Neural Networks 164 (2023) 38–48

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2022 Special Issue

Imitating the oracle: Towards calibrated model for class incremental learning

Fei Zhu, Zhen Cheng, Xu-Yao Zhang, Cheng-Lin Liu *

State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS), Institute of Automation of Chinese Academy of Sciences, Beijing 100190, China
School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing 100049, China

ARTICLE INFO

Article history:
Available online 23 April 2023

Keywords:
Class incremental learning
Continual learning
Lifelong learning

ABSTRACT

Class-incremental learning (CIL) aims to recognize classes that emerged in different phases. The joint-training (JT), which trains the model jointly with all classes, is often considered as the upper bound of CIL. In this paper, we thoroughly analyze the difference between CIL and JT in feature space and weight space. Motivated by the comparative analysis, we propose two types of calibration: feature calibration and weight calibration to imitate the oracle (ItO), i.e., JT. Specifically, on the one hand, feature calibration introduces deviation compensation to maintain the class decision boundary of old classes in feature space. On the other hand, weight calibration leverages forgetting-aware weight perturbation to increase transferability and reduce forgetting in parameter space. With those two calibration strategies, the model is forced to imitate the properties of joint-training at each incremental learning stage, thus yielding better CIL performance. Our *ItO* is a plug-and-play method and can be implemented into existing methods easily. Extensive experiments on several benchmark datasets demonstrate that *ItO* can significantly and consistently improve the performance of existing state-of-the-art methods. Our code is publicly available at https://github.com/Impression2805/ItO4CIL.

1. Introduction

Owing to the strong representation learning ability, deep neural networks have recently demonstrated impressive performance in various applications. The progress is mainly achieved based on the underlying assumption that the training data is stationary, independent and identically distributed. However, such an assumption can be unrealistic in practice because of the complex, dynamic and open world where new tasks or classes would emerge sequentially. For example, after deployment, an auto nomous car may encounter new objects that have to be recognized. Hence, there is a need for a machine learning system to learn new knowledge continually while maintaining the old knowledge, which is the goal of incremental learning (Delange et al., 2021; Maltoni & Lomonaco, 2019; Parisi et al., 2019) (IL, also known as lifelong learning or continual learning). Generally speaking, there are mainly three kinds of setting for IL (van de Ven & Tolias, 2019), i.e., domain-incremental learning (DIL), task-incremental learning (TIL) and class-incremental learning (CIL). For DIL, the label space of the tasks may not change, but the input-distribution (i.e., domain) is continually changing (Kirkpatrick et al., 2017; Simon et al., 2022; Tao, Hong, Chang, & Gong, 2020; Yang, Zhou, Zhan, Xiong, & Jiang, 2019). For example, an autonomous driving system has to survive in different environments where the observed domains come with an unpredictable sequence. For TIL and CIL, new classes continually appear, and the model has to recognize all seen classes. Specifically, as illustrated in Fig. 1, TIL assigns different classifiers for different tasks and needs task-identity (task-ID) at inference time, while CIL only has a single head that learns a unified classifier for all seen classes. This paper focuses on the more challenging and realistic CIL.

The major challenge of CIL is the catastrophic forgetting problem (Goodfellow, Mirza, Xiao, Courville, & Bengio, 2013): after learning new classes, the mode would suffer from serious performance degradation in previously learned classes. To alleviate this problem, some studies (Kirkpatrick et al., 2017; Zenke, Poole, & Ganguli, 2017) explicitly constrain the change of important parameters, while others (Li & Hoiem, 2018) use knowledge distillation (Hinton et al., 2015) to retain important knowledge implicitly. Data replay based methods (Rebuffi, Kolesnikov, Sperl, & Lampert, 2017) save and re-learn a small portion of old data when learning new classes, which serve as a strong baseline for CIL. However, there exists a natural imbalance problem: the number of training samples of new classes is far more than that of old

* Corresponding author at: State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS), Institute of Automation of Chinese Academy of Sciences, Beijing 100190, China.
E-mail addresses: zhufei2018@ia.ac.cn (F. Zhu), chengzhen2019@ia.ac.cn (Z. Cheng), xyz@nlpr.ia.ac.cn (X.-Y. Zhang), liucl@nlpr.ia.ac.cn (C.-L. Liu).

https://doi.org/10.1016/j.neunet.2023.04.010
0893-6080/

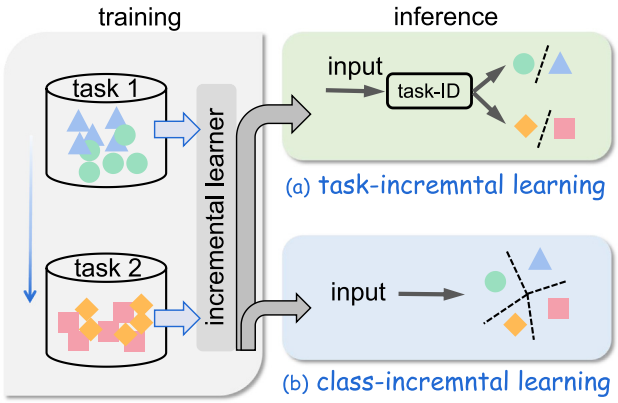

Fig. 1. Illustration of task-incremental learning (TIL) and class-incremental learning (CIL). TIL assigns different classifiers for different tasks and needs task-ID for inference, while CIL only has a single head that learns a unified classifier for all classes. This paper focuses on CIL.

classes. Consequently, a variety of techniques have been proposed to further mitigate the imbalance problem in data replay based CIL. For example, some methods (Ahn et al., 2021; Castro et al., 2018; Douillard et al., 2020; Lee, Lee, Shin, & Lee, 2019) include a fine-tuning step on a balanced subset of all classes at the end of each training phase. Other methods (Castro et al., 2018; Rebuffi et al., 2017; Zhao, Xiao, Gan, Zhang, & Xia, 2020) use post-hoc techniques to rectify the imbalance problem.

In this paper, we propose a novel perspective to tackle the forgetting problem in CIL by rethinking the different properties between CIL and joint-training (JT), which learns all classes jointly and is considered as the upper bound of CIL. Specifically, as shown in Fig. 2, we compare CIL with JT in both *feature space* and *weight space*. **(1) Feature space.** As illustrated in Fig. 2(a), In JT, all the classes are learned jointly with the balanced number of training samples. Therefore, the feature distributions of different classes are balanced. Besides, for each class, the feature distributions of training and test set are well matched. Therefore, the model can be well generalize to all classes. While in CIL, old and new classes are learned sequentially, and only a few samples of old classes are saved in memory. Consequently, the spanned feature distributions of old and new classes are imbalanced. Moreover, for each old class, the model can easily overfit the saved samples. Those two issues indicate severe feature deviation between the training and test instances in CIL. **(2) Weight space.** As illustrated in Fig. 2(b), in JT, with balanced and joint learning objective function, all classes could have small loss (i.e., R_{JT}) and the local minima is relatively flat. However, in CIL, learning new classes will result in a severe performance drop in old classes. This indicates worse weight smoothness: the loss increase ($R_2 - R_1$) is significant when updating θ_1 to θ_2.

Motivated by the above analysis, we propose to imitate the properties of JT during the incremental learning process. Therefore, the simple and effective *feature calibration* and *weight calibration* are proposed, as illustrated in Fig. 2. **(1) Feature calibration.** On the one hand, in feature space, to mimic the balanced (between different classes) and matched (between training and test set) feature distribution in JT, we propose deviation compensation to eliminate the feature deviation of old classes, thus maintaining the feature distribution of old classes. To this end, a self-paced linear transformation is applied to the output logits for each class during training, which encourages larger margins between old and new classes, as shown in Fig. 2(a). **(2) Weight calibration.** On the other hand, in parameter space, to mimic the minima that can well generalize to all classes in JT, we propose to smooth the weights of the model at each incremental learning stage (as shown in Fig. 2(a)), so that updating the model on new classes would lead to less forgetting of old knowledge. To this end, one simple technique is making the model robust to random noise injected into weights. To be more efficient and effective, we propose forgetting-aware adversarial weight perturbation. With large margin and flat minima properties, when learning new tasks, the new classes would be less overlapped with old classes, which can not only benefit new class generalization but also reduce the forgetting of old classes. Therefore, the proposed method can lead to a forward compatible effect that prepares the classifier to predict novel classes.

Through comprehensive experiments on several benchmark datasets, we verify that our plug-and-play method can consistently and significantly improve existing CIL approaches, achieving state-of-the-art performance. Our main contributions are summarized as follows:

- We explore and study how CIL differs from joint training (i.e., the oracle), and identify the crucial difference in both feature space and weight space. Therefore, we propose to improve CIL by imitating the oracle (ItO).
- The proposed ItO method consists of two simple and effective strategies to calibrate the feature distribution and model parameters, respectively.
- Extensive experiments demonstrate that our proposed method yields consistent improvements over previous state-of-the-art approaches.

The rest of this paper is organized as follows. In Section 2, we present related work. Section 3 provides the preliminaries of CIL. In Section 4, we describe the proposed ItO method in detail. In Section 5, we verify the effectiveness of our method by comprehensive experiments. Finally, Section 6 gives the concluding remarks.

2. Related work

We give a brief review of representative methods for deep learning based incremental learning. More information can be found in surveys (Lesort et al., 2020; Lomonaco & Maltoni, 2016; Maltoni & Lomonaco, 2019; Masana et al., 2020).

Non-exemplar based methods. Methods in this category do not store any old data. Parameter regularization such as EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), VCL (Nguyen, Li, Bui, & Turner, 2018), OWM (Zeng, Chen, Cui, & Yu, 2019), OGD (Farajtabar, Azizan, Mott, & Li, 2020) and Adam-NSCL (Wang, Li, Sun, & Xu, 2021) constrain the change of the important network parameters. Rather than directly avoiding forgetting in the parameter space, knowledge distillation (KD) (Hinton et al., 2015) based methods (Dhar et al., 2019; Li & Hoiem, 2018; Liu et al., 2020; Zhu, Zhang, & Liu, 2021) implicitly tackle the forgetting problem by keeping the model's input–output behavior on new data. Recently, another line of work, e.g., PASS (Zhu, Zhang, Wang, Yin, & Liu, 2021) and IL2A (Zhu, Cheng, Zhang, & Liu, 2021), constrain the distributions of old classes in the deep feature space to maintain previously learned decision boundary. They yield strong performance and become state-of-the-art non-exemplar CIL methods.

Generative replay based methods. Methods in this category generate and learn pseudo-samples of old classes when learning new classes. Early approaches (Kemker & Kanan, 2018; Shin, Lee, Kim, & Kim, 2017; Xiang, Fu, Ji, & Huang, 2019) simultaneously train an additional generative model to consolidate old knowledge. Then, when updating the model on new classes, pseudo-samples of old classes are generated for joint-training. Recently, some works (Smith et al., 2021; Yin et al., 2020) explore using the classification model itself to generate samples of old classes.

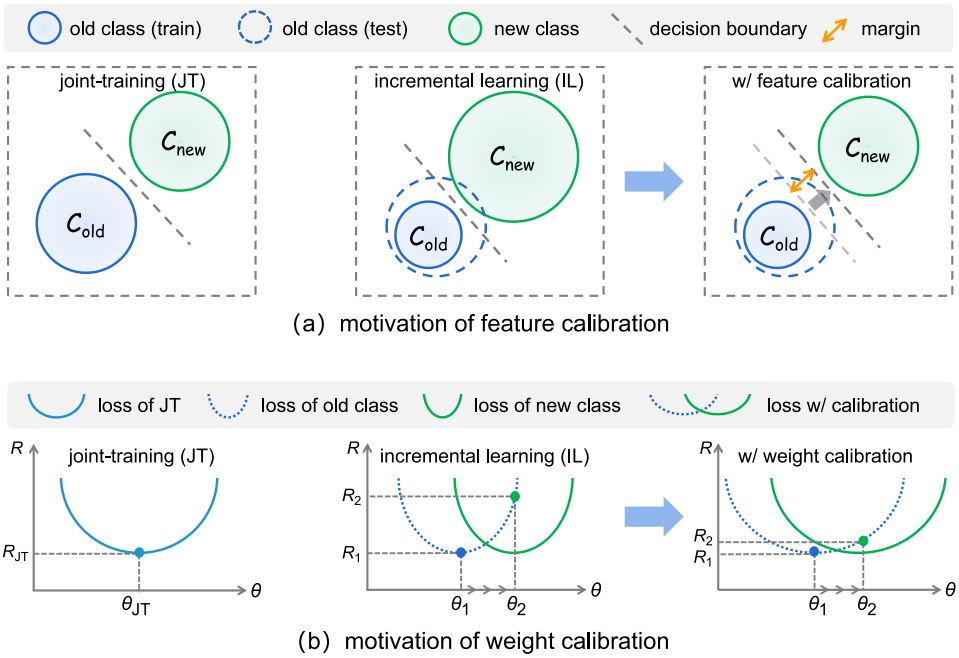

Fig. 2. Illustration of our motivation. (a) Feature calibration. Compared with JT, there exists a severe feature deviation between the training and test set for each old class in CIL. We propose deviation compensation to alleviate the feature deviation. (b) Weight calibration. In CIL, when updating the model (from θ_1 to θ_2), the loss of old classes is significantly increased (from R_2 to R_2). We calibrate an incremental learner by learning smoother weights.

Exemplar and bias correction based methods. Exemplar based methods (Douillard et al., 2020; Hou, Pan, Loy, Wang, & Lin, 2019; Rebuffi et al., 2017; Wu et al., 2019) alleviate forgetting by storing and replaying some training samples of each old class. Methods in this category have shown superior CIL performance. However, they suffer from imbalance problem: new classes often have much more training samples than that of old classes. As a result, the model can easily overfit the stored samples. To alleviate the imbalance problem, some methods like EEIL (Castro et al., 2018), UCIR (Hou et al., 2019), SS-IL (Ahn et al., 2021) and GD (Lee et al., 2019) use training-time strategies, while others like iCaRL (Rebuffi et al., 2017), BiC (Wu et al., 2019), IL2M (Belouadah & Popescu, 2019), WA (Zhao et al., 2020), and ScaIL (Belouadah & Popescu, 2020) leverage post-hoc techniques. For example, EEIL (Castro et al., 2018), GD (Lee et al., 2019) and PODnet (Douillard et al., 2020) include a fine-tuning step on a balanced subset of all classes at the end of each incremental learning stage. Rather than directly storing raw samples, Pellegrini, Graffieti, Lomonaco, and Maltoni (2020) proposed to store and replay activations volumes at some intermediate layer, which can reduce the computation and storage requirement. Flashcards (Gopalakrishnan, Singh, Fayek, Ramasamy, & Ambikapathi, 2022) leverages and replays some predefined random image patterns to capture the encoded knowledge of an incremental learning model.

Dynamic architectures. Many dynamic architecture based methods (Fernando et al., 2017; Hung et al., 2019) have been developed for TIL. Specifically, to prevent the forgetting of old tasks, the old parameters are frozen and new branches are allocated to learn new tasks. However, those methods are often impractical for CIL. Recently, DER (Yan, Xie, & He, 2021), FOSTER (Wang, Zhou, Ye, & Zhan, 2022) and Dytox (Douillard, Ramé, Couairon, & Cord, 2022) dynamically expand feature extractor to preserve old knowledge while learning new concepts, which yield strong CIL performance.

3. Preliminaries: Notations and problem statement

CIL involves many sequentially arrived tasks, and each new task consists of some new classes. Formally, for each incremental task t, the training data is donated by $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$, where N_t donate the number of training samples in task t, $\boldsymbol{x}_i^t$ is an sample in the input space $\mathcal{X}$ and $y_i^t \in \mathcal{C}_t$ is its corresponding target label. $\mathcal{C}_t$ is the class set of task t and the class sets of different task are disjoint, i.e., $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ if $i \neq j$. Following Douillard et al. (2020), Hou et al. (2019), Rebuffi et al. (2017), Wu et al. (2019), we assume an exemplar-memory $\mathcal{M}_{t-1}$ is used to store a tiny subset of old samples, i.e., $|\mathcal{M}_{t-1}| \ll N_t$. Therefore, at each incremental learning stage t, both the reserved old data and new data, i.e., $\mathcal{M}_{t-1} \cup \mathcal{D}_t$ are used to jointly train the model.

The classification model typically consists of a feature extractor and a unified classifier. Specifically, the feature extractor (e.g., deep convolutional neural network) maps the input into a feature vector in the deep feature space. The classifier is the final fully-connected layer with softmax option which produces a probability distribution as the prediction for each input. To facilitate analysis, we donate the deep neural network based model with two components: a feature extractor $f_{\boldsymbol{\theta}_t}(\cdot)$ and a unified linear classifier $\boldsymbol{w}_t$. Note that the unified classifier $\boldsymbol{w}_t$ includes the classification weights of all learned classes (i.e., $c \in \mathcal{C}_{1:t}$) by stage t. At inference time, the prediction for a test sample $\boldsymbol{x}_{\text{test}}$ is obtained by

$$\hat{y}_{\text{test}} = \arg\max_{c \in \mathcal{C}_{1:t}} \boldsymbol{w}_{t,c}^{\top} f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_{\text{test}}), \tag{1}$$

where $\boldsymbol{w}_t^{\top} f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_{\text{test}})$ represents the outputted logit vector (i.e., the score before softmax) by the model $\boldsymbol{\theta}_t$ for all learned class $\mathcal{C}_{1:t}$ so far .

4. Main method

To imitate the properties of JT in both feature space and parameter space, we propose ItO method which mainly consists of two simple and effective components. For each component, we introduce the motivation, and then describe the realization in detail.

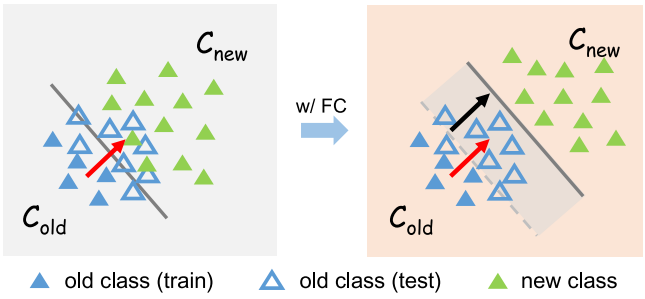

Fig. 3. Illustration of the effectiveness of our proposed feature calibration (FC). The red arrow represents feature deviation: the test data of old class tend to move to regions of new class. Feature calibration compensates for the effect by forcing large margin (black arrow) between old class and decision boundary. Thus, even after feature deviation, test samples of old class can still be correctly classified.

4.1. Feature calibration by deviation compensation

4.1.1. Motivation

An ideal way to remedy the catastrophic forgetting issue in CIL is to fully maintain the feature distribution of old classes when updating the model on new classes, which appears difficult with only a small fraction of old data available. Specifically, as illustrated in Fig. 2(a) and Fig. 3, the feature distribution of old and new classes are severely imbalanced, and the distributions of training and test set are mismatched in the feature space. Consequently, at inference time, many instances of old classes would be mapped to the positions near the decision boundaries, making it hard to correctly classify them. Consequently, the model cannot well generalize to old classes after learning new classes.

Since directly maintaining each old class's distribution is difficult, we therefore take an alternative way to design a training-time calibration strategy that can compensate for the effect of feature deviation of old classes. Our high-level idea is to increase the margin between the distribution of old classes and decision boundaries. In this way, even with some feature deviation, the test instances of old classes can still be mapped to the right decision region, as illustrated in Fig. 3. Technically, we propose to artificially *reduce* the logit values on old class nodes, i.e., $\boldsymbol{w}_{c,t}^{\top} f_{\theta_t}(\boldsymbol{x}_{\text{test}})$, $c \in \mathcal{C}_{1:t-1}$ during training, which could force or encourage the model itself to yield *larger* decision values for old classes. To this end, we design two different forms of calibration technique, i.e., additive feature calibration (AFC) and multiplicative feature calibration (MFC).

4.1.2. Realization

Additive feature calibration. Additive feature calibration reduces the logit values on old class nodes of a training sample by an additive term β, resulting in a modified cross-entropy loss as follows:

$$\ell_{ce-afc}((\boldsymbol{x}, y); \boldsymbol{\theta}_t, \boldsymbol{w}_t) = -\log\left(\frac{\exp[\boldsymbol{w}_{y,t}^{\top} f_{\theta_t}(\boldsymbol{x}) - \beta_y]}{\sum_c \exp[\boldsymbol{w}_{c,t}^{\top} f_{\theta_t}(\boldsymbol{x}) - \beta_c]}\right), \tag{2}$$

where $(\boldsymbol{x}, y) \in \mathcal{M}_{t-1} \cup \mathcal{D}_t$. If $c \in \mathcal{C}_t$, we set $\beta_c = 0$; if $c \in \mathcal{C}_{1:t-1}$, we set $\beta_c = -\log[(M_t + \lambda N_t)/m]$, in which M_t is the number of saved samples in $\mathcal{M}_{t-1}$, N_t is the number of training instances in new classes set $\mathcal{D}_t$, m donates the number of saved samples of each old class and λ is a hyperparameter. As can be seen, we have $\beta_c > 0$ for each old class and $\beta_c = 0$ for each new class.

Multiplicative feature calibration. Multiplicative feature calibration reduces the logit values on old class nodes via multiplication, leading to a modified cross-entropy loss as follows:

$$\ell_{ce-mfc}((\boldsymbol{x}, y); \boldsymbol{\theta}_t, \boldsymbol{w}_t) = -\log\left(\frac{\exp[\alpha_y \cdot \boldsymbol{w}_{y,t}^{\top} f_{\theta_t}(\boldsymbol{x})]}{\sum_c \exp[\alpha_c \cdot \boldsymbol{w}_{c,t}^{\top} f_{\theta_t}(\boldsymbol{x})]}\right), \tag{3}$$

If $c \in \mathcal{C}_t$, we set $\alpha_c = 1$; if $c \in \mathcal{C}_{1:t-1}$, we set $\alpha_c = m/n$, where n donates the number of samples in each new class. That is, we have $\alpha_c < 1$ for each old class and $\alpha_c = 1$ for each new class.

4.1.3. Discussion

Large margin effect. Our feature calibration method has the effect of enlarging the margin between old class and decision boundary. Specifically, for additive calibration, the large margin effect can be directly seen from Eq. (2), in which all old classes have the same large margin because each old class has the same number of saved samples in a memory buffer. While for multiplicative calibration, the margin is adaptive based on the output logit values. Concretely, donate the logit value on the old class node c as $o_{c,i} = \boldsymbol{w}_{c,t}^{\top} f_{\theta_t}(\boldsymbol{x}_i)$, then the calibrated value is $\alpha_c \cdot o_{c,i} = o_{c,i} - [o_{c,i}(1 - \alpha_c)]$. Therefore, the margin is $o_{c,i}(1 - \alpha_c)$, which depends on the original logit value $o_{c,i}$. Larger $o_{c,i}$ results in a larger distance to decision boundary, which is reasonable and desirable.

Comparison with other margin loss. In the area of face recognition, several kinds of margin loss such as L-softmax (Liu, Wen, Yu, & Yang, 2016), AM-softmax (Wang, Cheng, Liu, & Liu, 2018), CosFace (Wang, Wang, et al., 2018) and ArcFace (Deng, Guo, Xue, & Zafeiriou, 2019), have been proposed to enlarge the inter-class margin. Our feature calibration strategies are indeed different from them. (1) **Class-dependent.** In the previous work (Deng et al., 2019; Liu et al., 2016; Wang, Cheng, Liu, & Liu, 2018; Wang, Wang, et al., 2018) where only a single learning phase is involved and all classes are balanced, the margin between classes is chosen to be a class-independent constant. However, in our method, the margin is class-dependent and different for old and new classes in CIL. Specifically, in Eqs. (2) and (3), we only enlarge the margin between each old class to its decision boundary. (2) **Self-adaptive.** The value of the margin depends on the number of samples of old and new classes, and would change during the incremental learning process (see Section 4.1.2 for more details). While in Deng et al. (2019), Liu et al. (2016), Wang, Cheng, Liu, and Liu (2018), Wang, Wang, et al. (2018), the margin is a predefined constant. (3) **Effective.** As will be shown in Section 5.4, the class-independent margin in Deng et al. (2019), Liu et al. (2016), Wang, Cheng, Liu, and Liu (2018), Wang, Wang, et al. (2018) yields no improvement in CIL.

Comparison with other bias correction methods. As described in Section 2, many CIL methods leverage post-hoc strategy to correct the bias problem at the end of each IL phase. For example, some Ahn et al. (2021), Castro et al. (2018), Douillard et al. (2020), Lee et al. (2019) use balanced fine-tuning and others (Belouadah & Popescu, 2019; Wu et al., 2019; Zhao et al., 2020) use class weight normalization or alignment. While our proposed feature calibration technique can automatically learn a balanced feature space between old and new classes. Compared with another training-time bias correction method UCIR (Hou et al., 2019), our feature calibration is much more effective.

4.2. Weight calibration by forgetting-aware weight perturbation

4.2.1. Motivation

In JT, all classes are learned jointly and supposed to be well separated. However, in CIL, updating the model on new classes

would inevitably introduce weight shift, which results in significant performance drop of old classes. To facilitate the generalization ability in CIL, we propose to prepare for the weight shift in CIL by smoothing the weights. Particularly, CIL could benefit from smooth weights from two aspects: **(1) Less forgetting.** With smooth weights, the weight loss landscape (Li, Xu, Taylor, Studer, & Goldstein, 2018; Neyshabur, Bhojanapalli, McAllester, & Srebro, 2017) would be flat. That is, the loss would be uniformly low in the neighborhood area of each weight. Therefore, the loss of old classes would not increase significantly when learning new classes, resulting in less forgetting of previous knowledge. **(2) Better adaption.** With smooth weight space, the model trained on previous classes would be a better initialization for the next task, which would facilitate new class learning and in turn, mitigate the forgetting of old knowledge.

4.2.2. Realization

A common way to smooth the weight is to minimize the loss under weight perturbation. Specifically, small random noise can be added to the model parameters and the loss functions with different noise are minimized to find flat minima region. Formally, we can minimize the empirical loss given by

$$\ell_{ce-wc}((\boldsymbol{x}, y); \boldsymbol{\theta}_t, \boldsymbol{w}_t) = \frac{1}{K}\sum_{j=1}^{K} \ell\left((\boldsymbol{x}, y); \boldsymbol{\theta}_t + \boldsymbol{\epsilon}_j, \boldsymbol{w}_t\right), \tag{4}$$

where $\boldsymbol{\epsilon}_j$ is a noise vector sampled from a pre-defined noise distribution (e.g., Gaussian distribution) $P(\boldsymbol{\epsilon})$, K is the sampling times, $\boldsymbol{x}$ is a training sample. However, direct minimizing Eq. (4) could be inefficient and less effective due to the following reasons. Firstly, Eq. (4) involves multi-sampling process and the weight perturbation is based on each training sample, which is inefficient considering the large number of training data. Secondly, weight perturbation with random noise could be less effective.

Therefore, in this paper, we choose to minimize the batch-level loss with the worst-case perturbation. Suppose the batch size is n, the training loss in a batch is $\mathcal{L} = \frac{1}{n}\sum_{j=1}^{n} \ell_{ce}((\boldsymbol{x}, y); \boldsymbol{\theta}_t, \boldsymbol{w}_t)$, then we optimize the following objective:

$$\min_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}^*(\boldsymbol{\theta}_t)), \quad \text{where } \boldsymbol{\epsilon}^*(\boldsymbol{\theta}_t) \triangleq \arg\max_{\|\boldsymbol{\epsilon}\| \le \rho} \mathcal{L}(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}), \tag{5}$$

where $\rho \ge 0$ is a hyperparameter. To efficiently seek the worst-case perturbation, we approximate inner maximization problem in Eq. (5) via a first-order Taylor expansion of $\mathcal{L}(\boldsymbol{\theta}_t + \boldsymbol{\epsilon})$ w.r.t. $\boldsymbol{\epsilon}$ around 0, obtaining

$$\begin{aligned}\boldsymbol{\epsilon}^*(\boldsymbol{\theta}_t) &\triangleq \arg\max_{\|\boldsymbol{\epsilon}\| \le \rho} \mathcal{L}(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}) \approx \arg\max_{\|\boldsymbol{\epsilon}\| \le \rho} (\mathcal{L}(\boldsymbol{\theta}_t) + \boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t)) \\ &= \arg\max_{\|\boldsymbol{\epsilon}\| \le \rho} \boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t) \approx \rho \frac{\nabla \mathcal{L}(\boldsymbol{\theta}_t)}{\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|}.\end{aligned} \tag{6}$$

Optimizing above objective helps find the flat regions with smooth weights in the weight space that can be generalize well. While in CIL, reducing the forgetting of old knowledge is an important purpose. Therefore, we propose the **forgetting-aware** weight perturbation, in which the knowledge distillation (Hinton et al., 2015) loss ℓ_{kd} is also involved. Formally, the following total loss is used in Eq. (5):

$$\begin{aligned}\mathcal{L} = &\frac{1}{n}\sum_{j=1}^{n} \ell_{ce}((\boldsymbol{x}, y); \boldsymbol{\theta}_t, \boldsymbol{w}_t) \\ &+ \frac{1}{n}\sum_{j=1}^{n} \ell_{kd}[\mathrm{O}(\boldsymbol{x}_j; \boldsymbol{\theta}_{t-1}, \boldsymbol{w}_{t-1}), \mathrm{O}(\boldsymbol{x}_j; \boldsymbol{\theta}_t, \boldsymbol{w}_t)],\end{aligned} \tag{7}$$

where $\mathrm{O}(\cdot)$ donates the outputted probability distribution (Castro et al., 2018; Rebuffi et al., 2017) or feature vector (Douillard et al., 2020; Hou et al., 2019). Intuitively, the forgetting-aware weight perturbation guides the model to learn weights that not only generalize well to current training classes but also lead to less forgetting of old classes.

Algorithm 1: One CIL step with ItO

Input: Previous model $\boldsymbol{\Theta}_{t-1} = \{\boldsymbol{\theta}_{t-1}, \boldsymbol{w}_{t-1}\}$, training data $\mathcal{M}_{t-1} \cup \mathcal{D}_t$, number of saved samples per class m, batch size n, scheduled learning γ, number of iterations K, neighborhood size ρ

Output: Updated model $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_t, \boldsymbol{w}_t\}$, memory set $\mathcal{M}_t$

for $k \leftarrow 1$ *to* K **do**
- Sample a mini-batch data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$
- # Modify the cross-entropy (CE) loss for feature calibration
- CE loss with feature calibration $\mathcal{L}_{ce} = \frac{1}{n}\sum_{j=1}^{n} \ell_{ce-afc}(\boldsymbol{x}_j; \boldsymbol{\Theta}_t)$
- # Knowledge distillation (KD) loss
- KD loss $\mathcal{L}_{kd} = \frac{1}{n}\sum_{j=1}^{n} \ell_{kd}[\mathrm{Out}(\boldsymbol{x}_j; \boldsymbol{\Theta}_{t-1}), \mathrm{Out}(\boldsymbol{x}_j; \boldsymbol{\Theta}_t)]$
- Total loss $\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{kd}$
- # Weight calibration with forgetting-aware weight perturbation
- Compute worst-case perturbation $\hat{\boldsymbol{\epsilon}} \leftarrow \rho \frac{\nabla \mathcal{L}(\boldsymbol{\Theta})}{\|\nabla \mathcal{L}(\boldsymbol{\Theta})\|}$
- Gradient update $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \gamma \nabla \mathcal{L}(\boldsymbol{\Theta} + \hat{\boldsymbol{\epsilon}})$

Update memory set $\mathcal{M}_t \longleftarrow \mathcal{M}_{t-1} \cup \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$

4.2.3. Discussion

Our weight calibration is motivated from the perspective of weight loss landscape, which is a widely used to characterize the generalization gap in standard training scenario (Li et al., 2018; Neyshabur et al., 2017), however, there are few explorations under incremental learning. A similar weight perturbation technique has been used for adversarial robustness (Wu, Xia, & Wang, 2020) or standard single-task learning scenario (Foret, Kleiner, Mobahi, & Neyshabur, 2020). However, in Wu et al. (2020), the weight perturbation is based on adversarial samples, while ours is based on clean samples. Besides, ours is motivated to reduce the forgetting in CIL, therefore, we propose the forgetting-aware weight perturbation, which is designed for CIL and different from Foret et al. (2020), Wu et al. (2020).

Comparison with F2M. Recently, Shi et al. proposed F2M (Shi, Chen, Zhang, Zhan, & Wu, 2021) random noise based weight perturbation to find flat local minima for incremental few-shot learning. Our weight calibration is different from F2M (Shi et al., 2021) in several aspects: (1) **Forgetting-aware.** F2M only involves classification loss of base classes when searching for flat minima. In CIL, the knowledge distillation loss can indicate the forgetting of previous knowledge. Therefore, our weight perturbation is forgetting-aware by searching and optimizing the direction that forgets previous knowledge mostly. (2) **Worst-case.** F2M adds random noise to the weights, which is less effective and efficient than worst-case weight perturbation in our method. The effect of worst-case optimization has been widely demonstrated in previous work (Wu et al., 2020). (3) **Different phase.** Our method is used in all CIL phases, while Shi et al. only perform F2M at the first (base) learning stage. (4) **Better performance.** As shown in Section 5.4, the worst-case weight perturbation performs better than the random noise based perturbation used in F2M.

4.3. Overall learning objective

The proposed ItO method, which comprises two strategies named feature calibration and weight calibration, is plug-and-play and can be implemented into existing data replay based CIL methods easily by only modifying the cross-entropy loss in those methods. Algorithm 1 provides pseudo-code for plugging ItO into CIL methods, e.g., UCIR (Hou et al., 2019) or PODnet (Douillard et al., 2020). Note that the additive feature calibration based CE loss ℓ_{ce-afc} is used as default in line 4, while one can also use the multiplicative form ℓ_{ce-mfc}. Besides, we do not specify the KD loss $\mathcal{L}_{kd}$ in Algorithm 1 since it may have different forms in different baselines.

4.4. Comparison with CwD

Recently, Shi et al. (2022) found that the models trained with more classes have more uniform representations. Based on this observation, they proposed a Class-wise Decorrelation (CwD) method that enforces data representations to be more uniformly scattered by minimizing the Frobenius norm of the correlation matrix at the initial phase. Our method has the same purpose to mimic the properties of the oracle model. However, our method is different from CwD (Shi et al., 2022) in several aspects: (1) **Different insight.** CwD focuses on the uniformity of the representation of each class, while we focus on both the feature distribution balance in feature space and the weight smoothness in parameter space. (2) **Different strategy.** CwD minimizes the Frobenius norm of the correlation matrix, while we use deviation compensation in feature space and forgetting-aware adversarial weight perturbation in parameter space. (3) **Different phase.** CwD only performs in the initial phase of CIL, while our method performs in all phases during the course of CIL. (4) **Better performance.** As shown in Section 5.4, our method can consistently outperform CwD on different baselines and benchmarks.

5. Experiments

In this section, we first describe our experimental setups in Section 5.1. Then in Section 5.2, we show the proposed ItO can significantly boost strong CIL methods. In Section 5.3, we conduct ablation study to verify the effect of feature calibration and weight calibration individually. Finally, in Section 5.4, we provide further experiments to understand the proposed method for CIL.

5.1. Experimental setup

Dataset and networks. We conduct extensive experiments on three benchmark datasets. CIFAR-100 (Krizhevsky et al., 2009) has 50,000 training and 10,000 test images of 32×32 size from 100 classes. ImageNet-Sub (Wu et al., 2019) is a subset of ImageNet (ILSVRC 2012) (Deng et al., 2009) that contains 100 classes. To verify the effectiveness of our method on large-scale dataset, we also conduct experiments on the larger and more difficult ImageNet-Full (Deng et al., 2009) dataset, which comprises 1000 classes and over 1.2 million images with 224×224 size. Following (Douillard et al., 2020; Hou et al., 2019), we train ResNet-32 for CIFAR-100 and ResNet-18 (He, Zhang, Ren, & Sun, 2016) for ImageNet-Sub/ImageNet-Full from scratch.

Comparison approaches. We apply our proposed ItO to the following strong methods: UCIR (Hou et al., 2019) and PODnet (Douillard et al., 2020). Besides, we also provide results on non-exemplar based methods such as LwF-MC (Li & Hoiem, 2018) and PASS (Zhu, Zhang, Wang, et al., 2021), as well as popular data replay based baselines such as ER (van de Ven & Tolias, 2019), iCaRL (Rebuffi et al., 2017), BiC (Wu et al., 2019) and WA (Zhao et al., 2020). Besides, we also integrate our method with dynamic architecture based approaches like DER (Yan et al., 2021) and FOSTER (Wang et al., 2022). In Section 5.4, we compare our method with two recently proposed regularization techniques DDE (Hu, Tang, Miao, Hua, & Zhang, 2021) and CwD (Shi et al., 2022). For each data replay based method, we apply both the original classifier and near-class-mean (NME) classifier, and report the results of the better one.

Evaluation protocol and metrics. Following Douillard et al. (2020), Hou et al. (2019), Zhu, Zhang, Wang, et al. (2021), we train the model on half of classes for the first task, and equal classes in the rest phases. For each dataset, three different incremental settings, i.e., 5, 10 and 25 IL phases are conducted to evaluate the CIL performance in both short and long incremental steps. For example, on CIFAR-100, 10 phases setting means that the first task includes 50 classes, and the following 10 incremental phases (or tasks) have 5 classes in each task. For data replay based method, we save and replay R samples for each class. In summary, the setting for CIL can be denoted as **T-phase-R-replay**. For evaluation metrics, we use two metrics to measure the performance of a CIL model: (1) *Final accuracy* a_{final} is defined as the top-1 accuracy of all learned classes at the final incremental stage in the CIL process. (2) *Average accuracy* is computed as $A_t = \frac{1}{t}\sum_{i=1}^{t} a_i$, in which a_i is the top-1 accuracy of all the classes that have already been learned at stage i.

Implementation details. Among the compared methods, PASS (Zhu, Zhang, Wang, et al., 2021) is implemented using the official code. For other methods, the experiments are conducted based on the open-sourced code framework in Zhou, Wang, Ye, and Zhan (2021). Since our method is plug-and-play, we use the original hyper-parameters (e.g., learning schedule, training epochs) as that of the above codebase. For data replay based approaches, we use *herd selection* (Rebuffi et al., 2017) to select and store 10 samples for each old class in our main experiments. For hyper-parameter λ in feature calibration, we use $\lambda = 0.2$ for CIFAR-100 and $\lambda = 0.05$ for ImageNet-Sub and ImageNet-Full.

5.2. Results and analyses

Results on CIFAR-100 and ImageNet-Sub. To verify the calibration effectiveness of our method in CIL, we implement it on UCIR (Hou et al., 2019) and PODnet (Douillard et al., 2020), and compare with other strong CIL methods. Note that for data replay based methods, we store and replay 10 samples for each old class. Comparative results on CIFAR-100 and ImageNet-Sub are shown in Tables 1 and 2. Firstly, we confirm the strong performance of PASS (Zhu, Zhang, Wang, et al., 2021), which can outperform several data replay based methods without saving old samples. Secondly, as we can observe, across various CIL benchmarks, our ItO can consistently and significantly improve the final incremental accuracy and average incremental accuracy of UCIR and PODnet. Take 10 phases setting as an example: on CIFAR-100, our ItO achieves 55.19% final accuracy and 64.60% average accuracy, resulting in 6.42% final accuracy and 3.82% average accuracy improvement on PODnet; On ImageNet-Sub, we boost PODNet by 6.84% on final accuracy and 3.91% on average accuracy, achieve the new state-of-the-art performance.

Table 1
Comparisons of final and average incremental accuracies (%) on CIFAR-100 under 5, 10 and 25 phases. Data replay based methods store 10 samples for each class.

Method	5 phases		10 phases		25 phases	
	Final	Average	Final	Average	Final	Average
LwF-MC (Li & Hoiem, 2018)	23.19	39.83	15.20	27.79	4.27	15.21
PASS (Zhu, Zhang, Wang, et al., 2021)	55.67	63.84	49.03	59.87	44.12	55.07
ER (van de Ven & Tolias, 2019)	37.53	44.95	29.77	34.35	30.68	36.00
iCaRL-CNN (Rebuffi et al., 2017)	31.04	41.51	26.56	35.41	25.01	32.04
iCaRL-NME (Rebuffi et al., 2017)	40.68	51.80	36.95	44.72	34.11	39.49
BiC (Wu et al., 2019)	38.42	54.46	32.29	49.88	26.84	43.53
WA (Zhao et al., 2020)	49.16	58.11	40.69	46.98	35.06	41.78
UCIR (Hou et al., 2019)	49.93	60.58	47.38	57.59	42.54	52.33
+ ItO (ours)	**56.87**$_{+6.94}$	**65.80**$_{+4.95}$	**54.28**$_{+6.90}$	**63.46**$_{+5.86}$	**46.17**$_{+3.63}$	**56.53**$_{+4.20}$
PODnet (Douillard et al., 2020)	51.97	63.09	48.94	60.78	40.30	53.23
+ ItO (ours)	**55.85**$_{+3.88}$	**65.23**$_{+2.14}$	**55.19**$_{+6.24}$	**64.60**$_{+3.82}$	**49.15**$_{+8.85}$	**60.97**$_{+7.74}$

Table 2
Comparisons of final and average incremental accuracies (%) on ImageNet-Sub under 5, 10 and 25 phases settings. Data replay based methods store 10 samples for each class.

Method	5 phases		10 phases		25 phases	
	Final	Average	Final	Average	Final	Average
LwF-MC (Li & Hoiem, 2018)	34.34	54.18	17.98	40.49	5.10	18.35
PASS (Zhu, Zhang, Wang, et al., 2021)	56.68	67.33	52.24	62.09	36.22	48.93
ER (van de Ven & Tolias, 2019)	44.04	53.38	39.94	48.26	25.52	25.46
iCaRL-CNN (Rebuffi et al., 2017)	30.72	44.87	25.98	36.90	19.62	27.72
iCaRL-NME (Rebuffi et al., 2017)	47.20	59.62	40.52	51.37	32.94	40.38
BiC (Wu et al., 2019)	41.88	61.74	30.34	54.17	22.22	39.37
WA (Zhao et al., 2020)	50.32	61.18	40.42	52.23	32.22	40.52
UCIR (Hou et al., 2019)	59.12	71.89	54.10	68.35	45.92	57.61
+ ItO (ours)	**67.84**$_{+8.72}$	**75.05**$_{+3.16}$	**66.38**$_{+12.28}$	**74.45**$_{+6.09}$	**54.38**$_{+8.46}$	**63.37**$_{+5.76}$
PODnet (Douillard et al., 2020)	67.96	76.68	63.18	73.70	47.44	59.09
+ ItO (ours)	**70.54**$_{+2.58}$	**78.14**$_{+1.46}$	**70.02**$_{+6.84}$	**77.61**$_{+3.91}$	**61.34**$_{+13.90}$	**71.78**$_{+12.69}$

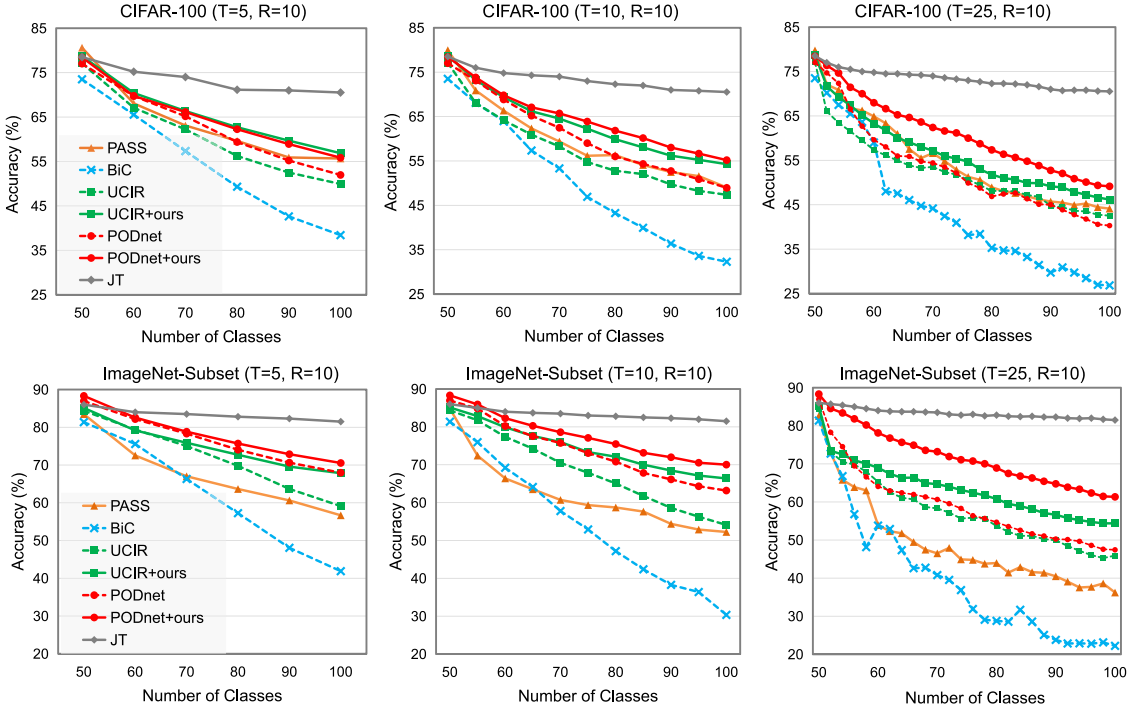

Fig. 4. Comparisons of the step-wise accuracies on CIFAR-100 and ImageNet-Subset under three different settings. For UCIR and PODnet, dashed lines present the baseline, and solid lines present the methods with our proposed ItO.

Long step CIL performance. Particularly, in long incremental steps setting, e.g., 25 phases setting, data replay method without bias calibration (e.g., ER and iCaRL-CNN) performs much worse than that of 5 or 10 phases. iCaRL-NME uses near-class-mean classifier for inference, which can be viewed as an under-sampling based post-hoc bias correction strategy. It remarkably

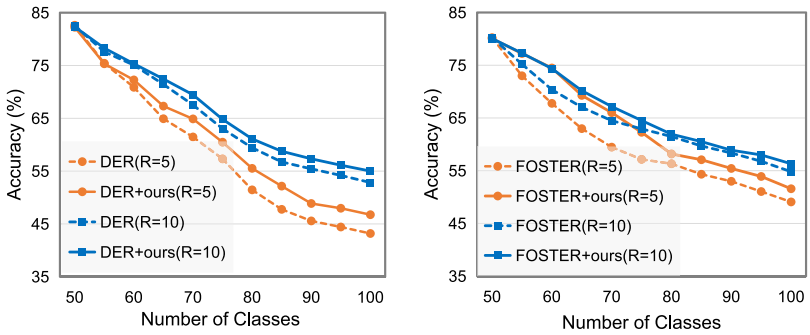

Fig. 5. Step-wise accuracies on CIFAR-100 with 10 incremental phases.

Table 3
Comparisons of final and average incremental accuracies (%) on ImageNet-Full under 10 and 25 phases settings. Data replay based methods store 10 samples for each class.

Method	10 phases		25 phases	
	Final	Average	Final	Average
iCaRL-CNN (Rebuffi et al., 2017)	21.32	31.14	17.23	24.91
iCaRL-NME (Rebuffi et al., 2017)	32.25	42.53	27.08	34.83
UCIR (Hou et al., 2019)	52.64	60.43	47.71	56.87
+ ItO (ours)	**55.71**$_{+3.07}$	**62.61**$_{+2.17}$	**51.87**$_{+4.16}$	**59.92**$_{+3.05}$
PODnet (Douillard et al., 2020)	54.77	63.75	48.82	59.19
+ ItO (ours)	**55.45**$_{+0.68}$	**63.95**$_{+0.20}$	**52.55**$_{+3.73}$	**61.43**$_{+2.24}$

Table 4
ItO can give remarkable performance boosts to the representative dynamic architectures based methods DER (Yan et al., 2021) and FOSTER (Wang et al., 2022). CIFAR-100 with 10 incremental phases.

Method	R=5		R=10	
	Final	Average	Final	Average
DER (Yan et al., 2021)	43.19	58.63	52.80	65.07
+ ItO (ours)	**46.75**$_{+3.56}$	**61.25**$_{+2.62}$	**55.01**$_{+2.21}$	**66.45**$_{+1.38}$
FOSTER (Wang et al., 2022)	49.08	60.39	54.81	64.67
+ ItO (ours)	**51.58**$_{+2.50}$	**64.14**$_{+3.75}$	**56.32**$_{+1.51}$	**66.29**$_{+1.62}$

improves the performance of iCaRL-CNN. Latter, BiC, WA, UCIR and PODnet use different bias correction strategies, which also improve the performance to some extent. Our proposed ItO further calibrates both the feature space and weight space, leading to significant improvement over UCIR and PODnet. For example, ItO achieves 71.87% average accuracy under 25 phases and boosts PODNet by 12.69%. Fig. 4 shows the step-wise results of incremental accuracy.

Results on ImageNet-Full. In Table 3, we present the results on ImageNet-Full dataset. Our method remarkably boosts the UCIR. For example, UCIR+ItO has 59.92% average incremental accuracy under 25 phases setting, outperforming UCIR by 3.05%. For PODnet, ItO has similar results under 10 phases setting, while has better performance under 25 phases setting. Note that the baseline PODnet uses sophisticated balanced fine-tuning to correct the bias at the end of each IL phase, while our method can calibrate the bias automatically.

Integrating ItO with dynamic architectures. Recently, dynamic architecture based methods have been successfully applied in CIL. For example, DER (Yan et al., 2021), FOSTER (Wang et al., 2022) and Dytox (Douillard et al., 2022) yield state-of-the-art performance by dynamically expanding the network. Here we integrate our proposed ItO with DER and FOSTER. The implementation is based on their official code. Specifically, experiments on CIAFR-100 with two settings, i.e., 10-phase-5-replay and 10-phase-10-replay are conducted respectively. The results are summarized in Table 4 and Fig. 5. Firstly, we find that FOSTER is more robust to the number of saved samples than DER. Secondly, as can be observed, our method can give large performance boosts to the above methods, which verifies that ItO can be successfully integrated with dynamic architecture based methods.

Table 5
Ablation study: feature calibration (FC) and weight calibration (WC). Experiments are conducted on CIFAR-100 with 10 incremental steps (T = 10, R = 10).

Method	FC	WC	UCIR		PODnet	
			Final	Average	Final	Average
Baseline	✗	✗	47.38	57.59	48.94	60.78
+ FC	✓	✗	52.00$_{+4.62}$	61.02$_{+3.43}$	53.50$_{+4.56}$	63.31$_{+2.53}$
+ WC	✗	✓	47.47$_{+0.09}$	59.07$_{+1.48}$	51.24$_{+2.30}$	61.62$_{+0.84}$
+ ItO	✓	✓	**54.28**$_{+6.90}$	**63.46**$_{+5.86}$	**55.19**$_{+6.24}$	**64.60**$_{+3.82}$

5.3. Ablation study

The proposed ItO method is comprised of two components: feature calibration (FC) by deviation compensation and weight calibration (WC) by forgetting-aware wright perturbation. Here the effect of individual aspects is analyzed. Fig. 6 and Table 5 show the results of using FC and WC alone and together. One can observe that: (1) FC has a remarkable effect on mitigating the imbalance problem and achieves much better results than Baseline, e.g., FC improves the performance of PODnet with a margin of 4.56% on final accuracy. (2) WC has a relatively small effect without FC since the imbalance problem of the classifier is severe. (3) The performance of baseline could be significantly improved by combining FC with WC, which demonstrates that FC and WC could benefit from each other and both the calibration effects play an important role in CIL.

5.4. Further understanding of proposed method

Comparisons of additive and multiplicative feature calibration. For feature calibration, we have designed two simple but different forms of calibration technique, i.e., additive feature calibration (AFC) and multiplicative feature calibration (MFC). In our previous experiments, AFC is used as default. Here we conduct experiments to compare AFC with MFC. Fig. 7 shows the comparison of AFC and MFC on improving the baseline methods, i.e., UCIR and PODnet. One can see that the boost of using AFC is higher than using MFC.

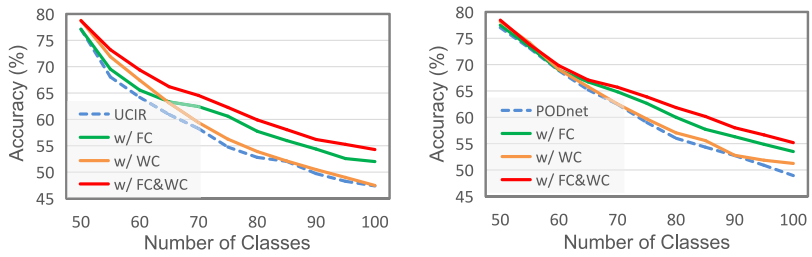

Fig. 6. Ablation study: comparisons of the step-wise accuracies on CIFAR-100 (T = 10, R = 10) with feature calibration (FC) and weight calibration (WC) strategies alone and together.

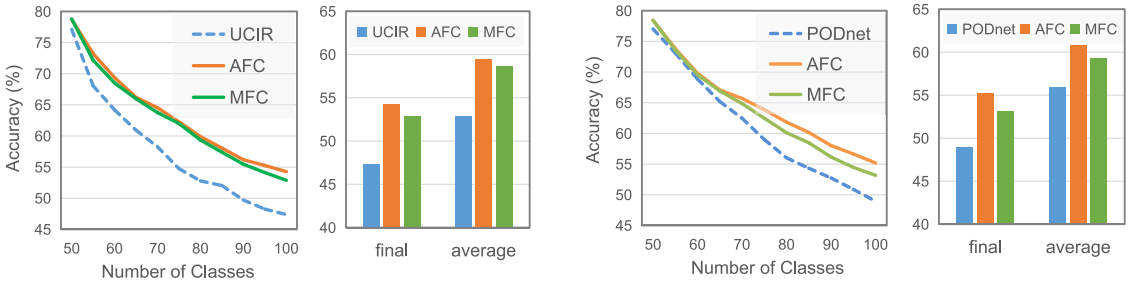

Fig. 7. Comparisons of additive feature calibration (AFC) and multiplicative feature calibration (MFC) on CIFAR-100 (T = 10, R = 10).

Table 6
Compare our method with DDE and CwD with 10 incremental steps.

Method	CIFAR-100				ImageNet-Sub			
	UCIR		PODnet		UCIR		PODnet	
	Final	Average	Final	Average	Final	Average	Final	Average
Baseline	46.34	58.31	48.28	58.92	51.06	64.04	60.03	70.40
+ DDE (Hu et al., 2021)	51.31	62.00	49.22	60.52	59.06	69.05	63.02	73.00
Baseline	47.38	57.59	48.94	60.78	54.10	68.35	63.18	73.70
+ CwD (Shi et al., 2022)	48.58	58.67	50.70	61.31	56.13	69.60	64.70	74.13
+ ItO (ours)	**54.28**	**63.46**	**55.19**	**64.67**	**66.38**	**74.46**	**70.02**	**77.61**

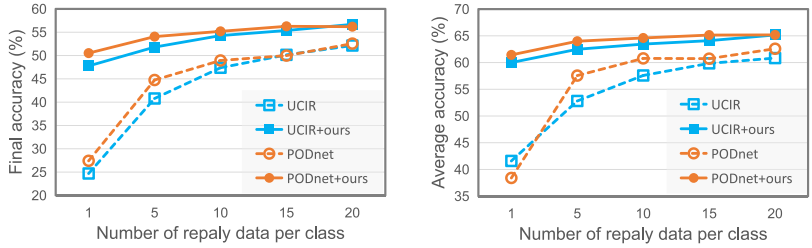

Fig. 8. Comparisons of final and averaged accuracy with different number of replay data per class. Experiments are conducted on CIFAR-100 with 10 incremental steps.

Robustness to different replay numbers. To demonstrate the effect and robustness of our proposed ItO, we vary the number of saved and replayed samples for each old class, which is donated by R. Specifically, we implement baseline methods and our ItO on more challenging cases where rarer old data are replayed (R = 1, 5), as well as easier cases where each old class has 20 samples (R = 20). As can be observed in Fig. 8, original methods have severe performance degradation when only 1 or 5 samples are replayed per class. For example, when reducing the number of the replayed samples from 10 to 1, the final accuracy drops 22.68% for UCIR and 21.52% for PODnet. In contrast, our method shows its robustness to the number of replay data. Particularly, in the more challenging setting with fewer old samples replayed, the improvements provided by our ItO is significantly increased. For example, it obtains up to 23.07% improvement of PODnet on final accuracy when replaying only 1 data per class.

Weight perturbation: worst-case v.s. random noise. In our weight calibration, we add worst-case weight perturbation to find the forgetting-aware direction for calibration. Here we compare the worst-case perturbation with random noise based perturbation. As shown in Fig. 9, random noise is less effective than worst-case based perturbation. Besides, perturbation with random noise is inefficient because it involves multi-sampling process and many of the perturbed directions are useless.

Large margin: class-dependent v.s. class-independent. As discussed in Section 4.1.3, our feature calibration (FC) strategy has the large margin effect for old classes in each CIL phase. Partially, the margin introduced by FC is class-dependent, which is different from existing margin losses (Deng et al., 2019; Liu et al., 2016; Wang, Cheng, Liu, & Liu, 2018; Wang, Wang, et al., 2018) that use a class-independent (CID), constant margin. For example, in additive FC (Eq. (2)), the margin for a old class $c \in \mathcal{C}_{1:t-1}$

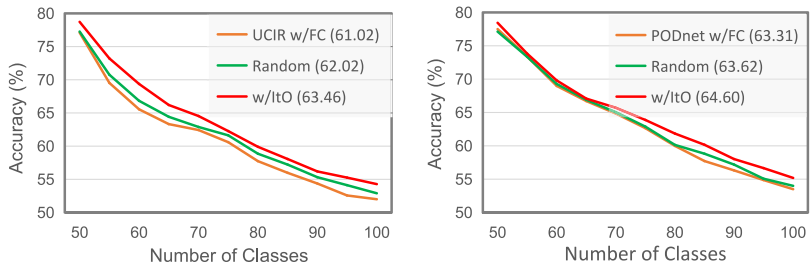

Fig. 9. Worst-case v.s. random weight perturbation. Before comparison, we apply feature calibration (FC) to UCIR and PODnet. ItO uses the worst-case based perturbation. The numbers in the legends represent the average incremental accuracy.

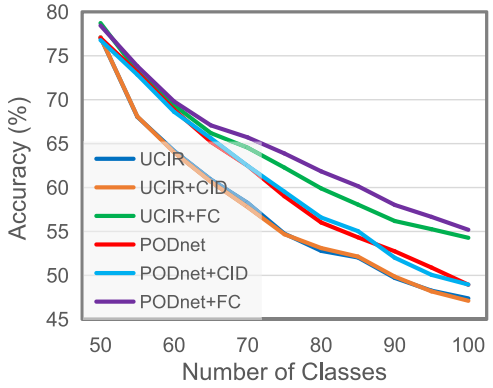

Fig. 10. Class-independent (CID) margin yields no improvement for CIL.

is $\beta_c > 0$ and depends on the number of saved old samples; For a new class $c \in \mathcal{C}_t$, the margin $\beta_c = 0$. When assigning a constant margin for both new and old classes, we observe in Fig. 10 that the CID margin yields no improvement for CIL. Actually, the relative position of old and new classes is crucial and the feature imbalance problem is still existing when using the CID margin.

Compare with CwD and DDE. We noticed that there are other regularization techniques that have been proposed for CIL more recently. Here we compare our proposed ItO with DDE (Hu et al., 2021) and CwD (Shi et al., 2022). Specifically, DDE (Hu et al., 2021) implicitly constrains the neighborhood relation by predicting a sample with its neighborhood. Besides, it also contains an incremental momentum effect removal module to reduce the bias in CIL. CwD (Shi et al., 2022) regularizes representations of each class to scatter more uniformly for improving the feature transferability in CIL. In Table 6, we show the comparison among DDE, CwD and our ItO. As can be observed, our method can fairly outperform DDE and CwD with different baselines on both CIFAR-100 and ImageNet-Sub datasets.

6. Conclusion

In this paper, we study CIL problem from a new viewpoint: improving CIL by mimicking the oracle model (joint-training) in both feature space and weight space. Motivated by the analysis, we propose ItO, which consists of two types of calibration: feature calibration (FC) and weight calibration (WC). Specifically, FC is realized by additive or multiplicative deviation compensation, and WC is achieved by forgetting-aware weight perturbation. Our ItO is plug-and-play and can be implemented into existing methods easily. Through extensive experiments, we show that our method yields consistent and significant performance improvements over previous SOTA methods by a large margin. Future works will consider more challenging scenarios like few-shot and federated CIL.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used is open-sourced, and I have shared the link to our code.

Acknowledgments

This work has been supported by the National Key Research and Development Program (2018AAA0100400), National Natural Science Foundation of China (61836014, 62222609, 62076236, 61721004), Key Research Program of Frontier Sciences of Chinese Academy of Sciences (ZDBS-LY-7004), Youth Innovation Promotion Association of Chinese Academy of Sciences (2019141).

References

Ahn, H., Kwak, J., Lim, S., Bang, H., Kim, H., & Moon, T. (2021). SS-IL: Separated softmax for incremental learning. In *ICCV* (pp. 844–853).

Belouadah, E., & Popescu, A. (2019). Il2m: Class incremental learning with dual memory. In *ICCV* (pp. 583–592).

Belouadah, E., & Popescu, A. (2020). Scail: Classifier weights scaling for class incremental learning. In *WACV* (pp. 1266–1275).

Castro, F. M., Marín-Jiménez, M. J., et al. (2018). End-to-end incremental learning. In *ECCV* (pp. 233–248).

Delange, M., Aljundi, R., Masana, M., et al. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Deng, J., Dong, W., Socher, R., et al. (2009). ImageNet: A large-scale hierarchical image database. In *CVPR* (pp. 248–255).

Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *CVPR* (pp. 4690–4699).

Dhar, P., Singh, R. V., Peng, K., et al. (2019). Learning without memorizing. In *CVPR* (pp. 5138–5146).

Douillard, A., Cord, M., Ollion, C., et al. (2020). PODnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV* (pp. 86–102).

Douillard, A., Ramé, A., Couairon, G., & Cord, M. (2022). Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR* (pp. 9285–9295).

Farajtabar, M., Azizan, N., Mott, A., & Li, A. (2020). Orthogonal gradient descent for continual learning. In *AISTATS* (pp. 3762–3773).

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., et al. (2017). Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734.

Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211.

Gopalakrishnan, S., Singh, P. R., Fayek, H., Ramasamy, S., & Ambikapathi, A. (2022). Knowledge capture and replay for continual learning. In *WACV* (pp. 10–18).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770–778).
Hinton, G., Vinyals, O., Dean, J., et al. (2015). Distilling the knowledge in a neural network, vol. 2, no. 7. arXiv preprint arXiv:1503.02531.
Hou, S., Pan, X., Loy, C. C., Wang, Z., & Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *CVPR* (pp. 831–839).
Hu, X., Tang, K., Miao, C., Hua, X.-S., & Zhang, H. (2021). Distilling causal effect of data in class-incremental learning. In *CVPR* (pp. 3957–3966).
Hung, C.-Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M., & Chen, C.-S. (2019). Compacting, picking and growing for unforgetting continual learning. In *NeurIPS, vol. 32*.
Kemker, R., & Kanan, C. (2018). FearNet: Brain-inspired model for incremental learning. In *ICLR*.
Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*, 3521–3526.
Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*: *Tech. rep.*, Citeseer.
Lee, K., Lee, K., Shin, J., & Lee, H. (2019). Overcoming catastrophic forgetting with unlabeled data in the wild. In *ICCV* (pp. 312–321).
Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., & Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, *58*, 52–68.
Li, Z., & Hoiem, D. (2018). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 2935–2947.
Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *NeurIPS, vol. 31*.
Liu, Y., Parisot, S., Slabaugh, G., Jia, X., Leonardis, A., & Tuytelaars, T. (2020). More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *ECCV* (pp. 699–716).
Liu, W., Wen, Y., Yu, Z., & Yang, M. (2016). Large-margin softmax loss for convolutional neural networks. In *ICML* (pp. 507–516).
Lomonaco, V., & Maltoni, D. (2016). Comparing incremental learning strategies for convolutional neural networks. In *IAPR workshop on artificial neural networks in pattern recognition* (pp. 175–184).
Maltoni, D., & Lomonaco, V. (2019). Continuous learning in single-incremental-task scenarios. *Neural Networks*, *116*, 56–73.
Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & van de Weijer, J. (2020). Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277.
Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. In *NeurIPS, vol. 30*.
Nguyen, C. V., Li, Y., Bui, T. D., & Turner, R. E. (2018). Variational continual learning. In *ICLR*.
Parisi, G. I., Kemker, R., Part, J. L., et al. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71.
Pellegrini, L., Graffieti, G., Lomonaco, V., & Maltoni, D. (2020). Latent replay for real-time continual learning. In *IROS* (pp. 10203–10209).
Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). iCaRL: Incremental classifier and representation learning. In *CVPR* (pp. 5533–5542).
Shi, G., Chen, J., Zhang, W., Zhan, L.-M., & Wu, X.-M. (2021). Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. In *NeurIPS, vol. 34* (pp. 6747–6761).
Shi, Y., Zhou, K., Liang, J., Jiang, Z., Feng, J., Torr, P., et al. (2022). Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *CVPR*.
Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. In *NeurIPS* (pp. 2990–2999).
Simon, C., Faraki, M., Tsai, Y.-H., Yu, X., Schulter, S., Suh, Y., et al. (2022). On generalizing beyond domains in cross-domain continual learning. In *CVPR* (pp. 9265–9274).
Smith, J., Hsu, Y.-C., Balloch, J., Shen, Y., Jin, H., & Kira, Z. (2021). Always be dreaming: A new approach for data-free class-incremental learning. In *ICCV* (pp. 9374–9384).
Tao, X., Hong, X., Chang, X., & Gong, Y. (2020). Bi-objective continual learning: Learning new while consolidating known. In *AAAI, vol. 34, no. 04* (pp. 5989–5996).
van de Ven, G. M., & Tolias, A. S. (2019). Three scenarios for continual learning. arXiv preprint arXiv:1904.07734.
Wang, F., Cheng, J., Liu, W., & Liu, H. (2018). Additive margin softmax for face verification. *IEEE Signal Processing Letters*, *25*(7), 926–930.
Wang, S., Li, X., Sun, J., & Xu, Z. (2021). Training networks in null space of feature covariance for continual learning. In *CVPR* (pp. 184–193).
Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., et al. (2018). Cosface: Large margin cosine loss for deep face recognition. In *CVPR* (pp. 5265–5274).
Wang, F.-Y., Zhou, D.-W., Ye, H.-J., & Zhan, D.-C. (2022). FOSTER: Feature boosting and compression for class-incremental learning. In *ECCV*.
Wu, Y., Chen, Y.-J., Wang, L., et al. (2019). Large scale incremental learning. In *CVPR* (pp. 374–382).
Wu, D., Xia, S.-T., & Wang, Y. (2020). Adversarial weight perturbation helps robust generalization. In *NeurIPS, vol. 33* (pp. 2958–2969).
Xiang, Y., Fu, Y., Ji, P., & Huang, H. (2019). Incremental learning using conditional adversarial networks. In *ICCV* (pp. 6618–6627).
Yan, S., Xie, J., & He, X. (2021). DER: Dynamically expandable representation for class incremental learning. In *CVPR*.
Yang, Y., Zhou, D.-W., Zhan, D.-C., Xiong, H., & Jiang, Y. (2019). Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In *KDD* (pp. 74–82).
Yin, H., Molchanov, P., Alvarez, J. M., Li, Z., Mallya, A., Hoiem, D., et al. (2020). Dreaming to distill: Data-free knowledge transfer via deepinversion. In *CVPR* (pp. 8715–8724).
Zeng, G., Chen, Y., Cui, B., & Yu, S. (2019). Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, *1*(8), 364–372.
Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. In *ICML* (pp. 3987–3995).
Zhao, B., Xiao, X., Gan, G., Zhang, B., & Xia, S. (2020). Maintaining discrimination and fairness in class incremental learning. In *CVPR* (pp. 13205–13214).
Zhou, D.-W., Wang, F.-Y., Ye, H.-J., & Zhan, D.-C. (2021). Pycil: A python toolbox for class-incremental learning. arXiv preprint arXiv:2112.12533.
Zhu, F., Cheng, Z., Zhang, X.-y., & Liu, C.-l. (2021). Class-incremental learning via dual augmentation. In *NeurIPS, vol. 34*.
Zhu, F., Zhang, X.-Y., & Liu, C.-L. (2021). Calibration for non-exemplar based class-incremental learning. In *ICME* (pp. 1–6).
Zhu, F., Zhang, X.-Y., Wang, C., Yin, F., & Liu, C.-L. (2021). Prototype augmentation and self-supervision for incremental learning. In *CVPR* (pp. 5871–5880).