0.001 for the last 2. As for Sparse R-CNN, we use the default hyper-parameters with 100 proposal boxes. Specifically, we use $3\times$ training schedule (36 epochs) on COCO and VOC and the learning rate is set to 2.5×10^{-5} for early training stages, divided by 10 at epoch 27 and 33, respectively. On account of the small sizes of KITTI and Kitchen, we train the Sparse R-CNN for 9k iterations instead. Our gated module is applied to the last three stages of the ResNet50 [1] backbone for both Faster R-CNN and Sparse R-CNN while the first two stages are fixed during training. The class embeddings C^t in the gated module are implemented by the word embeddings of class labels of task t. Here we generate the word embeddings using the "en_core_web_lg" model of spacy module in https://github.com/explosion/spacy-models/releases. We first pre-train a non-sparse detector without gated module and use it to guide the training of the gated detector with distillation loss. As mentioned in the main paper, the gated model is further fine-tuned for 2 epochs for better co-adaptation between the binary gates and the channel weights. The Pascal VOC mean average precision (mAP) is used as our evaluation metric following [4].

B. Overall optimization objective and sensitivity analysis of hyper-parameters

Our overall optimization objective is $\mathcal{L} = \mathcal{L}_{det} + \lambda_s\mathcal{L}_{sparsity} + \lambda_{kd}\mathcal{L}_{kd} + \lambda_{div}\mathcal{L}_{diversity}$. $\mathcal{L}_{det}$ indicates the standard detection loss of Faster/Sparse R-CNN. We provide sensitivity analysis on Kitchen(task2) of KITTI→Kitchen for λ_s, λ_{kd}, λ_{div} in Tab. 1.

λ_s	1	**2**	4	8
Kitchen(task2)	76.3	**78.3**	77.4	74.8
λ_{kd}	10	100	**500**	1000
Kitchen(task2)	75.2	76.1	**78.3**	76.3
λ_{div}	1	2	**4**	8
Kitchen(task2)	75.0	76.3	**78.3**	76.5

Table 1. Sensitivity analysis of λ_s, λ_{kd} and λ_{div}.

As for the threshold hyper-parameter η in Eq.(8) of the main paper, we also provide its sensitivity analysis in Tab. 2.

*Equal contribution. †Corresponding author.

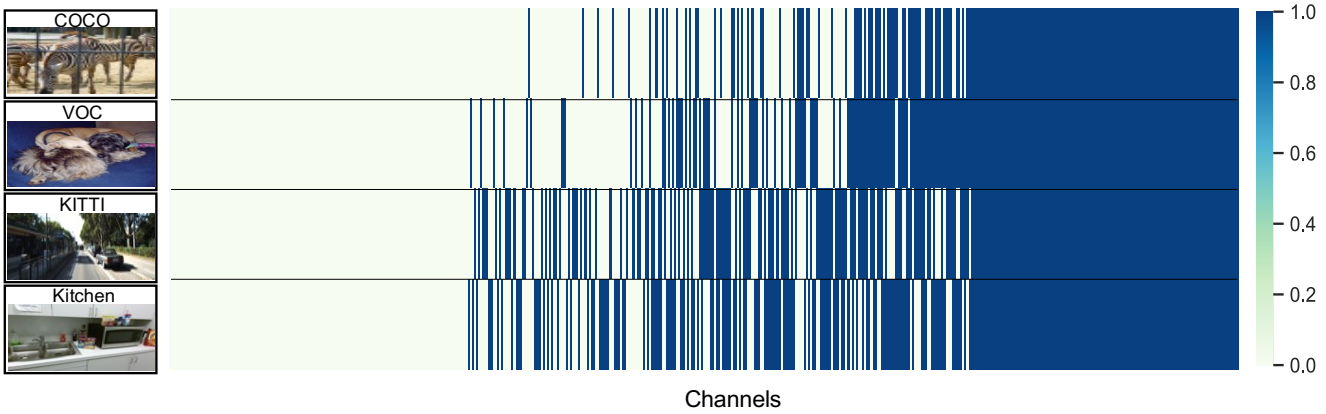


Figure 1. Visualization of the learned binary gates for the sequential tasks: COCO→VOC→KITTI→Kitchen. Yellow and blue channels indicate the inactivated and activated channels for specific tasks. For better illustration, gates are sorted by the overall execution rate over all tasks.

η	0.1	0.3	**0.5**	0.7	0.9
Kitchen(task2)	75.7	75.9	**78.3**	77.8	77.5

Table 2. Sensitivity analysis of η.

C. Evaluation Protocol for Class-incremental Setting

During the inference stage of class-incremental detection, our proposed ROSETTA *does not need a task identifier* to know which task an input image comes from. Here we simply take a "15+5" setting as an example. In the test stage for evaluating total 20 classes, given an arbitrary image, we can equip two sub-models with their corresponding class embeddings for the first 15 and last 5 classes, which are responsible for detecting whether those classes of objects appear in this image, respectively. In this way, making *two inferences of lightweight sub-models* with their corresponding class embeddings can detect any objects of total 20 classes, without exactly knowing which task this image belongs to. Actually, our proposed *model-based routing approach* and the *property of object detection* make this evaluation protocol reasonable.

D. Memory Budget

In addition to the storage of the detector backbone, *e.g.*, a Faster R-CNN or Sparse R-CNN, ROSETTA demands extra storage requirements in the memory bank, *e.g.*, historical gate lists, prototypes and class embeddings of different tasks. Taking the detection task on Pascal VOC as an example, there are totally about 154KB memory to be used: gate list = 110KB, prototypes = 20KB (1KB per class), class embeddings = 24KB. By comparison, the continual object detection method in [4] needs to store 100 exemplars (images) and at least 10MB memory is required (1 exemplar = 100KB), without considering other budgets. Hence, ROSETTA only requires limited memory budget compared to other methods.

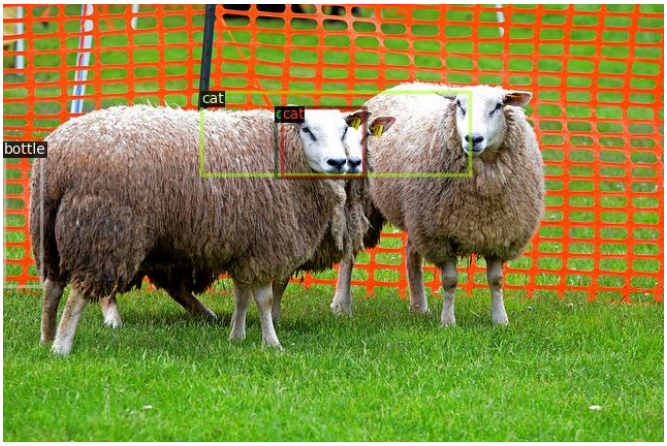
Remark: No extra memory budget is required for a historical teacher model of our knowledge distillation strategy. This is because our non-gated teacher model is only used for guiding the gate learning of the current task and will be discarded after training the gated student model.

E. Visualization of Binary Gates

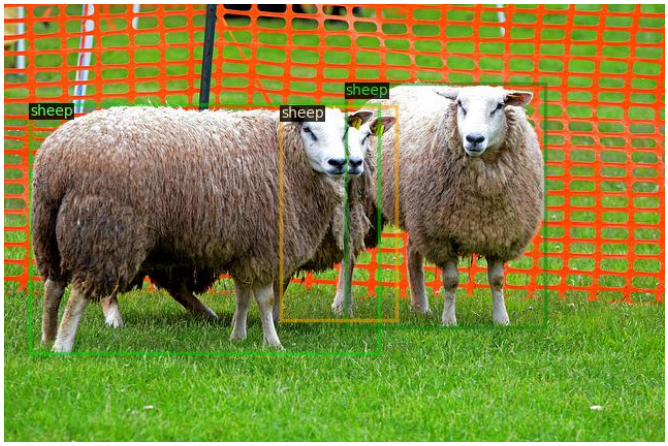
As shown in Fig. 1, we visualize the learned static binary gates of the backbone's last layer for the sequential tasks: COCO→VOC→KITTI→Kitchen. We can conclude that ROSETTA can capture the task correlations and automatically activate more exclusive channels if it observes a significant domain gap.

F. Examples of Continual Detection Results

Here we visualize the detection results of the sequential tasks: COCO→VOC→KITTI→Kitchen in Fig. 2. The comparison between a baseline model, *i.e.*, fine-tuning and ROSETTA is provided. The detection results in the left column of Fig. 2 illustrate that the baseline strategy, *i.e.*, fine-tuning is confronted with the problem of catastrophic forgetting, which leads to the mistakes of localization (Fig. 2a, Fig. 2c), classification (Fig. 2a) and the problem of low recall (Fig. 2e) when inferred on previously seen tasks. For example, no sheep is successfully detected in Fig. 2a; the bounding boxes of "bicycle" and "person" are not accurate in Fig. 2c; 3 cars are missed in Fig. 2e. By comparison, the right column of Fig. 2 shows that ROSETTA can well tackle the issue of catastrophic forgetting and memorize the previously learned knowledge.



(a) Fine-tuning (`task1`: COCO)

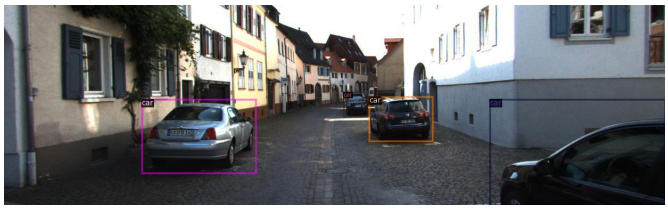


(b) ROSETTA (`task1`: COCO)

(c) Fine-tuning (`task2`: VOC)

(d) ROSETTA (`task2`: VOC)

(e) Fine-tuning (`task3`: KITTI)



(f) ROSETTA (`task3`: KITTI)

(g) Fine-tuning (`task4`: Kitchen)

(h) ROSETTA (`task4`: Kitchen)

Figure 2. Comparison between fine-tuning and ROSETTA on COCO→VOC→KITTI→Kitchen.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[2] KJ Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Towards open world object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5840, 2021. 1

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[4] X. Liu, H. Yang, A. Ravichandran, R. Bhotika, and S. Soatto. Multi-task incremental learning for object detection. *arXiv:2002.05347*, 2020. 1, 2

[5] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*, 2017. 1

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[7] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14454–14463, 2021. 1