Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 3044155, 21 pages
https://doi.org/10.1155/2023/3044155

WILEY | Hindawi

Research Article

A New Multinetwork Mean Distillation Loss Function for Open-World Domain Incremental Object Detection

Jing Yang,[1,2,3] **Kun Yuan**,[1] **Suhao Chen**,[4] **Qinglang Li**,[3] **Shaobo Li**,[2] **Xiuhua Zhang**,[5] **and Bin Li**[3]

[1]*State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China*
[2]*Key Laboratory of Advanced Manufacturing Technology of the Ministry of Education, Guizhou University, Guiyang 550025, China*
[3]*School of Mechanical Engineering, Guizhou University, Guiyang 550025, China*
[4]*Department of Industrial Engineering, South Dakota School of Mines and Technology, Rapid City 57701, USA*
[5]*College of Mechanical and Electronic Engineering, Guizhou Minzu University, Guiyang 550025, China*

Correspondence should be addressed to Kun Yuan; gs.kyuan20@gzu.edu.cn and Xiuhua Zhang; zhangxiuhua@gzmu.edu.cn

Received 12 April 2023; Revised 1 September 2023; Accepted 11 October 2023; Published 6 November 2023

Academic Editor: Gennaro Vessio

The development of object detection networks has reached a high point, and there have been significant improvements in accuracy and detection speed. Object detection is widely used in intelligent robots, self-driving cars, and other edge-intelligent terminals. Unfortunately, when a detector is allowed to learn new objects in an unfamiliar environment, it can catastrophically forget the objects it has already learned. In particular, reliable and stable knowledge cannot be extracted from old models. Based on this, a new multinetwork mean distillation loss function for open-world domain incremental object detection is presented. To better extract reliable and stable knowledge from old models, we enhanced the distillation output of the detector with a ResNet50 backbone and an output RoI head. The distillation output of the intermediate RPN is softened by adaptive distillation. To obtain more stable results, the ResNet50 backbone and RPN on the channel are zero-averaged. Various incremental steps and stability experiments are performed on two benchmark datasets, PASCAL VOC and MS COCO. The experimental results show the excellent performance of our method in different experimental scenarios, and it is superior to the most advanced methods. For example, in the setting of the batch task, incremental object detection on the PASCAL VOC and MS COCO datasets is improved by 3.4% and 2.1%, respectively.

1. Introduction

Object detection models, which are currently the most representative models for vision tasks, play a significant role in fields such as intelligent robot tasks [1], autonomous driving [2], and other edge intelligent terminal schemes [3]. However, existing supervised models can only be trained on labeled task data from existing categories in the training dataset. Furthermore, to adapt to a new task, the network parameters of the model need to be adjusted, and it is difficult for existing object detection models to adjust to dynamic real-world environments, causing them to forget old knowledge [4].

In this work, we investigate the problem of class-increment multinetwork object detection based on a catastrophic forgetting mechanism. In the incremental setting, task queues are introduced sequentially to the object detector, and a high-performance agent should maintain the old task performance during the new task learning process. Therefore, the model adaptive parameter update process that is executed when new task input is limited by setting knowledge distillation [4] at the model parameter level [5–7]. For Faster-RCNN [8] with a multinetwork structure, it is difficult to alleviate the catastrophic forgetting problem with the distillation of only a single network [9], and it is more effective to use multinetwork distillation [5, 10] to

retain old knowledge for the whole network. Furthermore, existing incremental object detection methods based on multinetwork knowledge distillation make the model more prone to learning the old task and minimize the new task learning effect.

Based on these considerations, Peng et al. [10] proposed an incremental learning approach to multinetwork adaptive distillation, where distillation is set up in multiple networks and the teacher network is used as a lower bound for adaptive knowledge extraction. However, forcing a comparison of the outputs of the teacher network and the student network as an adaptive extraction condition may lead to significant loss caused by past knowledge being identified as more important for new task learning and zeroed out for distillation loss; the weights related to this output value may be equally important for the old task. In contrast, Joseph et al. [5] performed meta-learning by specifying an RoI head layer and setting a certain number of iterations to optimize the gradient update direction to better learn the new task. However, because the region proposal network (RPN) for the class is agnostic, the method does not include a distillation loss term. The correct rate of RPN classification and anchor regression of an object and the background of an old task directly affect the accuracy of RoI head prediction and degeneration of the old class in the next stage because the candidate regions learned by the RoI head are generated from the previous step after pooling. This results in a lack of candidate regions for the old task in the learning process of the RoI head, which affects the ability of the object detection model to recognize new and old classes. In addition, by changing the network parameters during training to fit the new task, a direct calculation of the distillation loss of the network output at each stage will cause the output of the new model to be very different from the output data distribution of the old model. This will make the network output difficult to fit and unstable.

To address the aforementioned challenges based on the Faster-RCNN incremental object detector, we propose a new distillation scheme for the Faster-RCNN detector. We improve the distillation output of the ResNet50 backbone at the input level and the RoI head network at the output level, and we use adaptive distillation to maintain the past knowledge of the RPN. Moreover, we adopt the meta-learning strategy in [5] to mitigate the degradation of model learning performance for new tasks caused by knowledge distillation. In addition, to address the problem of bias in the output data distribution of the new and old models, we perform zero averaging on the output data of the ResNet50 backbone and RPN of the new and old models to mitigate bias in the output data distribution of the new model. Consequently, the primary contributions of our work are as follows:

(i) We propose a new scalable Faster-RCNN detector-based multinetwork distillation scheme that uses enhanced distillation values for the ResNet50 backbone and RoI head network and adaptive distillation for the RPN to mitigate the catastrophic forgetting problem.

(ii) To alleviate the instability of the network output caused by the differences in the old and new network outputs, we perform zero averaging on the output of the backbone network and RPN of the old and new models and consider the RoI head network averaged over the class to produce a new set of distillation losses.

(iii) We extensively evaluate the PASCAL VOC and COCO benchmark datasets and compare two advanced baseline methods. The experimental findings demonstrate the superior performance of our approach in various incremental scenarios.

2. Related Works

Incremental learning is a special machine learning paradigm that simulates the human brain's learning of sequential task streams, where the model can continuously learn new tasks and maintain old task performance. However, to maintain such properties, it is necessary to address the forgetting problem of the model due to new task learning [11]. On this basis, this section proposes incremental learning techniques for knowledge distillation and loss minimization.

2.1. Incremental Learning Approach for the Knowledge Distillation Strategy. Knowledge distillation [4] methods have been extended to mutual distillation learning [12–14], assisted distillation learning [15–17], spatial location distillation learning [18, 19], and dataset distillation learning [20–22]. In addition, knowledge distillation can be used in incremental learning because of its ability to transfer knowledge from one model to another [8–13]. Knowledge distillation in incremental learning typically transfers old information from the teacher's network to the student's network to alleviate the forgetting of old knowledge. As a traditional incremental learning method for knowledge distillation, LwF [23] mitigates forgetting by freezing the old model as the teacher network, using a temperature factor to soften the softmax output of the logit, then adding the factor to the current task loss as a regularization term, and constraining the model to mitigate forgetting through parameter updates. However, LwF is vulnerable to a significant learning bias when there is an imbalance between the old and new classes. To address this problem, Zhao et al. [24] combined weight aligning (WA) with knowledge distillation by utilizing WA to balance the weights of the old and new class information in the final fully connected layer while using knowledge distillation to maintain the model's discrimination of the old classes. In contrast, Dong et al. [25] used a dual-teacher distillation framework to mitigate the class imbalance problem using sampled unlabeled data to extract knowledge from the base class teacher and new class teacher models and transfer the knowledge to the student model. Similarly, Abdelsalam et al. [26] used a dual-teacher distillation strategy with regular and superclass teachers to solve the incremental implicitly refined classification (IIRC) problem. In feature relationship exploration, Yang et al. [9]

explored important correlations with old and new classes in the feature space. They maintained the ability to learn to detect new classes by passing correlations to retain the close relationships within important learning knowledge. Similarly, Dong et al. [27] proposed the exemplar relationship distillation incremental learning (ERDIL) method, which mines exemplar relationship information in old tasks through exemplar relationship graphs (ERG) and uses graph relationship-based knowledge distillation to transfer old knowledge to CNN models for learning new tasks. The above knowledge distillation-based methods all distill the model structure. The distilled old model is used to constrain the new model to update the new task learning process and maintain the performance of the old and new tasks. Additionally, the multinetwork structure of the object detection model gives us inspiration for model structure distillation.

2.2. Loss Optimization Method for Knowledge Distillation. In studies on knowledge distillation loss, existing distillation loss optimization methods [28–32] mainly optimize the deficiencies of the incremental learning processes by combining them with other techniques. Li et al. [30] prevented the features extracted from the intermediate neural network layers from changing drastically by adding feature distillation loss terms and minimizing the feature differences using a smoothed L1 loss function. EEIL [28] combines cross-entropy and distillation loss into an end-to-end learning network, using cross-entropy to learn new classes and distillation to retain knowledge corresponding to old classes. Xiang et al. [29] proposed a dynamic correction vector algorithm that combines representational memory and knowledge distillation loss to optimize cross-entropy and knowledge distillation loss functions to alleviate knowledge distillation bias and model overfitting problems. Douillard et al. [33] combined representation learning with distillation to mitigate the impact of feature extraction network changes by using a multiagent classifier through spatially based distillation loss-constrained representation evolution. To address the old-new data imbalance problem, Wu et al. [31] proposed a BiC algorithm for large-scale data processing based on distillation loss to correct old-new class bias. Similarly, Hou et al. [32] combined cross-entropy loss, feature-based distillation loss, and marginal ranking loss, which separates old and new classes, to mitigate the adverse effects of class imbalance. For the object detection problem, ILOD [9] uses knowledge distillation to regularize the output of the final classification and regression layers to retain the performance of the old task. Chen et al. [6] used cue learning to maintain the initial model feature information and added it to the distillation loss calculation while setting the confidence loss to extract the confidential information of the initial model to mitigate further forgetting. In the detector feature space, Yang et al. [34] investigated the applicability of both old and new classes and set a distillation loss term for two-stage Faster-RCNN using three perspectives—channel-based, point-based, and instance-based perspectives. Notably, the introduction of knowledge distillation aggravates the model's focus on new tasks and reduces their performance. Based on this, Peng et al. [10] applied adaptive distillation to multiple networks in the Faster-RCNN detector, using the teacher network as a lower bound and adaptively extracting knowledge to improve new task learning. In contrast, Joseph et al. [5] set the Warp loss to optimize the gradient update direction by specifying a layer of the RoI head network for meta-learning to make it better adapted to learning new tasks. In conclusion, the design of the loss function has optimization effects both on the degradation of the learning performance of new tasks caused by the introduction of knowledge distillation and on the alleviation of forgetfulness, prompting us to place a greater emphasis on our work on optimizing the distillation loss.

2.3. Gradient Mate Learning. In contrast to the aforementioned methodologies, contemporary scholars have focused their efforts on investigating the potential of meta-learning to facilitate enhanced computational efficiency in models. The initial work by Andrychowicz et al. [35] established the groundwork for the advancement of gradient meta-learning. Their proposal involved the automatic learning of hyperparameters for model optimizers by specifying particular optimizers. However, this approach poses difficulties in the selection of suitable optimization algorithms and parameter settings. Furthermore, model-agnostic meta-learning (MAML) [36] is a prominent method in gradient meta-learning that aims to enhance the initial model parameters for various tasks by performing gradient meta-updates on multiple tasks. This approach has garnered significant attention in the domain of few-sample learning. Nevertheless, the effectiveness of MAML is contingent upon the availability of high-quality task data and its sensitivity to hyperparameters, which impose certain restrictions. To address the challenges, Franceschi et al. [37] introduced differentiable convex optimization techniques in the field of meta-learning. This approach aimed to enhance the stability of the meta-update strategy and resolve the sensitivity issues associated with previous methods. In addition, Snell et al. [38] proposed a gradient-based meta-learning method utilizing a prototype network. This method proved effective in scenarios with limited training samples and overcame the challenges posed by sparse data through the concept of category prototyping. Similarly, Kedia and Chinthakindi [39] employed the reptile algorithm in meta-learning, combined with an inductive bias on pretrained weights, to enhance the generalization performance of the model. Notably, the meta-gradient of the reptile algorithm incorporates a gradient component that maximizes the inner product between different batch sizes from the same task, thereby facilitating greater adaptability to new tasks. Furthermore, Xu et al. [40] and other researchers have integrated reinforcement learning with gradient-based meta-learning techniques to enhance the effectiveness of deep reinforcement learning in large-scale applications. This is achieved by considering the payoff function as a parametric function with adjustable meta-parameters and addressing the optimization of task-specific objectives. Consequently, the integration of meta-learning has expanded the scope of incremental learning applications. Furthermore, Joseph et al. [5]

introduced a meta-learning approach for incremental target detection. This method involves learning from intermittent data inputs. During the meta-learning process, newly acquired data are incorporated into the Faster-RCNN network, and the RoI head modules with unfrozen weights are adjusted to better align with the new task. This approach addresses the issue of performance degradation in the model when attempting to distill knowledge for new tasks.

Inspired by the above work, a common strategy for incremental object identification methods to retain acquired knowledge is to simulate considerable activation of the original model by minimizing the first-order distillation loss. A new Faster-RCNN-based multinetwork structure distillation strategy for object detectors is devised as a result. Consideration is given to the impact of the new task's degraded performance because of knowledge distillation. In contrast, the problem of model output data distribution bias resulting from the instructor and student model tasks during learning is investigated.

3. Proposed Method

Incremental object detectors do not require all data classes to be available in advance. When new data are input, the unique structure of the detectors can prevent catastrophic instances of forgetfulness. Incremental object detection (iOD) is a commonly used approach for target detection, and it is characterized by its multinetwork structure. In this approach, a new model, referred to as the student network, is trained to learn a new task while keeping the weights of the previous model, known as the teacher model, fixed. This is achieved by setting the distillation loss, which helps mitigate the performance degradation of the student network on the new task caused by distillation. However, iOD's method of directly calculating the distillation loss on the input parameters can result in an imbalanced data distribution, leading to instability in the model's output. As shown in Figure 1, our method mitigates catastrophic forgetting via multinetwork knowledge distillation and experience replay. Specifically, we reinforce the focus on past tasks for the beginning and end of the model and consider the problem of poor RoI head training in the next phase due to the lack of RPN training on old tasks. We use adaptive distillation in the intermediate RPN phase. We employ adaptive distillation in the intermediate RPN stage to conditionally maintain the model's focus on the old task suggestion area; moreover, during the knowledge distillation process, we average the input to improve the stability of the model output. Furthermore, to prevent knowledge distillation from overprotecting past tasks and limiting the learning of new tasks, we use the gradient preprocessing meta-learning method in [5]. Specifically, the model learning process can be divided into two distinct phases: the incremental learning phase and the fine-tuning phase. In the incremental learning phase, the model learns the image features of the task by optimizing the specified loss function. On the other hand, the fine-tuning phase involves further training the model using microdata. This process allows the model parameters to be adjusted to effectively adapt to both previous and new tasks. The process of incremental learning can be divided into two stages: the initial stage involves learning a new task (referred to as new task loss), while the learning process is constrained by the distillation loss to limit changes in model parameters related to previous tasks (referred to as multinetwork mean distillation loss). The second stage utilizes a meta-learning gradient matrix to adjust the direction of the model's learning gradient (referred to as warp loss). In the succeeding exposition of the experimental findings, all experimental outcomes are refined, except those that lack a particular reference to the phase of incremental learning.

3.1. Problem Formulation. For a continuous task stream $\mathbb{T}$, task T_t ($T_t \in \mathbb{T}$) is delivered to the object detector at moment t, where T_t is composed of the incremental subtask set T_i ($i = 1, 2, \cdots, n$; $T_i \in T_t$) and task T_t is composed of the image dataset $\mathbb{D}$ with labels at moment t. The images contain several objects from different classes, but the labels are only valid for objects in task T_i. Moreover, we set the update rule of $\mathbb{M}_{OD}$ to be determined by θ. The parameter θ, which is used to define $\mathbb{M}_{OD}$, is divided into the task parameter C and the warp parameter ϖ; that is, $\theta = \psi \cup \varpi$, and $\psi \cap \varpi \neq \varnothing$. In terms of the learning process, the model learns the task parameter ψ in the first stage and the warp parameter ϖ in the second stage.

The specific learning process can be described as follows: at time t, when there is a task T_t, the object detector needs to learn the input picture $\mathbb{M}_{\mathrm{OD}}(I)$, which can be regarded as an aggregate function. For two-stage Faster-RCNN, $\mathbb{M}_{\mathrm{OD}}(I)$ can be formulated as a set function consisting of a backbone network $\mathbb{M}_{\mathrm{Backbone}}$, regional proposal network $\mathbb{M}_{\mathrm{RPN}}$, and RoI head $\mathbb{M}_{\mathrm{RoI\_Head}}$; i.e., $\mathbb{M}_{\mathrm{OD}}(I) = (\mathbb{M}_{\mathrm{Backbone}}\mathbb{M}_{\mathrm{RPN}}\ \mathbb{M}_{\mathrm{RoI\_Head}})(I)$.The input I is subjected to feature extraction by $\mathbb{M}_{\mathrm{Backbone}}$ to generate the feature map F. $\mathbb{M}_{\mathrm{RPN}}$ uses these features to generate N candidate regions that may contain objects and the corresponding scores, and each candidate region is subjected to $\mathbb{M}_{\mathrm{RoI\_Head}}$ to calculate the probability of being assigned to one of the classes in task $T_{i\leq t}$ and to perform regression calculations on its border positions. For incremental object detection, it is challenging to maintain the old task performance in a continuous task stream T without accessing all the data, and the incremental target detector needs to consider the classification of multiple networks with border regression on old knowledge memory compared to the normal incremental classification problem, such as the classification regression problem of $\mathbb{M}_{\mathrm{Backbone}}$ and the old class feature extraction problem of $\mathbb{M}_{\mathrm{RPN}}$ and $\mathbb{M}_{\mathrm{RoI\_Head}}$ on old knowledge in the Faster-RCNN mode. In our method, we employ a knowledge distillation strategy by freezing the past network model as the teacher network to guide the current task model as the student model. For the purpose of subsequent theoretical elaboration, elements labeled with "te" are defined as elements related to the teacher network, such as the teacher goal detector $\mathbb{M}_{\mathrm{OD}}^{\mathrm{te}}$, and elements labeled with "st" are defined as elements related to the student network, such as the student goal detector $\mathbb{M}_{\mathrm{OD}}^{\mathrm{st}}$.

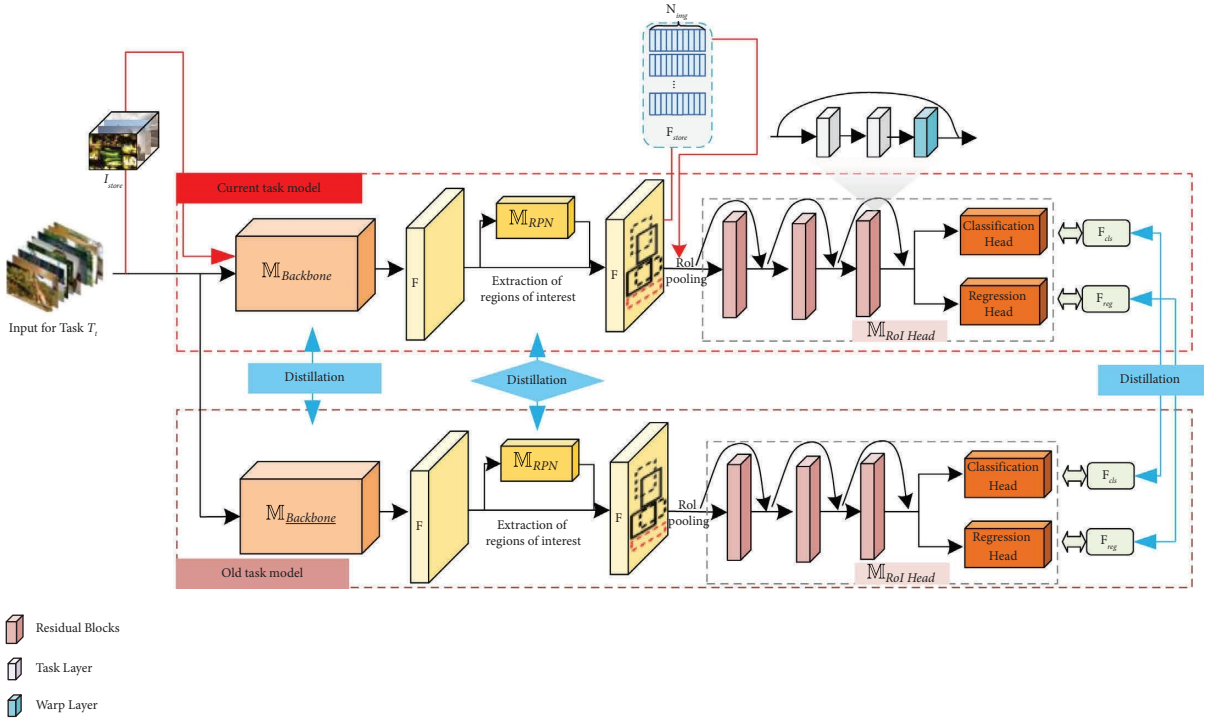

FIGURE 1: Implementation principle (the model uses different data inputs through different learning stages in the learning process; ➞ denotes the second stage of the meta-learning process; ➞ denotes the first stage of the new task learning process, which is included in the distillation process, and we set up the model to employ different distillation processes in different network structures).

3.2. New Task Loss. The learning of new tasks by the model can be viewed as the application of a loss function to learn model parameters. Specifically, the object detector uses the loss function to minimize the classification error and the bounding box positioning error for learning. Let $p = (p_0, \ldots, p_k)$ denote the predicted probability of $k + 1$ classes (k real classes and 1 background class); $l = (l_x; l_y; l_w; l_h)$ denotes the predicted bounding box position after pooling features for each RoI. The true labels are p^* (true class) and l^* (true class bounding box position), and $\mathscr{L}_{\text{RoI\_head}}$ is defined as follows:

$$\mathscr{L}_{\text{RoI\_Head}} = \mathscr{L}_{\text{cls}}(p, p^*) + \lambda[p^* \geq 1]\mathscr{L}_{\text{loc}}(l, l^*), \quad (1)$$

where $\mathscr{L}_{\text{cls}}(p, p^*) = -\log p/p^*$ is the log loss of the predicted class versus the true class, and $\mathscr{L}_{\text{loc}}$ is the smooth $L1$ loss; when $p^* = 0$, i.e., the background, the bounding box regression loss does not need to be calculated. Similarly, the training loss $\mathbb{M}_{RPN}$ yields a prediction score $o \in [0; 1]$, which indicates whether the selected region contains instances and corresponds to the bounding box prediction r. This loss is defined as follows:

$$\mathscr{L}_{\text{RPN}} = \mathscr{L}_{\text{cls}}(o, o^*) + \lambda o^* \mathscr{L}_{\text{loc}}(r, r^*), \quad (2)$$

where o^* indicates whether the region contains real labels; if the region contains real labels, $o^* = 1$, and it is 0 if the region does not contain real labels. r^* is the real bounding box regression target. The weighting parameter λ is set to 1 in all subsequent experiments.

3.3. Multinetwork Mean Distillation Loss. Similar to the way new tasks are learned, our model retains the performance of previous tasks, while new tasks are learned by calculating the mean distillation loss of multiple networks. Similar to Faster ILOD [10] and iOD [5], we use knowledge distillation to soften the softmax output by inserting a temperature factor T into the log output in equation (3) to maintain the model's performance on past tasks in a continuous task stream. However, unlike Faster ILOD, which uses multinetwork adaptive distillation, and iOD, which only distills the feature map and RoI head, we strengthen the distillation output at both ends $\mathbb{M}_{\text{Backbone}}$ and $\mathbb{M}_{\text{RoI\_Head}}$ to ensure the accuracy of backbone feature extraction at the very first input and RoI head detection at the final output. On the other hand, to ensure that the anchor of the $\mathbb{M}_{\text{RoI\_Head}}$ input contains past memory, we use adaptive zero mean distillation in the middle RPN layer to alleviate the overprotection of past knowledge while preserving the RPN's memory of past knowledge by adaptively increasing the distillation loss.

$$S_i^T = \frac{\exp(z_w/T)}{\sum_j \exp(z_d/T)}. \quad (3)$$

In addition, we consider that the direct introduction of knowledge distillation will cause the new model to update the network parameters adaptively during the training process to adapt to the new task, resulting in a large deviation of the output from the output data distribution of the old model,

which makes the network output difficult to fit and unstable. Therefore, we first apply zero-mean filtering to the inputs of $\mathbb{M}_{\text{Backbone}}$ and $\mathbb{M}_{\text{RPN}}$ to obtain the new outputs before calculating the distillation loss. Specifically, the zero mean is obtained by subtracting the mean of all pixels for each pixel f_i, as shown in equation (4). The model output after zero averaging has all pixel points distributed with the origin as the center point, preventing the situation where the data distribution is all negative or all positive at a certain time while keeping the original data distribution's shape and the mean value of all pixels after zero averaging zero. This makes the model output more stable by facilitating the convergence of the model weights during the back-propagation process.

$$\widetilde{y}_i = y_i - \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{4}$$

Therefore, to retain the model performance on the previous task during the learning of the new task, we perform multinetwork distillation on the backbone network, RPN, and RoI head network and add a mean value strategy to remember past information.

3.3.1. Backbone Distillation Losses. F is the layer containing the extracted object pixel features from the image, and F contains each feature pixel f_i. To obtain the object features associated with the old and new classes, a distillation loss constraint needs to be applied to $\mathbb{M}_{\text{Backbone}}$. We learn $\mathbb{M}^{\theta_t}_{\text{Backbone}}$ by freezing the weight parameters of $\mathbb{M}^{\theta_{t-1}}_{\text{Backbone}}$ and using the teacher network to teach the student network. For the same input I, the teacher network and the student network obtain outputs F_{te} and F_{st}, respectively. Furthermore, $\mathbb{M}_{\text{Backbone}}$ serves as input to the subsequent classification and regression steps, and an accurate description of the old and new class features is particularly important for the subsequent steps, so we strengthen the distillation of $\mathbb{M}_{\text{Backbone}}$. In addition, for faster convergence, we obtain $\widetilde{F}_{\text{te}}$ and $\widetilde{F}_{\text{st}}$ by zero averaging the features obtained by equation (4). Distillation loss is defined as follows:

$$\mathscr{L}_{\text{Backbone\_dis}} = \min \Sigma\left(\widetilde{f}_{\text{te},i} - \widetilde{f}_{\text{st},i}\right)^2, \tag{5}$$

where $\widetilde{f}_{\text{te},i}$ and $\widetilde{f}_{\text{st},i}$ are defined as each of the pixel features in $\widetilde{F}_{\text{te}}$ and $\widetilde{F}_{\text{st}}$, respectively, $\widetilde{f}_{\text{te},i} \in \widetilde{F}_{\text{te}}$, and $\widetilde{f}_{\text{st},i} \in \widetilde{F}_{\text{st}}$.

3.3.2. RPN Distillation Loss. $\mathbb{M}_{\text{RPN}}$ suggests regions $r = (r_1, r_2, \ldots, r_j)$ for the old and new class features that I extracted by $\mathbb{M}_{\text{Backbone}}$ in the previous stage and determines whether the corresponding region has a class score $o = (o_1, o_2, \ldots, o_i)$. As the first stage of the two-stage target detector, whether the proposed region network extracted by $\mathbb{M}_{\text{RPN}}$ contains old and new classes is particularly important for the next stage of $\mathbb{M}_{\text{RoI\_Head}}$ in the classification and regression of the old and new classes; however, if the distillation constraint on $\mathbb{M}_{\text{RPN}}$ is excessive, it will lead to an increase in $\mathbb{M}_{\text{RPN}}$'s focus on the old class member and affect the learning process of the new task, so we adopt the idea of Peng et al. [10], who suggested using the teacher network as a lower bound to adaptively choose whether to apply distillation constraints to $\mathbb{M}_{\text{RPN}}$. In contrast, we subject the output scores from $\mathbb{M}^{\text{te}}_{\text{RPN}}$ and $\mathbb{M}^{\text{st}}_{\text{RPN}}$ to distillation softening in equation (3) and zero averaging in equation (4) and use the KL dispersion loss as the classification loss. We determine the value on each dimension regarding l for the anchor regression of $\mathbb{M}_{\text{RPN}}$, and we regulate the regression by setting a threshold ξ. We take the empirical value $\xi = 1$ and use the sum of the o value of $\mathbb{M}^{\text{te}}_{\text{RPN}}$ over $\mathbb{M}^{\text{st}}_{\text{RPN}}$ and ξ as the activation value in the distillation loss calculation; however, higher values of $\mathbb{M}^{\text{st}}_{\text{RPN}}$ may be more important for new task learning and are therefore not involved in the distillation loss calculation. The definition of RPN distillation loss is as follows:

$$\mathscr{L}_{\text{RPN\_dis}} = \min \frac{1}{\mathscr{N}} \begin{cases} \left[\sum_i \widetilde{o}^T_{\text{te},i} \log \frac{\widetilde{o}^T_{\text{te},i}}{\widetilde{o}^T_{\text{st},i}} + \sum_j \mathscr{P}_j \left(r_{\text{te},j} - r_{\text{st},j}\right)^2\right], & \text{if } \widetilde{o}^T_{\text{te},i} > \widetilde{o}^T_{\text{st},i}, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

$$\mathscr{P}_j = \begin{cases} 1, & \text{if } \widetilde{o}^T_{\text{te},i} > \left(\widetilde{o}^T_{\text{st},i} + \xi\right), \\ 0, & \text{otherwise,} \end{cases}$$

where $\widetilde{o}^T_{\text{st},i}$ and $\widetilde{o}^T_{\text{te},i}$ are the scores of the output of $\mathbb{M}_{\text{RPN}}$ successively after the zero averaging treatment of equation (4) and the distillation softening treatment of equation (3), respectively, and the empirical value is $T = 6$ in our experiments. $\mathscr{N}$ is the total number of anchors.

3.3.3. RoI Head Distillation Loss. $\mathbb{M}_{\text{RPN}}$ generates proposals for the old and new classes passed through pooling $\mathbb{M}_{\text{RoI\_Head}}$ to obtain the final classification probabilities $(p_{\text{te}}, p_{\text{st}})$ and border regression values $(l_{\text{te}}, l_{\text{st}})$ for the teacher and student networks. In our approach, we focus more on the classification and regression of the old classes, considering $\mathbb{M}_{\text{RoI\_Head}}$ as the final stage of the two-stage object detector. In addition to the normal distillation loss calculation, we calculate the mean of each channel with respect to the class by equation (7) to increase $\mathbb{M}_{\text{RoI\_Head}}$ and to focus on the overall trend of the final classification probability and border regression. The temperature factor T is also introduced into

the log output of equation (3) to soften the "softmax" output and obtain more information about past knowledge.

$$\overline{h}_i = \frac{1}{\mathscr{C}} \sum_{j=1}^{\mathscr{C}} h_{i,j}, \tag{7}$$

where $\mathscr{C}$ denotes the number of channels and i denotes the i-th parameter of the j-th channel. The $\mathbb{M}_{\text{RoI\_Head}}$ loss is therefore defined as follows:

$$\begin{aligned}\mathscr{L}_{\text{RoI\_dis}} = \min\Bigg[&\frac{1}{2}\left(\sum_i \overline{p}^T_{\text{te},i} \log \frac{\overline{p}^T_{\text{te},i}}{\overline{p}^T_{\text{st},i}} + \sum_i p^T_{\text{te},i} \log \frac{p^T_{\text{te},i}}{p^T_{\text{st},i}}\right) \\ &+ \frac{1}{2}\left(\sum_j \left(\overline{l}_{\text{te},j} - \overline{l}_{\text{st},j}\right)^2 + \sum_j \left(l_{\text{te},j} - l_{\text{st},j}\right)^2\right),\end{aligned} \tag{8}$$

where $\overline{p}^T_{\text{te},i}$, $\overline{p}^T_{\text{st},i}$, $\overline{l}_{\text{te},j}$, and $\overline{l}_{\text{st},j}$ are all variables processed by the mean value of equation (7). The parameters labeled with T are all variables processed through equation (1).

3.4. Total Mission Loss in the First Phase. The first stage of our model's learning process for the task parameters can be characterized as learning the current task through each stage, followed by correcting the model parameters to maintain past task performance through the distillation loss in each stage. This allows the loss of the overall task to be defined as a linear combination of the loss of the new task and the multinetwork mean distillation loss. To balance the model performance in the past and present tasks, we employ a convex combination similar to that in [5] to set the stability and plasticity trade-off parameter α. Here, the total loss in the first stage is defined as follows:

$$\begin{aligned}\mathscr{L}_{\text{all\_task}} = &(1-\alpha)\left(\mathscr{L}_{\text{RoI\_Head}} + \mathscr{L}_{\text{RPN}}\right) \\ &+ \alpha\left(\mathscr{L}_{\text{Backbone\_dis}} + \mathscr{L}_{\text{RPN\_dis}} + \mathscr{L}_{\text{RoI\_dis}}\right),\end{aligned} \tag{9}$$

where α is defined as 0.1 in our experiments and is defined in Section 4.3.

3.5. Gradient Matrix Warp Loss. For the second phase of learning, the warp parameter ω, as depicted in Figure 2, is configured in the network warp layer in $\mathbb{M}_{\text{RoI\_Head}}$ to learn the preprocessing matrix $(P(\theta; \varnothing))$. By establishing an image store, a small number of images are saved for each class during the distillation learning procedure I_{store}. Images are taken from I_{store} and put in the set feature store F_{store}.

In the distillation learning process, a small number of images are stored for each class by setting up an image store I_{store}, and the images stored in I_{store} are stored in the set feature store F_{store} after feature extraction by $\mathbb{M}_{\text{Backbone}}$. Notably, F_{store} defines a fixed size queue N_{feat} for each class to mitigate the class imbalance problem. The stored queue characteristics are incorporated directly into the task learner by utilizing the meta-learning parameterization of $P(\theta; \varnothing)$, which warps the gradient toward the steepest direction and enables the parameters to be updated in the most suitable direction for different learning tasks.

Each image in I_{store} is passed through $\mathbb{M}_{\text{Backbone}}$ and $\mathbb{M}_{\text{RPN}}$ to generate the RoI pooled features and associated labels, which are then queued into F_{store}. Let f be the RoI pooled features, where f generates the predicted classification value p and the border prediction $\boldsymbol{l}$ through $\mathbb{M}_{\text{RoI\_Head}}$. The warp loss can then be calculated from the features and labels stored in F_{store}, and l_{wrap} is calculated as follows:

$$\begin{aligned}&\mathscr{L}_{\text{warp}} = \sum_{(\mathbf{f}, p^*, l^*) \sim F_{\text{Store}}} \left(\mathscr{L}_{\text{cls}}(\mathbf{p}, p^*) + [p^* \geq 1]\mathscr{L}_{\text{loc}}(\mathbf{l}, \mathbf{l}^*)\right), \\ &\text{s.t., } (\mathbf{p}, \mathbf{l}) = \mathbb{M}_{\text{RoI\_Head}}(\mathbf{f}),\end{aligned} \tag{10}$$

where L_{cls} is the log regression loss and L_{loc} is the smooth $L1$ loss.

4. Experimental Analysis

4.1. Datasets and Evaluation Metrics. We evaluated our method on the PASCAL VOC 2007 [41] and MS COCO 2014 [42] datasets. PASCAL VOC 2007 contains 9963 images with 24640 instances of annotations and 20 classes. According to the setup in [41], 50% of the dataset is divided into training and validation sets, and the rest is used for testing. MS COCO 2014 contains objects from 80 classes with 83,000 images in the training set and 41,000 images in the validation set.

For the evaluation metrics, the average accuracy of the 50% IOU threshold (mAP@50) was used as the main evaluation metric for both datasets. For MS COCO, we set multiple IOUs (AP, AP50, and AP-0.75) and sizes (APs: small, APm: medium, and APl: large) as evaluation metrics.

4.2. Incremental Experimental Scenario Setting. Similar to [5], we simulate incremental scenarios for PASCAL VOC and MS COCO, where the dataset D_t provides a set of selected classes C to be used for task T_t and passed to the learner at moment t. For each image in the dataset D_t that may contain multiple classes, one or several classes belonging to C will be learned as the task object class, and those instances of classes that do not belong to C will not be marked for learning.

According to the different difficulty levels of the classification task, we considered the effect of the learning intensity of initial base class tasks and incremental tasks on the model output results and defined class flow tasks and batch tasks. Class flow tasks can be interpreted as having incremental tasks T_i that flow into the model after learning the base class task T_0 with 1 to 2 additional classes per task, whereas batch tasks contain only one incremental task with 1 to 2 additional classes. As indicated in Table 1, we devised seven incremental scenarios based on the degree of difficulty of the incremental experiment scenarios according to the divided class flow tasks and batch tasks. The dataset used in experiments *a* to *f* includes the first 20 classes of PASCAL VOC and the dataset used in experiment *g* includes 80 classes from the MS COCO dataset.

4.3. Discussion and Analysis

4.3.1. Stability Analysis of the Zero Mean. To validate the influence of the zero mean on the stability of the model output, we conducted five consecutive replications of experiments (*d*)

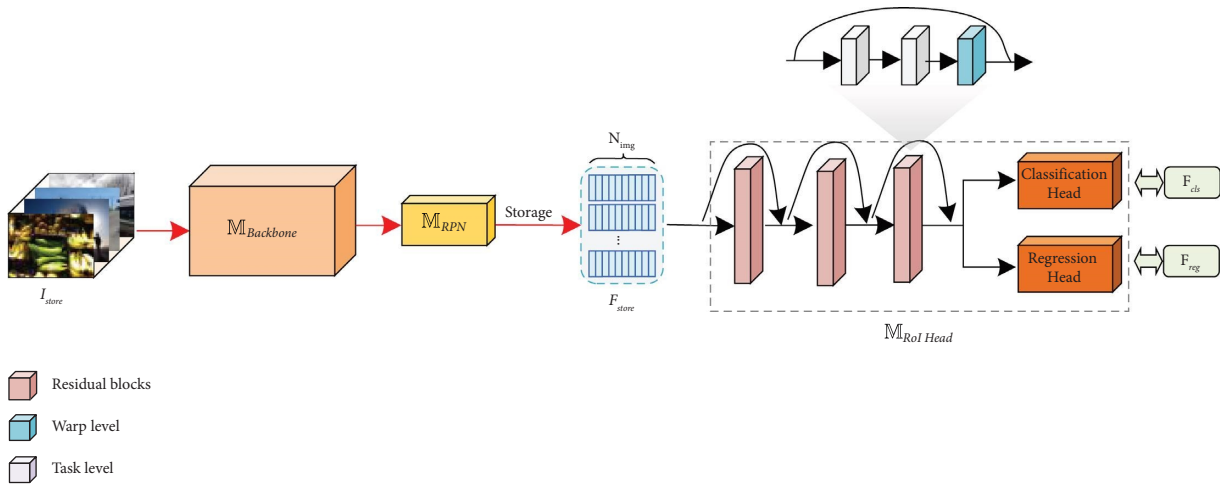

FIGURE 2: Gradient meta-learning mechanism (the images included within I_{store} were inputted into the student network via the I_{store}. During this process, all network weights, except for those in the warp layer of the RoI head network, were kept fixed. This was done to enhance the model's performance on the new task using the newly acquired image data).

TABLE 1: Experimental setup.

Task	Experiment	Number of classes in T_0	Incremental tasks T_i	Number of classes in T_i
Class_flow tasks	a	10	$T_1, \ldots, T_{10}$	1
	b	10	$T_1, \ldots, T_5$	2
	c	15	$T_1, \ldots, T_5$	1
Batch tasks	d	10	T_1	10
	e	15	T_1	5
	f	19	T_1	1
	g	40	T_1	40

through (f) and created box plots, as depicted in Figure 3, based on the output data. As seen from the figure, in experiment (d), our method has a more obvious advantage; its lowest mAP50 value is 64.7%, but this value is higher than the highest value of 63.63% obtained by the iOD method. In experiment (e), our approach's stability is not only comparable to but substantially superior to that of the iOD method, which exhibits an outlier (62.68%). In experiment (f), the gap between our method and iOD in terms of stability is wider, and the stability of our model is still better than that of iOD, although the maximum value of 68.28% obtained by iOD is higher than the lowest value of 68.18% obtained by our method in terms of accuracy. This indicates that in the incremental task containing only one class, the gap between different methods fluctuates slightly, but the results from Figure 3(c) show that our method still outperforms iOD in terms of overall accuracy. The results of the stability experiments reported in Figure 3 show that our method outperforms the iOD method in terms of output stability for all iOD methods without a zero mean, and the overall accuracy of the five experiments in all three scenarios is higher than that of the iOD method, which fully demonstrates the reliability and stability of our method.

To search for the optimal stability equilibrium parameter α of equation (9), we conducted several experiments on the value of α based on the incremental task approach of experiment (d). Table 2 displays the individual experimental outcomes. The table demonstrates that as α gradually increases, our model's experimental accuracy gradually declines from 65.0% to 60.7%. Based on the experimental results, we ultimately set the value to 0.1 in all tests.

4.3.2. Ablation Experiment. To confirm the increase in accuracy of the approaches introduced by our method (the RPN and zero mean), we carried out ablation experiments. As shown in Table 3, we still used the task form based on the incremental experiment (d). The results of the experiment are the mAP values of the base class T_0 (first 10 classes), the incremental task T_1 (last 10 classes), and the overall 20 classes. As seen from the table, when only zero averaging is introduced, the learning ability of the new task is improved, and the mAP of T_1 reaches 67.33%. When RPN adaptive distillation loss is introduced, the stability of past knowledge is improved, and the mAP of T_0 reaches 62.20%. When both strategies are implemented, the highest mAP values are attained for the new task T_1, and all 20 classes (68.28% and 65.00%, respectively), and the experimental results are consistent with the conclusions of our theoretical analysis in Section 3.3. In regard to the remaining two approaches [9, 10], it is worth noting that Faster ILOD [10] and the method by Shmelkov et al. [9] exhibit a comparative advantage in preserving performance on previous tasks. However, in practical production scenarios, emphasis is placed on the significance of new tasks. Consequently, our proposed method not only enhances the model's performance on new tasks but also

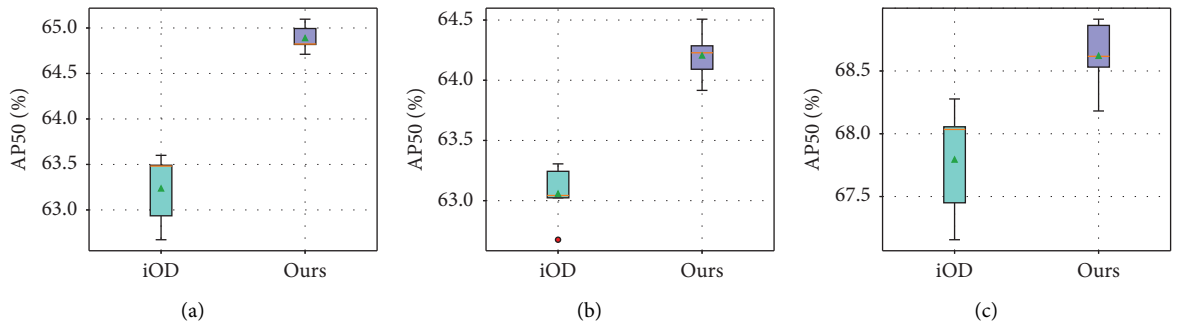

Figure 3: Stable output experiment (the height of the box plot represents the stability of the model output). (a) Experiment (*d*) stability experiment. (b) Experiment (*e*) stability experiment. (c) Experiment (*f*) stability experiment.

Table 2: α stability analysis (%).

α	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	T_0	
0.1	64.4	69.4	58.1	42.0	50.1	69.0	82.9	68.9	47.8	64.5	61.7	
0.2	66.1	70.0	61.4	41.0	46.9	70.8	78.6	73.1	41.9	72.7	62.3	
0.4	64.0	69.9	60.9	39.2	46.5	71.7	77.6		42.7	72.0	61.7	
0.6	67.7	69.5	61.6	41.5	48.9	73.1	79.4	73.6	46.0	70.2	63.2	
0.8	68.7	69.9	63.2	43.3	50.4	73.1	79.2	71.0	45.4	66.1	63.0	
A	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	tv	T_1	All
0.1	58.1	70.7	79.1	75.0	82.6	45.5	66.2	69.4	68.4	67.8	68.3	65.0
0.2	57.4	73.6	78.8	74.5	79.5	43.1	60.1	67.5	70.6	65.5	67.1	64.7
0.4	56.7	73.3	79.5	74.6	80.1	44.3	58.7	66.3	70.8	67.8	67.2	64.4
0.6	50.3	71.9	75.5	70.9	75.0	40.8	59.0	64.4	64.2	64.0	63.6	63.4
0.8	41.9	67.5	67.7	67.0	70.8	34.5	55.9	56.6	60.4	61.1	58.3	60.7

Table 3: Ablation experiments (%).

Methods/experiment		RPN	Zero_meaning	T_0	T_1	All
Faster ILOD [10]		—	—	69.76	54.47	62.12
Shmelkov et al. [10]		—	—	63.16	63.14	63.15
Experiment	1	×	×	57.77	65.32	61.55
	2	×	√	61.50	67.33	64.41
	3	√	×	62.20	66.30	64.25
	4	√	√	61.71	68.28	65.00

maintains a certain level of performance on previous tasks. This improvement results in a more substantial enhancement of overall task performance compared to the aforementioned approaches [10, 11], with superior performance demonstrated on all tasks. Furthermore, the results from Experiment 3 and Experiment 4 in Table 3 demonstrate that the inclusion of the zero mean in Experiment 3 leads to a decrease of approximately 0.5% in T_0 accuracy in Experiment 4, while the T_1 accuracy shows an improvement of approximately 2%. This improvement can be attributed to the utilization of the RPN adaptive distillation loss, which enhances the model's focus on the old task (T_0: 62.20% in Experiment 3). The introduction of zero mean learning causes the model to focus on the new input data, thereby allowing the model weights to be better adapted to the new task during distillation computation, resulting in a 2% enhancement in the performance of the new task. However, importantly, the performance of the old task is still maintained to some extent, albeit with a decrease of 0.5%. In comparison to Faster ILOD [10] and the model by Shmelkov et al. [9], our method achieves superior performance in T_1, surpassing them by 13.81% and 5.14%, respectively. In addition, our method outperforms both approaches in terms of overall performance, surpassing them by 2.88% and 1.85%, respectively.

To provide additional evidence for the effectiveness of our approach, we conducted a more detailed analysis of the incremental learning phase in the ablation experiments (refer to Table 4). The results indicate that the average precision (AP) values obtained by the model for identifying the old classes during the incremental class learning phase exceed 9.09. This can be attributed to the model's ability to generate probability distributions that are highly similar for these classes while still exhibiting subtle differences that enable partial identification of the old classes. This phenomenon may arise from the inherent limitations of the student model in accurately replicating the probability distribution of the teacher model during the knowledge distillation procedure. The student model can only approximate the output distribution of the teacher model.

Table 4: Ablation comparison experiments in incremental learning stages (%).

Experiments	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	T_0	
1	0.00	9.09	0.00	0.00	9.09	0.00	9.09	9.09	9.09	0.00	4.55	
2	9.09	9.09	0.00	3.22	9.09	0.62	9.09	8.08	9.09	9.09	6.65	
3	9.09	9.09	6.06	4.06	9.09	0.00	9.09	9.09	9.09	9.09	7.38	
4	9.09	9.09	9.09	0.00	9.09	0.00	9.09	9.09	9.09	9.09	7.27	
Experiments	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	tv	T_1	All
1	60.21	71.85	80.81	76.25	84.69	50.05	74.70	71.60	69.25	65.29	70.47	37.51
2	62.81	72.58	82.15	76.29	86.56	49.86	76.61	73.63	69.78	69.79	72.01	39.33
3	60.54	70.09	81.95	75.94	85.81	49.29	75.92	72.39	70.69	68.91	71.15	39.26
4	61.22	71.56	82.40	77.38	86.20	51.47	75.59	74.43	72.40	70.03	72.27	39.77

Hence, as depicted in Table 4, the incorporation of RPN adaptive distillation (Experiment 3) during the incremental learning phase enables the model to effectively recognize the old classes, resulting in the identification of 9 classes. Moreover, the overall accuracy of recognizing the old classes is enhanced by 2.83% compared to the mAP value obtained in Experiment 1, incorporating the zero mean (Experiment 4) results in an enhancement of the model's performance on the new task, reaching 72.27. Moreover, the performance on the old task experienced only a marginal decrease of 0.11%. This outcome indicates that our approach successfully achieves a better trade-off between plasticity and stability, effectively addressing the requirements of both maintaining performance on the old task and facilitating adaptation to the new task.

4.4. Analysis of the Experimental Results of Incremental Object Detection. We employ stochastic gradient descent (SGD) with 0.9 momentum. The initial learning rate is set to 0.02 and is then decreased to 0.0002. Each job receives 18,000 iterations of base-class training on the PASCAL VOC dataset, followed by 100 iterations per image and a total of 90,000 iterations for each of the two tasks. For 2080Ti, the model training procedure is executed on a single GPU, and since each GPU simultaneously analyses two images, the batch size is two. The N_{feat} and N_{img} queue sizes of the feature store F_{store} and image store I_{store} are set to 10. The evaluation process considers 100 detections per image, and the NMS threshold is 0.4. The coefficient of stability α is 0.1.

4.4.1. Class Flow Tasks. Incremental simulations, in which the model learns the top 10 or first 15 classes from the PASCAL VOC dataset as base classes, are performed, and the detector is fed one or two classes at a time. Tables 4–6 show the experimental results for the class flow task. The first row displays the joint learning of 20 classes as the incremental learning upper bound; the second row displays the model learning the base classes, where the base class is the first 10 classes in experiments (*a*) and (*b*) and the first 15 classes in experiment (*b*); and the following rows display each class in turn according to its ordinal number in the class flow task. The table shows the change in mAP values for all classes as well as the AP values of each class incrementally for each task. Figure 4 depicts the trend in the effects of each incremental task carried out by the model during the class flow incremental task on the base class, the old class, and all classes.

As seen from the data in Table 5, in the increment scenario of experiment (*a*), we set the base class $T_0 = 10$, and our mAP value is higher than that of the iOD method for all class increment processes. However, the overall task mAP value gradually decreases with the input of T_i at each increment step when $T_i = 1$. The largest mAP difference can reach 3.6%, while the average mAP value is 2.36% larger than that of the iOD as the class increments increase. The unique changing process of experiment (*a*) is depicted in Figure 4(a). As shown, the difference in mAP between our model and the base class, old class, and all classes steadily widens as T_i is added, demonstrating that our model is more stable in the case of an incremental task flow. In the incremental scenario of experiment (*b*), we enhanced the complexity of the incremental task by setting T_i to 2. Table 6 shows that as the difficulty of the incremental task increases, our model has a clear advantage over experiment (*a*) in terms of the overall accuracy, with an average mAP difference of 4.5%. In the details of experiment (*b*) depicted in Figure 4(b), the gaps in mAP between our model and experiment (*a*) for the base class, old class, and all classes are more pronounced. Specifically, in learning the T_2 task, the all-class gap reaches 5.1%. In comparison to experiment (*a*), it can be observed that as the incremental task learning difficulty grows, the mAP value of the model for the overall task declines at a faster rate. Experiment (*c*) enhances the difficulty of learning basic classes by increasing T_0 to 15 and T_i to 1. When the number of base classes learned increases to 15 classes, the partial gap of our models' mAP over that of iOD gradually increases with the learning of new classes, reaching a maximum of 3.9% in Table 7. In the details of experiment (*c*) shown in Figure 4(c), the mAP of our model is better than that of iOD for the base class, old class, and all classes. In the class flow task experiment, we can observe that relative to $T_i = 1$ in experiment (*a*), when the difficulty of learning T_i is increased, as in experiment (*b*), the model learning effect decreases significantly for each learned T_i, as depicted in Figure 4(b); however, the final mAP of the all-class scenario is comparable to that obtained in experiment a and remains at 46.9%. Compared with $T_0 = 10$ learned classes in experiment (*a*), increasing the base class task learning difficulty, as in experiment (*c*), decreases the learning effect of the very first task while keeping the number of total tasks and T_i learned classes constant. However, as the incremental task stream decreases, the model's final learning effect is better than that of the class stream with many tasks, as in Figure 4(c), and it remains at 54.8%.

Table 5: Experiment (*a*) number of base classes 10, number of incremental task learning classes 1 (%).

Class order	Methods	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	tv	mAp
1~20	—	79	81.4	73.8	54.2	61	79.1	90.7	78.6	53.6	81.5	64.1	77.8	80.9	77.4	82.7	47.3	68.1	68	77.1	72.2	72.4
1~10	—	74.5	78.4	68.4	52.5	59.3	75.5	84.7	77.9	53.7	81.9	—	—	—	—	—	—	—	—	—	—	70.7
+11	iOD	74.9	77.7	70.3	49.4	56.9	74.0	84.1	82.4	52.0	82.6	63.6	—	—	—	—	—	—	—	—	—	69.8
	Ours	75.7	77.7	70.2	46.1	60.7	73.9	84.4	82.0	51.0	82.8	63.5	—	—	—	—	—	—	—	—	—	69.8
+12	iOD	64.9	75.8	62.9	42.2	55.1	73.0	78.2	74.0	45.3	74.2	55.6	76.8	—	—	—	—	—	—	—	—	64.8
	Ours	66.6	74.6	64.3	43.0	55.4	73.3	82.2	76.2	46.2	74.7	59.3	78.0	—	—	—	—	—	—	—	—	66.1
+13	iOD	58.7	71.7	56.8	39.8	52.8	71.3	77.5	73.0	42.2	69.2	52.7	71.4	79.1	—	—	—	—	—	—	—	62.8
	Ours	63.2	69.3	61.4	41.6	53.1	71.6	78.1	75.1	43.7	68.0	56.6	74.6	80.8	—	—	—	—	—	—	—	64.4
+14	iOD	54.9	67.6	53.0	36.3	48.8	68.9	75.9	64.2	41.2	69.1	53.4	69.8	77.4	72.7	—	—	—	—	—	—	60.9
	Ours	56.8	70.7	56.0	39.5	48.6	70.3	77.0	70.2	42.8	68.7	53.2	72.3	78.7	72.0	—	—	—	—	—	—	62.6
+15	iOD	57.8	68.9	50.6	36.4	47.2	67.7	75.4	66.7	36.2	64.8	55.1	67.3	75.4	70.6	74.5	—	—	—	—	—	61.0
	Ours	60.5	71.5	54.1	40.6	49.0	70.0	77.1	68.4	40.5	66.7	52.1	69.8	78.1	71.6	74.6	—	—	—	—	—	63.0
+16	iOD	48.2	63.6	43.4	29.7	37.4	60.3	70.5	61.7	35.4	63.5	51.9	61.7	73.0	67.4	72.1	30.5	—	—	—	—	54.4
	Ours	55.3	64.8	47.9	33.7	44.0	64.9	75.4	63.7	38.2	66.9	52.0	67.0	75.6	69.9	72.7	32.3	—	—	—	—	57.8
+17	iOD	44.4	60.8	39.5	28.2	34.8	56.0	69.0	52.3	30.5	53.6	44.8	54.8	71.9	64.1	70.2	29.5	37.8	—	—	—	49.5
	Ours	49.7	63.9	39.9	31.4	39.8	61.1	73.2	55.2	32.7	56.0	48.1	65.1	73.1	65.8	71.0	33.5	36.9	—	—	—	52.7
+18	iOD	44.1	57.3	37.0	26.3	33.0	51.7	69.0	51.2	28.3	50.2	41.8	55.0	65.9	60.5	68.2	25.9	36.0	37.9	—	—	46.6
	Ours	48.2	64.0	38.5	28.8	39.6	60.8	69.7	52.7	30.3	54.3	43.4	59.7	69.5	63.6	71.2	29.6	38.0	42.6	—	—	50.2
+19	iOD	44.3	56.1	34.5	23.8	32.4	56.1	68.6	45.4	27.0	47.8	39.8	52.5	63.9	60.1	66.0	24.2	38.3	37.6	44.5	—	45.4
	Ours	45.7	59.8	36.3	28.1	37.7	59.6	69.4	51.9	28.4	53.1	42.4	56.5	65.5	63.3	67.5	26.7	40.3	43.5	46.0	—	48.5
+20	iOD	38.2	53.9	34.1	24.5	25.2	52.1	64.2	46.1	26.2	48.3	34.9	47.1	63.6	58.2	64.9	24.7	35.7	31.2	37.9	50.1	43.1
	Ours	44.5	57.5	34.3	27.7	32.2	57.1	68.1	49.0	28.7	53.0	41.9	53.2	64.8	62.1	65.8	25.9	37.5	38.3	40.6	52.8	46.8

Table 6: Experiment (*b*) number of base classes 10 and number of incremental tasks learning classes 2 (%).

Class order	Methods	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	tv	mAp
1–20	—	79	81.4	73.8	54.2	61	79.1	90.7	78.6	53.6	81.5	64.1	77.8	80.9	77.4	82.7	47.3	68.1	68	77.1	72.2	72.4
1–10	—	74.5	78.4	68.4	52.5	59.3	75.5	84.7	77.9	53.7	81.9	—	—	—	—	—	—	—	—	—	—	70.7
10–12	iOD	66.6	74.0	57.2	42.4	55.8	75.1	78.7	69.6	43.7	70.6	49.4	62.5	—	—	—	—	—	—	—	—	62.1
	Ours	73.7	76.2	63.4	45.4	57.7	75.2	83.2	72.6	44.1	73.5	55.7	68.1	—	—	—	—	—	—	—	—	**65.7**
12–14	iOD	57.6	60.5	49.8	37.5	43.4	61.0	76.1	60.8	36.4	45.3	32.4	52.1	56.9	65.1	—	—	—	—	—	—	52.5
	Ours	64.0	64.0	56.2	42.2	50.4	67.6	77.6	65.6	40.6	52.9	41.2	58.9	60.1	64.7	—	—	—	—	—	—	**57.6**
14–16	iOD	52.9	58.9	42.9	34.5	33.0	62.7	72.0	58.5	35.0	48.6	29.7	50.0	56.4	63.3	70.3	32.2	—	—	—	—	50.1
	Ours	57.7	64.6	45.1	37.0	45.4	66.4	76.4	62.5	37.8	58.8	34.5	56.5	58.7	68.2	70.7	34.1	—	—	—	—	**54.7**
16–18	iOD	48.2	54.1	36.1	26.7	35.2	59.1	68.7	51.4	29.8	47.6	26.5	48.1	57.0	58.0	67.8	27.9	34.6	46.7	—	—	45.7
	Ours	50.6	63.1	43.6	33.6	42.6	64.3	75.9	58.4	32.7	55.4	33.1	50.7	58.4	62.3	68.1	30.7	34.3	45.6	—	—	**50.2**
18–20	iOD	42.6	51.0	33.4	25.9	28.2	53.1	65.4	49.0	28.2	47.1	25.4	43.4	53.5	55.6	62.8	27.0	33.0	43.5	30.0	46.4	42.2
	Ours	44.8	59.6	41.3	32.0	37.7	61.8	68.4	54.6	31.8	55.2	30.0	46.1	55.3	61.3	64.7	30.6	34.4	48.5	36.1	44.5	**46.9**

The bold values represent the mAP optimal values of iOD compared with our method.

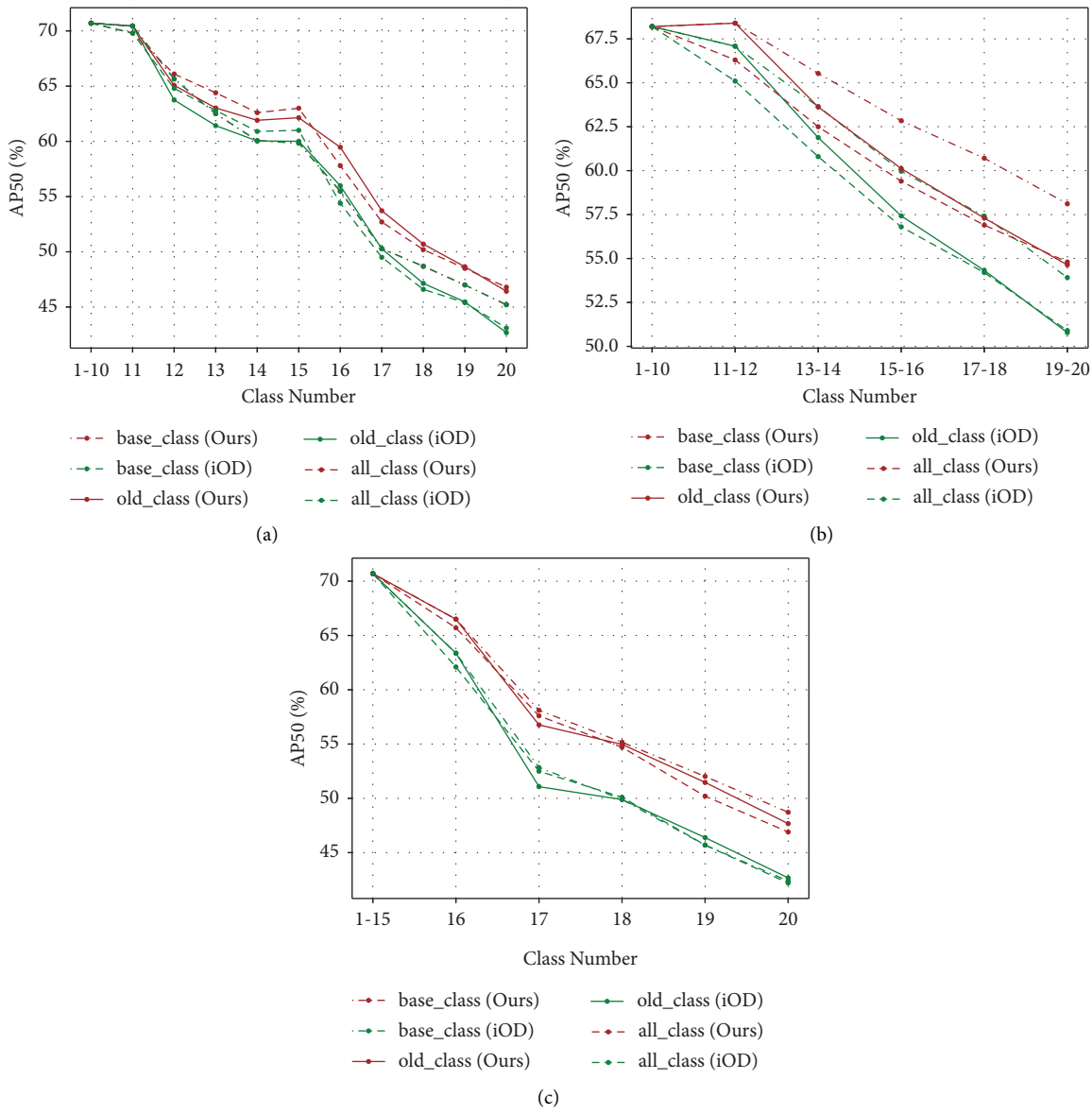

Figure 4: The effect of class flow incremental task learning on base classes, older classes, and all classes. (a) 10-1++. (b) 10-2++. (c) 15-1++.

4.4.2. Batch Tasks. In the batch task settings, we considered class batch learning of the model on the PASCAL VOC and MS COCO datasets with different numbers of base classes and incremental classes. T_0 validates the accuracy of our method.

Table 8 shows our results on the COCO dataset for the experiment (g). Specifically, we set up incremental scenarios with $T_0 = 40$ and $T_1 = 40$ and used the standard COCO dataset evaluation method with multiple IOU metrics (AP, AP50, and AP-0.75) and sizes (Aps: small, Apm: medium, and Apl: large) for a comprehensive evaluation. As seen in Table 9, our model continues to show excellent performance even for the high-volume class learning scenarios in the complex COCO dataset. It outperforms the iOD technique by more than 2% across all scales of evaluation, and its AP50 performance is 4.7% greater than that of the iOD method.

On the PASCAL VOC dataset, we report the results compared with those of the model by Shmelkov et al. [9], Faster ILOD [10], and iOD [5] in terms of mAP, while on the MS COCO dataset, we compare our results with those of iOD [5] with the standard COCO dataset evaluation method. Tables 9–11 show the experimental results of our comparisons.

In experiment (d), we set up a batch task increment scenario with T_0: $T_1 = 10:10$. Table 9 reveals that our model achieved the best learning effect in all experiments compared to the other methods; the mAP score reached 65.0%, and the best learning effect was that on the new tasks, where mAP reached 68.3%. Our method is also superior to iOD in the maintenance of old task performance, with the old class mAP reaching 61.0%.

In experiment (e), we adjusted the class learning ratio of T_0 and T_1 (T_0: $T_1 = 15:5$). In the results of experiment (e)

Table 7: Experiment (*c*) number of base classes 15 and number of incremental task learning classes 1 (%).

Class order	Methods	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	tv	mAp
1–20	—	79	81.4	73.8	54.2	61	79.1	90.7	78.6	53.6	81.5	64.1	77.8	80.9	77.4	82.7	47.3	68.1	68	77.1	72.2	72.4
1–15	—	71.5	76.1	67.5	46.7	56.6	69.4	84.4	74.4	50.3	74.5	53.8	69.2	77.5	73.2	78.3	—	—	—	—	—	68.2
+16	iOD	72.1	75.2	64.3	48.5	49.2	67.6	82.5	74.1	46.8	68.9	55.9	76.2	77.7	71.7	75.5	34.8	—	—	—	—	65.1
	Ours	76.7	76.5	67.0	50.0	53.1	66.9	82.9	75.0	47.3	68.4	57.3	76.5	79.2	73.6	75.5	34.7		—	—	—	**66.3**
+17	iOD	63.1	71.8	57.4	48.5	47.3	67.6	78.6	68.1	41.0	67.6	54.9	69.3	75.7	69.7	74.1	35.4	43.6	—	—	—	60.8
	Ours	67.5	75.6	59.1	49.4	50.2	69.0	78.8	70.9	41.7	69.5	56.3	70.7	76.9	72.9	74.5	35.0	44.6	—	—	—	**62.5**
+18	iOD	59.0	66.9	54.9	41.9	41.5	66.0	77.5	64.5	38.5	65.2	47.1	65.8	71.9	66.7	72.1	32.9	43.8	46.6	—	—	56.8
	Ours	63.2	73.7	58.5	43.6	47.7	69.4	78.1	65.9	37.8	68.4	52.5	68.2	74.0	68.2	73.4	32.9	46.5	46.6	—	—	**59.4**
+19	iOD	52.6	64.6	49.8	36.7	40.2	66.1	76.5	64.2	36.3	62.1	46.8	61.1	68.6	64.5	71.1	30.2	42.4	44.0	52.7	—	54.2
	Ours	56.5	68.2	54.8	39.5	42.6	69.6	77.8	70.4	37.1	64.4	48.1	66.6	73.1	69.6	72.3	30.6	44.1	46.1	49.1	—	**56.9**
+20	iOD	51.1	62.2	45.9	32.4	34.2	63.4	73.8	60.2	33.0	58.3	42.5	56.9	65.6	63.7	65.5	27.1	37.0	43.8	48.3	53.4	50.9
	Ours	54.3	66.1	52.5	35.3	40.2	67.1	76.2	64.4	34.7	64.0	46.7	60.8	72.1	66.3	71.0	27.6	39.8	47.0	52.0	57.0	**54.8**

The bold values represent the mAP optimal values of iOD compared with our method.

Table 8: Experiment *g*: COCO increment experiment (%).

Methods	AP	AP50	AP75	Aps	Apm	Apl
All 80	33.1	53.1	35.3	15.9	36.9	44.0
iOD	18.0	33.0	18.0	7.3	20.4	25.2
Ours	20.0	37.7	19.4	9.0	22.9	28.0

Table 9: Experiment (d): 10-10.

Methods	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	T_0	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	tv	mAp
1-20-	79	81.4	73.8	54.2	61	79.1	90.7	78.6	53.6	81.5	—	64.1	77.8	80.9	77.4	82.7	47.3	68.1	68	77.1	72.2	72.4
1-10-	74.5	78.4	68.4	52.5	59.3	75.5	84.7	77.9	53.7	81.9	70.7	—	—	—	—	—	—	—	—	—	—	70.7
Shmelkov et al.	69.9	70.4	69.4	54.3	48.0	68.7	78.9	68.4	45.5	58.1	63.2	59.7	72.7	73.5	73.2	66.3	29.5	63.4	61.6	69.3	62.2	63.1
Faster ILOD	72.8	75.7	71.2	60.5	61.7	70.4	83.3	76.6	53.1	72.3	69.8	36.7	70.9	66.8	67.6	66.1	24.7	63.1	48.1	57.1	43.6	62.2
iOD	60.4	65.8	53.9	40.2	37.5	66.5	80	66.2	43	64.2	57.8	53.6	64.4	75	72.9	80.5	44.2	62.9	66.6	64.2	68.9	61.6
Ours	64.4	69.4	58.1	42.0	50.1	69.0	82.9	68.9	47.8	64.5	61.7	58.1	70.7	79.1	75.0	82.6	45.5	66.2	69.4	68.4	67.8	65.0

Table 10: Experiment (*e*): 15-5.

Methods	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Bike	Person	T_0	Plant	Sheep	Sofa	Train	tv	mAp
1–20	79	81.4	73.8	54.2	61	79.1	90.7	78.6	53.6	81.5	64.1	77.8	80.9	77.4	82.7	—	47.3	68.1	68	77.1	72.2	72.4
1–15	71.5	76.1	67.5	46.7	56.6	69.4	84.4	74.4	50.3	74.5	53.8	69.2	77.5	73.2	78.3	68.2	—	—	—	—	—	68.2
Shmelkov et al.	70.5	79.2	68.8	59.1	53.2	75.4	79.4	78.8	46.6	59.4	59.0	75.8	71.8	78.6	69.6	68.3	33.7	61.5	63.1	71.7	62.2	65.9
Faster ILOD	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	71.6	36.7	61.7	59.3	67.9	59.1	67.9
iOD	66.2	73.4	60.0	47.5	46.5	73.0	78.2	71.6	46.0	72.3	55.0	69.4	75.6	73.0	74.6	65.5	42.5	50.1	59.7	65.2	66.6	63.3
Ours	73.4	75.4	60.8	46.2	49.7	71.1	78.3	74.2	46.6	69.8	59.2	72.5	76.9	73.7	74.9	66.9	41.4	50.8	61.8	69.3	64.2	64.5

Table 11: Experiment (f): 19-1.

Methods	Aero	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	T_0	tv	mAp
1-20-	79	81.4	73.8	54.2	61	79.1	90.7	78.6	53.6	81.5	64.1	77.8	80.9	77.4	82.7	47.3	68.1	68	77.1	—	72.2	72.4
1–19	66.5	71.6	62.9	41.6	50.2	63.7	80.1	69.8	43.6	66.6	50.4	67.7	72.2	65.3	73.3	37.4	58.5	53.9	64.0	60.9	—	61.0
Shmelkov et al.	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	68.5	62.7	68.3
Faster ILOD	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	68.9	61.1	68.6
iOD	75.4	76.6	68.9	47.6	52.5	75.4	79.1	76.5	45.7	77.2	61.3	79.6	83.1	73.4	76.1	43.7	71.6	64.9	73.1	68.2	73.1	68.1
Ours	72.9	78.0	69.1	46.8	54.5	73.5	86.3	76.8	51.6	77.8	58.0	73.7	80.9	77.2	82.3	45.7	69.0	60.4	74.3	68.9	68.7	68.9

Table 12: Time spent on batch tasks (/s).

Experiment	Ours	iOD
d	1084	1044
e	182	168
f	35	32

shown in Table 10, our method is slightly inferior to the methods in [9, 10] in terms of overall task mAP, but it is superior to the most recent method, iOD, in terms of retaining the old task performance and new task learning effect. It obtained 64.5% overall task mAP, 66.9% old task mAP, and 64.5% new task mAP.

In experiment (*f*), we increased the number of classes learned in T_0 to 19 classes. In the results reported in Table 11, our model is comparable to the other methods in overall task learning and old-class task performance, but it is optimal in terms of mAP. It obtains 68.9% mAP for the overall task and 68.9% mAP for the old-class task.

The class flow task experiments and batch task experiments demonstrate that the smaller the number of classes learned by T_i in the incremental task phase, the smaller the fluctuation of the model on the total mAP value during each new task T_i learning process is, where the fluctuation is the smallest for the scenario when $T_i = 1$. In contrast, when the number of classes learned by T_i increases, the gap in the overall mAP score of various methods increases significantly, and our method significantly outperforms the other methods in various incremental scenarios in the experiment.

4.4.3. Time Performance Analysis. Table 12 presents a comparison of the training time between our method and the iOD method during the incremental learning phase. The results indicate that our method requires slightly more time for training, primarily due to the increased computational parameters involved in the incremental learning phase. However, the difference in training time between the two methods is relatively small. Notably, our method achieves a higher mean average precision (mAP) for the performance on both the old and new classes compared to the iOD method, with an improvement of 4.4%. In future studies, we will continue to conduct additional investigations to decrease the time necessary for model training while maintaining the assurance of optimal model performance.

5. Conclusion

The catastrophic forgetting problem is mainly addressed by knowledge distillation in existing target detection models; however, object detection models with multiple network architectures require distinct types of distillation procedures. In this work, we present a novel multiple networks mean distillation method for object detection that uses zero averaging to process the model output parameters. Then, it adds the parameters to the distillation loss to further improve the model output stability while strengthening the distillation loss at the input and output sides of the model network structure and adaptively distilling the intermediate network structure to better obtain accurate outputs. We combine meta-learning with a multiple network mean distillation method. On the two basic datasets, we set up numerous incremental tests, and the outcomes show that our model performs better than the alternative comparison models.

Data Availability

All data are generated by relevant algorithms. If you need to reproduce the experimental results, please contact the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Jing Yang was responsible for conceptualization, methodology, investigation, formal analysis, writing the original draft, and revising the draft. Kun Yuan was responsible for data curation, methodology, and writing the original draft. Suhao Chen was responsible for resources, methodology, and supervision. Qinglang Li performed revision and checked the draft. Shaobo Li was responsible for visualization and investigation. Xiuhua Zhang was responsible for resources, supervision, formal analysis, revision, and checking the draft. Bin Li was responsible for methodology and validation.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 62166005), the Guizhou Provincial Key Technology R&D Program (Grant nos. QKH [2023]368, QKH[2022]003, and QKH[2021]335), Developing Objects and Projects of Scientific and Technological Talents in Guiyang City (Grant no. ZKHT[2023]48-8), Joint Open Fund Project of Key Laboratories of the Ministry of Education (Grant nos. QJH[2020]245 and QJH[2020]248), and the Guizhou Provincial Science and Technology Projects (Grant no. PTRC[2020]6007-2).

References

[1] N. H. Tasnim, S. Afrin, B. Biswas, A. A. Anye, and R. Khan, "Automatic classification of textile visual pollutants using deep learning networks," *Alexandria Engineering Journal*, vol. 62, pp. 391–402, 2023.

[2] X. Song, S. Gao, and C. Chen, "A multispectral feature fusion network for robust pedestrian detection," *Alexandria Engineering Journal*, vol. 60, no. 1, pp. 73–85, 2021.

[3] L. Xue, W. Yan, P. Luo et al., "Detection and localization of hand fractures based on GA_Faster R-CNN," *Alexandria Engineering Journal*, vol. 60, no. 5, pp. 4555–4562, 2021.

[4] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A survey on machine learning for recurring concept drifting data streams," *Expert Systems with Applications*, vol. 213, Article ID 118934, 2023.

[5] K. J. Joseph, J. Rajasegaran, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Incremental object detection via

meta-learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9209–9216, 2022.

[6] L. Chen, C. Yu, and L. Chen, "A new knowledge distillation for incremental object detection," in *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, Budapest, Hungary, July 2019.

[7] Y. Hao, Y. Fu, Y. G. Jiang, and Q. Tian, "An end-to-end architecture for class-incremental object detection with knowledge distillation," in *Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, Shanghai, China, July 2019.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[9] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental Learning of Object Detectors without Catastrophic forgetting," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3400–3409, IEEE, Venice, Italy, August 2017.

[10] C. Peng, K. Zhao, and B. C. Lovell, "Faster ILOD: incremental learning for object detectors based on faster RCNN," *Pattern Recognition Letters*, vol. 140, pp. 109–115, 2020.

[11] J. Kirkpatrick, R. Pascanu, N. Rabinowitz et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[12] A. Hao, Y. Min, and X. Chen, "Self-mutual distillation learning for continuous sign language recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11303–11312, Montreal, Canada, October 2021.

[13] K. Li, L. Yu, S. Wang, and P. A. Heng, "Towards Cross-Modality Medical Image Segmentation with Online Mutual Knowledge distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 775–783, New York, NY, USA, January 2020.

[14] P. Xie and X. Du, "Performance-Aware Mutual Knowledge Distillation for Improving Neural Architecture Search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11922–11932, New Orleans, LA, USA, June 2022.

[15] W. Son, J. Na, J. Choi, and W. Hwang, "Densely guided knowledge distillation using multiple teacher assistants," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9395–9404, Montreal, Canada, October 2021.

[16] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved Knowledge Distillation via Teacher assistant," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5191–5198, New York, NY, USA, January 2020.

[17] D. Y. Park, M. H. Cha, D. Kim, and B. Han, "Learning student-friendly teacher networks for knowledge distillation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13292–13303, 2021.

[18] J. Lee, Y. Jeong, S. Kim, J. Min, and M. Cho, "Learning to Distill Convolutional Features into Compact Local Descriptors," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 898–908, Waikoloa, HI, USA, January 2021.

[19] K. Yue, J. Deng, and F. Zhou, "Matching Guided distillation," in *Proceedings of the European Conference on Computer Vision*, pp. 312–328, Springer, Glasgow, UK, January 2020.

[20] T. Nguyen, R. Novak, L. Xiao, and J. Lee, "Dataset distillation with infinitely wide convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5186–5198, 2021.

[21] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J. Y. Zhu, "Dataset distillation by matching training trajectories," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4750–4759, Washington DC, USA, June 2022.

[22] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J. Y. W. Zhu, "Synthesizing tileable textures via dataset distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2278–2282, New Orleans, LA, USA, June 2022.

[23] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.

[24] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S. T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13208–13217, IEEE, New Orleans, LA, USA, June 2020.

[25] N. Dong, Y. Zhang, M. Ding, and G. H. Lee, "Bridging non Co-occurrence with unlabeled in-the-wild data for incremental object detection," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30492–30503, 2021.

[26] M. Abdelsalam, M. Faramarzi, S. Sodhani, and S. Chandar, "Iirc: incremental implicitly-refined classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11038–11047, Washington DC, USA, June 2021.

[27] S. Dong, X. Hong, X. Tao, X. Chang, X. Wei, and Y. Gong, "Few-shot class-incremental learning via relation knowledge distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1255–1263, Washington DC, USA, June 2021.

[28] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 233–248, Springer, Munich, Germany, July 2018.

[29] Y. Xiang, Y. Miao, J. Chen, and Q. Xuan, "Efficient incremental learning using dynamic correction vector," *IEEE Access*, vol. 8, pp. 23090–23099, 2020.

[30] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, and L. R. I. L. O. D. Heck, "Near real-time incremental learning for object detection at the edge," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pp. 113–126, Arlington, VA, USA, November 2019.

[31] Y. Wu, Y. Chen, L. Wang et al., "Large scale incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, Washington DC, USA, June 2019.

[32] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 831–839, IEEE, Long Beach, CA, USA, August 2019.

[33] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "Podnet: pooled outputs distillation for small-tasks incremental learning," in *Proceedings of the European Conference on Computer Vision*, pp. 86–102, Springer, Glasgow, UK, August 2020.

[34] D. Yang, Y. Zhou, A. Zhang et al., "Multi-View correlation distillation for incremental object detection," *Pattern Recognition*, vol. 131, Article ID 108863, 2022.

[35] M. Andrychowicz, M. Denil, S. Gomez et al., "Learning to learn by gradient descent by gradient descent," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[36] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the International Conference on Machine Learning PMLR*, pp. 1126–1135, Sydney, Australia, August 2017.

[37] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proceedings of the International Conference on Machine Learning*, pp. 1568–1577, PMLR, Baltimore, MD, USA, July 2018.

[38] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[39] A. Kedia and S. C. Chinthakindi, "Keep learning: self-supervised meta-learning for learning from inference," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Maintenant*, pp. 63–77, Europe, UK, June 2021.

[40] Z. Xu, H. P. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[42] T. Y. Lin, M. Maire, S. Belongie et al., "Microsoft coco: common objects in context," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 740–755, Springer, Tel Aviv, Israel, October 2014.