Artificial Intelligence 339 (2025) 104261

Contents lists available at ScienceDirect

Artificial Intelligence

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

Defying catastrophic forgetting via influence function

Rui Gao, Weiwei Liu *

The School of Computer Science, Wuhan University, Wuhan, 43000, China

ARTICLE INFO

Keywords:
Continual learning
Catastrophic forgetting
Influence function

ABSTRACT

Deep-learning models need to continually accumulate knowledge from tasks, given that the number of tasks are increasing overwhelmingly as the digital world evolves. However, standard deep-learning models are prone to forgetting about previously acquired skills when learning new ones. Fortunately, this catastrophic forgetting problem can be solved by means of continual learning. One popular approach in this vein is regularization-based method which penalizes parameters by giving their importance. However, a formal definition of parameter importance and theoretical analysis of regularization-based methods are elements that remain under-explored. In this paper, we first rigorously define the parameter importance by influence function, then unify the seminal methods (i.e., EWC, SI and MAS) into one whole framework. Two key theoretical results are presented in this work, and extensive experiments are conducted on standard benchmarks, which verify the superior performance of our proposed method.

1. Introduction

The digital world around us is continuously evolving [46]. Each day, huge amounts of data such as images and videos are produced on social media, which generates new emerging learning tasks [44,45]. It is important for learning systems to adapt quickly to changing environments [47,48,50]. For quick adaptation to occur, learning systems are required to memorize useful past experiences when new tasks appear. However, existing standard deep-learning methods quickly forget previously acquired experiences when learning new tasks [18]. This phenomenon, referred to as catastrophic forgetting [21], presents a significant challenge for some application scenarios in which previous training data cannot be obtained.

Continual learning enables the continual accumulation of knowledge over different tasks, referred to as lifelong learning [8], sequential learning [2] or incremental learning [1]. Many continual learning methods have been proposed to overcome the catastrophic forgetting problem. Among these, the regularization-based methods are popular due to their simplicity and promising experimental results. More specifically, when learning from new data, a regularization term is introduced in the loss function to consolidate previous knowledge. Furthermore, different metrics are employed by existing regularization-based methods to measure the parameter importance. For example, EWC [18] uses the Fisher Information Matrix to measure the importance of parameters; SI [43] maintains an online estimate of the synapse's importance toward solving problems encountered in the past; MAS [1] measures the importance according to the gradients of the squared L_2-norm of the learned network output function. However, these methods lack both a formal definition of parameter importance and substantive theoretical analysis.

This paper first rigorously defines the parameter importance via influence function, then generalizes the EWC, SI and MAS into one unified framework. We provide the asymptotic statistical property of a regularization-based method's estimator, the agnostic PAC

* Corresponding author.
E-mail addresses: grcswhu1002@gmail.com (R. Gao), liuweiwei863@gmail.com (W. Liu).

https://doi.org/10.1016/j.artint.2024.104261
Received 29 April 2023; Received in revised form 11 June 2024; Accepted 22 November 2024

Available online 27 November 2024

learning results, and how the variance achieves the Cramer-Rao Lower Bound. Extensive experimental results demonstrate that the proposed method consistently outperforms state-of-the-art baselines.

The remainder of this paper is organized as follows. We first discuss the related work in §2. §3 provides preliminaries of continual learning. Theoretical results of regularization-based methods are presented in §4, after which §5 introduces our proposed method. §6 evaluates the performance of our proposed method. Finally, we draw our conclusions in §7.

2. Related works

The problem of overcoming catastrophic forgetting has been addressed in many previous works and many recent surveys have focused on continual learning [31,23,32,13,11]. These works can be broadly divided into three categories: (1) Replay methods, (2) Regularization-based methods and (3) Parameter isolation methods.

Replay methods store samples or generate pseudo-samples using a generative model; these previous task samples are replayed while a new task is being learned [49]. iCaRL [33], the first of these replay methods, focuses on learning in a class-incremental way. Assuming fixed allocated memory, it selects and stores samples (exemplars) closest to the feature mean of each class. GEM [28] exploits exemplars to solve a constrained optimization problem, projecting the current task gradient in a feasible area, outlined by the previous task gradients. The major drawback of replay methods is that they require additional computation and storage of raw input samples.

Regularization-based methods add an extra regularization term in the loss function instead of storing raw inputs. These methods penalize changes to important parameters during the training of current task. Elastic weight consolidation (EWC) [18] is the first seminal work to establish this strategy. Synaptic Intelligence (SI) [43] breaks the EWC paradigm of determining new task importance weights in a separate phase after training. Memory Aware Synapses (MAS) [1] is a method that obtains gradients of the squared L_2-norm of the learned network output function instead of calculating the gradients of the loss function. Incremental Moment Matching (IMM) [22] estimates Gaussian posteriors for task parameters, in the same vein as EWC, but inherently differs in its use of model merging. FROMP [30] is based on functional-regularization, and regularizes the network outputs directly [4].

Parameter isolation methods dedicate different model parameters to each task so as to prevent any possible forgetting. When no architectural size constraints apply, it is possible to grow new branches for new tasks while freezing previous task parameters [42,35]. PackNet [29] iteratively assigns parameter subsets to consecutive tasks by constituting binary masks. There are two phases for training new tasks. First, the network is trained without altering previous task parameter subsets; second, it retrains this remaining subset of important parameters. For its part, HAT [36] requires only one training phase, incorporating task-specific embeddings for attention masking. The per-layer embeddings are gated through a Sigmoid to attain unit-based attention masks in the forward pass.

The influence function has a rich history in statistics and has recently been employed in the field of machine learning. Michael Wojnowicz et al. [41] introduce a method for approximating a quantity related to influence in generalized linear models. The influence function is used to trace a model prediction through the learning algorithm and identify the training points most responsible for a given prediction [19]. A multi-stage influence function score is developed to track predictions from a finetuned model all the way back to pretraining data and identify the pretraining examples in the pretraining task that make the highest contribution to prediction in the finetuning task with this score [7]. However, it is unclear whether the use of influence function is advantageous for continual learning. This work provides an answer in the affirmative.

3. Preliminaries

Continual learning consists of a sequence of disjoint tasks that are sequentially learned one at a time. These tasks often correspond to different datasets, or different splits of a dataset for which category labels do not overlap. This kind of setting is based on an assumption: only related data are allowed to be used for training the current task. While, under ideal circumstances, subsequent learning tasks would benefit from previous tasks, such a setting would result in the catastrophic forgetting of the knowledge learned from previous tasks for the model.

In this paper, we consider the use of regularization-based methods to address catastrophic forgetting in classification tasks. We train a shared neural network for all tasks. The model parameters vector is defined as $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_n)$.[1] The regularization-based methods aim to obtain the important value Ω_i for each parameter θ_i with respect to the previous tasks. We assume that the model parameters follow a distribution F. We identify the parameter $(\theta_1, ..., \theta_n)$ with its empirical parameter distribution F_n. N_{task} tasks are denoted as: $(\tau_1, \tau_2, ..., \tau_{N_{task}})$. Let (X_t, Y_t) be the training dataset for task τ_t, where X_t and Y_t represent the feature and label respectively. L_t is the loss function and $\hat{\boldsymbol{\theta}}^t = (\hat{\theta}_1^t, \hat{\theta}_2^t, ..., \hat{\theta}_n^t)$ is a local minimum for task τ_t.

The regularization-based methods use a regularization term to penalize changes to important parameters. Classical methods such as EWC and MAS adopt the following expression form:

$$L(\boldsymbol{\theta}) = L_t(\boldsymbol{\theta}) + \lambda \sum_i \Omega_i \left(\theta_i - \theta_i^*\right)^2 \tag{1}$$

where λ is the hyperparameter indicating the strength of the penalty, Ω_i represents the corresponding parameter importance with respect to the previous tasks, and θ_i^* denotes the "old" network parameters.

[1] The vectors in this paper are denoted as bold letters.

A fundamental question pertaining to regularization-based methods is that of which parameters are important for previous tasks. Assuming that model parameters follow specific distributions, we contend that if the parameter distribution with respect to a specific parameter is slightly perturbed, leading to a large change in performance, then this parameter can be considered important. The influence function has been used to figure out how the model's predictions change if a training input is modified [19]. This paper adopts the concept of influence function to measure the importance of parameters so as to address catastrophic forgetting in deep neural networks (DNNs). The important notations are shown in Table F.14 of Appendix F. In the following section, this paper analyzes the regularization-based methods in terms of the estimator's asymptotic statistical properties and agnostic PAC learning algorithm.

4. Theoretical results

The theoretical analysis of regularization-based methods is an element that remain under-explored. In this section, we contribute three main theoretical results: (1) that the estimator of a regularization-based method is asymptotically normal; (2) that the learning algorithm of a regularization-based method is successful, according to agnostic PAC learning; (3) that how the variance theoretically reaches the asymptotic minimum variance.

Typically, continual learning methods are effective in helping the model defy catastrophic forgetting on one task; however, this does not mean that the model can always achieve great performance on this task. For regularization-based methods, the reason may be that the estimator provided by methods for one task is far from the "correct value". In the following Theorem 1, we illustrate this point in terms of the asymptotic statistical properties of the estimator.

Theorem 1. *The estimator of a regularization-based method for task t can be represented as:*

$$T_\lambda(D^t_{m_t}) = \underset{\theta}{\operatorname{argmin}} \left\{ L_t(\theta; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1} (\theta_j - \hat{\theta}_j^{t-1})^2 \right\}. \tag{2}$$

We define $T(D^t) = \operatorname{argmin}_\theta \left\{ \int l_t(\theta;\ x) d D^t(x) \right\}$. The estimator $T_\lambda(D^t_{m_t})$ is then asymptotically normal. That is:

$$\sqrt{m_t} \left(T_\lambda(D^t_{m_t}) - T(D^t) \right) \to N(\sqrt{m_t} C_{m_t} \lambda, \sigma^2) \tag{3}$$

where $C_{m_t} = \left. \frac{d T_\lambda(D^t_{m_t})}{d\lambda} \right|_{\lambda=0}$ and $\sigma^2 = \mathbb{E} {\Phi_{D^t}}^2(X) = \int {\Phi_{D^t}}^2(x) d D^t(x)$. $\Phi_{D^t}(x)$ is the influence function of $T(\cdot)$ at D^t.

The proof of Theorem 1 can be found in Appendix C. In Theorem 1, $L_t(\theta; S^t) = \frac{1}{m_t} \sum_{i=1}^{m_t} l_t(\theta; x_i)$. l_t is the loss function. D^t is the distribution function and $D^t_{m_t}$ is the empirical distribution function for task τ_t. m_t is the size of the sample set S^t.

Remark 1. It is obvious that the estimator $T_\lambda(D^t_{m_t})$ is not unbiased. As m_t tends to be infinite, the bias approaches $C_{m_t}\lambda$. According to [38], the notion of "best" limit distribution is usually translated into asymptotic unbiasedness and minimum variance. Therefore, the asymptotic bias $C_{m_t}\lambda$ should be small. This implies that λ cannot be too large and that some restrictions of λ are needed. Moreover, we provide some conditions under which σ^2 can achieve the Cramér-Rao Lower Bound. More details can be found in the proof of Theorem 1.

Before we illustrate Theorem 2, we present some related definitions according to [37].

Definition 1 *(Agnostic PAC learnability).* A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\zeta, \delta \in (0,1)$ and for every distribution D over $X \times Y$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\zeta, \delta)$ i.i.d. examples generated by D, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_D(h) \leq \min_{h' \in \mathcal{H}} L_D\left(h'\right) + \zeta \tag{4}$$

where $L_D(h) \overset{\text{def}}{=} \underset{z \sim D}{\mathbb{E}}[l(h; z)]$. $l(h; z)$ is the loss function.

Definition 2 *(ζ-representative sample).* A training set S is called ζ-representative (w.r.t. domain Z, hypothesis class H, loss function l, and distribution D) if

$$\forall h \in H, \left| L_S(h) - L_D(h) \right| \leq \zeta \tag{5}$$

where $L_D(h) \overset{\text{def}}{=} \underset{z \sim D}{\mathbb{E}}[l(h; z)]$ and $L_S(h) \overset{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} l\left(h; z_i\right)$. $l(h; z)$ is the loss function and z_i is randomly drawn from D.

Definition 3 *(Uniform convergence).* We say that a hypothesis class H has the uniform convergence property (w.r.t. a domain Z and a loss function l) if there exists a function $m_{\mathcal{H}}^{\mathrm{UC}} : (0,1)^2 \to \mathbb{N}$ such that for every $\delta \in (0,1)$ and for every probability distribution D over Z, if S is a sample of $m \geq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta, \delta)$ examples drawn i.i.d. according to D, then, with probability of at least $1-\delta$, S is ζ-representative.

The following theorem states that the learning algorithm of a regularization-based method is a successful agnostic PAC learning algorithm and provides a clear limitation of λ.

Theorem 2. *We denote the pointwise multiplication operation using $\circ$. $\mathcal{H}^t$ is a finite hypothesis class for task τ_t. Z^t is a domain. D^t is the distribution over Z^t. $l_t : \mathcal{H}^t \times Z^t \to [0,1]$ is the loss function for task τ_t. Assume that $\mathcal{H}^t$ has the uniform convergence property (with sample complexity $m_{\mathcal{H}}^{\mathrm{UC}}(\zeta', \delta)$). Let $h_\lambda^t = \hat{\theta}_\lambda^t \in \mathcal{H}^t$ be the hypothesis that acts as the minimizer of $\hat{\theta}_\lambda^t = \mathrm{argmin}_\theta \left\{ L_t(\theta; S^t) + \lambda \sum_{j=1}^n \Omega_j^{t-1} (\theta_j - \hat{\theta}_j^{t-1})^2 \right\}$. A_λ^t is the algorithm that generates h_λ^t. S^t is the sample set of task τ_t with $m_t \geq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta', \delta)$ samples drawn independent and identical distributed (i.i.d.) according to D^t. If the condition*

$$\zeta \quad \in (0,1) \tag{6}$$

is satisfied, where

$$\zeta = 4\zeta' + 2\nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t) \cdot H_{\hat{\theta}^t}^{-1} \left\{ \boldsymbol{\Omega}^{\boldsymbol{t-1}} \circ (\hat{\theta}^{\boldsymbol{t-1}} - \hat{\theta}^{\boldsymbol{t}}) \right\}^T \lambda, \tag{7}$$

then A_λ^t is a successful agnostic PAC learning algorithm with sample complexity $m_{\mathcal{H}}(\zeta, \delta) \leq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta', \delta) \leq \left\lceil \frac{\log(2|\mathcal{H}^t|/\delta)}{2\zeta'^2} \right\rceil$. $H_{\hat{\theta}^t}$ is the Hessian matrix.

The proof of Theorem 2 can be found in Appendix D.

Remark 2. The restriction that $\zeta \in (0,1)$ is a condition and provides a limitation of λ. Under the conditions ($\zeta = 4\zeta' + 2\nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t) \cdot H_{\hat{\theta}^t}^{-1} \left\{ \boldsymbol{\Omega}^{\boldsymbol{t-1}} \circ (\hat{\theta}^{\boldsymbol{t-1}} - \hat{\theta}^{\boldsymbol{t}}) \right\}^T \lambda$, $\zeta \in (0,1)$), Theorem 2 proves that the learning algorithm of using a regularization-based method will be a successful agnostic PAC learning algorithm with sample complexity $m_{\mathcal{H}}(\zeta, \delta) \leq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta', \delta) \leq \left\lceil \frac{\log(2|\mathcal{H}^t|/\delta)}{2\zeta'^2} \right\rceil$. For convenience, we rewrite the condition as $\zeta = 4\zeta' + 2\lambda \sum_{i=1}^n \Omega_i^{t-1} \Delta_i^{diff} \boldsymbol{g} \boldsymbol{H}_i$, where Ω_i^{t-1} is an element of $\boldsymbol{\Omega}^{\boldsymbol{t-1}}$, Δ_i^{diff} is an element of $\hat{\theta}^{\boldsymbol{t-1}} - \hat{\theta}^{\boldsymbol{t}}$, $\boldsymbol{g} = \nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t)$, $\boldsymbol{H}_i$ is a column of the matrix $H_{\hat{\theta}^t}^{-1}$. ζ is the error difference upper bound of using the regularization-based method on task t. According to the proof of Theorem 2 in Appendix D, ζ' is the error difference upper bound of using **ERM** (i.e. empirical risk minimization) algorithm on task t. When $\lambda = 0$, ζ is totally dependent on ζ'. In this case, the learning algorithm of using a regularization-based method can be regarded as the **ERM** paradigm. This is consistent with setting $\lambda = 0$ of Eq. (1), i.e. $L(\theta) = L_t(\theta) + \lambda \sum_i \Omega_i \left(\theta_i - \theta_i^*\right)^2$, in the main file. When $\lambda > 0$, we first focus on ζ'. The condition implies that: the error of using the **ERM** algorithm is larger, the error of using the regularization-based method is larger. We thus let ζ' very small and ignore it. For $\sum_{i=1}^n \Omega_i^{t-1} \Delta_i^{diff} \boldsymbol{g} \boldsymbol{H}_i$, we find it is hard to be analyzed directly. We turn to analyze its upper bound. By Cauchy-Schwarz, we derive that $\sum_{i=1}^n \Omega_i^{t-1} \Delta_i^{diff} \boldsymbol{g} \boldsymbol{H}_i \leq \sqrt{\sum_{i=1}^n (\Omega_i^{t-1})^2 \sum_{i=1}^n (\Delta_i^{diff} \boldsymbol{g} \boldsymbol{H}_i)^2}$. It is evident that as the values of all Ω_i^{t-1} increase, the upper bound will be so large to violet the condition $\zeta \in (0,1)$. In this case, we have to limit λ to meeting $2\lambda \sum_{i=1}^n \Omega_i^{t-1} \Delta_i^{diff} \boldsymbol{g} \boldsymbol{H}_i \in (0,1)$.

Finally, we discuss how the variance in Theorem 1 theoretically reaches the asymptotic minimum variance.

Theorem 3. *Considering the variance σ^2 in Theorem 1, we prove that it can be bounded by:*

$$\sigma^2 \geq \frac{\left(\frac{\partial}{\partial \gamma} \left[T\left(D_\gamma\right) \right]_{\gamma_*} \right)^2}{J\left(D^t\right)}, \tag{8}$$

which is the Cramér-Rao Lower Bound. σ^2 can achieve the Cramér-Rao Lower Bound, if

$$\Phi_{D^t}(x) = \frac{A}{J\left(D^t\right)} B \tag{9}$$

where $A = \frac{\partial}{\partial \gamma} \left[\ln f_\gamma(x) \right]_{\gamma_*}$ and $B = \frac{\partial}{\partial \gamma} \left[T\left(D_\gamma\right) \right]_{\gamma_*}$, $f_\gamma(x)$ is the density of some sample distribution D_γ.

Remark 3. Theorem 1 shows the asymptotic statistical properties of estimator which results from the regularization-based method. According to [38], the notion of "best" limit distribution requires the asymptotic minimum variance. Theorem 3 shows how the variance theoretically achieve the Cramér-Rao Lower Bound. In this theorem, we assume that D^t is sample distribution for task t and is parameterized by γ_*. $J\left(D^t\right)$ is the Fisher information at D^t. To theoretically reach the lower bound. Eq. (9) should be satisfied. Please refer to the proof of Theorem 3 for details in Appendix E.

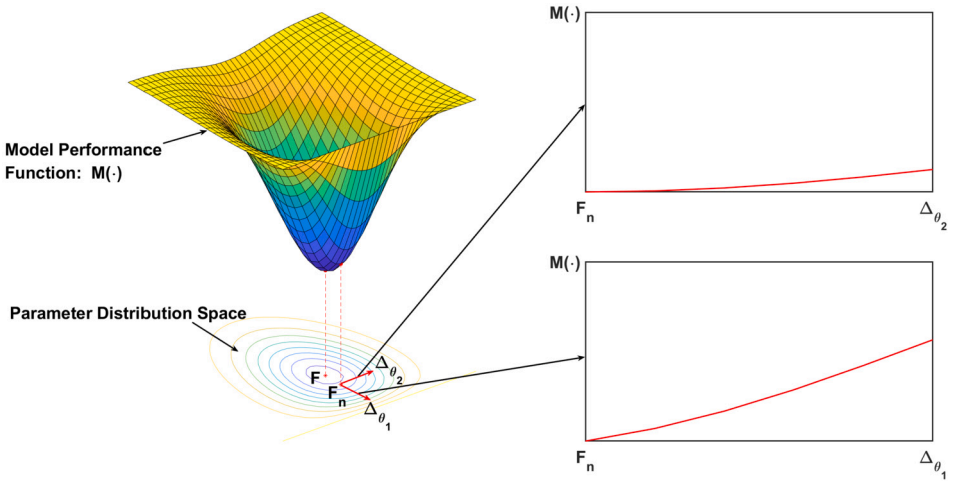


Fig. 1. Illustration of perturbing parameter distribution.

5. Parameters importance via influence function

In this section, we first define the influence function for continual learning. We then show how to calculate the influence of a parameter. We also explore the connection to the previous regularization-based methods and generalize them into a unified framework. Finally, according to the definition and calculation, we proposed a new method called Regularization based on Influence Function (**RIF**) to defy catastrophic forgetting.

5.1. Definition

Definition 4. Let model parameter θ follow a certain distribution F. M is a model performance function, which is a functional $M : domain(M) \to \mathbb{R}$. The domain of M is a convex set of parameter distributions containing F and all the empirical distributions F_n of F. The importance of this parameter θ can thus be defined as the influence function (IF) of M at F:

$$\begin{aligned} &IF(\theta; M, F) \\ &= \lim_{\epsilon \downarrow 0} \frac{M\left((1-\epsilon)F + \epsilon\Delta_\theta\right) - M(F)}{\epsilon} \\ &= \frac{d}{d\epsilon} M\left((1-\epsilon)F + \epsilon\Delta_\theta\right)\Big|_{\epsilon=0} \end{aligned} \tag{10}$$

where Δ_θ is the distribution of the point mass at θ, and ϵ is a small perturbation.

The influence function in our work is different from the typical influence function in statistics. We refer to literatures [15,16] to compare the typical influence function in statistics and the influence function in our work. The typical influence function in [16] is defined as $IF(x; T, F) = \lim_{t \downarrow 0} \frac{T((1-t)F + t\Delta_x) - T(F)}{t}$, while the influence function in our work is defined as $IF(\theta; M, F) = \lim_{\epsilon \downarrow 0} \frac{M\left((1-\epsilon)F + \epsilon\Delta_\theta\right) - M(F)}{\epsilon}$. They both describe the effect of an infinitesimal contamination at the point x (θ) on the estimate. Both T in $IF(x; T, F)$ and M in $IF(\theta; M, F)$ are essentially functionals, but their domains are different. The domain of T is the set of sample distributions, while the domain of M is the set of model parameter distributions. Meanwhile, F in $IF(x; T, F)$ represents the sample distribution while that in $IF(\theta; M, F)$ represents the model parameter distribution. According to Definition 4, we assume that the parameters of the model should follow a certain distribution F for one task. IF can reflect the influence of the model performance when the parameter distribution changes slightly, starting from F. As shown in Fig. 1, the true distribution F is typically agnostic, so we turn to the empirical distribution F_n. When the data is known, the model performance $M(\cdot)$ for one task depends on the parameter distribution. If a slight perturbation according to one parameter can significantly affect the model performance, we can conclude that this parameter plays an important role for the task. Therefore, we can measure the importance of each parameter to an individual task by perturbing the parameter distribution.

5.2. Calculation of influence function

This section outlines how the influence function is calculated. This paper takes M as the loss function. Recalling Section 5.1, we aim to determine the important parameters in terms of parameter distribution. According to Definition 4, we need to consider the perturbation of parameter distribution with respect to a specific parameter θ, which is $(1-\epsilon)F + \epsilon\Delta_\theta$. It is obvious that we cannot

perturb true parameter distribution F directly. We fill this gap by converting parameter distribution perturbation into parameter vector perturbation. Details can be found in Appendix G. The parameter vector perturbation is:

$$\hat{\theta}^t_{\epsilon,i} = \hat{\theta}^t + \epsilon(\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t) \tag{11}$$

where ϵ represents the small perturbation, while $\mathbf{1}$ is an all-ones vector of the same size as $\hat{\theta}^t$. $IF(\hat{\theta}^t_i; M, F_n)$ is written as $IF(\hat{\theta}^t_i)$ for convenience. According to Definition 4, we measure how a small ϵ perturbation affects the loss function of the current task τ_t:

$$\begin{aligned} IF^{(1)}(\hat{\theta}^t_i) &\overset{def}{=} \frac{d}{d\epsilon} L_t(\hat{\theta}^t_{\epsilon,i}; S^t)\Big|_{\epsilon=0} \\ &= \left(\frac{d(L_t(\hat{\theta}^t_{\epsilon,i}; S^t))}{d(\hat{\theta}^t_{\epsilon,i})} \cdot \frac{d(\hat{\theta}^t_{\epsilon,i})}{d\epsilon} \right)\Bigg|_{\epsilon=0} \\ &= \nabla_\theta L_t(\hat{\theta}^t; S^t) \cdot (\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t) \end{aligned} \tag{12}$$

where S^t is the sample set of task τ_t. From the derivation, we can determine that the calculation of $IF^{(1)}(\hat{\theta}^t_i)$ is both simple and fast.

We also provide an efficient method to calculate the second-order influence of parameters:

$$\begin{aligned} IF^{(2)}(\hat{\theta}^t_i) &\overset{def}{=} \left(\frac{d^2}{d\epsilon^2} L_t(\hat{\theta}^t_{\epsilon,i}; S^t) \right)\Big|_{\epsilon=0} \\ &= \frac{d}{d\epsilon} \left(\frac{d(L_t(\hat{\theta}^t_{\epsilon,i}; S^t))}{d(\hat{\theta}^t_{\epsilon,i})} \cdot \frac{d(\hat{\theta}^t_{\epsilon,i})}{d\epsilon} \right)\Bigg|_{\epsilon=0} \\ &= \frac{d}{d\epsilon} \left(\frac{d(L_t(\hat{\theta}^t_{\epsilon,i}; S^t))}{d(\hat{\theta}^t_{\epsilon,i})} \cdot (\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t) \right)\Bigg|_{\epsilon=0} \\ &= \left(\frac{d^2(L_t(\hat{\theta}^t_{\epsilon,i}; S^t))}{d(\hat{\theta}^t_{\epsilon,i})^2} \cdot \frac{d(\hat{\theta}^t_{\epsilon,i})}{d\epsilon} \cdot (\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t) \right)\Bigg|_{\epsilon=0} \\ &= \nabla^2_\theta L_t(\hat{\theta}^t; S^t) \cdot (\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t)^2 \\ &= (\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t) \cdot \nabla^2_\theta L_t(\hat{\theta}^t; S^t) \cdot (\hat{\theta}^t_i \mathbf{1} - \hat{\theta}^t)^T \end{aligned} \tag{13}$$

This calculation of second-order parameter influence involves calculating the second partial derivatives. When the model is complex and has a large number of parameters, such calculation may incur high computational cost. We therefore present an approximate calculation method for the second-order influence of the parameters in the classification tasks.

For simplicity, we let $A(\epsilon) = L_t(\hat{\theta}^t_{\epsilon,i}; S^t)$ where $\epsilon \in [0, 1]$. Since $\hat{\theta}^t_{\epsilon,i} \to \hat{\theta}^t$ as $\epsilon \to 0$, we perform a Taylor expansion of $A(\epsilon)$ at $\epsilon = 0$:

$$\begin{aligned} A(\epsilon) &\approx A(0) + \nabla A(0)\epsilon + \frac{\nabla^2 A(0)}{2!}\epsilon \\ &= A(0) + IF^{(1)}(\hat{\theta}^t_i)\epsilon + \frac{IF^{(2)}(\hat{\theta}^t_i)}{2!}\epsilon \end{aligned} \tag{14}$$

Let $\epsilon = 1$; we thus obtain the following expression according to Eq. (11):

$$\begin{aligned} IF^{(2)}(\hat{\theta}^t_i) &\approx 2 \cdot (A(1) - A(0) - IF^{(1)}(\hat{\theta}^t_i)) \\ &= 2 \cdot (L_t(\hat{\theta}^t_i \mathbf{1}; S^t) - L_t(\hat{\theta}^t; S^t) - IF^{(1)}(\hat{\theta}^t_i)) \end{aligned} \tag{15}$$

$L_t(\hat{\theta}^t; S^t)$ is the loss function, and $L_t(\hat{\theta}^t_i \mathbf{1}; S^t)$ can be easily calculated. Finally, we can approximate $IF^{(2)}(\hat{\theta}^t_i)$ in a very simple way.

5.3. Connection to EWC, SI and MAS

This paper studies continual learning from the IF perspective, which is based on the perturbation over the parameter distribution. If we perturbate a particular parameter, we can rewrite Eq. (11) as $\hat{\theta}^t_{\epsilon,i} = \hat{\theta}^t + \epsilon\boldsymbol{\delta}_i$; here, $\boldsymbol{\delta}_i$ is a vector for which the i-th entry is 1 and the remaining entries is 0. Combined with Eq. (12) and Eq. (13), we obtain the first-order and second-order influence of parameters as follows:

$$\begin{aligned} IF^{(1)}(\hat{\theta}^t_i) &= \left(\frac{d}{d\epsilon} L_t(\hat{\theta}^t_{\epsilon,i}; S^t) \right)|_{\epsilon=0} \\ &= \nabla_\theta L_t(\hat{\theta}^t; S^t) \cdot \boldsymbol{\delta}_i \end{aligned} \tag{16}$$

Algorithm 1 Regularization based on Influence Function.

Input: $(\tau_1, \tau_2, ..., \tau_T)$: All training tasks; T: total number of tasks; S^t: the sample set of task τ_t; L_t: the loss function for task τ_t; $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_n)$: the model parameters; $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, ..., \Omega_n)$: the parameter importance; k: calculating the first-order or second-order influence of parameters.

Initialization: $\boldsymbol{\Omega^0} = \mathbf{0}$
for $t = 1, \ldots, T$ **do**
  if $t == 1$ **then**
    $\hat{\theta}^t = \text{argmin}_\theta \, L_t(\theta; S^t)$
  else
    $\hat{\theta}^t = \text{argmin}_\theta \left\{ L_t(\theta; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1} (\theta_j - \hat{\theta}_j^{t-1})^2 \right\}$
  end if
  $\boldsymbol{\Omega^{new}} = CalculateIF(\hat{\theta}^t, S^t, L_t, k)$
  $\boldsymbol{\Omega^t} = \boldsymbol{\Omega^{t-1}} + \boldsymbol{\Omega^{new}}$
end for

Output: $(\hat{\theta}^1, \hat{\theta}^2, \ldots, \hat{\theta}^T)$

Algorithm 2 CalculateIF.

Input: $\hat{\theta}^t = (\hat{\theta}_1^t, \hat{\theta}_2^t, ..., \hat{\theta}_n^t)$: A local minimum for task τ_t; S^t: the sample set of task τ_t; L_t: the loss function for task τ_t; k: calculating the first-order or second-order influence of parameters.

Initialization: $\boldsymbol{\Omega_{new}} = \mathbf{0}$
for $i = 1, \ldots, n$ **do**
  $\Omega_i^{new} = | \nabla_\theta L_t(\hat{\theta}^t; S^t) \cdot (\hat{\theta}_i^t \mathbf{1} - \hat{\theta}^t) |$ which is based on Eq. (12)
  if k$==2$ **then**
    $\Omega_i^{new} = 2 \, | (L_t(\hat{\theta}_i^t \mathbf{1}; S^t) - L_t(\hat{\theta}^t; S^t) - \Omega_i^{new}) |$ which is Based on Eq. (15)
  end if
end for

Output: $\boldsymbol{\Omega^{new}}$

$$
\begin{aligned}
IF^{(2)}(\hat{\theta}_i^t) &= \left(\frac{d^2}{d\epsilon^2} L_t(\hat{\boldsymbol{\theta}}_{\epsilon,\boldsymbol{i}}^{\boldsymbol{t}}; S^t)\right)\Big|_{\epsilon=0} \\
&= \frac{d}{d\epsilon}\left(\frac{d(L_t(\hat{\boldsymbol{\theta}}_{\epsilon,\boldsymbol{i}}^{\boldsymbol{t}}; S^t))}{d(\hat{\boldsymbol{\theta}}_{\epsilon,\boldsymbol{i}}^{\boldsymbol{t}})} \cdot \frac{d(\hat{\boldsymbol{\theta}}_{\epsilon,\boldsymbol{i}}^{\boldsymbol{t}})}{d\epsilon} \right)\Big|_{\epsilon=0} \\
&= \boldsymbol{\delta_i} \cdot \nabla_\theta^2 L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) \cdot \boldsymbol{\delta_i}^T
\end{aligned}
\tag{17}
$$

In this case, EWC is equivalent to $IF^{(2)}(\hat{\theta}_i^t)$; this is because the importance of the parameters calculated by EWC is essentially the second-order gradient of each parameter. $IF^{(1)}(\hat{\theta}_i^t)$ is an important component of SI in calculating parameter importance. MAS also uses 1st-order parameter influence $IF^{(1)}(\hat{\theta}_i^t)$. The difference is that MAS focuses on the model output rather than the loss function. The model performance function M in Definition 4 can be seen as the output of the model, so we generalize EWC, SI and MAS in a unified framework.

5.4. Regularization based on influence function

According to the definition and calculation in Section 5, we propose a new method called Regularization based on Influence Function (**RIF**) to defy catastrophic forgetting. Algorithm 1 shows the entire process of training model with Regularization which is based on influence function. Algorithm 2 shows how to calculate the parameter importance Ω using Eq. (12) or (15). We return the absolute value of influence. We define RIF-1 to indicate that RIF calculates the first-order influence of parameters. We define RIF-2 to indicate that RIF calculates the second-order influence of parameters.

We emphasize that the influence function in our work is different from the typical influence function in statistics. The typical influence function in statistics concentrates on samples and is usually leveraged to identify the training points most responsible for a given prediction [19]. The regularization-based methods in continual learning focus on parameters and the fundamental question is that of which parameters in the prediction (model) are important for previous tasks. Definition 4 is proposed to fill this gap. Compared with the typical influence function in statistics [15,16], the influence function in Definition 4 concentrates on parameters and describes the effect of an infinitesimal contamination at the parameter θ on the estimate. The domain of M is the set of model parameter distributions. Based on Definition 4, we conclude a parameter vector perturbation, i.e. Eq. (11). According to this parameter vector perturbation, we derive how to measure the first-order and the second-order influence of a parameter, which are the bases of RIF. Therefore, RIF is the first to extend the concept of influence function to continual learning, which measures the influence of parameters from a high-level perspective of parameter distribution perturbation.

RIF also provides a novel way of generalizing the previous regularization-based methods in a unified framework. The previous regularization-based methods come from a straightforward parameter vector perturbation without rigorous derivation, which is empirical and intuitive. Our proposed method RIF starts from a clear definition (i.e. Definition 4), which contributes to the rigorous parameter vector perturbation (i.e. Eq. (11)). Therefore, RIF is more reasonable than the previous regularization-based methods to characterize the parameter importance information, which improves empirical performance.

6. Experiments

We conduct extensive experiments to verify the superior performance of our proposed method. In this paper, we mainly consider three scenarios in continual learning which are task-incremental learning (Task-IL) [39,40], class-incremental learning (Class-IL) [39,40] and class-incremental with repetition learning (Class-IRL) [10,27]. Task-incremental learning is best described as the case where an algorithm must incrementally learn a set of distinct tasks. The defining characteristic of task-incremental learning is that it is always clear to the algorithm—also at test time—which task must be performed. Class-incremental learning is best described as the case where an algorithm must incrementally learn to discriminate between a growing number of objects or classes. An often used set-up for this scenario is that a sequence of classification-based tasks is encountered, whereby each task contains different classes and the algorithm must learn to distinguish between all classes. Class-incremental with repetition learning is an extension of Class-incremental learning. In class-incremental with repetition learning, new training patterns belonging both to known and new classes become available in subsequent training batches.

6.1. Experimental setup

Baselines: In order to verify the validity of our proposed method, we select eleven baselines for comparison: **SI** [43], **MAS** [1], **EWC** [18], **IMM** [22], **FROMP** [30], **LwF** [24], **iCaRL** [33], **DER** [5], **GEM** [28], **A-GEM** [6], **PNN** [35], **CUBER** [25], **S-FSVI** [34] and **Finetuning**. **SI**, **MAS**, **EWC**, and **IMM** can be considered as the prior-focused methods, which estimate a distribution of model parameters as the prior when learning from new data. Typically, the importance of all neural network parameters is estimated, with parameters assumed to be independent to ensure feasibility. **FROMP** opens a new direction for continual learning methods in which regularization methods are naturally combined with memory-based methods. **LwF** is a data-focused method. Its basic building block is knowledge distillation from a previous model to the model being trained on the new data. **iCaRL**, **DER**, **GEM**, and **A-GEM** are replay methods which retain on a subset of stored samples while training on new tasks or constrain new task updates to not interfere with previous tasks. **PNN** is a parameter isolation method. **CUBER** characterizes the task correlation to identify the positively correlated old tasks in a layer-wise manner, and then selectively modifies the learnt model of the old tasks when learning the new task. **S-FSVI** frames continual learning as sequential function-space variational inference and adapts the variational objective to the continual-learning setting. **Finetuning** trains the model without considering the catastrophic forgetting in previous tasks. More details of these baselines can be seen in Appendix A.

Datasets: In Task-IL, we use four datasets: Split MNIST, Split Cifar100, Tiny Imagenet and the real-world dataset iNaturalist. Split MNIST consists of five binary classification tasks built from MNIST: 0/1, 2/3, 4/5, 6/7, and 8/9. Split Cifar100, a more difficult benchmark than MNIST, comprises of 10 tasks that consist of 10 consecutive classes from CIFAR-100. Tiny Imagenet is a subset of 200 classes from ImageNet [12], rescaled to an image size of 64×64. Each class contains 500 samples subdivided into training (80%) and validation sets (20%), along with 50 samples for evaluation. In order to construct a balanced dataset, we assign an equal amount of 20 randomly chosen classes to each task in a sequence of 10 consecutive tasks. This task-incremental setting allows for the use of an oracle at test time for our evaluation per task, ensuring that all tasks are roughly similar in terms of difficulty, size, and distribution. iNaturalist [11] is a real-world dataset with highly imbalanced classes. In this paper, we select eight super-categories from the total of 14 and only retain categories with at least 100 samples. We only use the training data, which is subdivided into training (70%), validation (20%) and evaluation (10%) sets. In Class-IL, we conduct experiments on two widely used datasets: Cifar100 and ImageNet [12]. In this scenario, we follow the work of [26] and give a similar division of datasets which has 1 initial task and 4 incremental tasks. For Cifar100, the initial task consists of 50 random classes, and all incremental tasks have same numbers of classes. The number of classes in one incremental task is set to be 5 or 10. ImageNet contains around 1.3 million samples of 224×224 color images from 1000 classes. Therefore, for ImageNet, the initial task consists of 500 random classes, and the number of classes in one incremental task is set to be 50 or 100. In Class-IRL, we use CORe50 [27] to conduct experiments. CORe50 has been collected in 11 distinct sessions (8 indoor and 3 outdoor) characterized by different backgrounds and lighting. It is a collection of 50 domestic objects belonging to 10 categories. Classification in this paper is performed at object level (50 classes). We follow up the work of [27] which generates 79 tasks from CORe50. The first task includes 10 classes, and each subsequent task contains 5 classes. The details of these datasets can be seen in Appendix A.

Models: In Task-IL, we use four different types of models to accommodate the four different datasets. It is also possible to examine the differences in the performance of methods across different models. For Split MNIST, we use a fully connected single-head network with two hidden layers, each comprising 256 hidden units. For Split Cifar100, we use the same model architecture as in [43]: a multi-head CNN with four convolutional layers, then two dense layers with dropout. A model based on a VGG configuration [9] and ResNet-18 [17] are used for Tiny Imagenet. The VGG model has a classifier consisting of two fully connected layers, each with 512 units. To explore the effect of model size on methods, we also adopt two VGG nets, one with wider layers and the other with more layers. Moreover, the size of iNaturalist is huge, which makes learning arduous; therefore we conduct the experiments solely for AlexNet [20], pretrained on ImageNet. The multi-head setting is adopted here to impose a separate fully connected layer with softmax for each task, with the number of outputs defined by the classes in that task. In Class-IL and Class-IRL, we follow up the work of [26], where models based on ResNet configurations are used.

Evaluation: We mainly use the accuracy or forgetting rates to measure the performance of the method. We first define $acc_{i,j}$ as the accuracy where learned model M_i is evaluated on the test data S^j_{test} of task j. In Task-IL, the final accuracy for task j is defined as $acc_{N_{task},j}$, where model $M_{N_{task}}$ is the final model. The forgetting rate for task j is defined as $acc_{j,j} - acc_{N_{task},j}$ where $acc_{j,j}$ can be seen as the initial accuracy for task j. In Class-IL, the evaluation metric is the accuracy $acc_{i,0:i}$, where learned model M_i is evaluated

Table 1
Final accuracy for each task after training all tasks on Split MNIST. (For interpretation of the references to colour please refer to the web version of this article.)

	Task 1	Task 2	Task 3	Task 4	Task 5	Avg
SI	99.57	90.60	90.82	98.89	98.79	95.74
EWC	99.43	60.14	55.39	96.78	99.14	82.18
RIF-1	100.00	96.91	98.99	98.59	96.32	98.16
RIF-2	99.95	95.84	98.56	98.74	96.02	97.82
MAS	99.91	88.83	89.54	98.99	98.64	95.18
FROMP	97.54	92.41	85.22	98.49	96.97	94.13
mode-IMM	80.05	91.28	75.45	97.08	82.20	85.21
mean-IMM	54.94	62.29	18.84	96.02	98.49	66.12
LwF	97.72	97.43	98.12	98.49	98.34	98.02
Finetuning	33.19	61.36	14.83	89.33	99.34	59.61
A-GEM	96.44	96.33	97.03	97.42	98.91	97.22
DER	96.16	95.72	96.22	96.98	98.78	96.77
GEM	95.49	95.28	95.88	96.01	98.18	96.17
iCaRL	96.86	96.93	97.45	97.76	99.15	97.63
PNN	94.99	94.81	95.58	95.67	98.59	95.93

Table 2
Forgetting rate for each task after training all tasks on Split MNIST.

	Task 1	Task 2	Task 3	Task 4	Avg
SI	0.38	9.01	8.59	0.40	4.60
EWC	0.52	39.08	44.34	2.72	21.67
RIF-1	-0.05	0.73	-0.37	0.35	0.17
RIF-2	0.00	0.49	0.00	0.10	0.15
MAS	0.05	10.43	10.19	-0.05	5.15
FROMP	2.41	6.46	14.35	0.50	5.93
mode-IMM	19.91	7.35	17.45	-0.30	11.10
mean-IMM	45.01	36.53	80.79	3.27	41.40
LwF	2.23	1.54	1.50	1.06	1.58
Finetuning	66.76	37.86	85.01	10.27	49.97
A-GEM	3.51	1.63	1.55	1.03	1.93
DER	3.79	1.71	1.81	0.96	2.07
GEM	4.46	1.43	1.64	1.32	2.21
iCaRL	3.09	1.08	1.25	1.28	1.68
PNN	4.97	1.34	1.32	1.00	2.16

on the test data $S_{test}^{0:i}$. $S_{test}^{0:i}$ denotes all seen data (classes) so far. In Class-IRL, we evaluate the learned model M_i by the accuracy $acc_{i,FTS}$ of M_i on a Full Test Set [27] which is fixed and includes patterns of all the classes.

6.2. Results in Task-IL scenario

Split MNIST: After training all tasks, the final accuracy of each task is presented in Table 1. We select the best three results in each column, which are marked in red, green and blue respectively. It is obvious that RIF achieves better performance than all other baselines. RIF with first-order influence achieves top three-ranked accuracy in tasks 1, 2, 3 with the best average accuracy of 98.16%. It can be seen that LwF also achieves good performance on Split MNIST: it has the second-best average accuracy, at 98.02%, and also demonstrates good performance on tasks 2 and 3. Finetuning performs the worst on all tasks and can be regarded as the minimum desired performance. In general, the accuracy of RIF is 0.53%–38.55% higher than all other approaches apart from LwF. Next, we focus on the forgetting rate, which is shown in Table 2. With its 0.15% and 0.17% average forgetting rate, RIF achieves the highest overall performance after training all tasks. The next highest is LwF with 1.58% forgetting. Overall, RIF consistently outperforms other approaches on Split MNIST. LwF shows competitive results. Fig. 2 plots the changes of all approaches when new tasks are added. It can be clearly seen that the accuracy of RIF changes very little, which indicates that RIF model learning can remember the previous tasks well.

Split Cifar100: The task in Split MNIST is a binary classification problem, which is too simple for our purposes. Split Cifar100 is a more difficult benchmark consisting of 10 tasks. All accuracies are summarized in Fig. 3. It is worth noting that the mode-IMM achieves the best performance on task 1. The curve of mode-IMM on task 1 is relatively flat and obtains the highest accuracy. The mode-IMM also shows recovery through backward transfer in subsequent tasks, which results in a negative forgetting rate. These results are listed in Table 4. This phenomenon (that the accuracy of mode-IMM begins at a low value and increases as new tasks are added) can also be seen in the following experiments on Tiny Imagenet. From Table 3 and Table 4, we can determine that RIF achieves a competitive result. The average accuracies of first- and second-order RIF are 56.44% and 57.27%, representing the best accuracies among the baselines. At same time, the forgetting rates of RIF are the second best across all approaches.

Tiny Imagenet: We first focus on the results by VGG model. Fig. 4 shows that results of all approaches (including A-GEM, DER, GEM, iCaRL and PNN), while the final accuracy and forgetting rates can be seen in Tables 5 and 6. PNN only trains models on 8 tasks

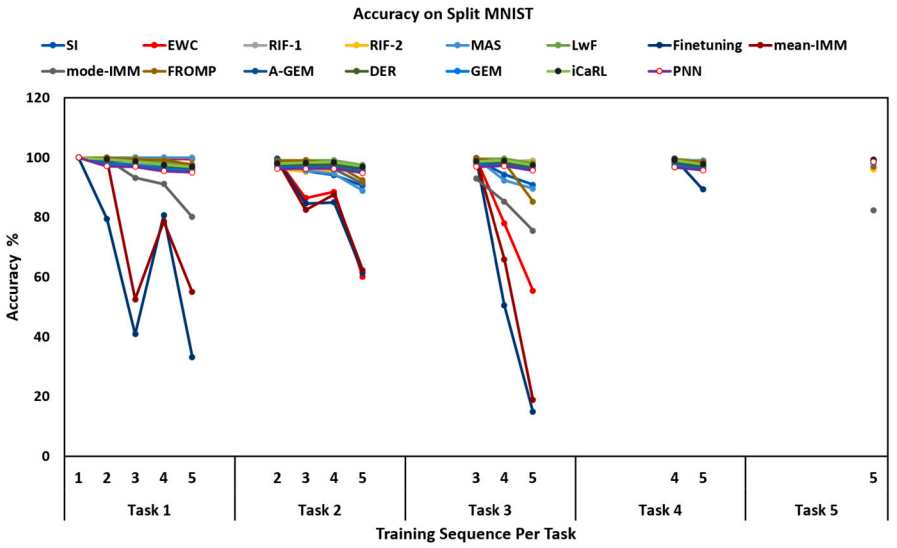


Fig. 2. Results on Split MNIST. The figure consists of five subpanels, with each subpanel showing the evolution of test accuracy for a specific task as more tasks are added for training. Figs. 3, 4 and 5 have the same structure. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 3
Final accuracy for each task after training all tasks on Split Cifar100.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Avg
SI	38.60	41.30	49.80	43.60	52.40	59.30	58.00	58.60	62.90	72.90	53.74
EWC	38.40	38.50	47.70	47.20	50.20	54.70	58.70	59.70	57.50	73.50	52.61
RIF-1	45.80	46.30	54.30	49.70	54.40	60.60	59.60	59.40	61.80	72.50	56.44
RIF-2	48.30	45.30	59.10	52.60	55.40	63.20	60.50	55.90	62.30	70.10	57.27
MAS	46.80	41.90	45.40	50.60	54.40	60.80	62.10	56.60	57.50	71.80	54.79
FROMP	44.30	29.40	31.50	34.90	45.50	51.30	53.70	51.10	50.60	67.30	45.96
mode-IMM	56.90	51.80	63.80	57.80	61.70	59.40	58.10	58.30	46.50	49.50	56.38
mean-IMM	37.70	32.20	38.20	42.40	48.90	55.60	53.40	57.60	61.20	74.90	50.21
LwF	47.00	43.10	50.30	51.00	54.10	57.70	60.60	63.30	62.00	72.80	56.19
Finetuning	9.60	5.30	9.40	8.60	5.60	11.60	21.20	13.30	14.90	76.20	17.57
AGEM	45.15	42.03	45.71	52.09	52.89	59.24	62.59	56.17	57.31	72.01	54.52
DER	44.27	39.90	43.53	47.49	50.69	59.51	58.96	53.82	56.84	71.65	52.67
GEM	42.89	37.22	40.73	45.96	48.65	55.98	57.90	52.92	53.55	71.37	50.72
iCaRL	44.17	39.73	42.25	48.26	53.13	57.02	59.54	54.53	54.30	71.08	52.40
PNN	41.01	37.65	40.68	45.79	49.27	56.05	58.38	50.35	52.53	71.56	50.33

Table 4
Forgetting rate for each task after training all tasks on Split Cifar100.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Avg
SI	21.80	16.50	18.80	19.60	15.20	9.40	11.70	9.00	1.20	13.69
EWC	22.00	20.00	22.10	16.30	17.00	13.20	10.30	8.20	5.90	15.00
RIF-1	14.60	10.60	14.30	14.20	11.80	6.80	10.00	8.20	2.60	10.34
RIF-2	12.10	11.90	9.80	10.80	10.00	5.00	9.00	8.80	0.90	8.70
MAS	13.60	14.60	21.90	12.50	11.10	6.80	6.70	9.00	5.30	11.28
FROMP	28.70	31.00	38.40	27.70	21.60	17.10	14.00	10.00	9.50	22.00
mode-IMM	5.80	3.40	-4.10	-3.90	-6.60	-8.90	-5.30	-5.50	-2.80	-3.10
mean-IMM	25.00	26.10	31.80	21.60	20.50	12.80	18.00	10.10	5.60	19.06
LwF	13.40	17.20	18.90	15.00	14.10	13.00	11.40	5.50	4.70	12.58
Finetuning	51.80	54.30	63.90	57.40	64.70	60.80	50.50	53.70	54.70	56.87
AGEM	15.85	15.36	20.73	11.78	12.34	8.35	7.08	9.08	4.81	11.71
DER	16.93	14.67	23.17	13.84	14.80	7.37	8.76	9.91	4.91	12.71
GEM	17.51	16.65	23.06	13.15	14.05	7.80	7.05	8.85	7.23	12.82
iCaRL	18.43	15.64	22.63	13.55	9.65	9.04	8.01	9.87	6.78	12.62
PNN	19.39	15.47	22.67	14.86	13.74	9.21	8.20	13.09	7.76	13.82

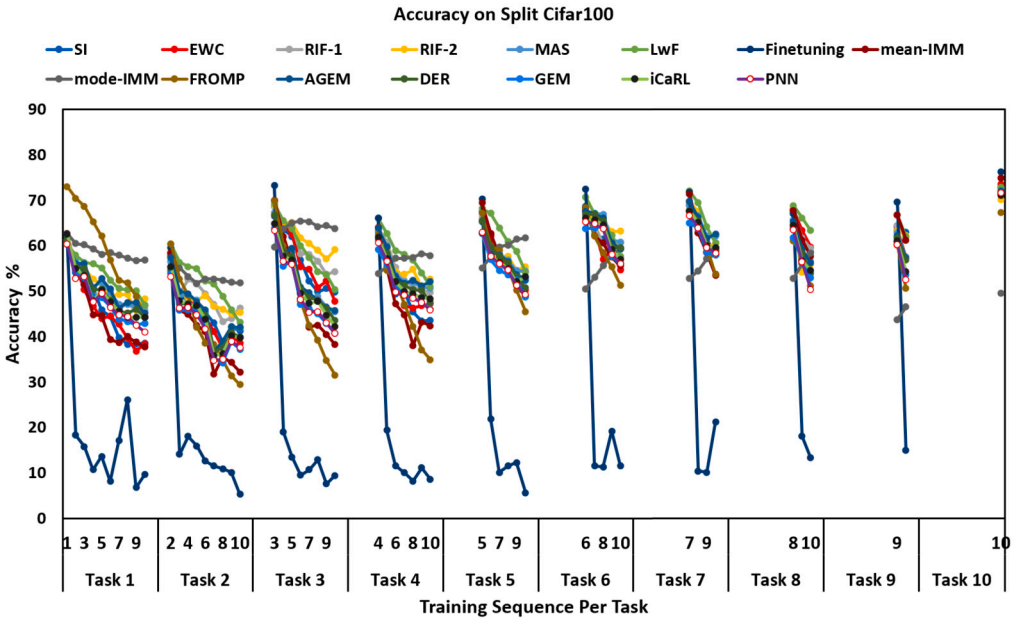


Fig. 3. Results on Split Cifar100.

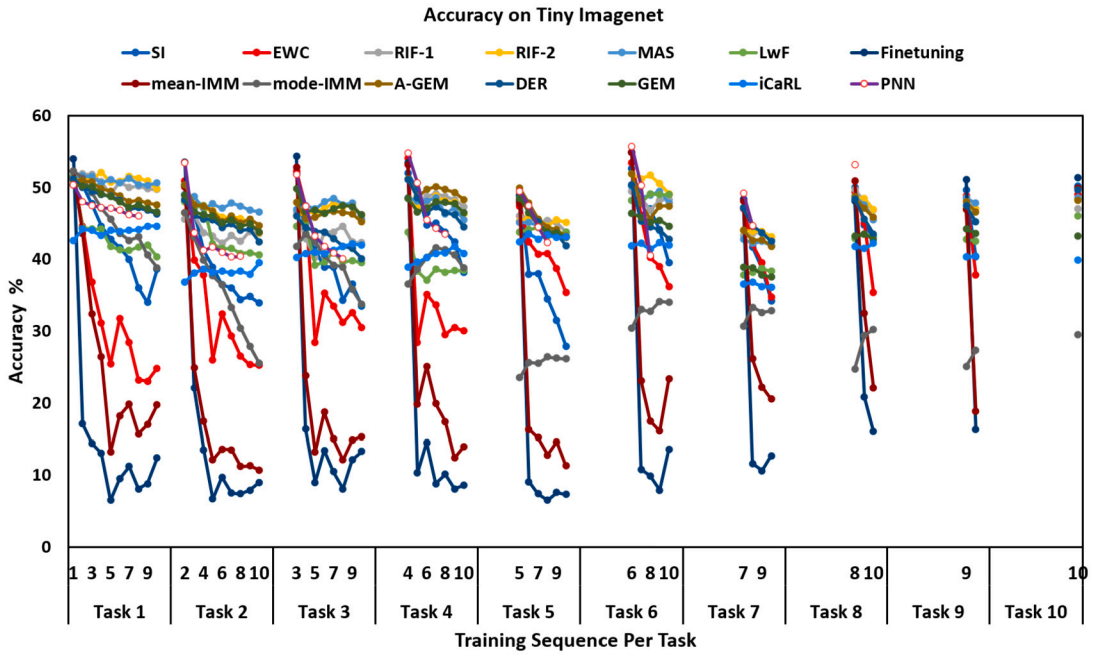


Fig. 4. Results by VGG model on Tiny Imagenet.

because of the limit of GPU memory. With 46.54% average accuracy, RIF-2 shows the highest overall performance after training all tasks on Tiny Imagenet. MAS achieves a competitive result with 46.30% average accuracy and 1.88% forgetting rate. A-GEM shows the best forgetting rates at -1.03%, but RIF also has the competitive forgetting rates at 1.37% and 1.97%. It is worth noting that mode-IMM and iCaRL also exhibits the phenomenon described in above experiment on Split Cifar100. Unfortunately, IMM obtains worse accuracies than other strategies for continual learning on Tiny Imagenet. iCaRL also has a poor performance on final accuracy for each task. The final accuracy and forgetting rates based on ResNet18 can be seen in Table 7. CUBER and S-FSVI achieve competitive results. The average final accuracies of them are all over 65% while those of EWC and MAS are 43.13% and 64.43%, respectively. CUBER achieves the best average final accuracy (67.73%). However, our proposed RIF with first-order and second-order influence achieve very competitive results with final accuracies of 67.52% and 67.43%, respectively. There are six tasks where RIF with first-order and second-order influence achieve the best final accuracy. The best baseline CUBER only outperforms our RIF on Tasks 9 and 10 in terms

Table 5
Final accuracy for each task by VGG model after training all tasks on Tiny Imagenet.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Avg acc
SI	38.60	33.90	33.50	38.20	27.90	39.50	34.20	43.00	40.30	50.20	37.93
EWC	24.80	25.20	30.50	30.00	35.40	36.20	34.70	35.40	37.80	49.00	33.90
RIF-1	49.70	43.60	42.30	47.30	43.80	48.30	42.80	46.80	47.00	47.00	45.86
RIF-2	49.80	43.70	46.00	46.30	45.10	49.10	43.10	46.90	47.00	48.40	46.54
MAS	50.60	46.60	45.60	45.50	43.10	47.80	41.90	45.50	47.80	48.60	46.30
mode-IMM	38.80	25.50	33.70	38.80	26.10	34.00	32.80	30.20	27.30	29.50	31.67
mean-IMM	19.70	10.70	15.30	13.90	11.30	23.30	20.50	22.10	18.80	49.90	20.55
LwF	40.30	40.60	39.50	38.40	43.80	49.00	38.30	43.20	42.50	46.00	42.16
Finetuning	12.40	9.00	13.30	8.60	7.30	13.60	12.70	16.00	16.30	51.40	16.06
A-GEM	47.60	44.70	45.20	48.30	43.20	47.40	41.80	45.80	46.60	48.20	45.88
DER	46.30	42.40	40.10	44.50	41.90	42.80	42.50	43.50	45.20	49.60	43.88
GEM	46.60	43.80	46.20	46.50	43.00	44.60	37.50	42.70	43.60	43.20	43.77
iCaRL	44.60	39.50	42.00	40.80	43.10	42.00	36.10	42.20	40.40	39.90	41.06
PNN	46.00	40.40	40.10	43.50	42.30	40.50	44.70	53.20			43.84

Table 6
Forgetting rate for each task by VGG model after training all tasks on Tiny Imagenet.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Avg
SI	12.90	16.50	18.40	13.80	21.00	13.10	13.00	7.00	9.30	13.89
EWC	26.70	25.70	21.80	24.10	12.00	17.20	12.30	14.00	11.00	18.31
RIF-1	1.80	2.00	4.30	1.20	2.20	1.20	0.50	2.90	1.60	1.97
RIF-2	1.70	2.80	0.00	2.30	0.20	2.80	0.90	1.60	0.00	1.37
MAS	0.90	2.40	1.30	3.10	2.00	2.40	0.90	3.20	0.70	1.88
mode-IMM	13.50	21.00	8.10	-2.30	-2.60	-3.60	-2.10	-5.50	-2.20	2.70
mean-IMM	32.60	39.40	37.50	39.30	36.50	31.60	27.60	28.80	28.10	33.49
LwF	11.20	7.40	5.10	5.40	0.00	-0.80	-0.50	-0.30	0.30	3.09
Finetuning	41.60	44.50	41.00	44.80	40.90	41.30	35.70	32.10	34.80	39.63
A-GEM	3.40	5.70	2.70	2.90	6.70	4.50	2.20	2.90	1.40	3.60
DER	4.90	5.90	5.90	6.60	7.00	7.60	4.60	4.90	2.30	5.52
GEM	3.80	5.10	3.60	2.00	5.50	1.80	1.40	0.50	0.60	2.70
iCaRL	-2.00	-2.70	-1.80	-1.90	-0.70	-0.10	0.40	-0.40	-0.10	-1.03
PNN	4.40	13.00	11.70	11.30	7.20	15.20	4.50			9.61

Table 7
Final accuracy and forgetting rate for each task by ResNet-18 after training all tasks on Tiny Imagenet.

		Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Avg acc
Accuracy	EWC	17.94	21.40	30.09	35.98	40.26	45.09	52.72	58.31	63.62	65.89	43.13
	RIF-1	57.35	58.53	62.82	66.75	66.12	66.71	73.74	73.24	74.58	75.37	67.52
	RIF-2	57.94	61.40	64.09	65.98	64.26	65.09	72.72	73.31	73.62	75.89	67.43
	MAS	55.19	55.95	59.83	61.08	60.71	64.85	68.59	70.62	72.04	75.39	64.43
	DER	52.11	59.11	62.97	66.25	63.76	67.37	70.36	71.07	72.54	77.89	66.34
	CUBER	56.71	60.36	62.39	64.69	65.64	67.28	73.38	72.63	75.87	78.32	67.73
	S-FSVI	55.96	57.17	63.57	61.26	59.99	60.74	70.29	74.29	74.17	76.27	65.37
Forgetting	EWC	62.31	58.15	50.50	44.27	38.49	32.48	24.08	20.36	10.17		30.31
	RIF-1	22.71	22.00	18.58	14.80	13.80	10.32	4.23	7.72	2.45		12.96
	RIF-2	22.11	18.33	16.41	13.78	13.36	11.82	3.53	2.22	0.38		11.33
	MAS	24.85	23.66	19.89	19.58	16.31	15.92	10.92	7.97	1.59		15.63
	DER	28.69	20.95	17.99	13.64	14.77	9.37	9.14	7.43	3.11		13.90
	CUBER	24.15	20.47	16.68	14.86	12.67	12.33	3.58	4.24	1.22		12.24
	S-FSVI	24.87	23.18	16.49	17.09	19.60	20.06	6.78	4.71	2.49		15.03

of final accuracy. Our proposed RIF with second-order influence achieves the best forgetting rate (11.33%) while CUBER also has the competitive forgetting rates at 12.24%. There are five tasks where RIF with first-order and second-order influence achieve the best forgetting rate. The best baseline DER only outperforms our RIF on Tasks 4 and 6 in terms of forgetting rate.

iNaturalist: Both Cifar100 and Tiny Imagenet are well balanced, with an equal amount of data and classes per task. Split MNIST can also be identified as a balanced task sequence, since the amount of data per task is very closed. Our goal is to scrutinize highly unbalanced task sequences, both in terms of classes and available data per task. We train AlexNet, pretrained on Imagenet, and track task performance for iNaturalist in Fig. 5.

The accuracy and forgetting rates are listed in Tables 8 and 9. With 62.57% and 62.25% average accuracies, RIF achieves the highest overall performance after training all tasks on iNaturalist, while GEM also achieves a competitive performance with 61.3% average accuracy. Other baselines like DER, SI, iCaRL, mean-IMM, PNN, EWC, MAS and A-GEM achieve similar average accuracies ranging from 59.59% to 61.25%. The forgetting rates in Table 9 also reflect the same trend.

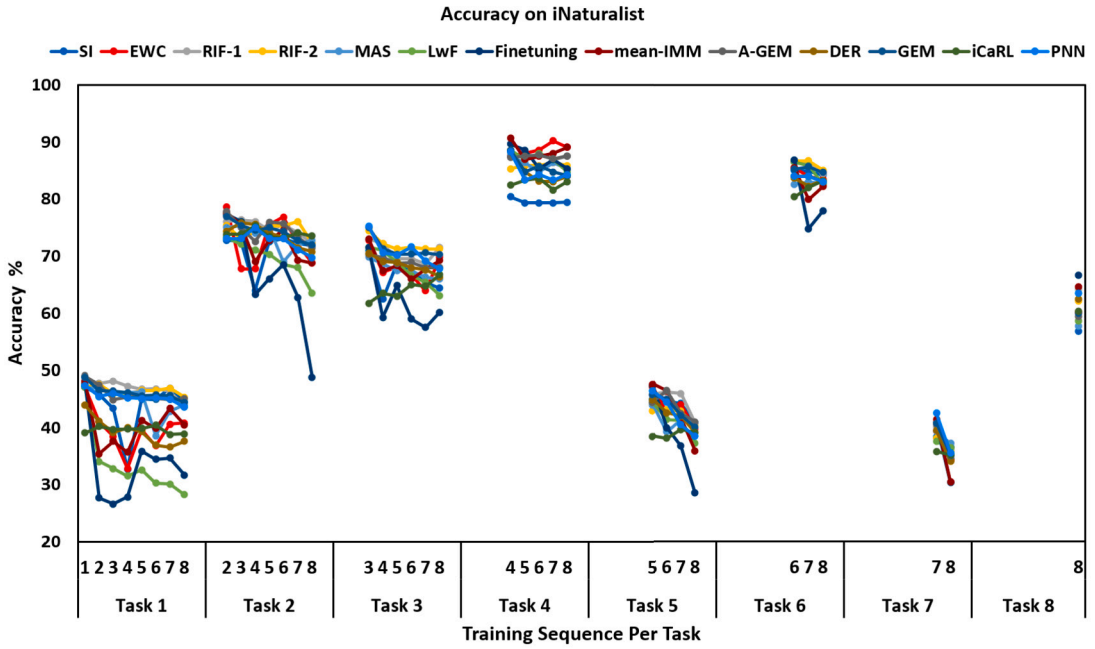


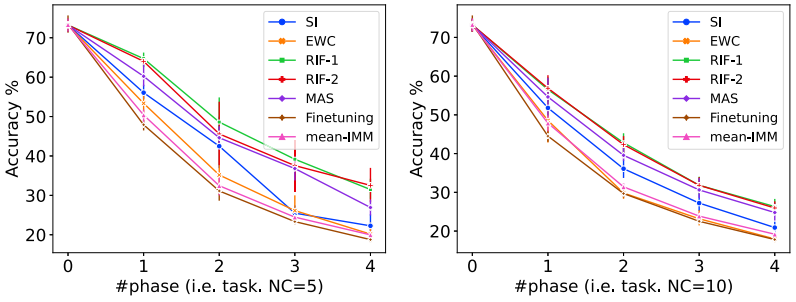
Fig. 5. Results on iNaturalist.

Table 8
Final accuracy for each task after training all tasks on iNaturalist.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Avg acc
SI	44.56	71.80	64.40	79.43	40.80	84.28	36.56	56.83	59.83
EWC	40.82	68.77	69.79	89.13	39.69	83.44	34.29	60.06	60.75
RIF-1	43.65	72.29	71.55	87.50	40.79	84.46	36.08	64.22	62.57
RIF-2	45.36	72.29	71.26	85.87	40.30	85.07	35.77	62.11	62.25
MAS	43.98	72.80	65.98	85.33	40.75	82.82	37.14	57.72	60.81
mean-IMM	40.49	68.77	69.21	89.13	35.88	82.21	30.36	64.65	60.09
LwF	28.24	63.48	63.05	84.78	37.18	82.62	36.35	58.63	56.79
Finetuning	31.67	48.87	60.12	85.33	28.47	77.91	30.26	66.69	53.66
A-GEM	45.09	71.41	67.97	87.58	40.94	83.08	34.50	59.44	61.25
DER	37.58	70.73	66.34	84.04	39.05	83.30	34.02	62.46	59.69
GEM	44.36	71.94	70.26	84.09	39.97	84.71	35.08	60.02	61.30
iCaRL	38.91	73.52	66.75	83.05	39.07	83.15	35.15	60.35	59.99
PNN	43.59	69.71	67.75	84.33	38.41	83.05	35.45	63.44	60.72

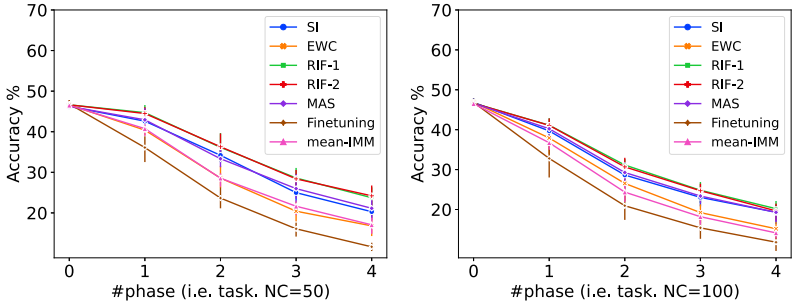
Table 9
Forgetting rate for each task after training all tasks on iNaturalist.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Avg
SI	2.51	1.00	6.57	1.00	4.16	0.80	1.79	2.55
EWC	6.25	9.82	2.93	-1.63	7.41	2.04	5.06	4.56
RIF-1	3.42	3.78	1.47	0.54	5.02	2.04	3.96	2.89
RIF-2	1.71	3.02	3.23	-0.54	2.67	1.64	2.31	2.00
MAS	3.09	2.27	3.81	2.17	3.24	-0.20	2.17	2.36
mean-IMM	7.64	8.82	3.81	1.63	11.75	3.27	11.05	6.85
LwF	18.83	9.82	8.50	3.80	7.21	3.89	1.14	7.60
Finetuning	16.06	28.46	11.44	4.35	18.91	9.00	10.53	14.11
A-GEM	3.95	6.43	2.29	-0.26	3.49	0.94	5.95	3.26
DER	6.35	3.64	4.16	4.58	5.87	0.39	5.42	4.34
GEM	4.42	5.05	4.80	4.50	5.76	0.57	5.65	4.39
iCaRL	0.20	0.30	-5.04	-0.59	-0.63	-2.72	0.60	-1.13
PNN	3.79	3.38	7.44	4.00	8.00	1.00	7.00	4.95



(a) Cifar100 NC=5 (b) Cifar100 NC=10

Fig. 6. Figs. 6(a) and 6(b) represent the results of baselines on Cifar100 in Class-IL. In the 0–th phase, there are 50 classes. The number of classes (NC) of each subsequent phase is set to 5 or 10.



(a) ImageNet NC=50 (b) ImageNet NC=100

Fig. 7. Figs. 7(a) and 7(b) represent the results of baselines on ImageNet in Class-IL. In the 0–th phase, there are 500 classes. The number of classes (NC) of each subsequent phase is set to 50 or 100.

From the above experimental results on four datasets, we can summarize that our proposed RIF consistently achieves higher accuracy and a lower forgetting rate than other baselines on four datasets; moreover, RIF is effective enough to defy the catastrophic forgetting of models. To better understand the performance of RIF, we visualize the results of RIF and Finetuning on Split MNIST. Details can be seen in Appendix B.

6.3. Results in Class-IL and Class-IRL scenarios

In this section, we challenge the more difficult scenarios in continual learning: Class-IL and Class-IRL.

Class-IL: To be consistent to related works [26], we call task as "phase" in this scenario. Figs. 6(a) and 6(b) represent the results of baselines on Cifar100. Figs. 7(a) and 7(b) represent the results of baselines on ImageNet in Class-IL. The performance of learned model M_i on $S_{test}^{0:i}$—all seen classes so far—drops with phases. As can be seen in Figs. 6(a) and 6(b), RIF with first-order or second-order influence performs better than other baselines in Cifar100 no matter what the number of classes of subsequent phases is. The results in ImageNet are same, which can be found in Figs. 7(a) and 7(b). Therefore, RIF performs better than other baselines in Class-IL.

Class-IRL: To be consistent to related works [27], we call task as "batch" in this scenario. As can be observed in Fig. 8, the performance of model trained by each baseline is not stable, but the accuracy ($acc_{i,FTS}$) tends to increase with new batches. This means that model can accumulate the knowledge of classes in Class-IRL. Overall, RIF performs better than other baselines. By checking the ending points of the curves in Fig. 8, we find that the model trained by RIF accumulates more knowledge about classes than that trained by other baselines, where the final accuracies ($acc_{78,FTS}$) of model trained by RIF-1 and RIF-2 are 28.04% and 29.84% respectively. The final accuracies of models trained by other baselines like SI, EWC, MAS, mean-IMM and Finetuning are 25.06%, 26.77%, 26.25%, 26.05% and 18.33% respectively. Therefore, RIF performs better than other baselines in Class-IRL.

6.4. Further analysis

This subsection analyzes the influence of other factors, including model size, task order and mixed information in Task-IL. We also count the time cost and memory usage of RIF and other methods.

6.4.1. Effect of model size

We design two additional models based on VGGnet, called DEEP and WIDE to explore the effect of model size. The model used in Section 6.2 is named BASE to distinguish it from the models used here. The BASE model has six convolutional layers. The WIDE

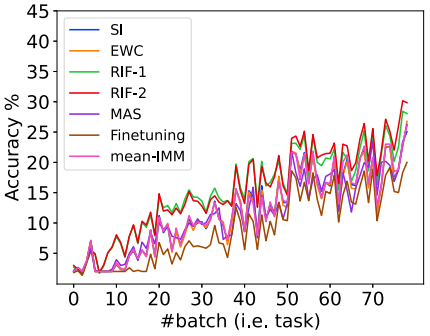
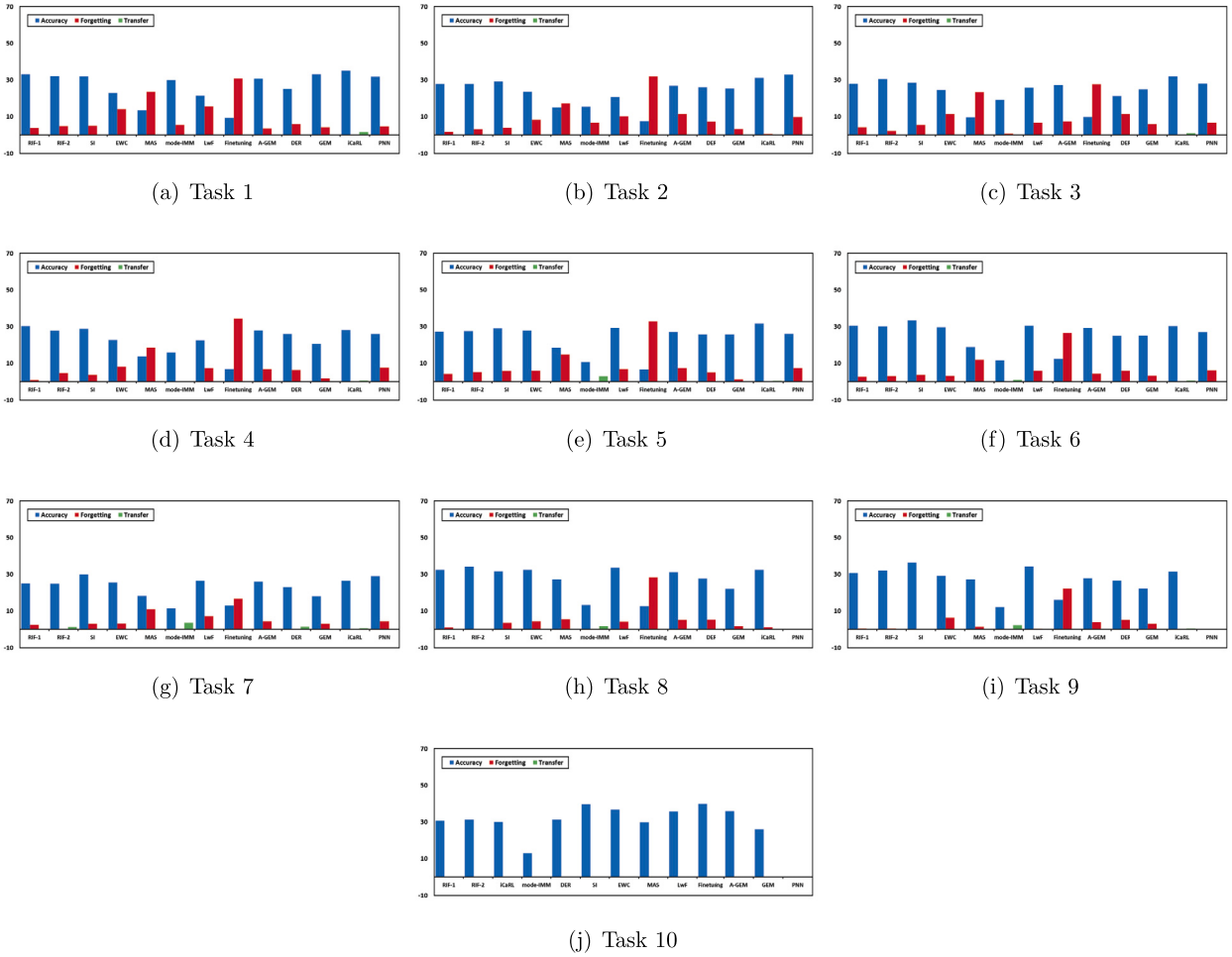


Fig. 8. Representation of the results of baselines on CORe50 in Class-IRL. There are 10 classes in the 0–th batch, and there are 5 classes in each subsequent phase.



(a) Task 1 (b) Task 2 (c) Task 3

(d) Task 4 (e) Task 5 (f) Task 6

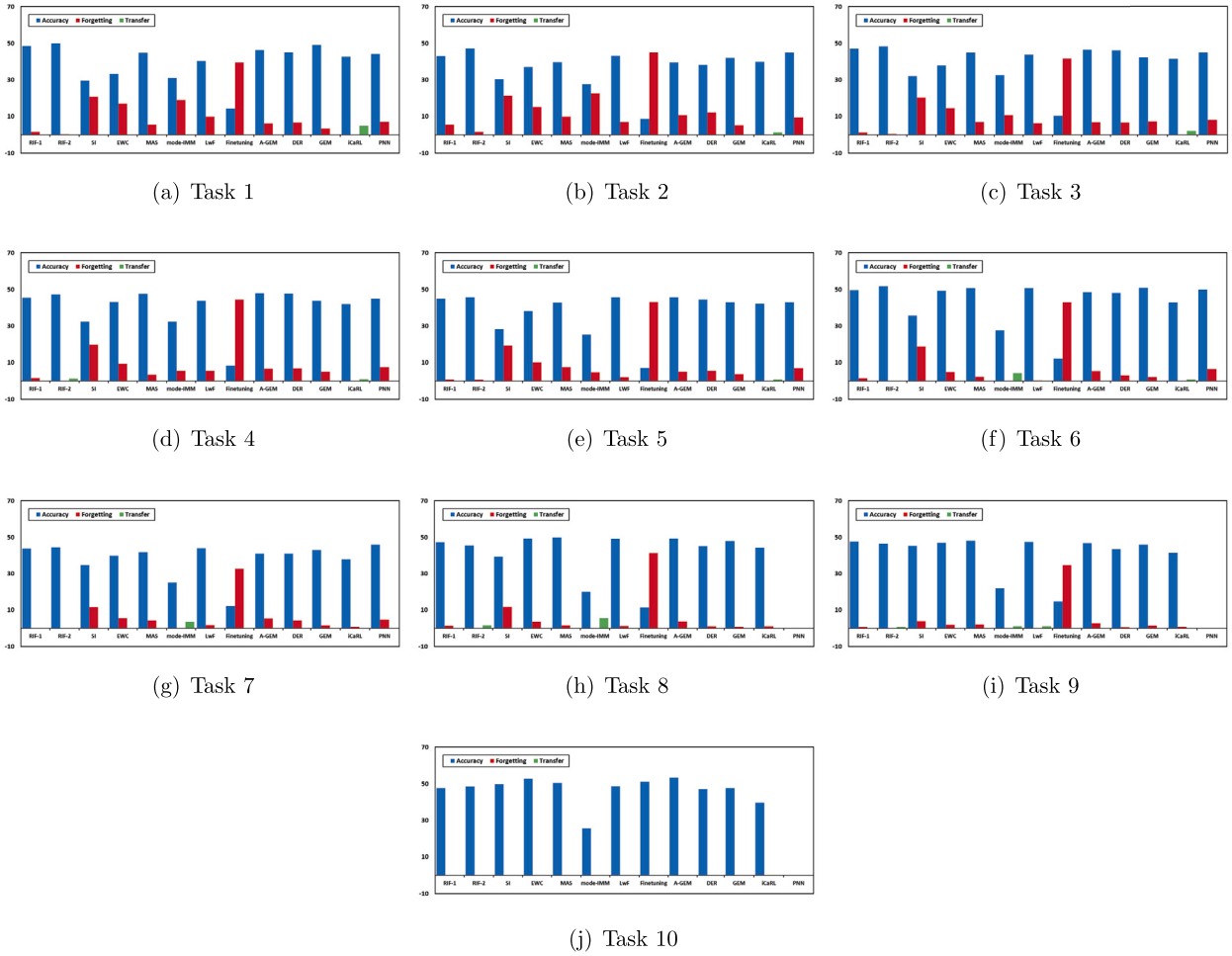
(g) Task 7 (h) Task 8 (i) Task 9

(j) Task 10

Fig. 9. Accuracy and forgetting rates for each task after training all tasks via deeper VGG net on Tiny Imagenet. The bars in blue represent the accuracies for each task. The bars in red represent the positive forgetting rates for each task. The bars in green represent the negative forgetting rates for each task, which indicates the knowledge transfer.

model has more units per layer (i.e., is wider), while the DEEP model has more layers. More details can be found in Appendix A. Here, we concentrate on the experiments for the DEEP and WIDE models.

All results for the DEEP model are presented in Fig. 9. We find that extending the BASE model with additional convolutional layers results in poorer performance. Additional layers may introduce extra unnecessary layers of abstraction, which results in overfitting for the DEEP model. RIF can also achieve great performance on the DEEP model. We consider a negative forgetting rate as knowledge transfer, which is reverse to forgetting rate and calculated as $acc_{N,j} - acc_{j,j}$. Overall, iCaRL seems to achieve the best performance with high accuracy and a little bit knowledge transfer, especially from task 1 to task 5. The possible reason is that the accuracy of



(a) Task 1 (b) Task 2 (c) Task 3

(d) Task 4 (e) Task 5 (f) Task 6

(g) Task 7 (h) Task 8 (i) Task 9

(j) Task 10

Fig. 10. Accuracy and forgetting rates for each task after training all tasks via wider VGG net on Tiny Imagenet. The bars in blue represent the accuracies for each task. The bars in red represent the positive forgetting rates for each task. The bars in green represent the negative forgetting rates for each task, which indicates the knowledge transfer.

iCaRL begins at a low value and increases as new tasks are added. RIF also demonstrates a remarkable advantage compared with other approaches from task 1 to task 5. From task 6 to task 10, RIF still achieves a competitive result close to that of iCaRL. RIF tends to remember knowledge of a larger number of past tasks. As more tasks are added during training, RIF can maintain its performance on previous tasks well. These results indicate that RIF is stable enough to remember the previous tasks.

All results for the WIDE model can be seen in Fig. 10. The WIDE model obtains significantly better results than the DEEP model. The results for the WIDE model resemble those of the BASE model. RIF outperforms other baselines, as it has the best accuracy and very small or even negative forgetting rates. iCaRL still achieves a little bit knowledge transfer, but the final accuracy for each task of iCaRL is smaller than that of RIF. We can further observe that LwF also performs well from task 6 to task 10. It plays the same role as SI, which performs well for the DEEP model. Under these circumstances, RIF still maintains good performance on previous tasks as more tasks are added in training, which is consistent with its performance on the DEEP model. We can also observe similar trends for the BASE model. This shows that RIF can effectively remember the previous tasks on Tiny Imagenet.

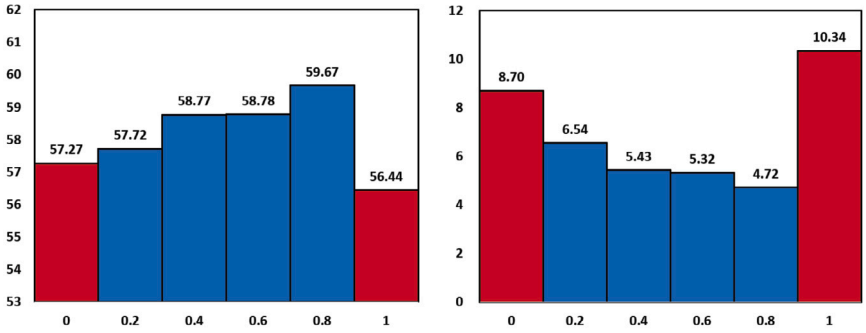
Speaking generally, RIF can outperform other methods and achieve stable performance on previous tasks, regardless of whether the model is wide or deep.

6.4.2. Effect of task order

This experiment examines how changing the task order affects the performance of the various approaches. All experiments are conducted on Tiny Imagenet with BASE model. The previously discussed results on Tiny Imagenet concern a random ordering. We change the task order based on task difficulty. By measuring the accuracy of each task over the three models (BASE, DEEP and WIDE), we can obtain the difficulty of each task. The random order discussed previously can be thought of as [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. The task difficulty results in an ordering from hard to easy: [5, 7, 10, 2, 9, 8, 6, 4, 3, 1]. The inverse order represents an ordering from easy to hard.

Table 10
Average accuracy (forgetting rate) of all methods with different task orders.

	SI	EWC	RIF-1	RIF-2	MAS	IMM_mode	LwF
random	37.93(13.89)	33.90(18.31)	45.86(1.97)	46.54(1.37)	46.30(1.88)	31.67(2.70)	42.16(3.09)
easy-to-hard	33.34(14.30)	31.20(18.70)	45.25(3.25)	44.48(3.90)	44.50(5.27)	32.50(2.60)	42.28(5.99)
hard-to-easy	40.83(3.91)	40.69(8.81)	42.56(4.18)	43.17(3.41)	35.31(10.12)	28.30(5.17)	40.33(3.13)
	finetuning	A-GEM	DER	GEM	iCaRL	PNN	
random	16.06(39.63)	45.88(3.60)	43.88(5.52)	43.77(2.70)	41.06(-1.03)	43.83(9.61)	
easy-to-hard	17.04(39.05)	43.02(7.01)	41.87(6.62)	41.51(5.79)	41.96(1.91)	43.12(9.11)	
hard-to-easy	16.53(35.09)	31.00(16.13)	30.93(15.97)	32.58(10.81)	30.66(6.76)	30.57(22.67)	



(a) Accuracy rate (b) Forgetting rate

Fig. 11. Results of mixing RIF importance weights on Split Cifar100.

The theory of curriculum learning [3] assumes that knowledge is better capture by starting with general and easier tasks before progressing to harder and more specific tasks, while the experimental results of this paper exhibit order-agnostic behavior. Final average accuracies and forgetting rates are listed in Table 10. RIF still achieves almost the best performance on both two orders. RIF, MAS, mode-IMM, LwF, A-GEM, DER, GEM, iCaRL and PNN follow our assumption that performances under easy-to-hard ordering are better than those for hard-to-easy ordering. However, an interesting result emerging from our research is that SI and EWC show unexpectedly better results for hard-to-easy ordering than for easy-to-hard ordering.

6.4.3. Mixed first-order and second-order influence

This subsection studies the following question: whether the simultaneous usage of the parameter importance with different orders can improve performance. To determine the answer to this question, we mix first-order influence and second-order influence via $\mu IF^{(1)} + (1-\mu)IF^{(2)}$ and conduct experiments on Split Cifar100. Under different values of μ, Fig. 11(a) plots the average accuracy while Fig. 11(b) shows the average forgetting rate. $\mu = 0$ and $\mu = 1$ represent second-order and first-order RIF respectively. It is obvious that a mixture of first- and second-order influence can improve accuracy and aid the model in remembering the knowledge of previous tasks.

6.4.4. Time cost and memory usage

Here we consider the time cost and memory usage of each baselines on the four datasets in Task-IL.

We count the time cost of all experiments on all four datasets in Task-IL and draw a box-plot representing the results (see Fig. 12). RIF with first-order influence enjoys the lowest time cost among all baselines except for Finetuning. Approximating second-order influence requires the first-order influence to be calculated first. Therefore, RIF with second-order influence requires a higher time cost. The time cost of A-GEM, DER, GEM, iCaRL and PNN is shown in Fig. 12. Generally speaking, these five baselines spend more time than other baselines due to the extra usage of memory and computational cost. The remained baselines like SI, EWC, MAS, mode-IMM, mean-IMM and LwF achieve similar time cost.

The memory of our GPU is 12 GB where available memory is 11019 MB. The memory usage of baselines can be found in Table 11. All results of regularization-based baselines seem to be close to each other. Similar to the situation in time cost, A-GEM, DER, GEM, iCaRL and PNN spend more memory cost. Due to the needs of computing kernel matrix and selecting the memorable past examples, FROMP requires too much memory. The memory of our GPU is unable to meet the needs of FROMP on Tiny Imagenet and Inaturalist. PNN will connect more old layers with training models on more tasks. Therefore, PNN only trains models on 8 tasks in Tiny Imagenet, and we set the memory cost of PNN in Tiny Imagenet as the max available memory (11019 MB).

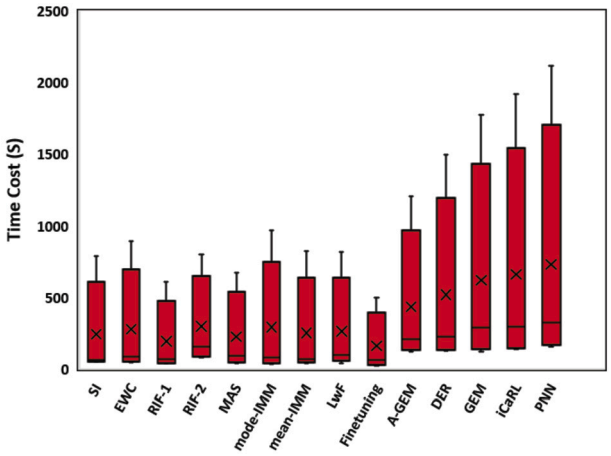


Fig. 12. The time cost of baselines.

Table 11
Memory usage of baselines (MB).

	Split MNIST	Split Cifar100	Tiny imagenet	Inaturalist
SI	1025	1519	3853	4869
EWC	1023	1511	2247	5077
RIF-1	1023	1511	2331	4983
RIF-2	1023	1511	2331	5015
MAS	1023	1511	2247	4863
mode-IMM	1023	1499	2223	5161
mean-IMM	1023	1499	2223	5053
LwF	1025	1509	2283	5233
Finetuning	1019	1491	2213	4649
FROMP	4017	9095		
A-GEM	2023	2812	4325	7152
DER	3102	3726	6230	9213
GEM	3230	3864	6456	9361
iCaRL	3418	3940	6676	9474
PNN	4003	7634	11019	10045

7. Conclusion

In this paper, we propose a novel regularization-based method called RIF for continual learning, which uses the influence function to measure the importance of weights. We provide an efficient method of calculating weight importance with first-order and second-order influence. We also reveal the connection between RIF and classical regularization-based methods, specifically EWC, SI and MAS. Moreover, three main theoretical results are presented in our paper. Our extensive experimental results illustrate that RIF achieves state-of-the-art performance across four benchmarks, and also outperforms other baselines given different model sizes, task orders, and mixed information.

CRediT authorship contribution statement

Rui Gao: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Weiwei Liu:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Weiwei Liu reports a relationship with The School of Computer Science, Wuhan University that includes: employment. Rui Gao reports a relationship with The School of Computer Science, Wuhan University that includes: employment. Rahaf Aljundi (rahaf.aljundi@kuleuven.be; rahaf.aljundi@esat.kuleuven.be) has conflict of interest by co-author - Rui Gao. Francesca Babiloni (francesca.babiloni@esat.kuleuven.be) has conflict of interest by co-author - Rui Gao. Mohamed Elhoseiny (elhoseiny@fb.com) has conflict of interest by co-author - Rui Gao. Marcus Rohrbach (mrf@fb.com) has conflict of interest by co-author - Rui Gao. Tinne Tuytelaars (tinne.tuytelaars@kuleuven.be; tinne.tuytelaars@esat.kuleuven.be) has conflict of interest by co-author - Rui Gao. Manohar Paluri

(mano@fb.com) has conflict of interest by co-author - Rui Gao. Georg Sperl (georg.sperl@ist.ac.at) has conflict of interest by co-author - Rui Gao. Christoph H. Lampert (chl@ist.ac.at) has conflict of interest by co-author - Rui Gao. Alexander Kolesnikov (akolesnikov@google.com) has conflict of interest by co-author - Rui Gao. Sylvestre-Alvise Rebuffi (srebuffi@robots.ox.ac.uk) has conflict of interest by co-author - Rui Gao. Yaoyao Liu (liuyaoyao@tju.edu.cn) has conflict of interest by co-author - Rui Gao. Yuting Su (ytsu@tju.edu.cn) has conflict of interest by co-author - Rui Gao. An-An Liu (liuanan@tju.edu.cn) has conflict of interest by co-author - Weiwei Liu. Qianru Sun (qianrusun@smu.edu.sg) has conflict of interest by co-author - Weiwei Liu. Bernt Schiele (schiele@mpi-inf.mpg.de) has conflict of interest by co-author - Weiwei Liu.

Acknowledgements

This work is supported by the Key R&D Program of Hubei Province under Grant 2024BAB038, National Key R&D Program of China under Grant 2023YFC3604702, and the Fundamental Research Fund Program of LIESMARS.

Appendix A. Additional information about baselines, datasets and models

Here, we provide more information about baselines, datasets and models used in our paper. There are 12 baselines in our paper. We first give some descriptions of these baselines.

SI: The synaptic state tracks the past and current parameter value, and maintains an online estimate of the synapse's importance toward solving problems encountered in the past.

MAS: Redefines the parameter importance measure to an unsupervised setting and obtains gradients of the squared L_2-norm of the learned network output function.

EWC: EWC introduces network parameter uncertainty in the Bayesian framework; the true posterior is estimated using a Laplace approximation with precision, determined by the Fisher Information Matrix (FIM), which shows equivalence to the positive semi-definite second order derivative of the loss near a minimum.

IMM: IMM estimates Gaussian posteriors for task parameters, in the same vein as EWC, but inherently differs in its use of model merging. In the merging step, the mixture of Gaussian posteriors is approximated by a single Gaussian distribution and corresponding covariances. Two methods for the merge step are proposed: mean-IMM and mode-IMM.

FROMP: FROMP is a functional-regularization approach that utilizes a few memorable past examples crucial to avoiding forgetting. It enables training while identifying both the memorable past and a functional prior by using a Gaussian Process formulation of deep networks.

LwF: Lwf uses only examples for the new task but optimizes both for high accuracy for the new task and for preservation of responses on existing tasks from the original network.

iCaRL: iCaRL is the first replay method, focused on learning in a class-incremental way. Assuming fixed allocated memory, it selects and stores samples (exemplars) closest to the feature mean of each class.

DER: DER mixes rehearsal with knowledge distillation and regularization and matches the network's logits sampled throughout the optimization trajectory to promot consistency with its past.

GEM: GEM exploits exemplars to solve a constrained optimization problem, projecting the current task gradient in a feasible area, outlined by the previous task gradients.

A-GEM: A-GEM is an improved version of GEM. It relaxes the problem to project on one direction estimated by randomly selected samples from a previous task data buffer.

PNN: PNN dynamically changes the network structure: instantiating a new neural network (a column) for each task being solved to prevent catastrophic forgetting, and making lateral connections to features of previously learned columns to enable transfer.

CUBER characterizes the task correlation to identify the positively correlated old tasks in a layer-wise manner, and then selectively modifies the learnt model of the old tasks when learning the new task.

S-FSVI frames continual learning as sequential function-space variational inference and adapts the variational objective to the continual-learning setting.

Table A.12
Details of datasets.

	Tasks	Classes/task	Train data/task	Task selection
Split MNIST	5	2	9365–10610	consecutive class
Split Cifar100	10	10	5000	consecutive class
Tiny Imagenet	10	20	8000	random class
iNaturalist	8	5–150	645–26059	supercategory
Cifar100 in Class-IL	5	$\{50, 5\}$ or $\{50, 10\}$	$\{25000, 2500\}$ or $\{25000, 5000\}$	random class
ImageNet	5	$\{500, 50\}$ or $\{500, 100\}$	$\{650000, 65000\}$ or $\{650000, 130000\}$	random class
CORe50	79	$\{10, 5\}$	$\{3000, 1500\}$	random class

Table A.13
VGG Models used for Tiny Imagenet.

	Feature Extractor								Classifier	
BASE	64	M	64	M	128*2	M	256*2	M	512*2	multi-head
WIDE	64	M	128	M	256*2	M	512*2	M	512*2	multi-head
DEEP	64	M	64*6	M	128*6	M	256*6	M	512*2	multi-head

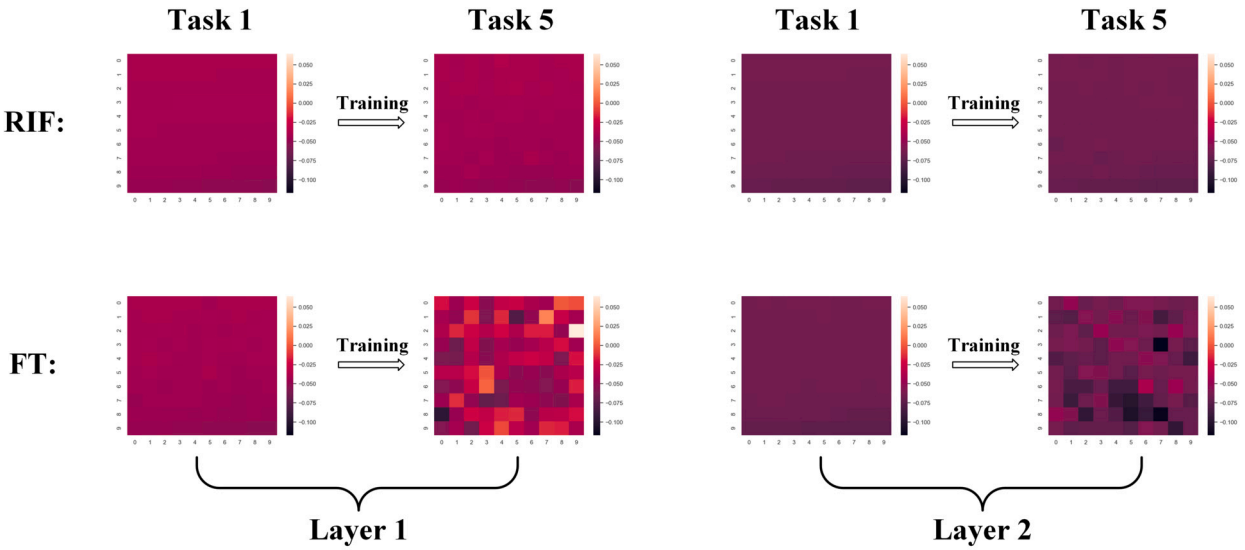


Fig. B.13. Visualization of the 100 most important parameters' changes on Split MNIST. FT represents the baseline Finetuning. The x-axis and y-axis of each sub-figure respectively represent the index. The scales of x-axis and y-axis in each sub-figure both range from 0 to 9.

Finetuning: Finetuning greedily trains each task without considering previous task performance—hence introducing catastrophic forgetting—and represents the minimum desired performance.

There are 7 benchmarks in our work, we divide them into several tasks which can be found in Table A.12. In further analysis, we construct two special models which are WIDE and DEEP models in order to figure out the effect of model size. The whole structures of used models are presented in Table A.13.

Appendix B. Visualization

To better understand the performance of RIF, we visualize the results of RIF and Finetuning on Split MNIST. For task 1, we select the 100 parameters with the largest influence and 100 parameters with the smallest influence in each layer, where the influence is calculated by RIF. The top and bottom of Fig. B.13 illustrate the changes in selected parameters after training by RIF and Finetuning, respectively. The parameters selected here are the 100 parameters with the largest influence. It is evident that these parameters undergo minimal changes under the RIF method while exhibiting significant changes under Finetuning. Combined with the fact that the model achieves the best accuracy on task 1 of Split MNIST using RIF, we can conclude that the important parameters considered by RIF can significantly affect the model performance. A similar observation can be made from Fig. B.14: specifically, that the parameters in blue change very little under the RIF method while changing significantly under Finetuning. Fig. B.14 also shows that the parameters considered by RIF to be irrelevant change freely. According to Tables 1 and 2, the model also achieves excellent performance on other tasks. These experimental results indicate that RIF accurately releases irrelevant parameters to help the model achieve great performance on the following tasks.

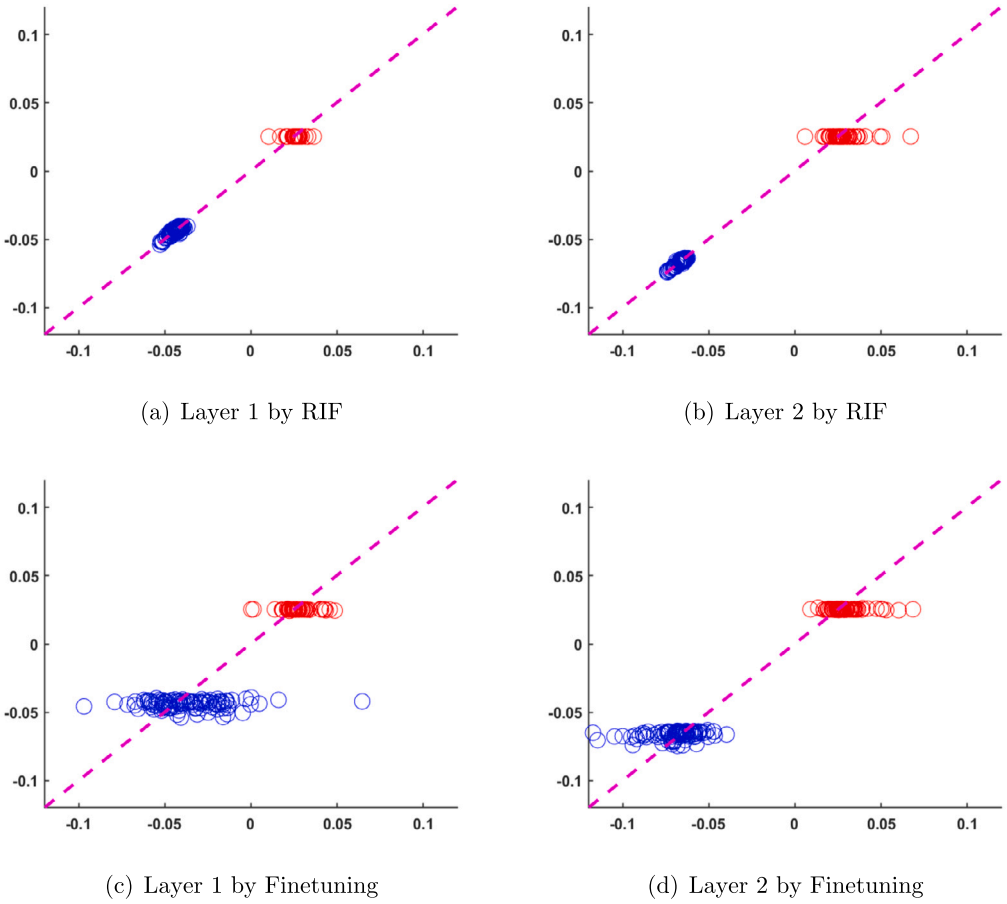
Appendix C. Proof of Theorem 1

In the following Theorem, we provide the asymptotic properties of a regularization-based method's estimator.

Theorem 1. *The estimator of a regularization-based method for task t can be represented as:*

$$T_\lambda(D^t_{m_t}) = \underset{\theta}{\operatorname{argmin}} \left\{ L_t(\theta; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1} (\theta_j - \hat{\theta}_j^{t-1})^2 \right\}. \tag{C.1}$$

We define $T(D^t) = \operatorname{argmin}_\theta \left\{ \int l_t(\theta;\ x) d\, D^t(x) \right\}$. The estimator $T_\lambda(D^t_{m_t})$ is then asymptotically normal. That is:



(a) Layer 1 by RIF (b) Layer 2 by RIF

(c) Layer 1 by Finetuning (d) Layer 2 by Finetuning

Fig. B.14. Visualization of changes in model parameters. The points in blue are the 100 most important parameters, while those in red are the 100 least important parameters. The parameters that are closer to the dashed line have fewer changes. In each sub-figure, the x-axis represents the parameter values before training model on a new task, and y-axis represents the parameter values after training model on a new task. The scales of x-axis and y-axis in each sub-figure both range from -0.1 to 0.1.

$$\sqrt{m_t}\left(T_\lambda(D^t_{m_t}) - T(D^t)\right) \to N(\sqrt{m_t}C_{m_t}\lambda, \sigma^2) \tag{C.2}$$

where $C_{m_t} = \left.\frac{dT_\lambda(D^t_{m_t})}{d\lambda}\right|_{\lambda=0}$ *and* $\sigma^2 = \mathbb{E}{\Phi_{D^t}}^2(X) = \int {\Phi_{D^t}}^2(x)d\,D^t(x)$. $\Phi_{D^t}(x)$ *is the influence function of* $T(\cdot)$ *at* D^t.

Proof. We use the fact that the function $A(\zeta) = T(D^t + \zeta(D^t_{m_t} - D^t))$ can be represented by a Taylor expansion at $\zeta = 0$,

$$A(\zeta) = A(0) + A'(0)\zeta + \ldots + \frac{A^{(k)}(0)t^k}{k!} + \mathbf{Rem_k} \tag{C.3}$$

where $\mathbf{Rem_k}$ is a remainder term and $\zeta \in [0,1]$. According to [14], when $\zeta = 1$, this corresponds to an expansion for D^t in the first order, this is given by the following:

$$\begin{aligned} T(D^t_{m_t}) &= T(D^t) + T'_{D^t}(D^t_{m_t} - D^t) + \mathbf{Rem}(D^t_{m_t} - D^t) \\ &= T(D^t) + \int \Phi_{D^t}(x)d(D^t_{m_t} - D^t)(x) + \mathbf{Rem}(D^t_{m_t} - D^t) \end{aligned} \tag{C.4}$$

where $T'_{D^t}(D^t_{m_t} - D^t)$ is the von Mises derivative of $T(\cdot)$ at D^t and $\Phi_{D^t}(x)$ is the influence function of $T(\cdot)$ at D^t. These are defined as follows:

$$\begin{aligned} T'_{D^t}(D^t_{m_t} - D^t) &= \frac{d}{d\zeta}T(D^t + \zeta(D^t_{m_t} - D^t))\,|_{\zeta=0} \\ &= \int \Phi_{D^t}(x)d(D^t_{m_t} - D^t)(x) \end{aligned} \tag{C.5}$$

$$\Phi_{D^t}(x) = \frac{d}{d\zeta} T(D^t + \zeta(\delta_x^t - D^t)) \mid_{\zeta=0} \tag{C.6}$$

where δ_x^t is the d.f. of the point mass one at x. Since $\int \Phi_{D^t}(x) d(D^t)(x) = 0$, we obtain that:

$$T(D_{m_t}^t) = T(D^t) + \int \Phi_{D^t}(x) d\, D_{m_t}^t(x) + \mathbf{Rem}(D_{m_t}^t - D^t) \tag{C.7}$$

which can be thought of as a von Mises expansion of $T(\cdot)$ at D_t. The linear term of the expansion is as follows:

$$\int \Phi_{D^t}(x) d\, D_{m_t}^t(x) = \frac{1}{m_t} \sum_{i=1}^{m_t} \Phi_{D^t}(X_i) \tag{C.8}$$

and therefore:

$$T(D_{m_t}^t) - T(D^t) = \frac{1}{m_t} \sum_{i=1}^{m_t} \Phi_{D^t}(X_i) + \mathbf{Rem}(D_{m_t}^t - D^t) \tag{C.9}$$

Since $T_\lambda(D_{m_t}^t) \to T(D_{m_t}^t)$ as $\lambda \to 0$, we perform a Taylor expansion of $T_\lambda(D_{m_t}^t)$ at $\lambda = 0$

$$T_\lambda(D_{m_t}^t) = T(D_{m_t}^t) + \frac{d T_\lambda(D_{m_t}^t)}{d\lambda}\bigg|_{\lambda=0} \lambda \tag{C.10}$$

where we drop the remainder term. Combining (C.9) with (C.10), we obtain

$$\begin{aligned} &\sqrt{m_t}\left(T_\lambda(D_{m_t}^t) - T(D^t)\right) \\ &= \frac{1}{\sqrt{m_t}} \sum_{i=1}^{m_t} \left(\Phi_{D^t}(X_i) + C_{m_t}\lambda\right) + \sqrt{m_t}\mathbf{Rem}(D_{m_t}^t - D^t) \end{aligned} \tag{C.11}$$

where $C_{m_t} = \frac{d T_\lambda(D_{m_t}^t)}{d\lambda}\Big|_{\lambda=0}$. The first term on the right-hand side is asymptotically normal according to the central limit theorem. According to [15], in most cases the remainder becomes negligible for $n \to \infty$, so that $T_\lambda(D_{m_t}^t)$ itself is asymptotically normal. That is:

$$\sqrt{m_t}\left(T_\lambda(D_{m_t}^t) - T(D^t)\right) \to N(\sqrt{m_t}C_{m_t}\lambda, \sigma^2) \tag{C.12}$$

where $C_{m_t} = \frac{d T_\lambda(D_{m_t}^t)}{d\lambda}\Big|_{\lambda=0}$ and the asymptotic variance equals:

$$\sigma^2 = \mathbb{E}{\Phi_{D^t}}^2(X) = \int {\Phi_{D^t}}^2(x) d\, D^t(x) \quad \square \tag{C.13}$$

Appendix D. Proof of Theorem 2

Here we want to prove that the learning algorithm of a regularization-based method can declare success under the definition of agnostic PAC learning. We first give a lemma and two corollaries. We also present the related definitions in the main paper.

Definition 1 *(Agnostic PAC learnability).* A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\zeta, \delta \in (0,1)$ and for every distribution D over $X \times Y$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\zeta, \delta)$ i.i.d. examples generated by D, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_D(h) \leq \min_{h' \in \mathcal{H}} L_D\left(h'\right) + \zeta \tag{D.1}$$

where $L_D(h) \overset{\text{def}}{=} \mathbb{E}_{z \sim D}[l(h; z)]$. $l(h; z)$ is the loss function.

Definition 2 *(ζ-representative sample).* A training set S is called ζ-representative (w.r.t. domain Z, hypothesis class H, loss function l, and distribution D) if

$$\forall h \in H, \left|L_S(h) - L_D(h)\right| \leq \zeta \tag{D.2}$$

where $L_D(h) \overset{\text{def}}{=} \mathbb{E}_{z \sim D}[l(h; z)]$ and $L_S(h) \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} l\left(h; z_i\right)$. $l(h; z)$ is the loss function and z_i is randomly drawn from D.

Definition 3 *(Uniform convergence).* We say that a hypothesis class H has the uniform convergence property (w.r.t. a domain Z and a loss function l) if there exists a function $m_{\mathcal{H}}^{\mathrm{UC}} : (0,1)^2 \to \mathbb{N}$ such that for every $\delta \in$(0,1) and for every probability distribution D over Z, if S is a sample of $m \geq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta, \delta)$ examples drawn i.i.d. according to D, then, with probability of at least $1-\delta$, S is ζ-representative.

Lemma 1. *Assume that a training set S is $\frac{\zeta}{2}$-representative (w.r.t. domain Z, hypothesis class $\mathcal{H}$, loss function l, and distribution D). Then, any output of $\mathbf{ERM}_{\mathcal{H}}(S)$, namely, any $h_S \in \operatorname{argmin}_{h\in\mathcal{H}} L_S(h)$, satisfies*

$$L_D(h_S) \leq \min_{h\in\mathcal{H}} L_D(h) + \zeta \tag{D.3}$$

Proof. For every $h \in \mathcal{H}$,

$$\begin{aligned} L_D\left(h_S\right) &\leq L_S\left(h_S\right) + \frac{\zeta}{2} \leq L_S(h) + \frac{\zeta}{2} \\ &\leq L_D(h) + \frac{\zeta}{2} + \frac{\zeta}{2} = L_D(h) + \zeta, \end{aligned} \tag{D.4}$$

where the first and third inequalities are due to the assumption that S is $\frac{\zeta}{2}$-representative (Definition 2), while the second inequality holds since h_S is an **ERM** predictor. □

The below corollary follows directly from Lemma 1 and the definition of uniform convergence.

Corollary 1. *If a class $\mathcal{H}$ has the uniform convergence property with a function $m_{\mathcal{H}}^{\mathrm{UC}}$, then the class is agnostically PAC learnable with the sample complexity $m_{\mathcal{H}}(\zeta,\delta) \leq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta/2,\delta)$. Furthermore, in that case, the $\mathbf{ERM}_{\mathcal{H}}$ paradigm is a successful agnostic PAC algorithm for $\mathcal{H}$.*

Corollary 2. *Let $\mathcal{H}$ be a finite hypothesis class, let Z be a domain, and let $l : \mathcal{H}\times Z \to [0,1]$ be a loss function. Then, $\mathcal{H}$ enjoys the uniform convergence property with sample complexity*

$$m_{\mathcal{H}}^{UC}(\zeta,\delta) \leq \left\lceil \frac{\log(2|\mathcal{H}|/\delta)}{2\zeta^2} \right\rceil. \tag{D.5}$$

Proof. Details can be found in [37]. □

We now establish Theorem 2.

Theorem 2. *We denote the pointwise multiplication operation using $\circ$. $\mathcal{H}^t$ is a finite hypothesis class for task τ_t. Z^t is a domain. D^t is the distribution over Z^t. $l_t : \mathcal{H}^t \times Z^t \to [0,1]$ is the loss function for task τ_t. Assume that $\mathcal{H}^t$ has the uniform convergence property (with sample complexity $m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta)$). Let $h_\lambda^t = \hat{\theta}_\lambda^t \in \mathcal{H}^t$ be the hypothesis that acts as the minimizer of $\hat{\theta}_\lambda^t = \operatorname{argmin}_\theta \left\{ L_t(\theta; S^t) + \lambda \sum_{j=1}^n \Omega_j^{t-1}(\theta_j - \hat{\theta}_j^{t-1})^2 \right\}$. A_λ^t is the algorithm that generates h_λ^t. S^t is the sample set of task τ_t with $m_t \geq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta)$ samples drawn independent and identical distributed (i.i.d.) according to D^t. If the condition*

$$\zeta \quad \in (0,1) \tag{D.6}$$

is satisfied, where

$$\zeta = 4\zeta' + 2\nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t) \cdot H_{\hat{\theta}^t}^{-1} \left\{ \boldsymbol{\Omega}^{\boldsymbol{t-1}} \circ (\hat{\boldsymbol{\theta}}^{\boldsymbol{t-1}} - \hat{\boldsymbol{\theta}}^{\boldsymbol{t}}) \right\}^T \lambda, \tag{D.7}$$

then A_λ^t is a successful agnostic PAC learning algorithm with sample complexity $m_{\mathcal{H}}(\zeta,\delta) \leq m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta) \leq \left\lceil \frac{\log(2|\mathcal{H}^t|/\delta)}{2\zeta'^2} \right\rceil$. $H_{\hat{\theta}^t}$ is the Hessian matrix.

Proof. Assume $h^t = \hat{\theta}^t \in \mathcal{H}^t$ is the output of the $\mathbf{ERM}_{\mathcal{H}^t}$ paradigm, i.e.

$$\hat{\theta}^t = \operatorname*{argmin}_\theta L_t(\theta; S^t) \tag{D.8}$$

Since $\mathcal{H}^t$ has the uniform convergence property (with sample complexity $m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta)$), we can determine that $\mathcal{H}^t$ is agnostic PAC learnable and hold that

$$L_{D^t}(h^t) \leq \min_{h\in\mathcal{H}} L_{D^t}(h) + 2\zeta' \tag{D.9}$$

$$\left| L_{S^t}(h^t) - L_{D^t}(h^t) \right| \leq \zeta' \tag{D.10}$$

Combining (D.9) and (D.10), we have

$$L_{S^t}(h^t) \le \min_{h\in\mathcal{H}} L_{D^t}(h) + 3\zeta' \tag{D.11}$$

The left side of (D.11) can be rewritten as follows:

$$L_{S^t}(h^t) = L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) = \frac{1}{m^t}\sum_{i=1}^{m^t} l_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; z_i^t) \tag{D.12}$$

where $z_i^t \in S^t$. We already have:

$$h_\lambda^t = \hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}} = \underset{\theta}{\operatorname{argmin}} \left\{ L_t(\boldsymbol{\theta}; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1}(\theta_j - \hat{\theta}_j^{t-1})^2 \right\} \tag{D.13}$$

According to Definition 2 and 3, since $\mathcal{H}^t$ has the uniform convergence property (with sample complexity $m_{\mathcal{H}}^{\mathrm{UC}}(\zeta', \delta)$), we have:

$$\left| L_{S^t}(h_\lambda^t) - L_{D^t}(h_\lambda^t) \right| \le \zeta'. \tag{D.14}$$

Then, we have:

$$L_{D^t}(h_\lambda^t) - \zeta' \le L_{S^t}(h_\lambda^t). \tag{D.15}$$

Next, since $h_\lambda^t \to h^t$ as $\lambda \to 0$, we have $L_{S^t}(h_\lambda^t) \to L_{S^t}(h^t)$ as $\lambda \to 0$. Therefore, we perform a Taylor expansion of $L_{S^t}(h_\lambda^t)$ at $\lambda = 0$:

$$\begin{aligned} L_{S^t}(h_\lambda^t) &\approx L_{S^t}(h^t) + \left.\frac{d L_{S^t}(h_\lambda^t)}{d\lambda}\right|_{\lambda=0} \lambda \\ &= L_{S^t}(h^t) + \nabla_\theta L_t(\hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}}; S^t) \left.\frac{d\hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}}}{d\lambda}\right|_{\lambda=0} \cdot \lambda \end{aligned} \tag{D.16}$$

where we have dropped the remainder term. We define the parameter change $\Delta_\lambda = \hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}} - \hat{\boldsymbol{\theta}}^{\boldsymbol{t}}$, then note that $\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}$ does not depend on λ; thus, we have:

$$\frac{d\hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}}}{d\lambda} = \frac{d\Delta_\lambda}{d\lambda} \tag{D.17}$$

Since $\hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}}$ is the minimizer of (D.13), it must satisfy the following (first order) optimality condition:

$$\nabla_\theta \{ L_t(\boldsymbol{\theta}; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1}(\theta_j - \hat{\theta}_j^{t-1})^2 \} \Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_\lambda^{\boldsymbol{t}}} = 0 \tag{D.18}$$

Since $h_\lambda^t \to h^t$ as $\lambda \to 0$, we can thus write the following Taylor expansion:

$$\nabla_\theta \sum_{k=0}^{\infty} \nabla_\theta^k \left\{ L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1}(\hat{\theta}_j^t - \hat{\theta}_j^{t-1})^2 \right\} \cdot \frac{\Delta_\lambda^k}{k!} = 0 \tag{D.19}$$

Now, by dropping the $o\left(\|\Delta_\lambda\|\right)$ terms, we have the following:

$$\begin{aligned} &\nabla_\theta \left\{ L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1}(\hat{\theta}_j^t - \hat{\theta}_j^{t-1})^2 \right\} + \\ &\nabla_\theta \left\{ \nabla_\theta \left\{ L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1}(\hat{\theta}_j^t - \hat{\theta}_j^{t-1})^2 \right\} \cdot \Delta_\lambda \right\} = 0 \end{aligned} \tag{D.20}$$

Since $\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}$ is a minimizer of (D.8), we have $\nabla_\theta L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) = 0$. Thus, (D.20) reduces to the following condition:

$$\left\{ 2\lambda \boldsymbol{\Omega}^{\boldsymbol{t-1}} \circ (\hat{\boldsymbol{\theta}}^{\boldsymbol{t}} - \hat{\boldsymbol{\theta}}^{\boldsymbol{t-1}}) \right\}^T + \nabla_\theta^2 L_t(\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}; S^t) \cdot \Delta_\lambda + 2\lambda \boldsymbol{Q} \cdot \Delta_\lambda = 0 \tag{D.21}$$

where $\boldsymbol{Q} = diag(\Omega_1^{t-1}, \cdots, \Omega_n^{t-1})$ and $\circ$ is the pointwise multiplication. By solving for ∇_θ, we obtain:

$$\Delta_\lambda = 2\lambda \left\{ H_{\hat{\boldsymbol{\theta}}^t} + 2\lambda \boldsymbol{Q} \right\}^{-1} \cdot \left\{ \boldsymbol{\Omega}^{\boldsymbol{t-1}} \circ (\hat{\boldsymbol{\theta}}^{\boldsymbol{t-1}} - \hat{\boldsymbol{\theta}}^{\boldsymbol{t}}) \right\}^T \tag{D.22}$$

which can be approximated by keeping only the $O(\lambda)$ terms as follows:

$$\Delta_\lambda = 2\lambda H_{\hat{\boldsymbol{\theta}}^t}^{-1} \cdot \left\{ \boldsymbol{\Omega}^{\boldsymbol{t-1}} \circ (\hat{\boldsymbol{\theta}}^{\boldsymbol{t-1}} - \hat{\boldsymbol{\theta}}^{\boldsymbol{t}}) \right\}^T \tag{D.23}$$

By combining (D.11), (D.16), (D.17) and (D.22), we have:

$$L_{S^t}(h_\lambda^t) \le \min_{h\in\mathcal{H}} L_{D^t}(h) + 3\zeta' + 2\nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t)\cdot H_{\hat{\theta}^t}^{-1}\left\{\mathbf{\Omega^{t-1}}\circ(\hat{\theta}^{t-1}-\hat{\theta}^t)\right\}^T\lambda \tag{D.24}$$

Combined with (D.15), we have

$$L_{D^t}(h_\lambda^t) \le \min_{h\in\mathcal{H}} L_{D^t}(h) + 4\zeta' + 2\nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t)\cdot H_{\hat{\theta}^t}^{-1}\left\{\mathbf{\Omega^{t-1}}\circ(\hat{\theta}^{t-1}-\hat{\theta}^t)\right\}^T\lambda \tag{D.25}$$

Letting

$$\zeta = 4\zeta' + 2\nabla_\theta L_t(\hat{\theta}_\lambda^t; S^t)\cdot H_{\hat{\theta}^t}^{-1}\left\{\mathbf{\Omega^{t-1}}\circ(\hat{\theta}^{t-1}-\hat{\theta}^t)\right\}^T\lambda, \tag{D.26}$$

if $\zeta\in(0,1)$, we conclude that A_λ^t is a successful agnostic PAC learning algorithm with sample complexity $m_{\mathcal{H}}(\zeta,\delta)$ for $\mathcal{H}^t$. As the derivation is based on S^t, we have $m_{\mathcal{H}}(\zeta,\delta)\le|S^t|=m_t$. The value of m_t is at least $m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta)$. Therefore, we have $m_{\mathcal{H}}(\zeta,\delta)\le m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta)$. According to Corollary 2, we obtain:

$$m_{\mathcal{H}}(\zeta,\delta)\le m_{\mathcal{H}}^{\mathrm{UC}}(\zeta',\delta)\le\left\lceil\frac{\log(2|\mathcal{H}^t|/\delta)}{2\zeta'^2}\right\rceil. \quad \square \tag{D.27}$$

Appendix E. Proof of Theorem 3

In the following Theorem, we establish the conditions under which σ^2 can achieve the Cramér-Rao Lower Bound.

Theorem 3. *Considering the variance σ^2 in Theorem 1, we prove that it can be bounded by:*

$$\sigma^2\ge\frac{\left(\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}\right)^2}{J\left(D^t\right)}. \tag{E.1}$$

which is the Cramér-Rao Lower Bound. σ^2 can achieve the Cramér-Rao Lower Bound, if

$$\Phi_{D^t}(x)=\frac{A}{J\left(D^t\right)}B \tag{E.2}$$

where $A=\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}$ and $B=\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}$, $f_\gamma(x)$ is the density of some sample distribution D_γ.

Proof. We provide an overview of the conditions under which σ^2 can theoretically achieve the Cramér-Rao Lower Bound. More details can be seen in [15]. Assuming that every distribution D can be parameterized by γ, we can denote D as D_γ. We denote the density of D_γ by f_γ, and set $D_*:=D_{\gamma_*}$ where γ_* is some fixed member of Γ. The Fisher information at D_* equals

$$J\left(D_*\right)=\int\left(\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}\right)^2 dD_*. \tag{E.3}$$

Recalling the proof of Theorem 1 in Appendix C, we derive that the asymptotic variance equals:

$$\sigma^2=\int{\Phi_{D^t}}^2(x)d\,D^t(x). \tag{E.4}$$

Similar to Eq. (C.7), we have

$$T(D_\gamma)=T(D_*)+\int\Phi_{D_*}(x)d\,D_\gamma(x). \tag{E.5}$$

Differentiating both sides of with respect to γ

$$\frac{\partial}{\partial\gamma}\left[\int\Phi_{D_*}(x)d\,D_\gamma\right]_{\gamma_*}=\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*} \tag{E.6}$$

Assuming that differentiation and interaction are changeable, we change the order of differentiation and interaction, and obtain:

$$\begin{aligned}\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}&=\int\frac{\partial}{\partial\gamma}\left[\Phi_{D_*}(x)d\,D_\gamma\right]_{\gamma_*}\\&=\int_d\Phi_{D_*}(x)\frac{\partial}{\partial\gamma}\left[f_\gamma(x)\right]_{\gamma_*}d\pi(x)\end{aligned} \tag{E.7}$$

where π is a probability measure. Using the fact that:

$$\begin{aligned}\frac{\partial}{\partial\gamma}\left[f_\gamma(x)\right]_{\gamma_*}d\pi(x)&=\frac{\partial}{\partial\gamma}\left[f_\gamma(x)\right]_{\gamma_*}\frac{f_{\gamma_*}(x)}{f_{\gamma_*}(x)}d\pi(x)\\&=\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}dD_*(x).\end{aligned}\tag{E.8}$$

Combining (E.7) and (E.8), we derive that:

$$\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}=\int\Phi_{D_*}(x)\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}dD_*(x).\tag{E.9}$$

Squaring both sides of (E.9), we have:

$$\left(\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}\right)^2=\left(\int\Phi_{D_*}(x)\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}dD_*(x)\right)^2.\tag{E.10}$$

Making use of Cauchy-Schwarz on the right side of (E.10), we derive that:

$$\begin{aligned}\left(\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}\right)^2&=\left(\int\Phi_{D_*}(x)\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}dD_*(x)\right)^2\\&\le\int{\Phi_{D_*}}^2(x)dD_*(x)\int\left(\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}\right)^2dD_*\\&=\int{\Phi_{D_*}}^2(x)dD_*(x)J\left(D_*\right).\end{aligned}\tag{E.11}$$

Then we obtain the Crameŕ-Rao inequality:

$$\int{\Phi_{D_*}}^2(x)dD_*(x)\ge\frac{\left(\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}\right)^2}{J\left(D_*\right)}.\tag{E.12}$$

To achieve the lower bound of (CR-inequality), we let γ_* here be the parameter of distribution D^t. Finally, we obtain the following condition:

$$\Phi_{D^t}(x)=\frac{A}{J\left(D^t\right)}B\tag{E.13}$$

where $A=\frac{\partial}{\partial\gamma}\left[\ln f_\gamma(x)\right]_{\gamma_*}$ and $B=\frac{\partial}{\partial\gamma}\left[T\left(D_\gamma\right)\right]_{\gamma_*}$. □

Appendix F. Important notations

Here we provide more descriptions of important notations in this paper. Details are presented in Table F.14.

Table F.14
The descriptions of important notations.

Notations	Descriptions
$\boldsymbol{\theta}=(\theta_1,\theta_2,...,\theta_n)$	The model parameters vector.
Ω_i	The importance of parameter θ_i with respect to the previous tasks.
λ	The hyperparameter indicating the strength of the penalty.
F	The parameter distribution.
F_n	The empirical parameter distribution of $\boldsymbol{\theta}=(\theta_1,\theta_2,...,\theta_n)$
N_{task}	The number of tasks.
$(\tau_1,\tau_2,...,\tau_{N_{task}})$	N_{task} tasks in continual learning. Task τ_t can also represented as Task t.
(X_t,Y_t)	(X_t,Y_t) be the training dataset for task τ_t, where X_t and Y_t represent the feature and label respectively.
$\hat{\boldsymbol{\theta}}^{\boldsymbol{t}}=(\hat{\theta}^t_1,\hat{\theta}^t_2,...,\hat{\theta}^t_n)$	A local minimum for task τ_t.
θ^*_i	The "old" network parameters.
D^t	The sample distribution for task t.
S^t	The sample set for task t.
m_t	The size of the sample set S^t.
$D^t_{m_t}$	The empirical sample distribution for task t.
L_t	L_t is the loss for task t. $L_t(\theta;S^t)=\frac{1}{m_t}\sum_{i=1}^{m_t}l_t(\theta;x_i)$.
l_t	The loss function for task t.
$T_\lambda(D^t_{m_t})$	The estimator of a regularization-based method for task t.
$T(D^t)$	The expected parameters to be evaluated for task t.
$C_{m_t}=\frac{dT_\lambda(D^t_{m_t})}{d\lambda}\Big\vert_{\lambda=0}$	The component of the bias of the estimator $T_\lambda(D^t_{m_t})$.
σ^2	The asymptotic variance.

Table F.14 (*continued*)

Notations	Descriptions
$T(\cdot)$	The functional whose domain is the set of sample distributions.
$\Phi_{D^t}(x)$	The influence function of $T(\cdot)$ at D^t. It is the typical influence function in [15,16].
$\mathcal{H}$	A hypothesis class.
$m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$	The sample complexity
ζ, δ	ζ represents an upper bound of the difference between the agnostic PAC learning algorithm's error and the best error achievable from the hypothesis class. δ is the probability.
$L_D(h) \overset{\text{def}}{=} \underset{z\sim D}{\mathbb{E}}[l(h;z)]$	The true loss for the returned hypothesis h from a learning algorithm.
$L_S(h) \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} l\left(h; z_i\right)$	The empirical loss.
$m_{\mathcal{H}}^{\text{UC}} : (0,1)^2 \to \mathbb{N}$	The sample complexity in Uniform Convergence.
$\circ$	The pointwise multiplication operation in Theorem 2.
$\mathcal{H}^t$	The finite hypothesis class for task t.
$h_\lambda^t = \hat{\boldsymbol{\theta}}_\lambda^t \in \mathcal{H}^t$	The hypothesis that acts as the minimizer of $\hat{\boldsymbol{\theta}}_\lambda^t = \text{argmin}_\theta \left\{ L_t(\boldsymbol{\theta}; S^t) + \lambda \sum_{j=1}^{n} \Omega_j^{t-1}(\theta_j - \hat{\theta}_j^{t-1})^2 \right\}$.
A_λ^t	The algorithm that generates h_λ^t.
$H_{\hat{\theta}^t}$	The Hessian matrix.
D_γ	The sample distribution which is parameterized by γ
γ_*	γ_* parameterizes the sample distribution D^t for task t.
$J\left(D^t\right)$	The Fisher information at D^t.
M	A model performance function, which is defined on a convex set containing the parameter distribution F and all the empirical distributions F_n of F.
$IF(\theta; M, F)$	The influence function (IF) of M at the parameter distribution F.
Δ_θ	The distribution of the point mass at θ.
ϵ	A small perturbation.
$\hat{\boldsymbol{\theta}}_{\epsilon,\boldsymbol{i}}^t$	The parameter vector perturbation
$\mathbf{1}$	An all-ones vector of the same size as $\hat{\boldsymbol{\theta}}^t$
$IF^{(1)}(\hat{\theta}_i^t)$	The first-order influence of parameter $\hat{\theta}_i^t$ for task t.
$IF^{(2)}(\hat{\theta}_i^t)$	The second-order influence of parameter $\hat{\theta}_i^t$ for task t.
RIF	Our proposed method, i.e. Regularization based on Influence Function.
RIF-1	RIF calculates the first-order influence of parameters.
RIF-2	RIF calculates the second-order influence of parameters.
$acc_{i,j}$	The accuracy where learned model M_i is evaluated on the test data S_{test}^j of task j.
S_{test}^j	The test data of task j.
$acc_{N_{task},j}$	The final accuracy for task j in Task-IL.
$acc_{j,j} - acc_{N_{task},j}$	The forgetting rate for task j in Task-IL.
$acc_{i,0:i}$	The accuracy where learned model M_i is evaluated on the test data $S_{test}^{0:i}$.
$S_{test}^{0:i}$	All seen data (classes) so far.
$acc_{i,FTS}$	The accuracy of M_i on a Full Test Set [27] which is fixed and includes patterns of all the classes.

Appendix G. Convert parameter distribution perturbation into parameter vector perturbation

Here we show how to convert parameter distribution perturbation into parameter vector perturbation.

Thinking from parameter distribution. A fundamental question pertaining to regularization-based methods is that of which parameters are important for previous tasks. Previous works like EWC, MAS think of this issue from the aspect of parameter. They usually perturb the loss or the model output with respect to a specific parameter to determine the importance of this specific parameter for current task. For example, the importance of the parameters calculated by EWC is essentially the second-order gradient with respect to each parameter. This perturbation is intuitive and straightforward. According to Definition 4 where we assume that the parameters of the model follow a distribution F, we think about the importance of parameters from a high-level perspective of parameter distribution perturbation which is $(1-\epsilon)F + \epsilon\Delta_\theta$. If this perturbation leads to a large change in performance, then this parameter can be considered as important.

Considering empirical parameter distribution F_n instead of true parameter distribution F. It is obvious that we cannot perturb true parameter distribution F directly. We use the empirical parameter distribution F_n to estimate the true parameter distribution F. Assume that the model arrives at the local minimum ($\hat{\boldsymbol{\theta}}^t = (\hat{\theta}_1^t, \hat{\theta}_2^t, ..., \hat{\theta}_n^t)$) for current task t. We identify $\hat{\boldsymbol{\theta}}^t$ with its empirical parameter distribution F_n. Therefore, we now consider the perturbation of F_n with respect to a specific parameter $\hat{\theta}_i^t$, i.e., $(1-\epsilon)F_n + \epsilon\Delta_{\hat{\theta}_i^t}$.

The corresponding parameters vector perturbation. Although we use the empirical parameter distribution F_n to estimate the true parameter distribution F, we still cannot perturb the empirical parameter distribution F_n. An alternative is to achieve the perturbation of parameter vector $\hat{\boldsymbol{\theta}}^t$ with respect to the parameter $\hat{\theta}_i^t$. By observing the perturbation $(1-\epsilon)F_n + \epsilon\Delta_{\hat{\theta}_i^t} = F_n + \epsilon(\Delta_{\hat{\theta}_i^t} - F_n)$, we find that the empirical parameter distribution F_n changes to $\Delta_{\hat{\theta}_i^t}$ starting from F_n. We thus conclude the parameter vector perturbation $\hat{\boldsymbol{\theta}}_{\epsilon,\boldsymbol{i}}^t = \hat{\boldsymbol{\theta}}^t + \epsilon(\hat{\theta}_i^t\mathbf{1} - \hat{\boldsymbol{\theta}}^t)$. ϵ represents the small perturbation. $\mathbf{1}$ is an all-ones vector of the same size as $\hat{\boldsymbol{\theta}}^t$.

Data availability

The data that has been used is confidential.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, T. Tuytelaars, Memory aware synapses: learning what (not) to forget, in: ECCV, 2018.
[2] R. Aljundi, M. Rohrbach, T. Tuytelaars, Selfless sequential learning, in: ICLR, 2019.
[3] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: ICML, 2009.
[4] A.S. Benjamin, D. Rolnick, K.P. Körding, Measuring and regularizing networks in function space, in: ICLR, 2019.
[5] P. Buzzega, M. Boschini, A. Porrello, D. Abati, S. Calderara, Dark experience for general continual learning: a strong, simple baseline, in: NeurIPS, 2020.
[6] A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with A-GEM, in: ICLR, 2019.
[7] H. Chen, S. Si, Y. Li, C. Chelba, S. Kumar, D.S. Boning, C. Hsieh, Multi-stage influence function, in: NeurIPS, 2020.
[8] Z. Chen, B. Liu, Lifelong Machine Learning, second edition, 2018.
[9] C. Chung, S. Patel, R. Lee, L. Fu, S. Reilly, T. Ho, J. Lionetti, M.D. George, P. Taylor, Implementation of an integrated computerized prescriber order-entry system for chemotherapy in a multisite safety-net health system, Bull. Am. Soc. Hosp. Pharm. (2018).
[10] A. Cossu, G. Graffieti, L. Pellegrini, D. Maltoni, D. Bacciu, A. Carta, V. Lomonaco, Is class-incremental enough for continual learning?, Front. Artif. Intell. 5 (2022) 829842.
[11] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: defying forgetting in classification tasks, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[12] J. Deng, W. Dong, R. Socher, L. Li, K. Li, F. Li, Imagenet: a large-scale hierarchical image database, in: CVPR, 2009.
[13] S. Farquhar, Y. Gal, Towards robust evaluations of continual learning, CoRR, arXiv:1805.09733, 2018.
[14] L.T. Fernholz, Von Mises Calculus for Statistical Functionals, 2012.
[15] F.R. Hampel, E.M. Ronchetti, P. Rousseeuw, W.A. Stahel, Robust Statistics: the Approach Based on Influence Functions, Wiley-Interscience, New York, 1986.
[16] F.R. Hampel, E.M. Ronchetti, P. Rousseeuw, W.A. Stahel, Robust Statistics: The Approach Based on Influence Functions, 2005.
[17] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.
[18] J. Kirkpatrick, R. Pascanu, N.C. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell, Overcoming catastrophic forgetting in neural networks, CoRR, arXiv:1612.00796, 2016.
[19] P.W. Koh, P. Liang, Understanding black-box predictions via influence functions, in: ICML, 2017.
[20] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: NeurIPS, 2012.
[21] D. Kumaran, D. Hassabis, J.L. McClelland, What learning systems do intelligent agents need? Complementary learning systems theory updated, Trends Cogn. Sci. (2016).
[22] S. Lee, J. Kim, J. Jun, J. Ha, B. Zhang, Overcoming catastrophic forgetting by incremental moment matching, in: NeurIPS, 2017.
[23] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N.D. Rodríguez, Continual learning for robotics, CoRR, arXiv:1907.00182, 2019.
[24] Z. Li, D. Hoiem, Learning without forgetting, in: ECCV, 2016.
[25] S. Lin, L. Yang, D. Fan, J. Zhang, Beyond not-forgetting: continual learning with backward knowledge transfer, in: NeurIPS, 2022.
[26] Y. Liu, Y. Su, A. Liu, B. Schiele, Q. Sun, Mnemonics training: multi-class incremental learning without forgetting, in: CVPR, 2020.
[27] V. Lomonaco, D. Maltoni, Core50: a new dataset and benchmark for continuous object recognition, in: CoRL, 2017.
[28] D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, in: NeurIPS, 2017.
[29] A. Mallya, S. Lazebnik, Packnet: adding multiple tasks to a single network by iterative pruning, in: CVPR, 2018.
[30] P. Pan, S. Swaroop, A. Immer, R. Eschenhagen, R.E. Turner, M.E. Khan, Continual deep learning by functional regularisation of memorable past, in: NeurIPS, 2020.
[31] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: a review, Neural Netw. (2019).
[32] B. Pfülb, A. Gepperth, A comprehensive, application-oriented study of catastrophic forgetting in dnns, in: ICLR, 2019.
[33] S. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, icarl: incremental classifier and representation learning, in: CVPR, 2017.
[34] T.G.J. Rudner, F.B. Smith, Q. Feng, Y.W. Teh, Y. Gal, Continual learning via sequential function-space variational inference, in: ICML, 2022, pp. 18871–18887.
[35] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, CoRR, arXiv:1606.04671, 2016.
[36] J. Serrà, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: ICML, 2018.
[37] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning - from Theory to Algorithms, 2014.
[38] A. van der Vaart, Asymptotic Statistics, 1998.
[39] G.M. van de Ven, A.S. Tolias, Three scenarios for continual learning, CoRR, arXiv:1904.07734, 2019.
[40] M. van der Ven, A.S. Tolias, Generative replay with feedback connections as a general strategy for continual learning, CoRR, arXiv:1809.10635, 2018.
[41] M. Wojnowicz, B. Cruz, X. Zhao, B. Wallace, M. Wolff, J. Luan, C. Crable, "Influence sketching": finding influential samples in large-scale regressions, in: BigData, 2016.
[42] J. Xu, Z. Zhu, Reinforced continual learning, in: NeurIPS, 2018.
[43] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in: ICML, 2017.
[44] W. Liu, I.W. Tsang, Making decision trees feasible in ultrahigh feature and label dimensions, J. Mach. Learn. Res. (2017).
[45] W. Liu, I.W. Tsang, On the optimality of classifier chain for multi-label classification, in: NeurIPS, 2015.
[46] P. Xu, W. Hu, J. Wu, W. Liu, Opinion maximization in social trust networks, in: IJCAI, 2020.
[47] Y. Mao, Z. Wang, W. Liu, X. Lin, P. Xie, MetaWeighting: learning to weight tasks in multi-task learning, in: ACL, 2022.
[48] Y. Mao, S. Yun, W. Liu, B. Du, Tchebycheff procedure for multi-task text classification, in: ACL, 2020.
[49] R. Gao, W. Liu, DDGR: continual learning with deep diffusion-based generative replay, in: ICML, 2023.
[50] X. Li, Z. Xin, W. Liu, Defending against adversarial attacks via neural dynamic system, in: NeurIPS, 2022.