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

Abstract

*Continual learning is a challenging real-world problem for constructing a mature AI system when data are provided in a streaming fashion. Despite recent progress in continual classification, the researches of continual object detection are impeded by the diverse sizes and numbers of objects in each image. Different from previous works that tune the whole network for all tasks, in this work, we present a simple and flexible framework for continual object detection via p**R**ot**O**typical ta**S**k corr**E**la**T**ion guided ga**T**ing mech**A**nism (ROSETTA). Concretely, a unified framework is shared by all tasks while task-aware gates are introduced to automatically select sub-models for specific tasks. In this way, various knowledge can be successively memorized by storing their corresponding sub-model weights in this system. To make ROSETTA automatically determine which experience is available and useful, a prototypical task correlation guided Gating Diversity Controller (GDC) is introduced to adaptively adjust the diversity of gates for the new task based on class-specific prototypes. GDC module computes class-to-class correlation matrix to depict the cross-task correlation, and hereby activates more exclusive gates for the new task if a significant domain gap is observed. Comprehensive experiments on COCO-VOC, KITTI-Kitchen, class-incremental detection on VOC and sequential learning of four tasks show that ROSETTA yields state-of-the-art performance on both task-based and class-based continual object detection.* [1]

1. Introduction

Thanks to the development of computer vision and deep learning, a great progress have been made in object detection [31, 32, 41]. The mainstream of existing works typically follows the offline training paradigm: an individual model is trained on a dataset and then evaluated on the test set with similar distribution. Nevertheless, online training on streaming data plays a more important role in real-world applications, especially large-scale industrial systems. More significantly, an artificial intelligent system is expected to continually learn different skills, *e.g.*, detect more and more objects in multiple scenarios, resembling the memorizing and learning abilities of humans instead of

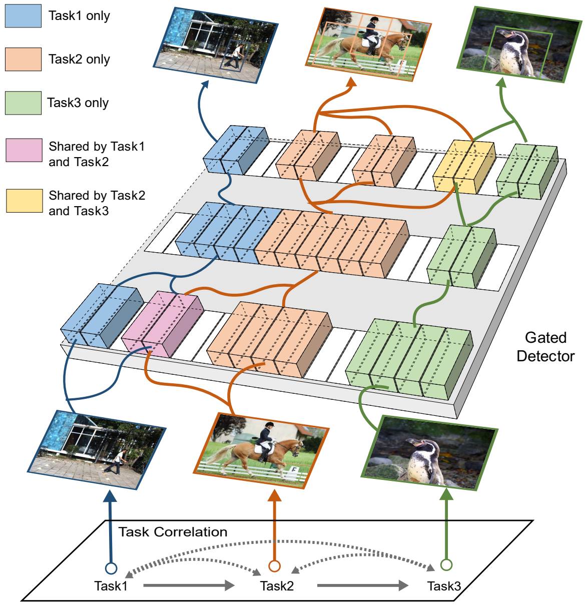

Figure 1. Overview of our p**R**ot**O**typical ta**S**k corr**E**la**T**ion guided ga**T**ing mech**A**nism (ROSETTA) for continual object detection. Sequential detection tasks share a unified backbone of the gated detector. Knowledge from previous tasks can be stored in the weights of the corresponding sub-models which are activated by the stored gates. Boxes in colors indicate the channels activated by task-aware gates for different tasks. Best viewed in color.

*Equal contribution. †Corresponding author.

[1] Codes are available at: https://github.com/dkxocl/ROSSETA.

learning from scratch every time. A common solution from the machine learning perspective is continual learning (lifelong learning) [23, 24, 34, 44], aiming to sequentially solve non-stationary tasks with ideally no performance drop when inferred on the previously seen tasks. Typically, an elastic model is required to achieve the equilibrium between continuously acquiring new knowledge and preserving existing knowledge.

Despite the increasing attention on continual image classification [1,24,44], few studies have been devoted to building a continual object detection [23, 34] framework due to the challenges of preserving the capability of localizing and recognizing multiple objects of diverse scales in streaming tasks. What is noteworthy is that, an image of the new task might simultaneously contain objects of novel classes and previously seen classes. Existing works on continual object detection mostly use knowledge distillation [34, 39, 45, 53] on features of the region proposal network (RPN) to mitigate catastrophic forgetting. Specifically, suppose an object detector is trained consecutively on two tasks, *i.e.,* `task1` and `task2`. When training on `task2`, the distillation-based methods will force the features of the RPN to be consistent with those produced by the saved model trained on `task1`. However, such methods suffer from domain shift: distillation over the `task2` data captures biased rather than the actual knowledge of `task1` due to the unavailability of samples from `task1`. Another line of researches [23, 34] on continual detection store a small number of samples(exemplars) for reviewing previous knowledge. Nevertheless, replaying exemplars also fails in capturing actual knowledge because not all samples are accessible and the sampling strategy plays a decisive role.

In this work, we explore a different way to solve continual object detection without any exemplar replay: *directly storing knowledge* via a sparse and dynamic framework. As illustrated in Fig. 1, a unified detector is shared by sequential tasks and the task-aware gates are designed to automatically determine which sub-models (channel-level) should be activated for specific tasks. To avoid the difficulty of jointly optimizing binary gates and channel weights, we propose a soften-and-discretize strategy for the gate learning. Specifically, dynamic soft gates are generated during training stage and then discretized to be static binary gates for inference stage. The channels' weights that have been activated for previous tasks by the binary gates are frozen and stored to keep existing knowledge. Each sub-model can dynamically choose whether to use the frozen channels to boost the learning for the current task. In this manner, the previous knowledge can be unbiasedly stored in the sub-models' weights and cross-task knowledge can be shared by their overlapped channels. Binary gates are used to retrieve such existing knowledge by activating sub-models' channel weights.

Although the proposed gating mechanism well preserves the previous knowledge, we observe degradation on the subsequent tasks when the domain gaps are significant, *e.g.*, KITTI→Kitchen, which have different foreground objects and backgrounds. We attribute this phenomenon to the unawareness of cross-task correlation. Confronted with a relatively large gap, sharing too much previous knowledge would limit the performance gain in subsequent tasks [26] and more exclusive channels should be activated for new tasks. Thus, we propose the Prototypical Task Correlation Guided Gating Mechanism to achieve a balance between sharing existing knowledge and exploiting exclusive knowledge (*i.e.,* identifying which knowledge in the current task is orthogonal to the existing one). Specifically, a task correlation guided Gating Diversity Controller (GDC) is proposed to adaptively adjust the diversity of gates for the new task based on class-specific prototypes. GDC computes a cross-task class-to-class prototypical correlation matrix to depict the inter-task affinity and hereby activates more gates for the `task2` when the domain gap between `task1` and `task2` is significant, and vice versa.

To verified its effectiveness, our proposed ROSETTA is evaluated on both task-based [34] and class-based [23] continual object detection scenarios. For task-based settings, our method equipped with Faster R-CNN backbone outperforms the state-of-the-art benchmarks by 11.8 mAP and 3.4 mAP on COCO and VOC for COCO→VOC, 5.8 mAP and 5.7 mAP on KITTI and Kitchen for KITTI→Kitchen. As for the class-incremental detection, our method surpasses the state-of-the-art method by 2.2 mAP for the "10+10" setting on VOC. Furthermore, comprehensive experiments demonstrate that our proposed gating mechanism successfully achieve an equilibrium between sharing knowledge and exploiting exclusive knowledge for multiple tasks by capturing their prototypical cross-task correlation.

2. Related Work

Continual Object Detection Continual learning [2, 6, 17, 24, 28] studies the problem of how an intelligent system sequentially learns from a stream of tasks. The ultimate goal of continual learning is to gradually digest new knowledge and preserve the acquired knowledge. Most existing studies on continual learning in visual system focus on continual image classification [3, 7, 12, 24, 35, 42] which aims to alleviate catastrophic forgetting for continually recognizing single object in an image. In comparison, object detection [31, 32, 41] is a high-level computer vision task involving object localization and object recognition, which has not been explored totally in the continual learning scenario. Due to the diverse sizes and numbers of objects in each image, continual object detection is a more challenging task than continual image classification. Existing works on continual object detection can be divided into two cat-

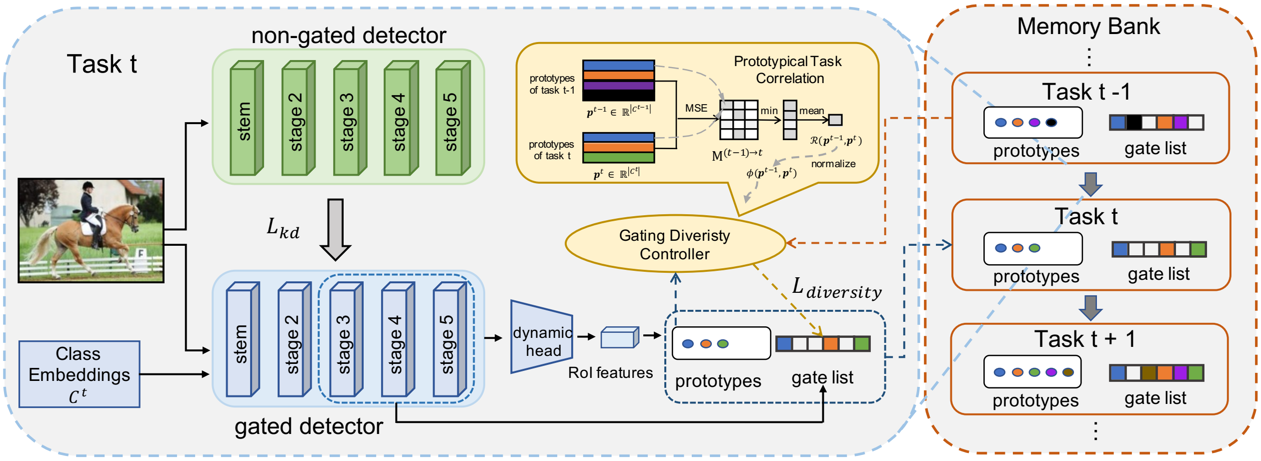

Figure 2. Pipeline of ROSETTA. (We take the Sparse R-CNN backbone as an example in this figure.) A non-gated detector is used to guide the learning of the gated detector via feature-level knowledge distillation. Class-specific prototypes are generated as the mean ROI (Region-of-Interest) feature of each class. A memory bank is used to store the historical gate lists and their corresponding prototypes. The gating diversity controller (GDC) captures the prototypical cross-task correlation based on the stored gate lists and prototypes, and then automatically adjusts the diversity of gates for the current task. Best viewed in color

egories: storing parts of samples as exemplars for experience replay [23] and knowledge distillation [34, 45]. [23] stores a balanced set of exemplars and fine-tunes the model after each incremental step. [45] leverages knowledge distillation for both object localization and object classification. [34] further exploits attentive feature distillation to distill important knowledge via both top-down and bottom-up attentions. In this work, we explore a different way for continual object detection without using exemplars via sparse and dynamic gating mechanism.

Gating Mechanism Dynamic inference methods [5, 22, 27, 47, 48] change network architecture based on the input data. As a general approach to achieve dynamic inference, gating mechanism takes in an intermediate feature map and outputs a binary vector as the decision of the candidate paths. It has been generally used to choose over different channels in dynamic pruning methods [4,8,10,13,19,21,29] and dynamic slimmable network [25]. Such strategy has also been used in other conditional inference methods including dynamic depth [47–49] and dynamic routing [50]. Gating mechanism is also applied in continual image classification for dynamic inference based on the inputs from different tasks [1, 36, 40, 44]. Differing from these works that rely on gradient estimation or other techniques for optimization of the binary gate, we use dynamic soft gates during training time and then discretize to static binary ones for inference.

3. Methodology

3.1. Continual Object Detection and its Challenges

Continual Object Detection The goal of continual object detection is to obtain an object detector performing well on $\mathcal{T}$ sequential Tasks. For the t^{th} task, the dataset $D^t = \{X^t, Y^t\}$ is provided to train the object detector, where X^t and Y^t denote the input images and the corresponding annotations, respectively. The categories to be recognized at time t are denoted as a set C^t.

Problem of Catastrophic Forgetting Catastrophic forgetting is a phenomenon that models suffer from rapid performance degradation on previously learned tasks [17]. This usually occurs when a model is continuously transferred among multiple datasets when the old data in history can not be accessed for review. Lots of works have explored to mitigate catastrophic forgetting for the problem of continual image classification, while a more delicate strategy is required for continual object detection due to the variety of numbers, sizes and classes of the objects of interests in different datasets [34].

In this section, we propose a new method for continual object detection to better tackle the problem of catastrophic forgetting via p**R**ot**O**typical ta**S**k corr**E**la**T**ion guided ga**T**ing mech**A**nism (ROSETTA). As illustrated in Fig. 1 and Fig. 2, ROSETTA can memorize multiple sequentially learned knowledge by storing the weights of one unified object detector, without any exemplar replay. For convenience of computing cross-task correlation, a memory bank is used to store each task's gate list and class-specific prototypes. Thanks to the gating strategy, knowledge of the previously seen tasks can be queried by their corresponding gates and retrieved by activating sub-models' channel weights. Moreover, a task correlation guided Gating Diversity Controller (GDC) is introduced to capture the cross-task correlation according to the stored gate lists and prototypes in memory banks and then make ROSETTA dynamically determine the gating diversity. In the following parts, elements of our proposed ROSETTA will be elaborated.

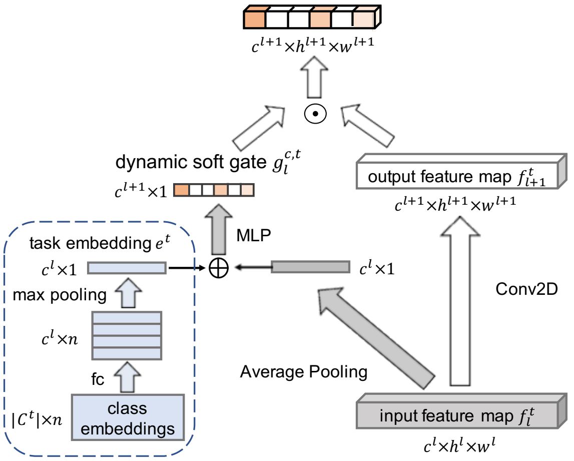
c^{l+1}×h^{l+1}×w^{l+1}
dynamic soft gate $g_l^{c,t}$
c^{l+1}×1
output feature map f_{l+1}^t
c^{l+1}×h^{l+1}×w^{l+1}
MLP
task embedding e^t
c^l×1
max pooling
c^l×n
fc
$|C^t|$×n
class embeddings
c^l×1
Conv2D
Average Pooling
input feature map f_l^t
c^l×h^l×w^l

Figure 3. Our proposed gated convolutional module (training stage). Class embeddings (word embeddings of the class labels) of the current task is fed into a fully connected layer and a max pooling layer to obtain task embedding $\boldsymbol{e}^t$. An MLP takes its inputs as task embedding $\boldsymbol{e}^t$ and input feature map $\boldsymbol{f}_l^t$ to generate the dynamic soft gate $g_l^{c,t}$.

3.2. Task-aware Gated Object Detector

As illustrated in Fig. 3, our proposed gating mechanism is applied to the convolutional layers of an object detector, which activates the sub-model of a specific task. Given an input feature map $\boldsymbol{f}_l^t \in \mathbb{R}^{c^l \times h^l \times w^l}$ of the l^{th} layer for task t, a convolutional operation F_l^t is performed and a gating module G_l generates the activated channels:

$$\boldsymbol{f}_{l+1}^t = G_l^t(\boldsymbol{f}_l^t, C^t) \odot (F_l^t(\boldsymbol{f}_l^t)), \tag{1}$$

where $\odot$ denotes the operation of channel-wise multiplication, $\boldsymbol{f}_{l+1}^t \in \mathbb{R}^{c^{l+1} \times h^{l+1} \times w^{l+1}}$ is the output feature map of gated convolution, $F_l^t(\boldsymbol{f}_l^t) \in \mathbb{R}^{c^{l+1} \times h^{l+1} \times w^{l+1}}$ is the output of convolution, $G_l^t(\boldsymbol{f}_l^t, C^t) = [g_l^{1,t}, g_l^{2,t}, ..., g_l^{c^{l+1},t}]$ represents channel gates of the l^{th} layer, $g_l^{c,t} \in [0, 1]$. The learnable task embedding $\boldsymbol{e^t}$ is generated based on class embeddings C^t, which is the concatenation of word embeddings [37] of the classes of current task. By introducing class embeddings C^t, the gated module can know exactly which categories of objects need to locate and recognize. Such meta-information provides a global contextual guidance for the gated module and makes it generate different gates when two tasks have similar images (input feature maps $\boldsymbol{f}_l^t$) but different classes to be detected, *e.g.*, class-incremental detection on VOC.

Previous works on gating mechanism [1, 25, 36, 44] employ binary gates, *i.e.* $g_l^{c,t} \in \{0, 1\}$, to control the selection of sub-models in both training and inference stages, which results in the difficulty of back-propagation. Empirically we found that training with binary gates easily collapses on complex and large-scale detection tasks. Inspired by the widely used continuous relaxation strategy in neural architecture search (NAS) [9, 33, 51], we propose a **soften-and-discretize** strategy for our gate learning to overcome the convergence hardship. Concretely, **dynamic soft gates** are generated by our gated module during training stage and then discretized to be **static binary gates** for inference stage via choosing appropriate thresholds $\gamma_l^{c,t}$:

$$\gamma_l^{c,t} = \mathbb{E}_{(x,y)\sim D_{val}^t}[g_l^{c,t}], \tag{2}$$

where D_{val}^t is the validation set of the t^{th} task, c denotes the c^{th} channel of the l^{th} layer and g_l^c is the corresponding channel gate mentioned above.

The static binary gates are obtained by thresholding as follows:

$$\hat{g}_l^{c,t} = \mathbb{I}[g_l^{c,t} \geq \gamma_l^{c,t}], \tag{3}$$

where $\mathbb{I}[\cdot]$ is an indicator function. It should be noticed that the dynamic soft gates are input-dependent while the static binary gates are *input-independent* and applicable to all samples in a task in inference stage. Similar to the coupling problem of discrete architecture encoding and the submodel weights [16, 33] in NAS (retraining or fine-tuning weights is needed), the activated channel weights and the discretized binary gates may not well accommodate to each other. To address this issue, we slightly fine-tune the newly activated channel weights to adapt it to the binary gates. Ultimately, the gated output feature for inference time is given as:

$$\boldsymbol{f}_{l+1}^t = \hat{\boldsymbol{g}}_l^t \odot (F_l^t(\boldsymbol{f}_l^t)), \tag{4}$$

where $\hat{\boldsymbol{g}}_l^t = [\hat{g}_l^{1,t}, \hat{g}_l^{2,t}, ..., \hat{g}_l^{c^{l+1},t}]$ are binary gates.

In addition to the soften-and-discretize strategy, **sparsity constraint** [1] and **knowledge distillation** [20] are incorporated to guide the learning of our sparse gated model. The sparsity loss encourages a smaller size of the sub-model in each task and reserves more inactivated channels for future tasks:

$$\mathcal{L}_{sparsity} = \mathbb{E}_{(x,y)\sim D^t}\left[\frac{1}{L}\sum_{l=1}^{L}\frac{\|G_l(\boldsymbol{f}_l^t, C^t)\|_1}{c^{l+1}}\right], \tag{5}$$

where L is the number of layers. Distinct from existing distillation-based continual learning methods [34, 39, 45, 53], we use feature-level knowledge distillation to avoid the training collapse of the sparse gated model rather than alleviating catastrophic forgetting. As shown in Fig. 2, the distillation loss is leveraged to let the gated student model generate similar feature maps to the non-gated teacher model:

$$\mathcal{L}_{kd} = \frac{1}{L}\sum_{l=1}^{L}\mathrm{MSE}(\boldsymbol{f}_l^t, \tilde{\boldsymbol{f}}_l^t), \tag{6}$$

where MSE is the mean square error between feature maps from student and teacher models, $\tilde{\boldsymbol{f}}_l^t$ is the feature map of teacher model, *i.e.* $\tilde{\boldsymbol{f}}_l^t = \tilde{F}_{l-1}^t(\tilde{\boldsymbol{f}}_{l-1}^t)$, $\tilde{F}_{l-1}^t$ is a traditional convolutional operation without gated module.

3.3. Task Correlation Guided Gating Diversity

With the help of the aforementioned task-aware gated module, existing knowledge can be fully stored in submodels' weights and easily retrieved by their binary gates.

When we move to solve the subsequent tasks, the previously activated channel weights will be frozen and the other inactivated ones will be still learnable. The sub-model for a new task can select parts of frozen channel weights of previous tasks to facilitate and promote its learning process, which can be seen as a scheme of knowledge sharing among tasks. In this way, the problem of catastrophic forgetting in continual object detection can be solved by the gating mechanism proposed in Sec. 3.2. However, we observe performance degradation when learning on the subsequent tasks if domain gaps are significant. We find knowledge sharing provided by gating mechanism benefits the subsequent task if the domain gap is small, *e.g.*, COCO→VOC. But it limits the performance gain when tasks have obviously different foreground objects and backgrounds, *e.g.*, KITTI is an outdoor autonomous driving dataset and Kitchen contains indoor kitchen scenes. We attribute this phenomenon to the unawareness of cross-task correlation and propose the Prototypical Task Correlation Gating Mechanism to achieve dynamic balance between sharing existing knowledge and exploiting exclusive knowledge (*i.e.* identifying the exclusive knowledge compared to the existing one).

To further exploit the potential of inactivated channels for a new task and enhance the ability to digest new knowledge, we introduce a **gating diversity loss** to allow more channel gates to be activated during training the new task. The diversity loss $\mathcal{L}_{diversity}$ is in the form of entropy:

$$\mathcal{L}_{diversity}^{l,t} = q_l^t \log q_l^t + (1 - q_l^t) \log(1 - q_l^t), \tag{7}$$

$$q_l^t = \frac{\sum_{i=1}^{m_l^t} g_l^{c,t} \mathbb{I}[g_l^{c,t} \geq \eta]}{\sum_{i=1}^{m_l^t} g_l^{c,t}}, \tag{8}$$

where m_l^t is the total number of previously inactivated channels in layer l reserved for task t, $\mathbb{I}$ is an indicator function, η is a hyper-parameter threshold. q_l^t in Eq. (8) is an estimated ratio of the newly activated gates for task t in layer l. The gating diversity loss in Eq. (7) is a negative entropy corresponding to such estimated ratio q_l^t. Minimizing the gating diversity loss means more gates will be activated if necessary while the saturation of channels is avoided thanks to the property of entropy.

Taking the correlation between different tasks into account, it can be intuitively assumed that few exclusive knowledge (newly activated gates) are necessary when two tasks are similar or almost identical, and vice versa. Going one step further, our ROSETTA is expected to automatically determine whether more gating diversity is needed for a new task. Hence, a task correlation guided **Gating Diversity Controller** (GDC) is integrated to adaptively adjust the gating diversity based on class-specific prototypes. For task t, the prototype of the i^{th} class $\boldsymbol{p}_i^t$ is stored in the memory bank as the mean RoI (Region-of-Interest) feature of class i, which is generated by an object detector. The class-to-class prototypical correlation matrix $\mathbf{M}^{m\to n} \in \mathbb{R}^{|C^m|\times|C^n|}$ can be used to depict the class-to-class correlation (distance) between task m and task n ($m \leq n$):

$$\mathbf{M}_{i,j}^{m\to n} = \mathrm{MSE}(\boldsymbol{p}_i^m,\ \boldsymbol{p}_j^n),\ i \in C^m \wedge j \in C^n, \tag{9}$$

where C^t denotes the categories to be recognized for task t. Based on $\mathbf{M}^{m\to n}$, we can define the class-to-task correlation between the j^{th} class of task n and the m^{th} task ($\boldsymbol{p}^m = [p_1^m, p_2^m, ..., p_{C^m}^m]$) as:

$$\mathcal{R}(\boldsymbol{p}_j^n, \boldsymbol{p}^m) = \begin{cases} \min\limits_{i\in C^m} & \mathbf{M}_{i,j}^{m\to n}, \quad m < n \\ \min\limits_{i\in C^m, i\neq j} & \mathbf{M}_{i,j}^{m\to n}, \quad m = n. \end{cases} \tag{10}$$

The task-to-task correlation between task m and task n can be induced by the average of the class-to-task correlations:

$$\mathcal{R}(\boldsymbol{p}^n, \boldsymbol{p}^m) = \frac{1}{C^n} \sum_{j\in C^n} \mathcal{R}(\boldsymbol{p}_j^n, \boldsymbol{p}^m), m \leq n. \tag{11}$$

Based on the prototypical task correlation $\mathcal{R}(\boldsymbol{p}^n, \boldsymbol{p}^m)$, the gating diversity controller (GDC) is used to give a weight for the diversity loss in Eq. (7) and maintain an equilibrium between sharing knowledge and exploiting exclusive knowledge:

$$\phi(\boldsymbol{p}^n, \boldsymbol{p}^m) = \max\{\frac{\mathcal{R}(\boldsymbol{p}^n, \boldsymbol{p}^m) - \mathcal{R}(\boldsymbol{p}^m, \boldsymbol{p}^m)}{\mathcal{R}(\boldsymbol{p}^n, \boldsymbol{p}^m)}, 0\}. \tag{12}$$

The controller in Eq. (12) provides a weight in the range of $[0, 1]$, which is a greater value when the knowledge transferring from task m to task n is more difficult and then more exclusive channels for task n should be activated.

By combining Eq. (7) and Eq. (12), the cross-task correlation guided gating diversity loss for task t ($t > 1$) is:

$$\mathcal{L}_{diversity} = \frac{1}{L} \frac{1}{(t-1)} \sum_{l=1}^{L} \sum_{i=1}^{t-1} \phi(\boldsymbol{p}^t, \boldsymbol{p}^i) \mathcal{L}_{diversity}^{l,t}. \tag{13}$$

To conclude, incorporating the cross-task correlation guided gating diversity loss into the continual learning pipeline of our task-aware gated object detector can achieve better equilibrium between acquiring new knowledge and preserving old knowledge by adaptively adjusting the diversity of channel gates.

4. Experiments

4.1. Experimental Setting

Datasets In general, there are two kinds of experimental settings for continual detection: task-incremental object detection [34] and class-incremental object detection [23]. In terms of task-incremental object detection, D^i and D^j ($i \neq j$) are two independent datasets. As for class-incremental object detection, the categories of objects we expect to detect is incrementally observed in a dataset, *i.e.* $C^t \subset C^{t+1}$. In brief, the former setting is dataset-level incremental detection and the latter one is in a class-level fashion. Without loss of generality, we will evaluate our model for both these two case of continual object detection. Following the continual object detection settings in [34] and [23], we conduct experiments on COCO [30], Pascal VOC [11], KITTI [14] and Kitchen [15]. For task-incremental object detection, the datasets we use are COCO-VOC and KITTI-Kitchen, following the settings in [34]. COCO and VOC have similar domains, and 20 categories of these two datasets are coincident. KITTI is an outdoor autonomous driving dataset while Kitchen contains indoor kitchen scenes. Therefore, domains of KITTI and Kitchen are significantly different and their categories are totally disjoint, which makes continual object detection on KITTI-Kitchen more challenging than COCO-VOC. For class-incremental continual object detection, ROSETTA is validated following the incremental protocol in [23]. The datasets for incremental detection come from Pascal VOC 2007 [11] and three settings ("10 + 10", "15 + 5", "19 + 1") are used for evaluation. We split Pascal VOC into two tasks as an incremental task sequence. For example, "10 + 10" means that a detector is firstly trained on the images only with the annotations of the first 10 categories, and then trained on the those containing the other 10 classes.

Implementation Details Our experiments are conducted on 8 GPUs with batch size of 16 and implemented by PyTorch [38]. Without loss of generality, our ROSETTA can be equipped different object detector backbones. In our experiment, we choose two representative object detectors as our backbones: Faster R-CNN [41] and Sparse R-CNN [46]. Faster R-CNN is a commonly used detector which heavily relies on dense object candidates. Sparse R-CNN is a recently proposed detector without NMS, which has a sparse-in sparse-out paradigm with a higher efficiency. For Faster R-CNN backbone, we use the same training scheme as [34] and generate our baseline results of joint training and fine-tuning. As for Sparse R-CNN, we use its default hyper-parameters with 100 proposal boxes. Specifically, we use 3× 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 proposed in Sec. 3.2 is applied to the last three stages of the ResNet50 [18] backbone for both Faster R-CNN and Sparse R-CNN while the first two stages are fixed during training. 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 Sec. 3.2, our 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 [34].

COCO-VOC				
Methods	COCO → VOC		VOC → COCO	
Joint Training (Faster R-CNN)	48.8	81.6	81.6	48.8
Fine-tuning (Faster R-CNN)	23.2	79.5	74.7	47.1
LwF Detection† [45]	26.6	73.0	-	
Feature Distillation† [43]	26.9	72.4	-	
Attention Distillation† [52]	28.5	73.0	-	
EWC† [24]	32.2	73.4	-	
MAS† [2]	32.7	73.4	-	
AFD† [34]	36.8	75.2	-	
EWC [24]	27.2	75.0	67.0	44.4
MAS [2]	28.1	74.8	69.0	43.9
AFD [34]	27.8	77.1	75.4	45.1
ROSETTA-Faster R-CNN(Ours)	**48.6**	**80.5**	**77.5**	**46.5**
Joint Training (Sparse R-CNN)	52.9	83.2	83.2	52.9
Fine-tuning (Sparse R-CNN)	35.7	81.2	74.9	49.9
ROSETTA-Sparse R-CNN(Ours)	49.5	82.3	79.5	48.3
KITTI-Kitchen				
Methods	KITTI → Kitchen		Kitchen → KITTI	
Joint Training (Faster R-CNN)	55.0	83.7	83.7	55.0
Fine-tuning (Faster R-CNN)	7.7	82.2	13.5	54.2
LwF Detection† [45]	39.4	69.9	59.9	54.7
Feature Distillation† [43]	35.0	69.4	62.7	54.4
Attention Distillation† [52]	39.8	71.0	64.2	52.8
EWC† [24]	48.3	65.5	68.4	52.8
MAS† [2]	42.8	71.7	67.7	**55.6**
AFD† [34]	48.1	72.4	68.6	53.4
EWC [24]	15.8	65.8	10.7	53.4
MAS [2]	8.9	70.3	11.5	54.0
AFD [34]	36.6	72.0	20.5	50.5
ROSETTA-Faster R-CNN(Ours)	**53.9**	**78.1**	**78.4**	54.7
Joint Training (Sparse R-CNN)	55.5	81.8	81.8	55.5
Fine-tuning (Sparse R-CNN)	20.2	79.8	18.6	53.6
ROSETTA-Sparse R-CNN(Ours)	53.3	78.3	78.2	54.5

Table 1. Comparisons with existing methods for task-incremental object detection, in terms of mAP (%). Arrows indicate the order of learning. Methods storing exemplars for experience replay are denoted by '†'. '-' indicates that results are not given by [34] and no public codes are provided. Results of ROSETTA equipped with Faster R-CNN and Sparse R-CNN backbones are both shown in this table. The best results with Faster R-CNN backbone are denoted in boldface.

4.2. Comparison with Existing Methods

For task-incremental object detection, we compare our ROSETTA with AFD [34], LwF Detection [45], Feature Distillation [43], Attention Distillation [52], EWC [24], MAS [2], the joint training and fine-tuning baselines. In terms of joint training which is usually seen as the empirical upper bound of continual learning, data of all tasks are accessible during training the object detector. As for fine-tuning, all tasks are sequentially trained without additional constraints. For class-incremental object detection, the methods we compare include ILOD [45], Faster ILOD [39], ORE [23]. The performance of our ROSETTA with Faster R-CNN and Sparse R-CNN backbones (termed as "ROSETTA-Faster R-CNN" and "ROSETTA-Sparse R-CNN" respectively) are given in Tab. 1 and Tab. 2.

10 + 10 setting	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	**mAP**
All 20	68.5	77.2	74.2	55.6	59.7	76.5	83.1	81.5	52.1	79.8	55.1	80.9	80.1	76.8	80.5	47.1	73.1	61.2	76.9	70.3	70.51
First 10	79.3	79.7	70.2	56.4	62.4	79.6	88.6	76.6	50.1	68.9	0	0	0	0	0	0	0	0	0	0	35.59
New 10	7.9	0.3	5.1	3.4	0	0	0.2	2.3	0.1	3.3	65	69.3	81.3	76.4	83.1	47.2	67.1	68.4	76.5	69.2	36.31
ILOD [45]	69.9	70.4	69.4	54.3	48	68.7	78.9	68.4	45.5	58.1	59.7	72.7	73.5	73.2	66.3	29.5	63.4	61.6	69.3	62.2	63.15
ILOD + Faster R-CNN	70.5	75.6	68.9	59.1	56.6	67.6	78.6	75.4	50.3	70.8	43.2	68.1	66.2	65.1	66.5	24.3	61.3	46.6	58.1	49.9	61.14
Faster ILOD [39]	72.8	75.7	71.2	60.5	61.7	70.4	83.3	76.6	53.1	72.3	36.7	70.9	66.8	67.6	66.1	24.7	63.1	48.1	57.1	43.6	62.16
ORE [23]	63.5	70.9	58.9	42.9	34.1	76.2	80.7	76.3	34.1	66.1	56.1	70.4	80.2	72.3	81.8	42.7	71.6	68.1	77	67.7	64.58
ROSETTA-Faster R-CNN	74.2	76.2	64.9	54.4	57.4	76.1	84.4	68.8	52.4	67.0	62.9	63.3	79.8	72.8	78.1	40.1	62.3	61.2	72.4	66.8	**66.80**
15 + 5 setting	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	**mAP**
First 15	74.2	79.1	71.3	60.3	60	80.2	88.1	80.2	48.8	74.6	61	76	85.3	78.2	83.4	0	0	0	0	0	55.03
New 5	3.7	0.5	6.3	4.6	0.9	0	8.8	3.9	0	0.4	0	0	16.4	0.7	0	41	55.7	49.2	59.1	67.8	15.95
ILOD [45]	70.5	79.2	68.8	59.1	53.2	75.4	79.4	78.8	46.6	59.4	59	75.8	71.8	78.6	69.6	33.7	61.5	63.1	71.7	62.2	65.87
ILOD + Faster R-CNN	63.5	76.3	70.7	53.1	55.8	67.1	81.5	80.3	49.6	73.8	62.1	77.1	79.7	74.2	73.9	37.1	59.1	61.7	68.6	61.3	66.35
Faster ILOD [39]	66.5	78.1	71.8	54.6	61.4	68.4	82.6	82.7	52.1	74.3	63.1	78.6	80.5	78.4	80.4	36.7	61.7	59.3	67.9	59.1	67.94
ORE [23]	75.4	81	67.1	51.9	55.7	77.2	85.6	81.7	46.1	76.2	55.4	76.7	86.2	78.5	82.1	32.8	63.6	54.7	77.7	64.6	68.51
ROSETTA-Faster R-CNN	76.5	77.5	65.1	56.0	60.0	78.3	85.5	78.7	49.5	68.2	67.4	71.2	83.9	75.7	82.0	43.0	60.6	64.1	72.8	67.4	**69.17**
19 + 1 setting	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	**mAP**
First 19	77.8	81.7	69.3	51.6	55.3	74.5	86.3	80.2	49.3	82	63.6	76.8	80.9	77.5	82.4	42.9	73.9	70.4	70.4	0	67.34
Last 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	3.2
ILOD [45]	69.4	79.3	69.5	57.4	45.4	78.4	79.1	80.5	45.7	76.3	64.8	77.2	80.8	77.5	70.1	42.3	67.5	64.4	76.7	62.7	68.25
ILOD + Faster R-CNN	60.9	74.6	70.8	56	51.3	70.7	81.7	81.5	49.45	78.3	58.3	79.5	79.1	74.8	75.7	42.8	74.7	61.2	67.2	65.1	67.72
Faster ILOD [39]	64.2	74.7	73.2	55.5	53.7	70.8	82.9	82.6	51.6	79.7	58.7	78.8	81.8	75.3	77.4	43.1	73.8	61.7	69.8	61.1	68.56
ORE [23]	67.3	76.8	60	48.4	58.8	81.1	86.5	75.8	41.5	79.6	54.6	72.8	85.9	81.7	82.4	44.8	75.8	68.2	75.7	60.1	68.89
ROSETTA-Faster R-CNN	75.3	77.9	65.3	56.2	55.3	79.6	84.6	72.9	49.2	73.7	68.3	71.0	78.9	77.7	80.7	44.0	69.6	68.5	76.1	68.3	**69.64**

Table 2. Comparisons with existing class-incremental object detectors with Faster R-CNN backbone on three different settings: "10 + 10", "15 + 5", "19 + 1". For example, "First 15" means training on the first 15 classes of Pascal VOC 2007 with Faster R-CNN backbone and "New 5" refers to fine-tuning on the new 5 classes. The best results are denoted in boldface.

Task-incremental object detection Here we divide task-incremental object detection into two case: (1) domains of tasks are relatively similar, (2) domains are significantly different. These two cases are corresponding to our experiments conducted on COCO-VOC and KITTI-Kitchen, respectively. Results are given in Tab. 1. Although ROSETTA does not need any exemplar for experience replay, it also outperforms state-of-the-art methods including exemplar-based methods which are denoted by '†'. Thanks to the gating mechanism, ROSETTA can better obviate the problem of catastrophic forgetting, even in the case of a significant domain gap. Specifically, on Kitchen→KITTI and KITTI→Kitchen, the performance of other methods on `task1` dramatically drop while our ROSETTA-Faster R-CNN can alleviate the catastrophic forgetting to a great extent. For example, on KITTI→Kitchen, ROSETTA-Faster R-CNN achieves 17.3 mAP and 6.1 mAP improvements on KITTI and Kitchen compared to AFD [34], respectively. Equipped with Sparse R-CNN, ROSETTA-Sparse R-CNN outperforms our fine-tuning baseline on `task1`, validating its effectiveness of avoiding catastrophic forgetting.

Class-incremental object detection For the setting of class-incremental object detection, we mainly compare our ROSETTA with other incremental detectors with a fair Faster R-CNN backbone, in terms of three settings ("10 + 10", "15 + 5", "19 + 1") following [23]. As shown in Tab. 2, our ROSETA-Faster R-CNN performs well on the class-incremental tasks against the state-of-the-art incremental detectors [23,39,45]. Notably, ROSETTA-Faster R-CNN surpasses ORE [23] by 2.2 mAP on the setting of "10 + 10".

	COCO →	VOC →	KITTI →	Kitchen	Average
Fine-tuning	4.3	16.3	45.9	87.0	38.4
ROSETTA(Ours)	49.5	82.3	62.5	87.3	**70.4**
	KITTI →	COCO →	VOC →	Kitchen	Average
Fine-tuning	24.9	5.9	24.6	85.2	35.2
ROSETTA(Ours)	53.3	48.9	80.3	84.5	**66.8**

Table 3. Results of sequential training on 4 tasks.

$\mathcal{L}_{sparsity}$	$\mathcal{L}_{kd}$	$\mathcal{L}_{diversity}$	KITTI →	Kitchen	Gates			
					only `task1`	overlap	only `task2`	not used
✗	✗	✗	51.7	70.2	6.8%	48.3%	9.4%	35.5%
✓	✗	✗	51.0	70.5	6.1%	26.4%	1.2%	66.3%
✓	✓	✗	53.3	73.3	9.6%	32.9%	8.1%	49.4%
✓	✓	✓	53.3	78.3	14.3%	28.2%	20.3%	37.2%

Table 4. Ablation studies on KITTI→Kitchen.

4.3. Results on Four Sequential Tasks

To verify the capability of solving multiple tasks, we compare ROSETTA to the fine-tuning baseline with the same Sparse R-CNN backbone on two task-incremental sequences: COCO → VOC → KITTI → Kitchen and KITTI → COCO → VOC → Kitchen. The sequential results shown in Tab. 3 illustrate that our ROSETTA outperforms the fine-tuning strategy by a large margin (32 mAP and 31.6 mAP in average of four tasks, respectively).

4.4. Ablation Study

The effect of $\mathcal{L}_{sparsity}$, $\mathcal{L}_{kd}$ and $\mathcal{L}_{diversity}$ To verify the effectiveness of each module in our proposed ROSETTA, we conduct ablation experiments on KITTI→Kitchen with Sparse R-CNN backbone to ablate $\mathcal{L}_{sparsity}$ $\mathcal{L}_{kd}$ and $\mathcal{L}_{diversity}$. As shown in Tab. 4, our baseline model, *i.e.* the trivial gated detector proposed in Sec. 3.2, can alleviate the catastrophic forgetting while two tasks both occupies lots of channel gates with a great overlap of 48.3%. Thanks to the sparsity constraint, we obtain sparser models for both `task1` and `task2`. Moreover, knowledge distillation helps the gated module achieve better convergence due to the guidance of non-gated teacher. In Tab. 4, $\mathcal{L}_{kd}$ benefits the gate learning of both `task1` and `task2` and we observe performance improvement on both two tasks. By introducing our proposed task correlation guided gating diversity loss $\mathcal{L}_{diversity}$, ROSETTA can automatically capture the significant domain gap between KITTI and Kitchen. Therefore, more exclusive knowledge is exploited for `task2` (20.3% only for `task2`) and less gates are overlapped by two tasks. With the gating diversity constraint on `task2`, we observe a remarkable improvement of 5.0 mAP on Kitchen.

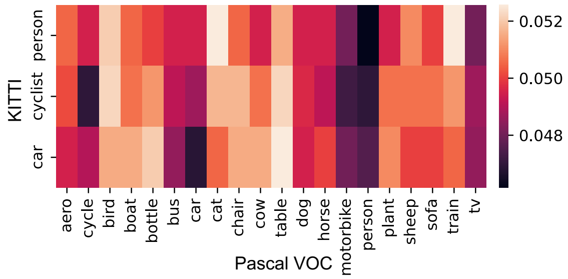

Figure 4. Visualization of the normalized prototypical correlation matrix of VOC→KITTI, *i.e.* $\mathbf{M}^{\text{VOC}\rightarrow\text{KITTI}}$. Darker color indicates higher similarity between the classes of these two tasks.

4.5. Cross-task Analysis

Cross-task correlation To understand how ROSETTA captures the cross-task correlation, we visualize the normalized class-to-class prototypical correlation matrix of VOC→KITTI, *i.e.* $\mathbf{M}^{\text{VOC}\rightarrow\text{KITTI}}$. For convenience, the visualized matrix in Fig. 4 is its transpose. As illustrated in Fig. 4, ROSETTA is able to find out the correct category correspondences between KITTI and VOC. For example, “cyclist” means people engaged in cycling. Thus, in Fig. 4, the class of “cyclist” in KITTI is close to “cycle”, “motorbike” which has similar appearance and semantics to “cycle” and “person” in VOC.

Gate analysis In Fig. 5, we provide the statistical information of the learned binary gates on KITTI-Kitchen and COCO-VOC. Here colors and the corresponding percentage represent the proportion of channel gates activated for different tasks. Since domains of COCO and VOC are similar, the gates learned on `task1` are almost inherited by `task2`. By comparison, our ROSETTA automatically finds that it needs to activate more channels for `task2` by observing the domain gap between KITTI and Kitchen.

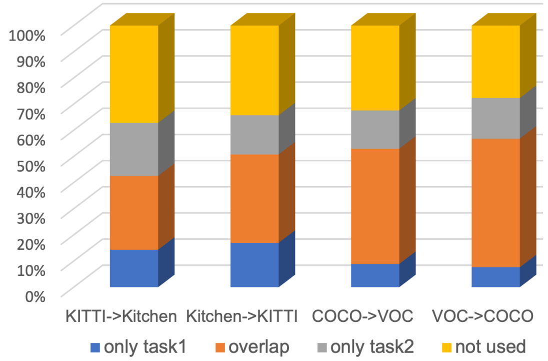

Figure 5. Analysis of gates for task-incremental detection.

5. Conclusion and Discussion

In this paper, we propose a sparse and dynamic framework for continual object detection via p**R**ot**O**typical ta**S**k corr**E**la**T**ion guided ga**T**ing mech**A**nism (ROSETTA) to memorize previous knowledge and further promote learning future tasks by knowledge sharing. We propose a task-aware gated module and integrate it with the backbone of object detectors. In this way, the gated detector is capable of avoiding catastrophic forgetting by storing the sub-models' weight and the corresponding binary gates. To further promote learning subsequent tasks, we propose a task correlation guided gating diveristy controller to capture the cross-task correlation. Comprehensive experiments on COCO-VOC, KITTI-Kitchen and class-incremental detection on VOC and sequential learning of four tasks verifies the superiority of our ROSETTA on both class-based and task-based continual object detection.

Potential negative social impact Our method has no ethical risk on dataset usage and privacy violation as all the benchmarks are public and transparent.

Limitation With regards to the limitation of our work, we only focus on one kind of task (modality) for the continual learning system *i.e* object detection in the visual modality. As our expectation, we aim to extend our ROSETTA to a more general continual system which can deal with different kinds of tasks and modalities in future work.

Acknowledgment

This work was supported in part by National Key RD Program of China under Grant No. 2020AAA0109700, National Natural Science Foundation of China (NSFC) No.61976233, Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, Guangdong Outstanding Youth Fund (Grant No. 2021B1515020061).

References

[1] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. Bejnordi. Conditional channel gated networks for task-aware continual learning. In *CVPR*, 2020. 2, 3, 4

[2] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018. 2, 6

[3] R. Aljundi, M. Rohrbach, and T. Tuytelaars. Selfless sequential learning. In *ICLR*, 2019. 2

[4] Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. In *ICLR*, 2020. 3

[5] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. *arXiv:1702.07811*, 2017. 3

[6] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020. 2

[7] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 2

[8] Jinting Chen, Zhaocheng Zhu, Cheng Li, and Yuming Zhao. Self-adaptive network pruning. In *ICONIP*, 2019. 3

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019. 4

[10] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In *CVPR*, 2019. 3

[11] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 2010. 6

[12] R. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 1999. 2

[13] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. In *International Conference on Learning Representations*, 2018. 3

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 6

[15] G. Georgakis, M. Reza, A. Mousavian, P. Le, and J. Košecká. Multiview rgb-d dataset for object instance detection. In *International Conference on 3D Vision*, 2016. 6

[16] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 4

[17] R. Hadsell, D. Rao, A. Rusu, and R. Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020. 2, 3

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[19] Charles Herrmann, Richard Strong Bowen, and Ramin Zabih. An end-to-end approach for speeding up neural network inference. *arXiv preprint arXiv:1812.04180*, 2018. 3

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015. 4

[21] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. In *NeurIPS*, 2019. 3

[22] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 3

[23] 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. 2, 3, 5, 6, 7

[24] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 2017. 2, 6

[25] Changlin L., Guangrun W., Bing W., Xiaodan L., Zhihui L., and Xiaojun C. Dynamic slimmable network. In *CVPR*, 2021. 3, 4

[26] Seungwon Lee, Sima Behpour, and Eric Eaton. Sharing less is more: Lifelong learning in deep networks with selective layer transfer. In *International Conference on Machine Learning*, pages 6065–6075. PMLR, 2021. 2

[27] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *CVPR*, 2020. 3

[28] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016. 2

[29] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *NeurIPS*, 2017. 3

[30] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[31] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 1, 2

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2

[33] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018. 4

[34] X. Liu, H. Yang, A. Ravichandran, R. Bhotika, and S. Soatto. Multi-task incremental learning for object detection. *arXiv:2002.05347*, 2020. 2, 3, 4, 5, 6, 7

[35] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017. 2

[36] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018. 3, 4

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 4

[38] 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. 6

[39] C. Peng, K. Zhao, and B. Lovell. Faster ilod: Incremental learning for object detectors based on faster rcnn. *Pattern Recognition Letters*, 2020. 2, 4, 6, 7

[40] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. *Advances in Neural Information Processing Systems*, 2019. 3

[41] 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, 2, 6

[42] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2018. 2

[43] A. Romero, N. Ballas, S. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv:1412.6550*, 2014. 6

[44] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018. 2, 3, 4

[45] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017. 2, 3, 4, 6, 7

[46] 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. 6

[47] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 3

[48] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 3

[49] Yue Wang, Jianghao Shen, Ting-Kuei Hu, Pengfei Xu, Tan Nguyen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. Dual dynamic inference: Enabling more efficient, adaptive and controllable deep inference. *JSTSP*, 2020. 3

[50] Wenhan Xia, Hongxu Yin, Xiaoliang Dai, and Niraj K Jha. Fully dynamic inference with deep neural networks. *IEEE Transactions on Emerging Topics in Computing*, 2021. 3

[51] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019. 4

[52] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 6

[53] Wang Zhou, Shiyu Chang, Norma Sosa, Hendrik Hamann, and David Cox. Lifelong object detection. *arXiv preprint arXiv:2009.01129*, 2020. 2, 4