# IncDet: In Defense of Elastic Weight Consolidation for Incremental Object Detection

Liyang Liu, Zhanghui Kuang, Yimin Chen, Jing-Hao Xue, Wenming Yang and Wayne Zhang

***Abstract*—Elastic weight consolidation (EWC) has been successfully applied for general incremental learning to overcome the catastrophic forgetting issue. It adaptively constrains each parameter of the new model not to deviate much from its counterpart in the old model during fine-tuning on new class datasets, according to its importance weight for old tasks. However, the previous study demonstrates that it still suffers from catastrophic forgetting when directly used in object detection. In this paper, we show EWC is effective for incremental object detection if with critical adaptations. *First*, we conduct controlled experiments to identify two core issues why EWC fails if trivially applied to incremental detection: 1) the absence of old class annotations in new class images makes EWC misclassify objects of old classes in these images as background, and 2) the quadratic regularization loss in EWC easily leads to gradient explosion when balancing old and new classes. *Then*, based on the above findings, we propose the corresponding solutions to tackle these issues: 1) utilize pseudo bounding box annotations of old classes on new datasets to compensate for the absence of old class annotations, and 2) adopt a novel Huber regularization instead of the original quadratic loss to prevent from unstable training. *Last*, we propose a general EWC-based incremental object detection framework and implement it under both Fast R-CNN and Faster R-CNN, showing its flexibility and versatility. In terms of either the final performance or the performance drop w.r.t. the upper bound of joint training on all seen classes, evaluations on the PASCAL VOC and COCO datasets show that our method achieves a new state-of-the-art.**

***Index Terms*—object detection, catastrophic forgetting, incremental detection, Bayesian online learning**

## I. Introduction

OBJECT detection is undoubtedly the cornerstone in computer vision and has facilitated many amazing applications related to image understanding, such as instance segmentation [1], pose estimation [2] and human detection/recognition [3]. Modern object detection systems equipped with deep convolutional networks can locate and classify object regions accurately and rapidly. Especially by the introduction of large-scale datasets such as ImageNet [4], COCO [5], OpenImages [6] and VisualGenome [7], we can train models capable of detecting up to several thousands of classes. Nevertheless, almost all of these models can only recognize objects which are restricted to predefined categories of interest, but we may always want to detect new classes that the models have not learned before. As humans learn in a life-long manner, a model that can evolve to detect fresh concepts is appealing. Incremental detection [8] aims at incrementally adapting the base model trained on old classes to detect new classes and meanwhile not to forget the old ones.

A naïve approach for incremental detection is to fine-tune the base model with new class training data [8], but this method notoriously suffers from catastrophic forgetting [9] when the old data is inaccessible during fine-tuning. Consequently, forgetting leads to severe performance drop on old categories, and also dramatic false detections on new classes due to misclassifying old class objects as new categories. As an alternative, training from scratch with the combination of old and new data seems feasible to avoid catastrophic forgetting, but sometimes we may not have access to old images or annotations because of privacy issues. Furthermore, if the amount of old data is much larger than that of the new data, we have to spend a lot of computation and storage resources on old data, and thus cannot expect low-cost model adaptation, which is attractive especially in circumstances with a constrained training budget.

Incremental object detection is supposed to benefit from incremental learning which has been widely investigated [10], [11], [12], [13], [14], [15]. Especially, the previous study [11] proposed Elastic Weight Consolidation (EWC) based on Bayesian online learning [16]. Although EWC achieves impressive empirical results on image classification, it still suffers from catastrophic forgetting when directly applied to object detection as reported in [8]. Through our pilot experiments, we identify the following two key issues leading to the failure of applying EWC to incremental detection. *First*, images in new datasets might contain both old class and new class objects, but with only new classes annotated. Treating all regions which do not belong to new classes as background would misclassify old class objects as non-objects, leading to disastrous missing detections of old classes. *Second*, the quadratic loss in EWC easily leads to gradient explosion and thus unstable training during the hyper-parameter tuning process to find the best trade-off between old and new classes.

To mitigate the *missing annotation* issue, we propose a simple yet effective strategy that generates pseudo bounding boxes of old classes to compensate for the absence of old class annotations. Specifically, for each image in new datasets,

This work was supported by the Natural Science Foundation of China (No.61771276 and No.61871258) and the Special Foundation for the Development of Strategic Emerging Industries of Shenzhen (No.JCYJ20170817161845824 and No.JCYJ20170817161056260).

L. Liu and W. Yang are with Shenzhen Key Lab. of Inf. Sci.&Tech./Shenzhen Engineering Lab. of IS.&DCP., Shenzhen International Graduate School/Department of Electronic Engineering, Tsinghua University, China (e-mail: {liu-ly14@mails, yang.wenming@sz}.tsinghua.edu.cn).

Z. Kuang, Y. Chen and W. Zhang are with SenseTime Research (e-mail: {kuangzhanghui, chenyimin, wayne.zhang}@sensetime.com).

J.-H. Xue is with the Department of Statistical Science, University College London, London WC1E 6BT, U.K. (e-mail: jinghao.xue@ucl.ac.uk).

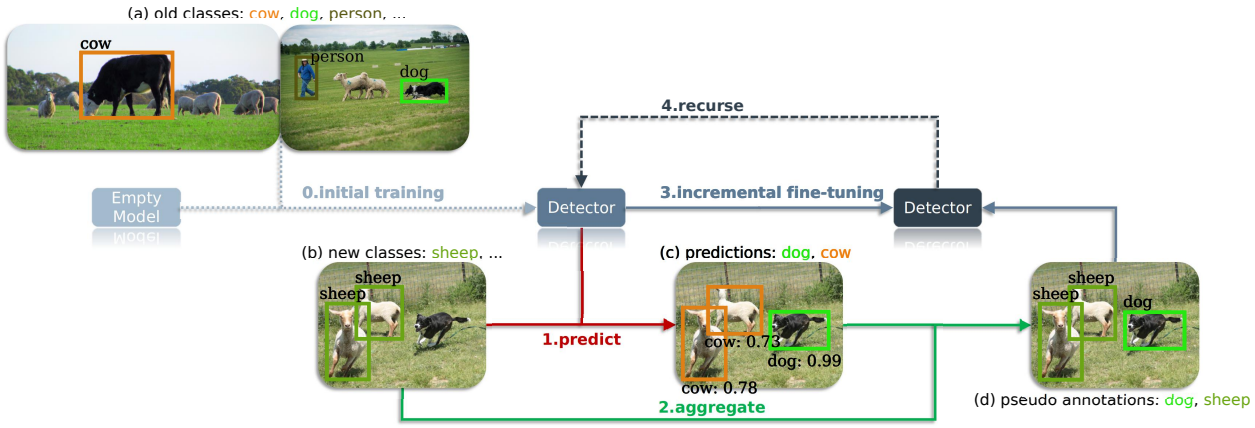



Fig. 1. Illustration of our incremental detection framework IncDet. (a) and (b) are the original annotations provided in the initial training and incremental fine-tuning phases, respectively. (c) shows the predicted results of the old model on new images. (d) manifests the aggregated annotations we use to fine-tune the old model and obtain the new model. Our incremental object detection pipeline consists of steps 0–3, in which step 0 needs to be run only once at the very beginning, and steps 1–3 can be recursively executed if more classes are to be added.

we predict old class objects with the old model and obtain pseudo bounding boxes for old classes. Then old class pseudo bounding boxes and new class annotated bounding boxes are jointly used to incrementally fine-tune the model as shown in Fig. 1. In this way, the fine-tuned model can not only learn to detect new classes but also retain the performance of old classes to a large extent.

To overcome the *unstable training* issue, instead of the quadratic loss in EWC, we propose a novel Huber regularization. It shares the same spirit with the regularization term in EWC by constraining each parameter in the new model around its counterpart in the old model. Meanwhile, our Huber regularization adaptively clips the gradient of each parameter according to its contribution to old classes. Compared with the conventional *holistic* gradient clipping [17], the proposed *adaptive* gradient clipping not only realizes stable training but also achieves better performance.

Based on the above two adaptations, we propose IncDet, an EWC-based incremental object detection framework that is general and flexible. Our framework IncDet is illustrated in Fig. 1 which consists of 4 main steps: **0**. initial training: an empty model (maybe pretrained on an image classification task) is trained with some old classes to obtain the base model; **1**. predict: the trained base model makes predictions on images collected for new classes; **2**. aggregate: predictions from the old model and manual annotations are aggregated; **3**. incremental fine-tuning: the base model is fine-tuned incrementally on images with partially labeled (new classes) boxes and partially predicted (old classes) pseudo annotations. Step 0 is done only once at the very beginning and steps 1 to 3 can be recursively executed if more classes need to be added.

To summarize, our **contributions** are listed as follows:

1) We instantiate EWC in the context of incremental object detection and identify the core issues that lead to the failure of directly applying EWC to detection: missing annotation and unstable training.
2) We propose pseudo annotation to compensate for the missing annotations of old classes in new class images. It mitigates the wrong supervisions of regarding old class objects as background during fine-tuning.
3) We propose the Huber regularization to alleviate the unstable training issue when fine-tuning the old model on new datasets, by adaptively clipping the gradient of each parameter according to its importance for old class detection tasks.
4) We present IncDet, an EWC-based incremental object detection framework, and implement it under both Fast R-CNN [18] and Faster R-CNN [19], demonstrating its versatility and flexibility.
5) To showcase the effectiveness of our approach, we conduct experiments on the PASCAL VOC [20] and COCO [5] datasets. By adding new classes all at once or sequentially, our results surpass the previous best [8], [21], [22], [23] in terms of both the final performance and the performance gap compared with the joint training upper bound.

## II. Related Work

### A. Incremental learning

Incremental learning has been of great interest for decades [24], [25] and has attracted much attention recently [26], [27]. Algorithms developed so far can be generally categorized into four types [28]. **Prior-based methods** regard the posterior over model parameters after finishing training one task as the prior for transferring to the new task. Distinguished prior-based methods differ in the way how prior knowledge is modeled and will be further discussed in Sec. II-B. **Rehearsal methods** are the second type, where memories about old classes are stored selectively or reproduced by generative models. These memories are then used together with new data to overcome forgetting. VCL [14] adopts the K-center algorithm [29] to construct a core-set by choosing exemplars from the dataset of each old class. DGR [13] alleviates the necessity of storing old data by training deep generative

TABLE I
ADVANTAGES OF INCDET COMPARED WITH PREVIOUS STATE-OF-THE-ARTS.

| methods | ILOD [8] | DMC [21] | RILOD [22] | CIFRCN [23] | IncDet (Ours) |
|---|---|---|---|---|---|
| w/o auxiliary models | ✗ | ✗ | ✗ | ✗ | ✓ |
| high memory efficiency | ✗ | ✗ | ✗ | ✗ | ✓ |
| high computational efficiency | ✗ | ✗ | ✗ | ✗ | ✓ |
| w/o extra data | ✓ | ✗ | ✓ | ✓ | ✓ |
| w/o external proposals | ✗ | ✓ | ✓ | ✓ | ✓ |

models to mimic the past data. **Ensemble-based methods** store a series of historical models and the final determination is made by ensemble of these models. ADAIN [30] transforms the knowledge of current data chunk into the learning process of future chunks through the specially designed mapping function. Learn$^{++}$.NSE [31] and its class imbalance variant [32] study incremental learning of concept drift in nonstationary environments. DTEL [33] preserves historical models that are diverse enough for better transferability to new concept. **Dynamic structure methods** progressively increase model capacity by adding sub-networks to compensate for the new tasks. PNN [34] completely fixes model parameters for old tasks while DEN [35] selectively retrains the old network and dynamically expand its capacity if necessary. Broad learning system (BLS) [36], [37] develops incremental learning algorithms for increment of input variables [38], feature nodes [39] and enhancement nodes. EM-ELM [40] and AG-ELM [41] achieve incremental learning by adding hidden nodes to the extreme learning machine. In learning without forgetting (LwF) [10], all tasks share feature extraction parameters and task-specific parameters are introduced for newly arrived tasks. When training on new datasets, task-specific parameters for new tasks are learned via fine-tuning, while the output of combined task-shared parameters and task-specific parameters for old tasks are regularized to be close to the output of a frozen copy of the old model.

### B. EWC and its variants

Among prior-based incremental learning methods, the one most related to our work is EWC [11], which has been successfully applied to image classification and reinforcement learning. Inspired by Laplace approximation [42], EWC adopts a Gaussian prior for every single parameter, and approximates the Hessian of the likelihood after training on each task to be diagonal. It can be considered as regularizing parameters that are more important for old tasks to have less discrepancy with the old parameters when fine-tuning on new datasets, where the importance is determined by the likelihood loss at the end of each training stage. SI [12] however computes the importance in an online fashion and thus takes the whole training trajectory into consideration. Instead of depending on the likelihood loss, MAS [43] approximates the importance by the gradient of the model output w.r.t. the model parameters, and it can also take advantage of the unlabeled data. RWalk [44] generalizes EWC and SI from a theoretically grounded KL-divergence based perspective. Ritter *et al.* [15] introduce the Kronecker factored online Laplace approximation to replace the diagonal Hessian in EWC with a block diagonal one, and thus model the interactions among the parameters. The above incremental learning methods mainly focus on incremental image classification while we devote ourselves to incremental object detection which is under-explored.

### C. General object detection

General object detection in the deep learning era can be divided into proposal-based and proposal-free methods. **Proposal-based** detectors first extract class-agnostic region proposals of the potential objects, and then conduct proposal-level classification and box regression. R-CNN [45] uses convolutional networks to classify the region proposals produced by selective search [46]. SPP-Net [47] and Fast R-CNN [18] speed up R-CNN by sharing the computation of feature extraction for each region. NoC [48] points out the importance of using proposal-level convolutional networks to classify each region instead of the widely used multi-layer perceptrons. Faster R-CNN [19] introduces a region proposal network (RPN) that shares features with the detection network, and thus enables nearly cost-free proposal generation. In contrast, **proposal-free** detectors attempt to predict bounding boxes and get the confidence of each category directly. YOLO [49] divides the image into grids and for each grid cell predicts several bounding boxes and class probabilities. SSD [50] extends YOLO with multi-scale feature maps and adopts diverse default boxes for various object shapes. RetinaNet proposes focal loss [51] to deal with dense detections on multi-scale feature maps based on FPN [52]. General object detection has been largely developed, but it cannot be trivially used to detect new classes without catastrophic forgetting.

### D. Incremental object detection

Shmelkov *et al.* study incremental detection without catastrophic forgetting in the pioneering ILOD [8]. They develop the dual-network learning under Fast R-CNN [18], where one fixed copy of the old model serves as the teacher for remembering old classes, and the other learnable model acts as the student for learning new classes, trying to minimize the discrepancy between outputs of the two models on new class data. Dual-network learning is inspired by learning without forgetting (LwF) [10] which adopts knowledge distillation [53]. Zhang *et al.* [21] propose deep model consolidation (DMC) and extend LwF [10] to proposal-free detector RetinaNet [51]. They propose triple-network learning, where two models for the old and new classes are trained separately and a third network is learned via knowledge distillation from the former two networks with the help of extra unlabeled data, with which to overcome the intrinsic bias toward either old or new classes. RILOD [22] introduces an extra feature distillation

loss besides the classification and regression distillation loss in ILOD [8], and it is implemented under both edge-cloud and edge-only setups. CIFRCN [23] extends ILOD [8] to Faster R-CNN [19] and proposes an end-to-end class incremental detection method via distilling RPN [19] simultaneously with R-CNN, without relying on external proposal generation.

Almost all of the previous incremental object detection methods are *auxiliary-based* and inspired by knowledge distillation [53], where knowledge about old classes is stored in the frozen copy of the trained models. This requires feed-forward passes through the auxiliary teacher networks during fine-tuning on new data, and thus consumes more computation and memory resources. In contrast, our *auxiliary-free* EWC-based IncDet embeds knowledge with parameter importance for old classes. When learning new classes IncDet only demands a single network fine-tuning, which is more efficient and light-weight. In addition, compared with ILOD [8], IncDet is more general and can be adopted in any conventional object detection method, not only those who depend on external proposal generation approaches. We also do not need extra unsupervised data as used in DMC [21] for knowledge distillation. Tab. I summarizes the advantages of IncDet compared with previous incremental detection methods.

### E. Self-training

Pseudo annotation proposed in this paper is reminiscent of self-training which is widely used in general machine learning, as well as object detection [54]. Data distillation [55] applies the model trained on manually labeled data to generate annotations on unlabeled images. A key difference is that they try to utilize a large amount of unlabeled data for self-promotion, but we aim at preventing the model from wrongly regarding old class regions from new class images as background due to the absence of manual labels. Moreover, in self-training, labeled and unlabeled images share the same set of categories so there will not be class confusion resulting from the emergence of new classes. While in incremental detection, the model may predict new classes in new images as old classes with high confidence if they share similar appearances, which should be dealt with carefully to generate accurate pseudo ground truth.

## III. EWC for Incremental Detection

In this section, we first review EWC [11] for general incremental learning from the perspective of Bayesian online learning [16]. Then we instantiate it in the context of incremental object detection. Given an old model $\boldsymbol{\theta}_t^\star$ optimized on datasets $\mathcal{D}_{1:t}$ for the previous $t$ tasks, EWC tries to learn a new model $\boldsymbol{\theta}_{t+1}^\star$ which performs well for all tasks 1 to $t+1$, using the new dataset $\mathcal{D}_{t+1}$ without the help of past data $\mathcal{D}_{1:t}$. It models optimization for the new model as maximizing the posterior on the collection of all data visited so far:

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t+1}\right) \propto p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right) p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t}\right), \tag{1}$$

in which $p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right)$ stands for the likelihood of new data. As the posterior $p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t}\right)$ is intractable without access to $\mathcal{D}_{1:t}$, EWC approximates it by a multivariate Gaussian distribution with Laplace approximation [42] at the optimized model for the first $t$ tasks $\boldsymbol{\theta}_t^\star$:

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t}\right) \approx \mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\theta}_t^\star, \boldsymbol{H}_t^{-1}\right), \tag{2}$$

where $\boldsymbol{H}_t$ is the Hessian matrix of $-\log p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t}\right)$ w.r.t. $\boldsymbol{\theta}$ computed at $\boldsymbol{\theta}_t^\star$. To obtain $\boldsymbol{H}_t$, we again apply Laplace approximation on the posterior probability $p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t+1}\right)$:

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}_{1:t+1}\right) \approx \mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\theta}_{t+1}^\star, \boldsymbol{H}_{t+1}^{-1}\right). \tag{3}$$

By substituting Eq. (3) and (2) into (1), and taking the second order derivative w.r.t. $\boldsymbol{\theta}$, we have the following recursive form of the Hessian matrix:

$$\boldsymbol{H}_{t+1} \approx -\nabla_{\boldsymbol{\theta}}^2 \log p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right) + \boldsymbol{H}_t. \tag{4}$$

As the Hessian matrix $\boldsymbol{H}$ whose size is square of the model size cannot be computed exactly, EWC assumes the diagonal structure [11], [15] of the Hessian matrix and thus the Gaussian distribution in Eq. (2) is fully factorized for each parameter $\theta_i$:

$$\log \mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\theta}_t^\star, \boldsymbol{H}_t^{-1}\right) \propto -\frac{1}{2} \sum_i H_t^{ii} \left(\theta_i - \theta_t^{i\star}\right)^2, \tag{5}$$

in which $H_t^{ii}$ denotes the diagonal entry of $\boldsymbol{H}_t$. So from Eq. (1), a model doing well for all seen tasks 1 to $t+1$ can be trained by minimizing the regularized loss:

$$\mathcal{L}_{t+1} = -\log p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right) + \frac{\lambda}{2} \sum_i H_t^{ii} \left(\theta_i - \theta_t^{i\star}\right)^2, \tag{6}$$

where $\lambda$ is a hyper-parameter introduced to balance the plasticity and stability of the model [11]. Diagonal entries of the Hessian matrix can be computed recursively from Eq. (4):

$$H_{t+1}^{ii} = H_t^{ii} - \partial_i^2 \log p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right)\big|_{\theta_i = \theta_{t+1}^{i\star}}, \tag{7}$$

in which $\partial_i^2$ is the second-order partial derivative w.r.t. $\theta_i$.

In the above we review the most general form of EWC, now we instantiate it for supervised learning, especially incremental object detection. Under the basic assumption that data samples are independent and identically distributed, in supervised learning, the likelihood $\log p\left(\mathcal{D}_t \mid \boldsymbol{\theta}\right)$ in Eq. (6) and (7) can be decomposed into

$$\log p\left(\mathcal{D}_t \mid \boldsymbol{\theta}\right) = \sum_j \log p\left(\boldsymbol{y}_t^{(j)} \mid \boldsymbol{x}_t^{(j)}; \boldsymbol{\theta}\right), \tag{8}$$

where $\boldsymbol{x}^{(j)}$ is the $j$-th data sample and $\boldsymbol{y}^{(j)}$ stands for its corresponding label. Then the derivatives in Eq. (7) is

$$\begin{aligned} -\partial_i^2 \log p\left(\mathcal{D}_t \mid \boldsymbol{\theta}\right) &= -\sum_j \partial_i^2 \log p\left(\boldsymbol{y}_t^{(j)} \mid \boldsymbol{x}_t^{(j)}; \boldsymbol{\theta}\right) \\ &\approx \sum_j \left(\partial_i \log p\left(\boldsymbol{y}_t^{(j)} \mid \boldsymbol{x}_t^{(j)}; \boldsymbol{\theta}\right)\right)^2 \\ &= \sum_j \left(\partial_i \mathcal{L}_t^{(j)}\right)^2, \end{aligned} \tag{9}$$

where $\mathcal{L}_t^{(j)}$ represents the loss of the $j$-th sample on task $t$. Above we use the property of the Fisher information to build a connection between the second-order derivative and the square of the first-order derivative, which can be regarded as the

importance of each parameter to the current task. Combining Eq. (6), (7) and (9), we have the regularized loss:

$$\mathcal{L}_{t+1} = \frac{\lambda}{2}\sum_i w_t^i \left(\theta_i - \theta_t^{i\star}\right)^2 - \log p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right), \quad (10)$$

with the recursive update of the importance weight (*i.e.*, the diagonal entry of the Hessian matrix):

$$w_{t+1}^i = w_t^i + \sum_j \left(\partial_i \mathcal{L}_{t+1}^{(j)}\right)^2\Big|_{\theta_i=\theta_{t+1}^{i\star}}, \ t = 0, 1, 2, ... \quad (11)$$

and $w_0^i = 0, \ \forall i$ for recursion initialization. Intuitively, the regularized objective not only depends on the likelihood of the new data, but also the weighted model parameter deviation from the old model, with the weight decided by the importance of each parameter for all seen tasks. Note that we first fine-tune the old model $\boldsymbol{\theta}_t^\star$ on the new data $\mathcal{D}_{t+1}$ to obtain the optimized new model $\boldsymbol{\theta}_{t+1}^\star$. Then at the end of an incremental fine-tuning phase which typically consists of about a dozen of epochs, the importance of each parameter for the current task $t+1$ is computed by iterating through the corresponding dataset $\mathcal{D}_{t+1}$ with the optimized model $\boldsymbol{\theta}_{t+1}^\star$ for an *extra* epoch. Each iteration consists of a forward pass and a backward pass but the model is not updated by gradient descent. The computed importance weights are accumulated to $w_{t+1}^i$, the importance for all experienced tasks. Then the accumulated importance $w_{t+1}^i$ is used for remembering the tasks 1 to $t+1$ when we fine-tune on the dataset of more new tasks.

For incremental object detection, we set the per-image loss $\mathcal{L}_t^{(j)}$ in Eq. (11) as the sum of per-region losses including region classification and box regression for each image $j$ and task $t$:

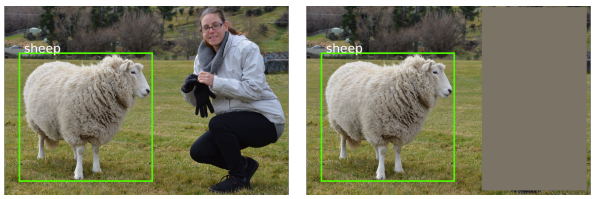
$$\mathcal{L} = \sum_r \mathcal{L}_{\text{cls}}^r + \mathcal{L}_{\text{reg}}^r. \quad (12)$$

The likelihood $-\log p\left(\mathcal{D}_{t+1} \mid \boldsymbol{\theta}\right)$ in Eq. (10) can also be represented by the classification and regression losses as in conventional detection. So with Eq. (10), (11) and (12), we instantiate EWC [11] for incremental object detection.

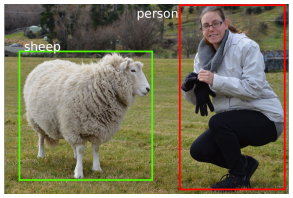
**Discussion.** The recursive update of the importance weight in Eq. (11) is significant, to which we eventually aggregate the memory about all past tasks, by involving one new task at a time. In practice, to incrementally adapt the old model for some new classes, we **first** add parameters corresponding to detectors of the new classes and initialize them randomly. Note that the newly-added parameters are negligible compared to the full model. **Then** we fine-tune the old model with the regularized loss in Eq. (10) to obtain the optimized new model. **Finally** we update the importance of each parameter as Eq. (11) using the optimized model over $\mathcal{D}_{t+1}$ to be prepared for the fine-tuning on possible added classes in the future.

## IV. Pseudo Annotation

**Motivation.** Although the instantiation of EWC provides a viable way for incremental object detection, we find it still suffers from catastrophic forgetting as reported in [8]. We hypothesize the failure of trivially applying EWC mainly results from the fact that old class objects may appear in new class images, which will not be encountered in incremental image classification [11]. Since regions without bounding box annotations are simply considered as background during training in currently popular detection frameworks [18], [19], so old class objects are misclassified as background when learning new classes. However, in the past training stages, old class regions have been labeled as foreground. During incremental fine-tuning, the regularization term in Eq. (10) suggests to classify old class regions as foreground, but the likelihood term forces the network to regard them as background. This contradiction leads to the optimization difficulty and finally makes the network struggle to converge well.




(a) origin (b) mask




(c) pseudo

Fig. 2. Comparisons among different annotation strategies. (a) shows the original annotations (*i.e.*, sheep) for the new dataset. (b) masks the old class objects (*i.e.*, person) by the mean pixel values. (c) adds pseudo annotations (*i.e.*, person) to the original manually labeled annotations.

To validate the conjecture above, we conduct pilot experiments VOC 2007 [20] with Fast R-CNN by incrementally learning 5 new classes one by one from a base model trained with 15 old classes. We fine-tune the model by masking the old class objects in the new images with mean pixel values as shown in Fig. 2 (b), and thus exclude the negative influence of old class objects. Compared with the model fine-tuned with the original annotations as shown in Fig. 2 (a), both mAP of 15 old classes and that of all 20 classes are significantly improved. Specifically, mAP of old classes increases from 46.9% to 59.7% while that of all classes boosts from 47.6% to 56.7%. We believe that ignoring old classes can improve recall of them during testing.

Since we do not have oracle annotations of old classes in the new images, the masking strategy cannot be implemented in practice. As a proxy, we propose to exploit predictions of the old model on the new images. In this way, not only do we alleviate wrong supervisions, but also we can fully utilize information in the new images by consolidating old class memories with the predicted regions. However, as the old model has only learned to detect old classes, it will classify new objects similar to old ones as its known categories (*e.g.* Fig. 1 (c) it regards *sheep* as *cow*). To deal with the wrong

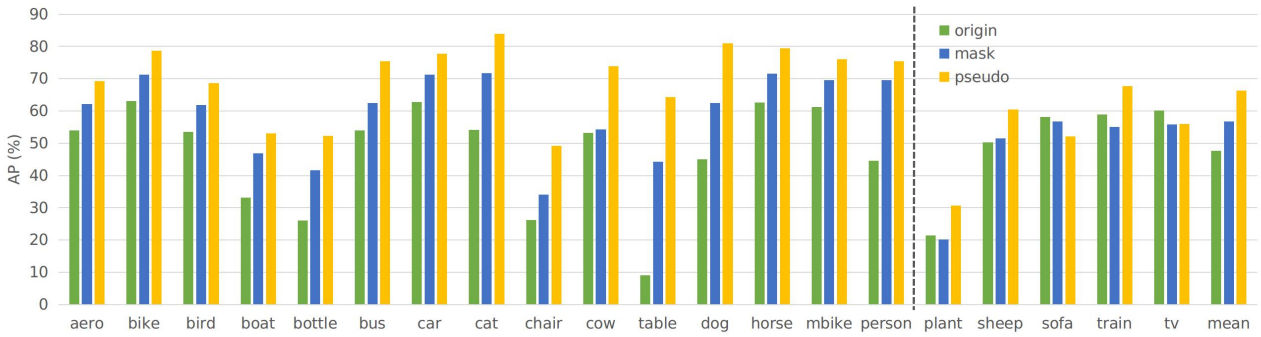



Fig. 3. Effect of our pseudo annotations on the AP for each class and the mAP across all seen classes after incremental fine-tuning, where we originally train the network on the first 15 classes (left to dashed line) and then fine-tune it with the remaining 5 ones (right to dashed line) **sequentially**. "origin" shows the result of fine-tuning with the original annotations including only the new classes as Fig. 2 (a). "mask" represents the result of masking old class regions with mean pixel values as Fig. 2 (b). "pseudo" is the result of fine-tuning with our pseudo annotations as Fig. 2 (c).

predictions, we further propose to correct predictions of the old model with the manually labeled new class annotations. Only those predicted old class boxes with maximal IoU with new class annotations less than 0.5 are kept (see Fig. 1 (d)). When we fine-tune the model with the ensemble of pseudo annotations and the manual labels, mAP of the 15 old classes and that of all classes increases to 70.6% and 66.3%, respectively, which is remarkably superior to its counterpart with masking training strategy (see Fig. 3).

**Discussion.** Although pseudo annotation seems intuitive, as far as we know, we are the first to identify that the failure of EWC in incremental detection originates from missing annotations. By adopting this simple strategy with certain improvements (correct the old class pseudo annotations with the new class manual labels), we effectively make EWC work well in incremental detection, in contrast with the conclusion drawn from ILOD [8]. Besides, solely depending on pseudo annotation is not enough due to the unstable training issue resulting from the quadratic loss in EWC, as discussed next.

## V. Huber Regularization

**Motivation.** The balancing parameter $\lambda$ in Eq. (10) acts as a trade-off between remembering old classes and learning new classes. When $\lambda = 0$, the optimization of the model degrades to conventional fine-tuning, and thus leads to catastrophic forgetting. Ideally, when $\lambda = +\infty$, the new model is kept the same as the old one, and thus no new classes can be learned. To find the best $\lambda$ for incremental object detection, we gradually increase $\lambda$. As shown in Tab. II, larger $\lambda$ leads to significantly better performance on old classes but that of new classes does not change notably. However, when $\lambda$ reaches a certain value (*e.g.* 10), the training process becomes dramatically unstable, which prevents us from finding the best trade-off between old and new classes. We empirically find that the gradient explosion of the regularization term is the reason as we notice that certain parameters own extremely large gradients before gradient explosion. Initially, we thought it was because the inaccurate pseudo annotations act as outlier data samples during incremental fine-tuning, but the gradient explosion still happens even if we use the oracle manual labels of old classes in the new images.

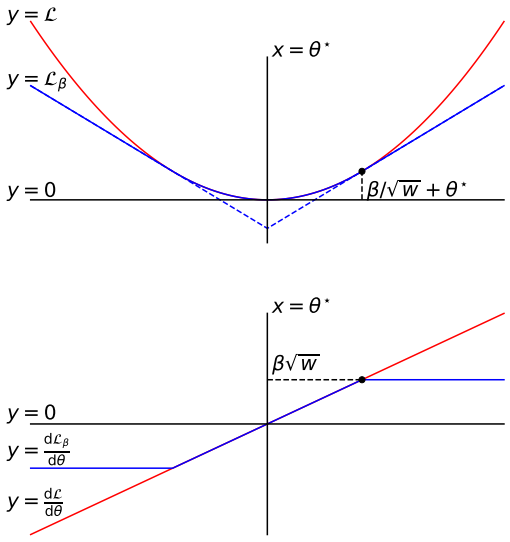



Fig. 4. **Upper:** Visualization of Huber regularization $\mathcal{L}_\beta$ (with threshold $\beta$) and quadratic regularization $\mathcal{L}$ for a single parameter $\theta$. $\theta^\star$ denotes its last optimized value and $w$ is its corresponding importance for previous tasks. When $|\theta - \theta^\star| \leqslant \beta/\sqrt{w}$, the two regularizations are the same. **Lower:** The gradient of $\theta$ in quadratic regularization $\mathcal{L}$ is unbounded, but the absolute gradient of $\theta$ in Huber regularization $\mathcal{L}_\beta$ is adaptively clipped to $\beta\sqrt{w}$ if $|\theta - \theta^\star| > \beta/\sqrt{w}$ according to its importance. More important a parameter is for old classes, more easily its gradient is clipped.

We next analyze the gradient mathematically. Take a single parameter $\theta$ in Eq. (10) as an example, the quadratic term $\mathcal{L} = w\left(\theta - \theta^\star\right)^2/2$ has a gradient $w\left(\theta - \theta^\star\right)$ w.r.t. $\theta$. Due to the distributional shifts of the images across different classes, the model weights will change drastically to accommodate the new classes at the beginning of incremental fine-tuning, and thus the gradients are very easy to grow dramatically and finally explode.

Therefore we propose a novel Huber regularization to

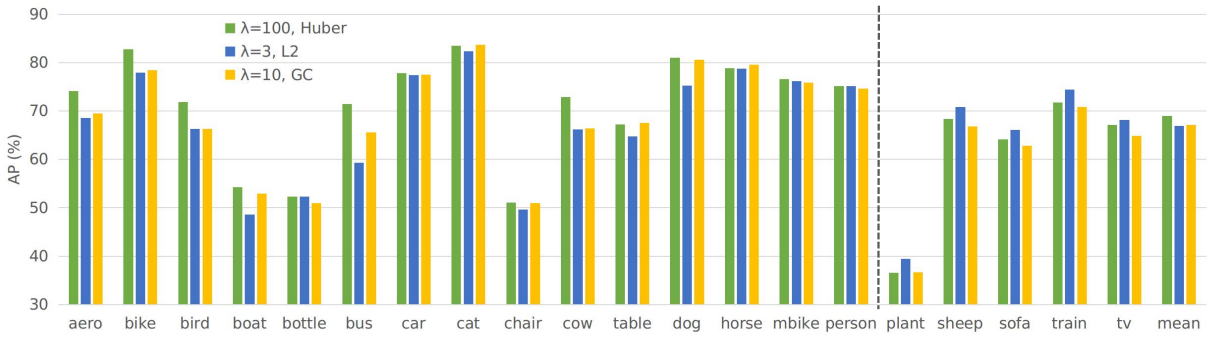



Fig. 5. Efficacy of our Huber regularization. It shows the AP result for each class and the mAP across all classes after incremental fine-tuning. We originally train the network on the first 15 classes (left to dashed line) and then fine-tune it with the remaining 5 ones (right to dashed line) **all at once**. "Huber" represents the result of our Huber regularization. "$L_2$" means to replace the Huber regularization with the quadratic loss originated from EWC. "GC" means to use gradient clipping [17] instead of our Huber regularization. $\lambda$ is tuned to get the best performance for each case.

TABLE II
EFFECT OF $\lambda$ ON MAP (%) OF OLD (15), NEW (5) AND ALL (20) CLASSES USING QUADRATIC LOSS WHEN 5 CLASSES ARE ADDED **ALL AT ONCE**.

| $\lambda$ | old | new | all |
|---|---|---|---|
| 0.1 | 62.7 | 63.6 | 62.9 |
| 0.3 | 64.0 | 63.5 | 63.8 |
| 1 | 66.8 | 63.0 | 65.9 |
| 3 | **67.9** | **63.8** | **66.9** |

constrain the gradient within a reasonable range. More specifically, we replace the quadratic regularization in Eq. (10) by

$$\mathcal{L}_\beta(\theta) = \begin{cases} \frac{1}{2} w (\theta - \theta^\star)^2 & \sqrt{w}\,|\theta - \theta^\star| \leqslant \beta \\ \beta \left( \sqrt{w}\,|\theta - \theta^\star| - \frac{\beta}{2} \right) & \sqrt{w}\,|\theta - \theta^\star| > \beta \end{cases}, \quad (13)$$

where $\beta$ is a pre-defined clipping threshold. By taking the derivative, we have

$$\frac{\mathrm{d}\mathcal{L}_\beta(\theta)}{\mathrm{d}\theta} = \begin{cases} w(\theta - \theta^\star) & \sqrt{w}\,|\theta - \theta^\star| \leqslant \beta \\ \beta \sqrt{w} \frac{\theta - \theta^\star}{|\theta - \theta^\star|} & \sqrt{w}\,|\theta - \theta^\star| > \beta \end{cases}. \quad (14)$$

It can be seen that the gradient is effectively restricted to be in the range of $[-\beta\sqrt{w}, \beta\sqrt{w}]$. Besides, as the Huber regularization is smooth so it is the same as the quadratic loss when $\sqrt{w}\,|\theta - \theta^\star| \leqslant \beta$. For each parameter $\theta$, the corresponding regularization loss and gradient are shown in Fig. 4. We can see that the absolute gradient of a parameter with importance $w$ to the previous tasks are clipped to $\beta\sqrt{w}$ if its deviation from the last optimal value $|\theta - \theta^\star| > \beta/\sqrt{w}$. As expected, more important a parameter is for old classes, more easily its gradient should be clipped.

Alternatively, we may adopt Gradient Clipping [17] to deal with the gradient explosion as

$$\hat{g}_i = \frac{\beta_g}{\max(\beta_g, \|\boldsymbol{g}\|)} g_i, \quad (15)$$

in which $\boldsymbol{g}$ is the gradients of the regularized loss w.r.t. model parameters $\boldsymbol{\theta}$ and $g_i$ stands for its $i$-th entry, and $\beta_g$ is the clipping threshold for the norm of $\boldsymbol{g}$. Our Huber regularization differs from [17]. First, in [17] parameters are *coupled* with each other when clipping the gradient, but we *decouple* the gradient clipping for different parameters. For example, if the gradient of a certain parameter excesses the clipping threshold, [17] will rescale the gradients of all the parameters, but we only clip the gradient of the specific parameter. Moreover, [17] down-scales the gradients of all the parameters in the model with a *fixed* scaling factor, but we *adaptively* clip the gradient for each single parameter according to its importance for old classes. Compared with [17] which may unnecessarily clip some parameters and thus limit the capacity of the entire model, our method is more flexible.

**Discussion.** The Huber regularization proposed here may seem simple, but it is based on our key findings that certain parameters own extremely large gradients during incremental fine-tuning. Through the Huber regularization, we equivalently adopt adaptive gradient clipping for the individual parameter to alleviate the unstable training issue of EWC in incremental detection. It also differs from and significantly outperforms the traditional gradient clipping method [17] (Fig. 5).

## VI. EXPERIMENTS

Based on the above adaptations, *i.e.* pseudo annotation and Huber regularization, we introduce our incremental object detection framework IncDet as shown in Fig. 1. In practice, we **first** train a base model that can detect multiple base classes. **Then** when new classes arrive, we produce the old class pseudo annotations which are corrected by the new class manual labels. **Next** we incrementally fine-tune the base model with pseudo annotations and Huber regularization, endowing it with the ability to detect the new classes and not to forget the old classes catastrophically. **Finally** the importance weight for each model parameter is updated to embrace more new classes. Note that the importance update step is merged into the *incremental fine-tuning* in Fig. 1 for compactness. Our framework is flexible thanks to its Bayesian online learning nature. So the new classes can be seamlessly added sequentially or all at once. Algorithm 1 gives the pseudo-code for the incremental detection pipeline of our proposed IncDet. We implement our framework under both Fast R-CNN [18] and Faster R-CNN [19] to demonstrate its versatility. We first carry out ablation studies to analyze the effectiveness of different components of our framework and then conduct comprehensive comparisons with state-of-the-art methods [8], [21], [22], [23].

**Algorithm 1** Incremental detection pipeline of our IncDet

**Input:** old model $\mathcal{M}$, new class datasets $\{\mathcal{D}_t\}$, $t \geqslant 2$
**Output:** new model $\mathcal{M}'$
**repeat**
1. predict on $\mathcal{D}_t$ with $\mathcal{M}$ to obtain pseudo annotations $\mathcal{D}'_t$
2. fine-tune $\mathcal{M}$ on $\mathcal{D}'_t$ with Huber regularization to get $\mathcal{M}'$
3. update the importance weight $w_i$ as Eq. (11)
4. $\mathcal{M} \leftarrow \mathcal{M}'$, $t \leftarrow t+1$

**until** all new classes are added

### A. Datasets and metrics

We conduct our main experiments on the PASCAL VOC 2007 [20] detection benchmark, including 5011 and 4952 images from 20 classes in the trainval and test subsets, respectively. Training is done on the trainval split and testing is done on the test split with mAP@0.5 as the evaluation metric. We also conduct experiments on the COCO 2014 [5] dataset to demonstrate IncDet's effectiveness on the large-scale dataset. We train our model in the 80K images from the training set and test on the 5K minival split. COCO style mAP at 0.5 IoU and mean mAP (mmAP) among different IoU thresholds ranging from 0.5 to 0.95 are used as evaluation metrics.

### B. Implementation details

Following ILOD [8], we choose ResNet-50 [56] as the backbone network and Batch Normalization [57] is replaced with the frozen channel-wise affine layer. We also use Edge Boxes [58] to extract 2000 proposals from every image. We set the optimizer as SGD with momentum 0.9. Images are scaled to 600 pixels for the shorter side and the longer side is constrained to be less than 1000 pixels. Notice that in each training stage, only images containing at least one instance from the classes of interest are used, and bounding boxes corresponding to irrelevant classes are excluded from the annotations. We implement IncDet with PyTorch [59] and train on servers with 8 GPUs for all experiments.

### C. Ablation Studies

**Pseudo annotation** is proposed to overcome the missing annotations of old classes in the new images. A comparison of our method with and without pseudo annotations is shown in Fig. 3. It shows that paradoxes produced by the supervision in different training stages bring a huge negative impact on old classes. Masking regions of old classes consistently and substantially leads to recall and thus AP improvement of old classes. By utilizing pseudo annotations, we can further boost the performance of old classes. Surprisingly, APs of some new classes, such as *plant*, *sheep* and *train*, are also improved remarkably. We believe this is because pseudo annotations of the old classes can prevent the model from misclassifying old class objects as the new classes, and hence reduce false alarms of the new classes during testing.

One concern about the pseudo annotation is how false detections of the old model affect the performance. In Tab. III we investigate the model performance when varying the detection score threshold $\sigma$. Predicted boxes whose scores lower than $\sigma$ are excluded from the pseudo annotations, and smaller $\sigma$ leads to more false detections. As shown, the mAP decreases as the number of false detections increases but the performance is fairly robust to the exact setting of $\sigma$. The last row with $\sigma = 1.0$ means not to use pseudo annotations and thus leads to much worse final performance.

TABLE III
EFFECT OF THE SCORE THRESHOLD $\sigma$ ON MAP (%) OF OLD (15), NEW (5) AND ALL (20) CLASSES WHEN 5 CLASSES ARE ADDED **ALL AT ONCE**.

| $\sigma$ | old | new | all |
|---|---|---|---|
| 0.3 | 63.0 | 56.5 | 61.4 |
| 0.4 | 63.2 | 57.5 | 61.8 |
| 0.5 | 63.5 | 57.7 | 62.1 |
| 0.6 | 63.5 | 57.9 | 62.1 |
| 0.7 | 63.5 | 57.6 | 62.1 |
| 0.8 | 63.7 | 58.1 | 62.3 |
| 0.9 | **64.5** | **58.3** | **63.0** |
| 1.0 | 39.6 | 56.7 | 43.8 |

**The Huber regularization** is extremely important in our approach, without which we cannot train the network successfully. We have tried linearly warming up [60] the learning rate to avoid the gradient explosion issue but still failed, as the quadratic loss quickly goes out of control in the warm-up phase. The effect of Huber regularization is shown in Fig. 5, where gradient clipping [17] is also compared. Note that the clipping threshold $\beta_g$ in Eq. (15) is adjusted properly so that the quadratic loss matches the Huber regularization loss and the gradient explosion vanishes. Compared with the quadratic loss both with and without gradient clipping [17], we are able to obtain better old and new class trade-off and thus superior final mAP across all learned classes.

TABLE IV
EFFECT OF $\lambda$ ON MAP (%) OF OLD (15), NEW (5) AND ALL (20) CLASSES WHEN 5 NEW CLASSES ARE ADDED **ALL AT ONCE**. RESULTS ARE OBTAINED VIA TRAINING ON THE TRAINING SET AND TESTING ON THE VALIDATION SET.

| $\lambda$ | old | new | all |
|---|---|---|---|
| 10 | 61.2 | 58.7 | 60.6 |
| 30 | 62.7 | **59.0** | 61.8 |
| 100 | 64.5 | 58.3 | **63.0** |
| 300 | 65.6 | 54.8 | 62.9 |
| 1000 | **66.1** | 51.8 | 62.5 |

**Impact of the balancing parameter $\lambda$.** As shown in Fig. 6, with our Huber Regularization, increasing $\lambda$ would consistently boost the APs of the old classes. But the APs of the new classes will be harmed because of the stronger intransigence of the model. We can even fine-tune the model stably with a huge $\lambda$ such as 1000 without gradient explosion. The best trade-off between old and new classes is achieved when $\lambda = 100$. For fair comparison with previous work [8], we investigate the effect of $\lambda$ when training on the trainval set and testing on the test set. In practice, to completely avoid the use of the test set, we can choose the best hyper-parameters (*e.g.*, $\lambda$) by training on the training set and testing on the validation set (results shown in Tab. IV). Then we can use the chosen optimal $\lambda$ to train on the combination of the training and validation (trainval) set to obtain the final model. Comparing Fig. 6 and Tab. IV, we are able to get the best $\lambda = 100$

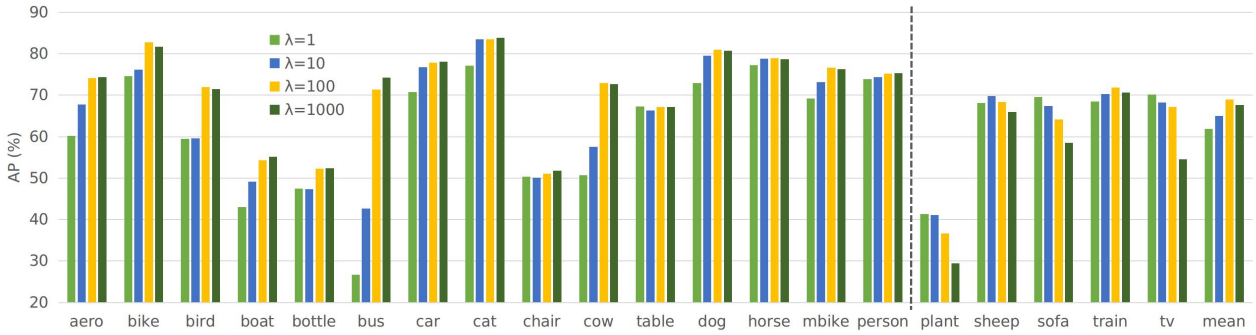



Fig. 6. Effect of $\lambda$ on the AP for each class and the mAP across all seen classes after incremental training, where we originally train the network on the first 15 classes (left to dashed line) and then fine-tune it with the remaining 5 ones (right to dashed line) **all at once**.

TABLE VI
COMPARISON OF MAP (%) ON VOC 2007 TEST SET WHEN 5 CLASSES ARE ADDED **SEQUENTIALLY**. NUMBERS IN EACH ROW REPRESENT MAP FOR OLD, NEW AND ALL CLASSES, AFTER ADDING A SINGLE CLASS ACCORDING TO THE ALPHABETICAL ORDER. COLUMNS WITH "-B" SUFFIX ARE THE RESULTS OF JOINT TRAINING ON ALL SEEN CLASSES, SHOWN AS UPPER BOUNDS FOR INCREMENTAL DETECTION. COLUMNS WITHOUT SUFFIX SHOW THE MAP AFTER INCREMENTAL FINE-TUNING. COLUMNS WITH "-△" SUFFIX SHOW MAP DROP AFTER ADDING CLASSES W.R.T. THE PREVIOUS BOUNDS, THE LOWER THE BETTER.

| added class | methods | old-B | new-B | all-B | old | new | all | old-△ | new-△ | all-△ |
|---|---|---|---|---|---|---|---|---|---|---|
| 16 (plant) | ILOD [8] | 70.9 | 47.7 | 69.5 | 69.7 | 26.3 | 67.0 | 1.2 | 21.4 | 2.5 |
| | IncDet-Fast | 74.0 | 45.2 | 72.2 | 73.3 | 32.1 | 70.7 | 0.7 | **13.1** | 1.5 |
| | IncDet-Faster | 75.1 | 47.7 | 73.3 | 75.4 | 33.3 | 72.7 | **-0.3** | 14.4 | **0.6** |
| 17 (sheep) | ILOD [8] | 70.9 | 58.5 | 69.4 | 67.4 | 37.4 | 63.9 | 3.5 | 21.2 | 5.5 |
| | IncDet-Fast | 74.0 | 59.8 | 72.3 | 72.3 | 45.8 | 69.2 | 1.6 | **14.0** | 3.1 |
| | IncDet-Faster | 75.1 | 60.9 | 73.4 | 74.2 | 42.3 | 70.4 | **0.9** | 18.6 | **3.0** |
| 18 (sofa) | ILOD [8] | 70.9 | 60.9 | 69.2 | 66.4 | 42.5 | 62.5 | 4.5 | 18.3 | 6.7 |
| | IncDet-Fast | 74.0 | 63.3 | 72.2 | 71.5 | 46.5 | 67.3 | 2.5 | **16.9** | 4.9 |
| | IncDet-Faster | 75.1 | 65.0 | 73.4 | 73.4 | 47.6 | 69.1 | **1.7** | 17.4 | **4.3** |
| 19 (train) | ILOD [8] | 70.9 | 64.9 | 69.6 | 66.0 | 48.8 | 62.4 | 4.9 | 16.1 | 7.2 |
| | IncDet-Fast | 74.0 | 65.9 | 72.3 | 71.2 | 53.4 | 67.4 | 2.8 | **12.6** | 4.9 |
| | IncDet-Faster | 75.1 | 69.0 | 73.8 | 72.8 | 54.8 | 69.0 | **2.3** | 14.3 | **4.8** |
| 20 (tv) | ILOD [8] | 70.9 | 66.7 | 69.8 | 66.0 | 51.6 | 62.4 | 4.9 | 15.0 | 7.4 |
| | IncDet-Fast | 74.0 | 67.4 | 72.3 | 70.6 | 53.4 | 66.3 | 3.4 | **14.1** | **6.0** |
| | IncDet-Faster | 75.1 | 69.9 | 73.8 | 72.2 | 53.7 | 67.6 | **2.9** | 16.2 | 6.2 |

consistently, indicating the robustness of the hyper-parameter tuning. Note that the results in Tab. IV are lower than those in Fig. 6 because less training data is used (the validation set is excluded).

TABLE V
EFFECT OF THE CLIPPING THRESHOLD ON MAP (%) OF OLD (15), NEW (5) AND ALL (20) CLASSES WHEN 5 CLASSES ARE ADDED **ALL AT ONCE**.

| $\beta$ | old | new | all | $\beta_g$ | old | new | all |
|---|---|---|---|---|---|---|---|
| $3\times10^{-7}$ | 62.3 | 63.5 | 62.6 | $10^{0}$ | **69.5** | 59.8 | **67.1** |
| $10^{-6}$ | 66.2 | **64.1** | 65.7 | $3\times10^{0}$ | 69.2 | 60.2 | 66.9 |
| $3\times10^{-6}$ | 68.9 | 62.9 | 67.4 | $10^{1}$ | 69.1 | 59.9 | 66.8 |
| $10^{-5}$ | 71.4 | 61.6 | **69.0** | $3\times10^{1}$ | 69.4 | 60.1 | **67.1** |
| $3\times10^{-5}$ | 71.6 | 60.4 | 68.8 | $10^{2}$ | 69.0 | **60.5** | 66.9 |
| $10^{-4}$ | **71.7** | 57.9 | 68.2 | $3\times10^{2}$ | 69.2 | 60.3 | 67.0 |
| $3\times10^{-4}$ | 71.5 | 58.1 | 68.1 | $10^{3}$ | 69.1 | 59.9 | 66.8 |

(a) Huber Regularization (b) Gradient Clipping [17]

**Impact of the clipping threshold $\beta$.** Generally speaking, as $\beta$ increases, mAP of the old classes increases and that of the new classes decreases (shown in Tab. V (a)). Since we only restrict the gradient of the regularization term not to cause gradient explosion, the actual gradient for each parameter comes from both the regularization and the likelihood term where the gradient is kept unconstrained. When $\beta$ is too small, the incremental fine-tuning degrades to the naïve fine-tuning without regularization as the regularization loss approaches zero, so the model will forget the old classes. However when $\beta$ is too large, the gradient cannot be constrained effectively and thus leads to worse performance on new classes. We fix $\beta$ to $10^{-5}$ in the rest experiments for its best trade-off. For a complete comparison, we also study the effect of the clipping threshold $\beta_g$ in gradient clipping [17] in Tab. V (b). Although gradient clipping can alleviate the gradient explosion issue, its best performance 67.1% lags behind that of our Huber regularization 69.0%, showing the superiority of our adaptive clipping method.

### *D. Results on VOC*

Now we compare our proposed IncDet with the previous state-of-the-art methods [8], [21], [22], [23]. The theoretical upper bounds for incremental detection are the results of joint training with all the seen classes. So we compare the methods by both the absolute mAP and the mAP drop w.r.t. these bounds after incremental fine-tuning.

**Add classes sequentially.** Tab. VI and Fig. 7 (b) compare our method with ILOD [8] when we first train on 15 old

TABLE VII
COMPARISON OF MAP (%) ON VOC 2007 TEST SET WHEN **GROUPS** OF CLASSES ARE ADDED **SEQUENTIALLY**. EACH ROW REPRESENTS MAP FOR THE **FOUR** GROUPS OF CLASSES AND ALL CLASSES, AFTER ADDING A GROUP CONSISTING OF 5 CLASSES. COLUMNS WITH "-B" SUFFIX ARE THE RESULTS OF JOINT TRAINING ON ALL SEEN CLASSES, SHOWN AS UPPER BOUNDS FOR INCREMENTAL DETECTION. COLUMNS WITHOUT SUFFIX SHOW THE MAP AFTER INCREMENTAL FINE-TUNING. COLUMNS WITH "-△" SUFFIX SHOW MAP DROP AFTER ADDING CLASSES W.R.T. THE PREVIOUS BOUNDS, THE LOWER THE BETTER.

| added classes | methods | a-B | b-B | c-B | d-B | all-B | a | b | c | d | all | a-△ | b-△ | c-△ | d-△ | all-△ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 – 10 | ILOD [8] | 65.2 | 73.2 | | | 69.2 | 44.8 | 59.2 | | | 52.0 | 20.4 | 14.0 | | | 17.2 |
| | CIFRCN [23] | 65.2 | 73.2 | | | 69.2 | 43.8 | 71.2 | | | 57.5 | 21.4 | 2.0 | | | 11.7 |
| | IncDet-Fast | 68.4 | 76.2 | | | 72.3 | 62.4 | 74.5 | | | 68.4 | **6.0** | **1.7** | | | **3.9** |
| | IncDet-Faster | 70.9 | 78.2 | | | 74.5 | 63.7 | 75.4 | | | 69.6 | 7.2 | 2.7 | | | 4.9 |
| 11 – 15 | ILOD [8] | 66.2 | 71.3 | 77.3 | | 71.6 | 49.0 | 43.4 | 48.6 | | 47.0 | 17.2 | 27.9 | 28.7 | | 24.6 |
| | CIFRCN [23] | 66.2 | 71.3 | 77.3 | | 71.6 | 35.3 | 49.0 | 68.4 | | 50.9 | 30.9 | 22.3 | **8.9** | | 20.7 |
| | IncDet-Fast | 68.5 | 75.7 | 77.8 | | 74.0 | 58.8 | 67.6 | 68.6 | | 65.0 | 9.7 | 8.1 | 9.2 | | 9.0 |
| | IncDet-Faster | 71.6 | 76.1 | 79.3 | | 75.7 | 62.4 | 70.6 | 69.9 | | 67.6 | **9.2** | **5.5** | 9.4 | | **8.1** |
| 16 – 20 | ILOD [8] | 65.1 | 70.4 | 76.4 | 67.8 | 69.9 | 40.2 | 35.5 | 42.5 | 38.8 | 39.3 | 24.9 | 34.9 | 33.9 | 29.0 | 30.6 |
| | CIFRCN [23] | 65.1 | 70.4 | 76.4 | 67.8 | 69.9 | 34.6 | 44.1 | 55.6 | 59.6 | 48.5 | 30.5 | 26.3 | 20.8 | **8.2** | 21.4 |
| | IncDet-Fast | 68.8 | 75.4 | 77.7 | 67.4 | 72.4 | 55.8 | 61.9 | 66.0 | 57.0 | 60.2 | 13.0 | 13.5 | 11.8 | 10.4 | 12.2 |
| | IncDet-Faster | 70.4 | 75.6 | 79.1 | 69.9 | 73.8 | 59.2 | 66.0 | 68.5 | 56.6 | 62.6 | **11.2** | **9.6** | **10.6** | 13.3 | **11.2** |

TABLE VIII
COMPARISON OF MAP (%) ON VOC 2007 TEST SET WHEN **5 OR 10** CLASSES ARE ADDED **ALL AT ONCE**. NUMBERS IN EACH ROW REPRESENT MAP FOR OLD, NEW AND ALL CLASSES, AFTER ADDING A SET OF NEW CLASSES SIMULTANEOUSLY. COLUMNS WITH "-B" SUFFIX ARE THE RESULTS OF JOINT TRAINING ON ALL SEEN CLASSES, SHOWN AS UPPER BOUNDS FOR INCREMENTAL DETECTION. COLUMNS WITHOUT SUFFIX SHOW THE MAP AFTER INCREMENTAL FINE-TUNING. COLUMNS WITH "-△" SUFFIX SHOW MAP DROP AFTER ADDING CLASSES W.R.T. THE PREVIOUS BOUNDS, THE LOWER THE BETTER.

| added classes | methods | old-B | new-B | all-B | old | new | all | old-△ | new-△ | all-△ |
|---|---|---|---|---|---|---|---|---|---|---|
| 16 – 20 | ILOD [8] | 70.9 | 66.7 | 69.8 | 68.3 | 58.4 | 65.9 | 2.6 | 8.2 | 3.9 |
| | IncDet-Fast | 74.0 | 67.4 | 72.4 | 71.4 | 61.6 | 69.0 | 2.6 | **5.8** | **3.4** |
| | IncDet-Faster | 75.1 | 69.9 | 73.8 | 72.7 | 63.5 | 70.4 | **2.3** | 6.5 | **3.4** |
| 11 – 20 | ILOD [8] | 68.4 | 71.3 | 69.8 | 63.2 | 63.1 | 63.1 | 5.2 | 8.1 | 6.7 |
| | DMC [21] | 75.3 | 74.0 | 74.7 | 70.5 | 66.2 | 68.3 | 4.8 | 7.9 | 6.4 |
| | RILOD [22] | 75.3 | 74.0 | 74.7 | 67.5 | 68.3 | 67.9 | 7.9 | 5.7 | 6.8 |
| | IncDet-Fast | 72.1 | 72.6 | 72.4 | 68.0 | 71.2 | 69.6 | 4.1 | **1.4** | **2.8** |
| | IncDet-Faster | 73.0 | 74.5 | 73.8 | 69.7 | 71.8 | 70.8 | **3.3** | 2.7 | 3.0 |

classes and then add 5 new classes sequentially. They show that our method instantiated with Fast R-CNN and Faster R-CNN outperforms the approach based on knowledge distillation [53] consistently, in terms of both the final performance and the performance drop compared with the upper bounds. We notice that our upper bound is a bit higher than ILOD [8] and this may be due to different implementation infrastructure (e.g. PyTorch vs. TensorFlow, multi- vs. single-gpu training). As we are concerned about the performance drop after incremental fine-tuning, so these results can still be compared reasonably. Both methods perform well on memorizing old classes but struggle to detect new classes. Overfitting on new category images can be the reason, since only hundreds of images are added for each class. We also perform adding 10 classes sequentially and compare the results with ILOD [8] (Fig. 7 (a)). The gap between the solid lines is almost always larger than that of the dashed lines, which means our mAP drop is consistently less than ILOD [8].

**Add groups of classes sequentially.** To compare fairly with the recent method CIFRCN [23], we follow their settings and split the 20 classes in the VOC dataset to 4 groups (a, b, c, and d) with 5 classes in each group. The incremental fine-tuning is conducted by adding one group of classes at a time until all 20 classes are learned. The results are shown in Tab. VII, where our method significantly outperforms its counterpart in terms of the mAP for each group, the mAP for all classes and the mAP drop compared with joint training upper bounds, showing that our method is also superior in the group-wise incremental detection setting.

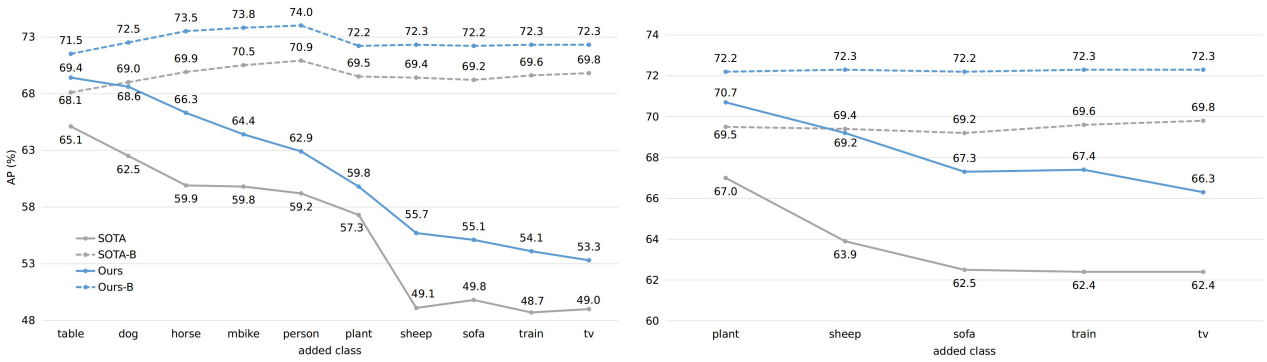
**Add classes all at once.** We report the results of adding 5 and 10 classes all at once in Tab. VIII. Compared with the auxiliary-based methods implemented with either proposal-based [8] or proposal-free framework [21], [22], our approach performs substantially better regarding mAP for new classes and meanwhile obtains comparable or better mAP for old ones, indicating we can leverage the capacity of the network to a greater extent. Besides, comparing Tab. VI and Tab. VIII, we see that adding multiple classes simultaneously manifests better mAP than adding a single class each time.

TABLE IX
COMPARISON OF MAP (%) ON COCO MINIVAL SPLIT. MAP OF ALL 80 CLASSES ARE SHOWN IN THIS TABLE. COLUMNS WITH "-B" SUFFIX ARE THE RESULTS OF JOINT TRAINING ON ALL SEEN CLASSES, SHOWN AS UPPER BOUNDS FOR INCREMENTAL DETECTION. COLUMNS WITHOUT SUFFIX SHOW THE MAP AFTER INCREMENTAL FINE-TUNING. COLUMNS WITH "-△" SUFFIX SHOW MAP DROP AFTER ADDING CLASSES W.R.T. THE PREVIOUS BOUNDS, THE LOWER THE BETTER.

| methods | .5-B | .5:.95-B | .5 | .5:.95 | .5-△ | .5:.95-△ |
|---|---|---|---|---|---|---|
| ILOD [8] | 38.1 | 22.6 | 37.4 | 21.3 | **0.7** | 1.3 |
| IncDet-Fast | 42.8 | 25.5 | 41.9 | 25.5 | 0.9 | 0.0 |
| IncDet-Faster | 50.4 | 29.6 | 49.3 | 29.7 | 1.1 | **-0.1** |

### *E. Results on COCO*

To validate the effectiveness of our proposed IncDet on the challenging large-scale dataset, we also conduct incremental




(a) adding 10 classes (b) adding 5 classes

Fig. 7. Comparison of AP for each new class between ILOD [8] and our IncDet after adding new classes **sequentially**, the dashed lines indicate joint training upper bounds and the solid lines are the results from incremental fine-tuning.

TABLE X
COMPARISON OF MAP (%) ON VOC 2007 TEST SET WHERE THE BASE MODEL IS TRAINED ON THE FIRST 10 VOC CLASSES ON VOC IMAGES AND FINE-TUNED WITH THE REST 10 VOC CLASSES ON **VOC OR COCO** IMAGES. "VOC" COLUMNS REPRESENT MAPS AFTER INCREMENTAL FINE-TUNING ON IMAGES FROM IN-DOMAIN DATASET (VOC), "COCO" COLUMNS ARE THE RESULTS AFTER INCREMENTAL FINE-TUNING ON IMAGES FROM CROSS-DOMAIN DATASET (COCO). COLUMNS WITH "-B" SUFFIX ARE THE RESULTS OF JOINT TRAINING ON ALL SEEN CLASSES WITH IMAGES FROM VOC, SHOWN AS UPPER BOUNDS FOR INCREMENTAL DETECTION. COLUMNS WITHOUT SUFFIX SHOW THE MAP AFTER INCREMENTAL FINE-TUNING. COLUMNS WITH "-△" SUFFIX SHOW MAP DROP AFTER ADDING CLASSES W.R.T. THE PREVIOUS BOUNDS, THE LOWER THE BETTER.

| methods | old-B | new-B | all-B | VOC | | | | | | COCO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | old | new | all | old-△ | new-△ | all-△ | old | new | all | old-△ | new-△ | all-△ |
| ILOD [8] | 68.4 | 71.3 | 69.8 | 63.2 | 63.1 | 63.1 | 5.2 | 8.2 | 6.7 | 61.4 | 48.2 | 54.8 | 7.0 | 23.1 | 15.0 |
| IncDet-Fast | 72.1 | 72.6 | 72.4 | 68.0 | 71.2 | 69.6 | 4.1 | **1.4** | **2.8** | 70.9 | 75.0 | 72.9 | 1.2 | **-2.4** | -0.5 |
| IncDet-Faster | 73.0 | 74.5 | 73.8 | 69.7 | 71.8 | 70.8 | **3.3** | 2.7 | 3.0 | 73.8 | 76.1 | 74.9 | **-0.8** | -1.6 | **-1.1** |

detection experiments on the COCO dataset and compare it with previous work [8]. The base model is trained on the first 40 classes and then fine-tuned with the rest 40 ones as done in [8]. Tab. IX compares our method with ILOD [8]. It shows that IncDet implemented with Fast R-CNN and Faster R-CNN only declines slightly in terms of mAP@0.5 w.r.t. the joint training upper bounds with all seen classes. And it achieves comparable or even better results in terms of mean mAP across different IoU thresholds compared with its corresponding joint training upper bounds, which suggests that our method can also overcome catastrophic forgetting on the large-scale dataset. Compared with ILOD [8], our method achieves comparable performance drop in terms of mAP@0.5, and much less performance drop in terms of average mAP across different IoU thresholds (0.5 to 0.95). This indicates our method produces more accurate bounding boxes. Also, thanks to the versatility of our IncDet, we can leverage advanced detection methods such as Faster R-CNN and thus achieve much better final performance in challenging datasets.

### F. Results of Cross-Domain

Since we are interested in domain drift during incremental detection, where the datasets used for old model training and incremental fine-tuning come from different domains, we initially train the old model on the first 10 classes from VOC, then fine-tune it on the rest 10 classes either from the VOC or COCO dataset, and finally test the new model on the VOC test set. Tab. X shows that the performance obtained via incrementally fine-tuning on COCO with our method is better than that of fine-tuning on VOC, even better than that of jointly-training on VOC with all seen classes. We believe that this is because COCO has much more appearance variations such as scale, pose and viewpoint than VOC. Compared with ILOD [8], our IncDet achieves much less performance drop in both in-dataset and cross-dataset scenarios. This reveals the generalization ability of our method both for the in-domain and cross-domain setting.

### G. Efficiency Investigation

As incremental detection aims at incrementally adapting the old model to detect new classes, last we study the efficiency of our IncDet in terms of both time and memory during fine-tuning and testing. Comparison is also made with previous knowledge distillation based method ILOD [8].

**Testing efficiency.** With the number of classes increasing, the model size grows as parameters for new classes are added. So we care about the time spent and memory consumed at testing time when more classes are involved. We train the base model on the first 5 VOC classes and then add the rest 15 classes to the model, each time 5 classes are added. As can be seen from Tab. XI adding 5 classes only spends about 1 more millisecond and consumes about 0.2 MB more memory. Compared with 0.260s inference time and 2382.6 MB memory consumption for the 5 old classes, our method is efficient when adding classes. This is intuitive because we only add about 51.2K parameters for the new classes in the last fully-connected layers (including region classification and

box regression) of the network, which are negligibly light-weight compared with the backbone network ResNet-50 that has around 25.6M parameters. All the time and memory are measured on a GTX 1080 Ti GPU.

TABLE XI
TIME SPENT AND MEMORY CONSUMED AT TESTING TIME FOR A SINGLE IMAGE WHEN THE NUMBER OF DETECTED CLASSES INCREASES.

| #classes | 5 | 10 | 15 | 20 |
|---|---|---|---|---|
| time (s) | 0.260 | 0.261 | 0.262 | 0.263 |
| memory (MB) | 2382.6 | 2382.8 | 2383.0 | 2383.2 |

**Training efficiency.** Compared with the *auxiliary-based* methods [8], [21], [22], [23] which need to conduct forward passes of the teacher networks during fine-tuning on new data for about a *dozen* of epochs, our EWC-based method is totally *auxiliary-free* and only needs to go through the new data with the optimized model at the end of each fine-tuning phase for a *single* epoch to update the importance weight of each parameter in the network. The extra epoch for importance update is the same as normal epochs for incremental fine-tuning, so the time spent for importance update is less than one tenth of the usual training time where more than ten epochs are typically needed. Tab. XII verifies that our IncDet is much more efficient and economic in terms of computation and memory. Specifically, our IncDet based on Fast R-CNN needs less than half of the time and memory compared with ILOD [8]. ILOD exploits one auxiliary teacher network during fine-tuning and thus needs to maintain a frozen copy of the whole model in memory while we do not. IncDet based on Faster R-CNN needs more time and memory compared with the Fast R-CNN variant because it includes an RPN module for online proposal generation, but it is still more efficient than the counterpart ILOD [8]. Moreover, our method is more efficient than DMC [21] where two different teacher networks corresponding to the old and new classes are used during incremental fine-tuning.

TABLE XII
TIME SPENT AND MEMORY CONSUMED WHEN FINE-TUNING ON 10 NEW CLASSES WITH 2 IMAGES IN A BATCH.

| methods | time (s) | memory (MB) |
|---|---|---|
| ILOD [8] | 0.98 | 9579 |
| IncDet-Fast | 0.41 | 4483 |
| IncDet-Faster | 0.70 | 4885 |

## VII. CONCLUSIONS

In this paper, we propose IncDet, an EWC-based incremental detection framework, which can theoretically enable any conventional detection methods to learn to detect new classes in a recursive and efficient manner. Key issues causing the failure of directly applying EWC to object detection are identified by controlled experiments and addressed by our simple but effective *pseudo annotation* and *Huber regularization*. Comprehensive experiments show that our general approach implemented with either Fast R-CNN or Faster R-CNN outperforms the previous best methods [8], [21], [22], [23] while requiring less memory and computation during incremental fine-tuning.

For future work, we may extend incremental detection to the setting of few shot detection [61], where the incremental fine-tuning only requires few new class images and annotations. In this way, we can further reduce the cost needed for manual labeling and the time spent on incremental fine-tuning. In addition, it is valuable to develop techniques on boosting the incremental detection performance especially when the new classes are added sequentially. Moreover, how to apply variants [43], [15] of EWC to incremental detection is still an open problem.

## REFERENCES

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.
[2] R. Alp Güler, N. Neverova, and I. Kokkinos, "DensePose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7297–7306.
[3] L. Qiang, W. Zhang, L. Hongliang, and K. N. Ngan, "Hybrid human detection and recognition in surveillance," *Neurocomputing*, vol. 194, pp. 10–23, 2016.
[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Cision and Pattern Recognition*, 2009, pp. 248–255.
[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.
[6] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.
[7] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.
[8] 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*, 2017, pp. 3400–3409.
[9] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.
[10] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.
[11] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
[12] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3987–3995.
[13] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.
[14] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *International Conference on Learning Representations*, 2018.
[15] H. Ritter, A. Botev, and D. Barber, "Online structured Laplace approximations for overcoming catastrophic forgetting," in *Advances in Neural Information Processing Systems*, 2018, pp. 3742–3752.
[16] M. Opper and O. Winther, "A Bayesian approach to on-line learning," *On-line learning in neural networks*, pp. 363–378, 1998.
[17] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[18] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.
[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.
[20] 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.
[21] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1131–1140.
[22] D. Li, S. Tasci, S. Ghosh, J. Zhu, J. Zhang, and L. Heck, "RILOD: near real-time incremental learning for object detection at the edge," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. ACM, 2019, pp. 113–126.
[23] Y. Hao, Y. Fu, Y.-G. Jiang, and Q. Tian, "An end-to-end architecture for class-incremental object detection with knowledge distillation," in *IEEE International Conference on Multimedia and Expo (ICME'19)*, 2019.
[24] L. Fu, H.-H. Hsu, and J. C. Principe, "Incremental backpropagation learning networks," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 757–761, 1996.
[25] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, 2001, pp. 409–415.
[26] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Sympoisum on Artificial Neural Networks (ESANN)*, 2016.
[27] T. Schaul, H. van Hasselt, J. Modayil, M. White, A. White, P.-L. Bacon, J. Harb, S. Mourad, M. Bellemare, and D. Precup, "The barbados 2018 list of open issues in continual learning," *arXiv preprint arXiv:1811.07004*, 2018.
[28] S. Farquhar and Y. Gal, "Towards robust evaluations of continual learning," in *Privacy in Machine Learning and Artificial Intelligence workshop, ICML*, 2018.
[29] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.
[30] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, 2011.
[31] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.
[32] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2283–2301, 2012.
[33] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832, 2018.
[34] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.
[35] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *International Conference on Learning Representations*, 2018.
[36] C. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 10–24, 2017.
[37] C. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1191–1204, 2018.
[38] Y. Xing, F. Shen, and J. Zhao, "Perception evolution network based on cognition deepening model—adapting to the emergence of new sensory receptor," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 3, pp. 607–620, 2015.
[39] C. Hou and Z.-H. Zhou, "One-pass learning with incremental and decremental features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 11, pp. 2776–2792, 2017.
[40] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352–1357, 2009.
[41] R. Zhang, Y. Lan, G.-b. Huang, and Z.-B. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 365–371, 2012.
[42] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.
[43] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.
[44] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *The European Conference on Computer Vision (ECCV)*, September 2018.
[45] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.
[46] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.
[47] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.
[48] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1476–1481, 2016.
[49] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
[50] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.
[51] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.
[52] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.
[53] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[54] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," *WACV/MOTION*, vol. 2, 2005.
[55] I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He, "Data distillation: Towards omni-supervised learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4119–4128.
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, 2015, pp. 448–456.
[58] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European Conference on Computer Vision*. Springer, 2014, pp. 391–405.
[59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
[60] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.
[61] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8420–8429.