Regularizing Second-Order Influences for Continual Learning

Zhicheng Sun[1], Yadong Mu[1,2*], Gang Hua[3]

[1]Peking University, [2]Peng Cheng Laboratory, [3]Wormpex AI Research

{sunzc,myd}@pku.edu.cn, ganghua@gmail.com

Abstract

Continual learning aims to learn on non-stationary data streams without catastrophically forgetting previous knowledge. Prevalent replay-based methods address this challenge by rehearsing on a small buffer holding the seen data, for which a delicate sample selection strategy is required. However, existing selection schemes typically seek only to maximize the utility of the ongoing selection, overlooking the interference between successive rounds of selection. Motivated by this, we dissect the interaction of sequential selection steps within a framework built on influence functions. We manage to identify a new class of second-order influences that will gradually amplify incidental bias in the replay buffer and compromise the selection process. To regularize the second-order effects, a novel selection objective is proposed, which also has clear connections to two widely adopted criteria. Furthermore, we present an efficient implementation for optimizing the proposed criterion. Experiments on multiple continual learning benchmarks demonstrate the advantage of our approach over state-of-the-art methods. Code is available at https://github.com/feifeiobama/InfluenceCL.

1. Introduction

The ability to continually accumulate knowledge is a hallmark of intelligence, yet prevailing machine learning systems struggle to remember old concepts after acquiring new ones. *Continual learning* has hence emerged to tackle this issue, also known as *catastrophic forgetting*, and thereby enable the learning on long task sequences which are frequently encountered in real-world applications [16, 18]. Amongst a variety of valid methods, the replay-based approaches [34, 41] achieve evidently strong results by buffering a small *coreset* of previously seen data for rehearsal in later stages. Due to the rigorous constraints on memory overhead, the replay buffer needs be carefully maintained such that only the samples deemed most critical

---
*Corresponding author.

Figure 1. (a) In continual learning, earlier coreset selection exerts a profound influence on subsequent steps through the data flow. (b) By ignoring this, a greedy selection strategy can degrade over time. Therefore, we propose to model and regularize the influence of each selection on the future.

for preserving overall performance are selected during the learning procedure.

Despite much effort in sophisticated design of coreset selection criteria [3,7,49,55], existing works are mostly developed for an oversimplified scenario, where each round of selection is considered isolated and primarily focused on refining single-round performance. For example, Borsos *et al.* [7] greedily minimize the empirical risk on the currently available subset of data. However, as shown in Fig. 1a, the actual selection process within continual learning is rather different in that each prior selection defines the input for subsequent selection and therefore has an inevitable impact on the later decision. Neglecting such interactions between successive selection steps, as in previous practice, may result in a degraded coreset after the prolonged selection process. To maximize overall performance, an ideal continual learning method should take into account the future influence of each round of selection, which remains unresolved due to the obscure role of intermediate selection results.

This work models the interplay among multiple selection steps with the classic tool of influence functions [22, 30], which estimates the effect of each training point by perturbing sample weights. Similarly, we begin by upweighting two samples from consecutive time steps, and uncover that the latter one's influence evaluation is biased due to the consequent perturbations in test gradient and coreset Hessian. This gives rise to a new class of second-order influences that interfere with the initial influence-based selection strategy. To be specific, the selection tends to align across different rounds, favoring those samples with a larger gradient projection along the previously upweighted ones. It further suggests that some unintended adverse effects of the early selection steps, such as incidental bias introduced into the replay buffer, will be amplified after rounds of selection and impair the final performance.

To address the newly discovered disruptive effects, we propose to regularize each round of selection beforehand based on its expected second-order influence, as illustrated in Fig. 1b. Intuitively, the selection with a large magnitude of second-order influence will substantially interfere with future influence estimates, and therefore should be avoided. However, the magnitude itself cannot be precalculated for direct guidance of the selection, since it is related to some unknown future data. We instead derive a tractable upper bound for the magnitude which results in the form of ℓ_2-norm, and integrated it with the vanilla influence functions to serve as the final selection criterion. The proposed regularizer can be interpreted with clarity as it equates to a couple of existing criteria ranging from gradient matching to diversity, under varying simplifications. Finally, we present an efficient implementation that tackles the technical challenges in selecting with neural networks.

Our contributions are summarized as below:

- We investigate the previously-neglected interplay between consecutive selection steps within an influence-based selection framework. A new type of second-order influences is identified, and further analysis states its harmfulness in continual learning.

- A novel regularizer is proposed to mitigate the second-order interference. The regularizer is associated with two other popular selection criteria and can be efficiently optimized with our implementation.

- Comprehensive experiments on three continual learning benchmarks (Split CIFAR-10, Split CIFAR-100 and Split *mini*ImageNet) clearly illustrates that our method achieves new state-of-the-art performance.

2. Related works

Continual learning studies the training of models on a sequence of tasks with potential data distribution shift. It is known for suffering from catastrophic forgetting [35],

where the model abruptly forgets past knowledge after being updated on new tasks. To overcome this effect, three main streams of methods have been developed: weight regularization [28, 33, 57], parameter isolation [2, 43, 45, 56] and memory replay [34, 41, 46]. However, the first two approaches exhibit over-regularization or unconstrained parameter growth on long task sequences [21]. In this work, we focus on the replay-based approach, which stores a small subset of previous data in a memory buffer, to be revisited when learning new tasks.

One of the key components in replay-based methods is the sample selection strategy, which undertakes the task of summarizing representative data into the replay buffer. The pioneering works [34, 41] featured a ring buffer for exemplar management, followed by reservoir sampling [42, 51] and its class-balanced variants [14, 40]. More sophisticated selection criteria have been proposed as well, such as matching the overall gradient [4, 49] or enhancing the diversity of samples [3, 5, 55]. Recently, Borsos *et al.* [7] and Sun *et al.* [47] introduced influence functions [30] to continual learning, providing a fresh and interpretable perspective on sample selection. Our method builds on an influence-based framework but also has clear connections to gradient matching and diversity.

Influence functions, known from robust statistics [22], were recently advocated in machine learning by Koh and Liang [30] to estimate the effect of upweighting a training sample to the model parameter and test loss. Since then, influence functions have served as sample selection criteria for various scenarios such as data-efficient learning [48, 53], class imbalanced learning [38] and noisy label learning [31]. Typically, samples with the highest influence scores are identified and then downweighted or discarded. Closely related to our work are their applications in continual learning [7, 47, 61] and domain adaptation [15], but they were still developed for the single-round selection scenario described earlier and overlooked the high-order interactions between multiple rounds of selection, leading to suboptimal results in the continual learning setup.

There are also efforts made toward more accurate and efficient calculation of influence functions. For example, Basu *et al.* [6] introduced second-order approximations to measure group effects [6, 29] of large, coherent groups of training points. However, their analysis is limited to jointly optimized samples and cannot be applied to continuously incoming data. In the other direction, various speedup tricks [7, 20, 44, 59] have been proposed for over-parameterized neural networks, among which a representative approach is to use neural tangent kernels [24] as the proxy model [7, 59]. We adapt some of these techniques to the proposed second-order influence functions and derive an efficient selection criterion for continual learning.

3. Influence functions for continual learning

3.1. Problem formulation

We consider learning on a continuous stream of data $\mathcal{Z} = \{z_i\}_{i=1}^n$ assuming that only a fraction of samples $\mathcal{Z}_t \subset \mathcal{Z}$ can be accessed at each time step t. The main challenge in such a learning paradigm is to retain performance on the seen data $\mathcal{Z}_{1:t} = \bigcup_{i=1}^t \mathcal{Z}_i$ in later stages. To achieve this, we adopt a most straightforward and effective approach that stores a few samples $\mathcal{C}_t$ in a replay buffer as the representatives of previous data $\mathcal{Z}_{1:t}$, where the memory size $|\mathcal{C}_t| \leq m$ and $m \ll n$. In consequence, it is crucial to maintain a high-quality subset throughout the training process [3, 55]. Formally, let θ be the model parameters and $L(z_i, \theta)$ be the loss on training point z_i. Then, our selection goal is to preserve model performance on $\mathcal{Z}_{1:t}$ by replaying on $\mathcal{C}_t$, which is formulated as:

$$
\min_{\mathcal{C}_t \subset \mathcal{C}_{t-1} \cup \mathcal{Z}_t, |\mathcal{C}_t| \leq m} \sum_{z_i \in \mathcal{Z}_{1:t}} L(z_i, \hat{\theta})
$$
$$
\text{s.t.} \quad \hat{\theta} = \arg\min_{\theta} \sum_{z_i \in \mathcal{C}_t} L(z_i, \theta). \tag{1}
$$

Though its main form is consistent with that of the well-studied coreset selection problem [7], note that the outer objective is intractable since $\mathcal{Z}_{1:t}$ is partly unavailable.

In the following sections, we first present a baseline selection strategy produced by the vanilla influence functions, then showcase its limitation in handling continual selection and propose our improved version.

3.2. Influence-based selection

Influence functions [30] provide an efficient approximation for solving the coreset selection problem by perturbing sample weights, for which the previous exact solution requires expensive leave-one-out retraining.

Before diving in, we first convert Eq. (1) into a tractable problem by leveraging the empirical risk on $\mathcal{C}_{t-1} \cup \mathcal{Z}_t$[1] as a proxy for the original test loss on $\mathcal{Z}_{1:t}$, presuming a close correlation between them:

$$
\min_{\mathcal{C}_t \subset \mathcal{C}_{t-1} \cup \mathcal{Z}_t, |\mathcal{C}_t| \leq m} \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} L(z_i, \hat{\theta})
$$
$$
\text{s.t.} \quad \hat{\theta} = \arg\min_{\theta} \sum_{z_i \in \mathcal{C}_t} L(z_i, \theta). \tag{2}
$$

Solving this involves uncovering the effect of selecting or discarding some training sample $z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t$, for which a small weight ϵ is added to the interested sample, and then the optimal point in the inner optimization becomes:

$$
\hat{\theta}_{\epsilon,z} = \arg\min_{\theta} \sum_{z_i \in \mathcal{C}_t} L(z_i, \theta) + \epsilon L(z, \theta). \tag{3}
$$

---

[1]The loss on $\mathcal{Z}_t$ needs to be reweighted by a constant to balance with the loss on $\mathcal{C}_{t-1}$. We omit the weight for simplicity unless otherwise stated.

Let $\hat{\theta}_t = \hat{\theta}_{\epsilon,z}|_{\epsilon=0}$ denote the initial optimal parameters and $H_{\hat{\theta}_t} = \sum_{z_i \in \mathcal{C}_t} \nabla_\theta^2 L(z_i, \hat{\theta}_t)$ denote the Hessian which by assumption is positive definite. A classic result by Cook and Weisberg [17] yields the change in model parameters as $\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\Big|_{\epsilon=0} = -H_{\hat{\theta}_t}^{-1} \nabla_\theta L(z, \hat{\theta}_t)$, from which we can derive the influence of upweighting z on the outer loss in Eq. (2):

$$
\begin{aligned}
\mathcal{I}(z) &= \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \frac{dL(z_i, \hat{\theta}_{\epsilon,z})}{d\epsilon}\Big|_{\epsilon=0} \\
&= -\sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z_i, \hat{\theta}_t)^\top H_{\hat{\theta}_t}^{-1} \nabla_\theta L(z, \hat{\theta}_t).
\end{aligned} \tag{4}
$$

We denote the inverse Hessian-vector product therein as $s_t = H_{\hat{\theta}_t}^{-1} \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z_i, \hat{\theta}_t)$ for future use.

To minimize the outer loss, the coreset $\mathcal{C}_t$ should contain the samples with the lowest negative influence scores. It can be efficiently solved by a greedy strategy, which starts from the full set $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \mathcal{Z}_t$ and discards the sample z with the largest influence at each iteration until the memory constraint $|\mathcal{C}_t| \leq m$ is met. While such a strategy is qualified in a variety of evaluations [7, 52, 53, 61], we argue that it is not the answer to continual learning by dissecting the joint effect of influence-based selection in a sequence.

3.3. Second-order influences in continual selection

In continual learning, each prior selection determines the input for subsequent steps and thus influences the effectiveness of future selection, *i.e.*, it may interfere with the later evaluation of the selection criterion.

To model such an interaction, we consider two samples z and z' from consecutive steps t and $t+1$. With the previous sample z upweighted by ϵ, the next round of selection is also affected, by a drift in the influence score of z'. Specifically, there are two cases depending on whether z and z' are jointly optimized in the following step:

(1) The samples z and z' are not jointly optimized, meaning that the previous sample z is excluded from the coreset in the next round and only serves as a test point. As a result, only the outer loss in Eq. (2) is affected, with the influence score of the subsequent candidate z' changed to:

$$
\begin{aligned}
\mathcal{I}_{\epsilon,z}(z') = -\Bigg(&\sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1}) + \epsilon \nabla_\theta L(z, \hat{\theta}_{t+1}) \Bigg)^\top \\
&H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}).
\end{aligned} \tag{5}
$$

By taking the derivative w.r.t. ϵ, it yields the influence of upweighting z on the influence score of z'. Alternatively, it can be viewed as the second-order influence of two samples

3

from consecutive selection steps on the outer loss:

$$
\begin{aligned}
\mathcal{I}^{(2)}(z, z') &= \left. \frac{d\mathcal{I}_{\epsilon,z}(z')}{d\epsilon} \right|_{\epsilon=0} \\
&= -\nabla_\theta L(z, \hat{\theta}_{t+1})^\top H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}).
\end{aligned}
\tag{6}
$$

We will soon analyze its effect using the geometric view of projection.

(2) The samples z and z' are jointly optimized by the inner objective in Eq. (2). In this case, the inner Hessian is also changed, in the direction of $H_{\hat{\theta}_{t+1},z} := \nabla_\theta^2 L(z, \hat{\theta}_{t+1})$. Hence, the influence score of the following sample z' is modified to:

$$
\begin{aligned}
\mathcal{I}_{\epsilon,z}(z') = -&\left(\sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1}) + \epsilon \nabla_\theta L(z, \hat{\theta}_{t+1}) \right)^\top \\
&(H_{\hat{\theta}_{t+1}} + \epsilon H_{\hat{\theta}_{t+1},z})^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}).
\end{aligned}
\tag{7}
$$

Similarly, the second-order influence of z' and z can be obtained as follows (see Supplementary for the derivation):

$$
\begin{aligned}
\mathcal{I}^{(2)}(z, z') &= \left. \frac{d\mathcal{I}_{\epsilon,z}(z')}{d\epsilon} \right|_{\epsilon=0} \\
&= -(\nabla_\theta L(z, \hat{\theta}_{t+1}) - H_{\hat{\theta}_{t+1},z} s_{t+1})^\top H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}).
\end{aligned}
\tag{8}
$$

Compared to the first case, there is an additional term related to the Hessian perturbation, whose effect will be discussed in Sec. 3.5. Nevertheless, the prior term in Eq. (6) is fully retained.

Second-order influences compromise sample diversity. For an intuitive understanding of the new influences, let $\langle u, v \rangle = u^T H_{\hat{\theta}_{t+1}}^{-1} v$ define an inner product over the gradient space and induce the notion of projection. In both cases, the sample z' with a larger gradient projection onto the previous z is estimated with a lower influence and thus more likely to be selected. This drives the influence-based selection strategy to include more similar samples to the buffer and expel dissimilar ones, leading to a lack of diversity.

Second-order influences amplify memory bias. While the above analysis is based on a real perturbation to sample weights, it could also be a simulated perturbation on the gradient statistics of the coreset. Such perturbations are common in the learning process, as each selection step may introduce an incidental bias to the buffer. Like real samples, they can produce the same second-order effect of attracting similar ones into the buffer. Although these perturbations are expected to have zero mean like white noises, the earlier perturbations have taken a greater effect in accumulating memory bias, which results in inaccurate influence estimates and degradation of future selection.

3.4. Regularizing second-order influences

To mitigate the harmful second-order effects without imposing additional memory overhead, we suggest regularizing each coreset selection step in advance, so that future influence estimates will be less error-prone.

Suppose we are at the t-th step, the goal is to minimize its interference with some sample z' of the next step, namely the total effect of discarding the subset $\overline{\mathcal{C}}_t = \mathcal{C}_{t-1} \cup \mathcal{Z}_t - \mathcal{C}_t$, which can be computed by summing Eq. (6) or Eq. (8)[2]. However, it cannot be predetermined which equation to be used, so a hyperparameter $\mu \in [0,1]$ is introduced as the ratio for the latter case, allowing us to consider a weighted sum of the two second-order influences:

$$
\begin{aligned}
\Delta\mathcal{I}(z') &\approx -\sum_{z \in \overline{\mathcal{C}}_t} \mathcal{I}^{(2)}(z, z') \cdot 1 \\
&= \sum_{z \in \overline{\mathcal{C}}_t} (\nabla_\theta L(z, \hat{\theta}_{t+1}) - \mu H_{\hat{\theta}_{t+1},z} s_{t+1})^T H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}).
\end{aligned}
\tag{9}
$$

Ideally, the magnitude of total influence $\Delta I(z')$ should be very small so that future selection will receive less interference. However, this term is intractable since it is associated with an unknown sample z', so we turn to optimize its upper bound given by the Cauchy–Schwarz inequality:

$$
\begin{aligned}
|\Delta\mathcal{I}(z')| \leq &\left\| \sum_{z \in \overline{\mathcal{C}}_t} (\nabla_\theta L(z, \hat{\theta}_{t+1}) - \mu H_{\hat{\theta}_{t+1},z} s_{t+1}) \right\| \\
&\times \left\| H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}) \right\|,
\end{aligned}
\tag{10}
$$

We omit the second norm that depends mainly on the upcoming data, and employ the first one to regularize the ongoing selection, during which the yet unknown θ_{t+1} and s_{t+1} are approximated with θ_t and s_t, respectively:

$$
\mathcal{R}(\mathcal{C}_t) = \left\| \sum_{z \in \overline{\mathcal{C}}_t} (\nabla_\theta L(z, \hat{\theta}_t) - \mu H_{\hat{\theta}_t,z} s_t) \right\|.
\tag{11}
$$

Proposed selection criterion. A weighted sum of the first-order influence and second-order regularizer balanced by a hyperparameter ν is adopted to guide the selection process. The final objective at the t-th step is as follows:

$$
\min_{\mathcal{C}_t \subset \mathcal{C}_{t-1} \cup \mathcal{Z}_t, |\mathcal{C}_t| \leq m} \sum_{z \in \mathcal{C}_t} \mathcal{I}(z) + \nu \mathcal{R}(\mathcal{C}_t).
\tag{12}
$$

Next, we will interpret it by unveiling its connection to two widely adopted selection criteria, and then present an efficient implementation for deep models in practical use.

---

[2]With first-order approximations here, the group effect [6] of dropped datapoints can be fairly neglected.

Effect of μ

| Full gradient | Feasible region, $\mu > 0$ | Feasible region, $\mu = 0$ |

Figure 2. Geometric interpretation of our proposed regularizer. As μ grows larger from zero, the regularizer shifts from the initial gradient matching to introducing more gradient diversity along the predominant direction.

3.5. Connection to gradient matching and diversity

The following case studies of Eq. (11) allow to establish clear connections between the proposed regularizer and two commonly used selection criteria: gradient matching [4, 27, 60] and sample diversity [3, 5, 55].

Start by setting $\mu = 0$, which corresponds to having full confidence that the currently discarded samples will not be favored in the future, and then the regularizer becomes:

$$\mathcal{R}(\mathcal{C}_t) = \left\| \sum_{z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z, \hat{\theta}_t) - \sum_{z \in \mathcal{C}_t} \nabla_\theta L(z, \hat{\theta}_t) \right\|. \quad (13)$$

It turns out to minimize the Euclidean distance between the full gradient and the coreset gradient, which is equivalent to gradient matching. For an intuitive look, we consider a hard constraint $\mathcal{R}(\mathcal{C}_t) < \delta$ and mark the feasible region of the coreset gradient with light blue in Fig. 2.

In the more general case $\mu > 0$, we temporarily ignore the sample information carried in Hessian for simplicity, by assuming that the Hessian is identical for all training samples. Then there is:

$$\mathcal{R}(\mathcal{C}_t) = \left\| (1 - \alpha\mu) \sum_{z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z, \hat{\theta}_t) - \sum_{z \in \mathcal{C}_t} \nabla_\theta L(z, \hat{\theta}_t) \right\|, \quad (14)$$

where α is a coefficient related to the coreset size only. As illustrated in Fig. 2, the feasible region is moved in the opposite direction of the total gradient. This promotes samples whose gradients are less aligned or even conflicting with the main direction to be included in the coreset, thus encouraging diversity. In a holistic view, the hyperparameter μ plays the role of balancing gradient matching and diversity.

While these simplifications equate our proposed regularizer to a couple of intuitive criteria, the original form excels in the additional incorporation of Hessian-related information, which will be empirically demonstrated in the experiments.

3.6. Implementation for neural networks

This section presents an efficient implementation of optimizing the selection criterion with deep neural networks. Since similar issues for the vanilla influence functions have been intensively addressed [7, 30, 47], the focus here is on the optimization of the newly proposed regularizer.

A key difference of our second-order regularizer is that it cannot be partitioned into independent terms associated to different samples, which hinders application of the aforementioned greedy selection strategy. We overcome this challenge by leveraging the first-order Taylor expansion of the regularizer. To this end, the original variable $\mathcal{C}_t$ is replaced by a continuous vector w_t, where each element $w_{t,i} = \mathbb{1}(z_i \in \mathcal{C}_t)$ indicates whether the corresponding sample is included in the coreset. After that, the regularizer can be transformed into an equivalent form:

$$\mathcal{R}(w_t) = \left\| \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} (1 - w_{t,i})(\nabla_\theta L(z_i, \hat{\theta}_t) - \mu H_{\hat{\theta}_t, z_i} s_t) \right\|, \quad (15)$$

whose first-order Taylor approximation w.r.t. w_t is a linear combination of uncorrelated terms (see Supplementary for details) and can therefore be addressed with greedy heuristics. We then implement an iterative optimization algorithm for selection that starts from an exterior point $w_t = \mathbf{1}$, and modifies w_t during each iteration to drop the sample z_i with the largest $I(z_i) + \nu \nabla_{w_{t,i}} \mathcal{R}(w_t)$, until the memory constraint $\|w_t\|_0 \leq m$ is satisfied. Such a strategy of greedily minimizing the proposed criterion is found to be sufficient in later experiments.

To further enable efficient inference on non-convex and over-parameterized deep neural networks, we are inspired by Borsos *et al.* [7] and employ neural tangent kernels [24] as the proxy model for computing the regularizer in Eq. (15). The inverse Hessian-vector product s_t therein can be effectively computed using the conjugate gradient method, with a small damping term added to the Hessian to ensure that it is positive definite.

Computational cost. The performance bottleneck lies in computing the inverse Hessian-vector product s_t, for which the conjugate gradient method takes $O(m'p^2)$ time to yield an exact solution [30], where $m' = |\mathcal{C}_{t-1} \cup \mathcal{Z}_t|$ and p is the number of parameters. Since this part of cost is shared with the first-order influence function, our regularizer introduces only a small overhead of $O(m'p)$ in evaluating the Hessian-vector product $H_{\hat{\theta}_t, z_i} s_t$ each time. In addition, the adopted neural tangent kernel approximation significantly eases the burden by reducing the number of parameters.

Our computational efficiency will be further verified by the runtime comparison in Sec. 4.3 and the per-step overhead analysis in the supplementary material.

Method		Class-incremental				Task-incremental			
		$m = 300$		$m = 500$		$m = 300$		$m = 500$	
		ACC (%)	BWT (%)	ACC (%)	BWT (%)	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Non-IF	GEM [34]	37.51	-70.48	36.95	-69.76	89.34	-9.09	90.42	-7.88
	A-GEM [13]	20.02	-95.68	20.01	-95.69	85.52	-14.07	86.45	-12.83
	ER [14]	34.19	-78.18	40.45	-70.36	88.97	-9.95	90.60	-7.74
	GSS [3]	35.89	-75.80	41.96	-68.24	88.05	-10.63	90.38	-7.73
	ER-MIR [1]	38.53	-72.72	42.65	-67.50	88.50	-10.33	90.63	-7.62
	GDUMB [40]	36.92	-	44.27	-	73.22	-	78.06	-
	HAL [12]	24.45	-83.56	27.94	-80.01	79.90	-14.39	81.84	-12.73
	GMED [25]	38.12	-73.16	43.68	-66.21	88.91	-9.76	89.72	-8.75
IF	Vanilla IF	41.76	-68.59	47.14	-62.20	90.67	-7.65	91.06	-7.36
	MetaSP [47]	43.76	-66.37	50.10	-58.39	89.91	-9.00	91.41	-7.36
	Ours	**48.62**	**-60.24**	**53.07**	**-54.44**	**91.52**	**-6.94**	**92.53**	**-5.46**

Table 1. Comparison with influence function (IF)-based methods and non-IF-based methods on Split CIFAR-10 under different memory constraints. For the compared methods, we report the results summarized in [47] in chronological order. The task-incremental setting asks the model to classify within each task, while the class-incremental setting requires predicting both task identity and class label.

4. Experiments

4.1. Experimental setup

Datasets. We conduct experiments on three continual learning benchmarks: (1) Split CIFAR-10 [58] splits the original CIFAR-10 [32] dataset into 5 disjoint subsets, where each subset comprises 2 classes. (2) Split CIFAR-100 [58] is constructed from the CIFAR-100 [32] dataset, containing 10 tasks with disjoint class labels. (3) Split *mini*ImageNet [14, 47] derives from the few-shot learning dataset *mini*ImageNet [50], a subset of ImageNet [19] with 100 classes and 600 images per class. The dataset is divided equally to create 5 sequential tasks, with each image resized to 32×32. For all three benchmarks, we follow their original papers in splitting training and test sets.

Metrics. Two evaluation metrics are employed, including Average Accuracy (ACC) and Backward Transfer (BWT) [34], where ACC is the average accuracy after the model has been trained on all tasks and BWT indicates the average forgetting of all previous tasks. Formally, they are defined as:

$$\text{ACC} = \frac{1}{T} \sum_{i=1}^{T} R_{T,i}, \quad \text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} R_{T,i} - R_{i,i},$$
(16)

where T is the number of tasks and $R_{i,j}$ is the accuracy of the model on the j-th task after learning i tasks. As done in [9, 47], we measure the metrics under both task-incremental and class-incremental settings. The latter is particularly difficult as it does not provide task identity for each sample at test time.

Baselines. Our method is compared against nine replay-based competitors: ER [14], GEM [34], A-GEM [13], GSS [3], ER-MIR [1], GDUMB [40], HAL [12], GMED [25] and MetaSP [47]. Additionally, a base strategy that uses the vanilla influence functions (IF) introduced in Sec. 3.2 is considered. To make a fair comparison, we first reproduce the main experiments of [47] and then report the results therein.

Implementation details. We adopt ResNet-18 [23] as the backbone architecture. The model is optimized by SGD for 50 epochs per task, with fixed batch size and replay batch size of both 32, as in [10]. The learning rate is set to 0.1 on Split CIFAR-10 and Split CIFAR-100, and 0.03 on Split *mini*ImageNet. The other hyperparameters are empirically set as $\mu = 0.5$ and $\nu = 0.01$ by default, whose sensitivity analysis will be given in Sec. 4.3. In the training, we use cross entropy loss for replay samples and apply standard data augmentations of random cropping and horizontal flipping. To limit the computational overhead of our method, the replay buffer is only updated during the last epoch of each task. The continual learning scenarios are provided by Mammoth [8, 9] which relies on PyTorch [39]. For calculating Neural Tangent Kernels, the library of [37] is employed.

4.2. Main results

The results on the Split CIFAR-10 benchmark are summarized in Tab. 1. Overall, our approach achieves the best continual learning performance in all metrics under various evaluation settings and memory constraints, suggesting its superiority over state-of-the-art methods. From the results, we also observe that: (1) The first-order influence-

Method	Class-incremental				Task-incremental			
	$m = 500$		$m = 1000$		$m = 500$		$m = 1000$	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Non-IF GEM [34]	15.91	-77.07	22.79	-68.32	68.68	-18.72	73.71	-12.81
A-GEM [13]	9.31	-85.18	9.27	-84.88	55.28	-34.10	55.95	-33.01
ER [14]	13.75	-81.64	17.56	-77.52	66.82	-22.73	71.74	-17.40
GSS [3]	14.01	-80.02	17.87	-76.04	66.80	-21.44	71.98	-16.06
ER-MIR [1]	13.49	-82.09	17.56	-77.59	66.18	-23.60	71.20	-18.10
GDUMB [40]	11.11	-	15.75	-	36.40	-	43.25	-
HAL [12]	8.20	**-65.70**	10.59	**-63.86**	44.98	-25.17	50.07	-20.61
GMED [25]	14.56	-80.68	18.67	-76.23	68.82	-20.53	73.91	-15.10
IF Vanilla IF	17.49	-77.54	22.75	-72.56	71.74	-17.90	73.25	-17.22
MetaSP [47]	19.28	-76.13	25.72	-68.69	70.81	-19.74	**76.14**	**-14.32**
Ours	**21.15**	-73.24	**27.99**	-64.56	**72.53**	**-17.22**	74.27	-16.37

(a) Split CIFAR-100

Method	Class-incremental				Task-incremental			
	$m = 500$		$m = 1000$		$m = 500$		$m = 1000$	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Non-IF A-GEM [13]	10.69	-49.22	10.69	-49.16	18.34	-39.65	18.78	-39.05
ER [14]	11.00	-50.84	11.35	-50.08	28.97	-28.40	31.59	-24.95
GSS [3]	11.09	-50.66	11.42	-49.91	28.67	-28.71	31.75	-24.56
ER-MIR [1]	11.07	-50.46	11.32	-49.92	29.10	-27.95	31.39	-24.89
GDUMB [40]	6.22	-	7.15	-	16.37	-	17.69	-
GMED [25]	11.03	-50.23	11.73	-48.93	30.47	-26.02	32.85	-22.69
IF Vanilla IF	12.08	-48.55	14.64	-47.15	33.74	-21.71	37.55	-19.28
MetaSP [47]	12.74	-48.84	14.54	-45.52	34.36	-21.70	37.20	-17.83
Ours	**13.63**	**-47.94**	**16.15**	**-43.78**	**36.46**	**-19.48**	**39.61**	**-16.01**

(b) Split *mini*ImageNet

Table 2. Comparison with state-of-the-art methods on more challenging benchmarks, including Split CIFAR-100 and Split *mini*ImageNet. While most setups follow the previous experiment, a different hyperparameter setting $\mu = 0.75$ is adopted on Split CIFAR-100 with memory size $m = 1000$ empirically.

based methods obtain significant performance gains over traditional methods, but our version almost doubles the improvement under the small memory setting of $m = 300$. By taking into account the second-order influences, it outperforms the state-of-the-art method by 4.86% in ACC and 6.13% in BWT in the challenging class-incremental evaluations, which in turn reflects the non-trivial effect of second-order influences. (2) Though the base strategy vanilla IF is slightly inferior to the state-of-the-art method MetaSP in most cases for it does not introduce any new losses in the model training, it can substantially surpass MetaSP with a simple regularization term on the selection criterion. This verifies the effectiveness of our proposed regularizer.

We further evaluate our method on the more difficult Split CIFAR-100 and Split *mini*ImageNet benchmarks. As presented in Tab. 2, our approach brings consistent perfor-

mance improvement over vanilla IF on both benchmarks, by up to 5.24% and 8.0% in terms of ACC and BWT, respectively. Compared to other approaches, it obtains state-of-the-art performance on Split *mini*ImageNet in all metrics, and leads ACC by about 2% on three different setups of Split CIFAR-100. Only in the simplest scenario of task-incremental learning on Split CIFAR-100 with $m = 1000$, our method fails to surpass MetaSP, for which we speculate that the high-order interference is alleviated to some extent by the large buffer itself. Nevertheless, our method remains competitive in most evaluations.

4.3. Ablation study and analysis

This section empirically justifies the design of our approach through a series of experiments conducted on Split CIFAR-10 with a fixed memory size of $m = 500$.

Method	Class-incremental		Task-incremental	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Vanilla IF	47.14	-62.20	91.06	-7.36
+ matching	49.86	-58.24	92.30	-5.78
+ diversity	50.59	-57.06	90.69	-7.28
+ both	51.61	-56.31	91.93	-6.11
Ours	**53.07**	**-54.44**	**92.53**	**-5.46**

Table 3. Comparison with alternative regularizers on Split CIFAR-10 with memory size $m = 500$. The diversity-based regularizer is implemented by minimizing the norm of the coreset gradient.

(a) Weight for the regularizer   (b) Weight for Hessian-related term

Figure 3. Sensitivity to hyperparameters on Split CIFAR-10 with $m = 500$. They are set to $\nu = 0.01$ and $\mu = 0.5$ by default.

Comparison with other regularizers.   To further validate the effectiveness of our regularizer, it is compared to two closely related criteria, namely gradient matching and diversity, in Tab. 3. Our regularizer draws on the strengths from both criteria, including the satisfactory task-incremental performance of gradient matching and the higher class-incremental ACC of gradient diversity. In addition, we compare it with a combination of both criteria, during which the considerable performance advantage of our Hessian-aware regularizer illustrates the benefits from incorporating Hessian-related information. More comparative studies can be found in the supplementary material.

Hyperparameter sensitivity.   In order to explore the sensitivity of our strategy to hyperparameter settings, we vary the two coefficients μ and ν and plot the response of model performance. Figure 3a shows that under both evaluation settings, ACC improves consistently with the growth of weight ν on the second-order regularizer until its stability. The situation is more complicated in Fig. 3b, where the two ACC curves exhibit inconsistent trends regarding the weight μ for the Hessian-related term. Recall that in Sec. 3.5 μ is associated with the degree of diversity, so it may be interpreted that the more challenging class-incremental setting requires higher memory diversity, while the simpler task-increment setting accommodates lower diversity.

Figure 4.   Evolution of the influence estimation accuracy measured by Kendall's τ [26].

Method	Runtime (h)
ER [14]	0.92
A-GEM [13]	1.22
GSS [3]	0.92
HAL [13]	1.73
Vanilla IF	1.69
MetaSP [47]	2.03
Ours	1.74

Table 4. Running time measured on Split CIFAR-10 using a single NVIDIA 2080 Ti GPU.

Accuracy of influence estimates.   To measure accuracy, the previous practice of comparing with the ground-truth given by leave-one-out retraining [6,30] imposes an impractical cost under the continual learning setup. Hence, we turn to a large reservoir sampling buffer to produce unbiased influence estimates and compute their correlation with other predictions. The Kendall rank correlation [26] is adopted since we are more concerned with each sample's rank during selection. As illustrated in Fig. 4, the buffer maintained by vanilla IF dissociates from the unbiased buffer after a short period, producing inaccurate influence estimates. In contrast, our strategy yields a relatively higher agreement, which verifies its effectiveness in mitigating interference.

Running time.   Table 4 compares our proposed method with several replay-based competitors in terms of efficiency. Though influence-based approaches generally incur higher time costs in the Hessian-related computations, our method imposes a limited overhead with the newly introduced second-order regularizer. This implies that the above improvements are achieved within comparable training time, demonstrating the high efficiency of our implementation.

5. Concluding remarks

We propose an effective coreset selection strategy for continual learning that addresses the interference within successive selection steps. By dissecting the interactions between consecutive rounds of influence-based selection, a new class of second-order influences is identified, with which the long selection process may gradually lose diversity and accumulate bias. To solve this problem, a novel regularizer is presented which is linked to two other popular criteria but incorporates additional Hessian-related information. Finally, we implement the proposed selection criterion with high efficiency and validate its effectiveness in a variety of comparative experiments.

References

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *NeurIPS*, pages 11849–11860, 2019. 6, 7, 13

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, pages 3366–3375, 2017. 2

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, pages 11817–11826, 2019. 1, 2, 3, 5, 6, 7, 8, 13

[4] Lukas Balles, Giovanni Zappella, and Cédric Archambeau. Gradient-matching coresets for continual learning. In *NeurIPS Workshops*, 2021. 2, 5

[5] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, pages 8218–8227, 2021. 2, 5

[6] Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In *ICML*, pages 715–724, 2020. 2, 4, 8, 12

[7] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In *NeurIPS*, pages 14879–14890, 2020. 1, 2, 3, 5, 13

[8] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended DER-verse. *TPAMI*, pages 1–16, 2022. 6

[9] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, pages 15920–15930, 2020. 6

[10] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *ICPR*, pages 2180–2187, 2021. 6

[11] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *ICLR*, 2022. 13

[12] Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI*, pages 6993–7001, 2021. 6, 7

[13] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 6, 7, 8, 13

[14] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. In *ICML Workshops*, 2019. 2, 6, 7, 8, 11, 13

[15] Hongge Chen, Si Si, Yang Li, Ciprian Chelba, Sanjiv Kumar, Duane Boning, and Cho-Jui Hsieh. Multi-stage influence function. In *NeurIPS*, pages 12732–12742, 2020. 2

[16] Zhiyuan Chen and Bing Liu. *Lifelong Machine Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2018. 1

[17] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982. 3

[18] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, 44(7):3366–3385, 2021. 1

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[20] Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *EMNLP*, pages 10333–10350, 2021. 2

[21] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2020. 2

[22] Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974. 2

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[24] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *NeurIPS*, pages 8580–8589, 2018. 2, 5

[25] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. In *NeurIPS*, pages 29193–29205, 2021. 6, 7

[26] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 8

[27] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *ICML*, pages 5464–5474, 2021. 5

[28] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 2, 13

[29] Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. In *NeurIPS*, pages 5254–5264, 2019. 2, 12

[30] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, pages 1885–1894, 2017. 2, 3, 5, 8, 11

[31] Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence-based data relabeling. In *ICLR*, 2022. 2

[32] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009. 6

[33] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017. 2

[34] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, page 6470–6479, 2017. 1, 2, 6, 7

[35] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. 2

[36] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. In *NeurIPS*, pages 7308–7320, 2020. 13

[37] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *ICLR*, 2020. 6

[38] Seulki Park, Jongin Lim, Younghan Jeon, and Jin Young Choi. Influence-balanced loss for imbalanced visual classification. In *ICCV*, pages 735–744, 2021. 2

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 6

[40] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. GDumb: A simple approach that questions our progress in continual learning. In *ECCV*, pages 524–540, 2020. 2, 6, 7

[41] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 1, 2, 13

[42] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019. 2

[43] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[44] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *AAAI*, pages 8179–8186, 2022. 2

[45] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4548–4557, 2018. 2

[46] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, page 2994–3003, 2017. 2

[47] Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring example influence in continual learning. In *NeurIPS*, 2022. 2, 5, 6, 7, 8

[48] Daniel Ting and Eric Brochu. Optimal subsampling with influence functions. In *NeurIPS*, pages 3654–3663, 2018. 2

[49] Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. GCR: Gradient coreset based replay buffer selection for continual learning. In *CVPR*, pages 99–108, 2022. 1, 2, 13

[50] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, pages 3637–3645, 2016. 6

[51] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985. 2

[52] Tianyang Wang, Jun Huan, and Bo Li. Data dropout: Optimizing training data for convolutional neural networks. In *IEEE International Conference on Tools with Artificial Intelligence*, pages 39–46, 2018. 3

[53] Zifeng Wang, Hong Zhu, Zhenhua Dong, Xiuqiang He, and Shao-Lun Huang. Less is better: Unweighted data subsampling via influence function. In *AAAI*, pages 6340–6347, 2020. 2, 3

[54] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 13

[55] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *ICLR*, 2022. 1, 2, 3, 5, 13

[56] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018. 2

[57] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019. 2

[58] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017. 6

[59] Rui Zhang and Shihua Zhang. Rethinking influence functions of neural networks in the over-parameterized regime. In *AAAI*, pages 9082–9090, 2022. 2

[60] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021. 5

[61] Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *AAAI*, pages 4714–4722, 2021. 2, 3

A. Notation

Table 5 summarizes the used notation for quick lookup.

B. Continual learning framework

The pseudocode for our learning procedure is presented in Alg. 1. Following ER [14], the model is trained on a mini-batch composed of the current task data and replay examples at each time step. Meanwhile, to reduce the computational cost imposed by the selection algorithm, the replay buffer is updated only in the last epoch of each task. For the settings of hyperparameters, please refer to Sec. 4.1.

C. Derivation of influence functions

As a background introduction, this section provides the derivation of the first-order influence score $\mathcal{I}(z)$ in Eq. (3), following the idea by Koh and Liang [30].

It begins with upweighting an interested sample z by an infinitesimal amount ϵ, after which the perturbed optimal point $\hat{\theta}_{\epsilon,z}$ can be written as follows:

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta} \sum_{z_i \in \mathcal{C}_t} L(z_i, \theta) + \epsilon L(z, \theta). \quad (17)$$

Its first-order optimality condition states that:

$$0 = \sum_{z_i \in \mathcal{C}_t} \nabla_\theta L(z_i, \hat{\theta}_{\epsilon,z}) + \epsilon \nabla_\theta L(z, \hat{\theta}_{\epsilon,z}). \quad (18)$$

To exploit the known optimal point $\hat{\theta}_t$, we apply the first-order Taylor expansion on the right-hand side:

$$0 \approx \left[\sum_{z_i \in \mathcal{C}_t} \nabla_\theta L(z_i, \hat{\theta}_t) + \epsilon \nabla_\theta L(z, \hat{\theta}_t) \right] \\ + \left[\sum_{z_i \in \mathcal{C}_t} \nabla_\theta^2 L(z_i, \hat{\theta}_t) + \epsilon \nabla_\theta^2 L(z, \hat{\theta}_t) \right] (\hat{\theta}_{\epsilon,z} - \hat{\theta}_t), \quad (19)$$

where $o(\|\hat{\theta}_{\epsilon,z} - \hat{\theta}_t\|)$ terms are dropped. It is also assumed that L is twice-differentiable and convex in θ. Using the optimality condition $\sum_{z_i \in \mathcal{C}_t} \nabla_\theta L(z_i, \hat{\theta}_t) = 0$ and the notation $H_{\hat{\theta}_t} = \sum_{z_i \in \mathcal{C}_t} \nabla_\theta^2 L(z_i, \hat{\theta}_t)$, it can be simplified to:

$$\hat{\theta}_{\epsilon,z} - \hat{\theta}_t \approx H_{\hat{\theta}_t}^{-1} \nabla_\theta L(z, \hat{\theta}_t) \epsilon, \quad (20)$$

where $o(\epsilon)$ terms are neglected. This yields the derivate of $\hat{\theta}_{\epsilon,z}$ w.r.t. ϵ:

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\bigg|_{\epsilon=0} = -H_{\hat{\theta}_t}^{-1} \nabla_\theta L(z, \hat{\theta}_t). \quad (21)$$

Finally, the influence of a particular sample z on the test

Symbol	Description
$\mathcal{Z}_t$	Available data at the t-th step
$\mathcal{Z}_{1:t}$	Seen data till the t-th step
$\mathcal{C}_t$	Coreset at the t-th step
m	Maximum coreset size
$L(z, \theta)$	Loss of parameter θ on sample z
$\hat{\theta}_t$	Optimal point at the t-th step
$\hat{\theta}_{\epsilon,z}$	Optimal point after z is upweighted by ϵ
$H_{\hat{\theta}_t}$	Hessian of $\hat{\theta}_t$ on coreset $\mathcal{C}_t$
$H_{\hat{\theta}_t,z}$	Hessian of $\hat{\theta}_t$ on sample z
s_t	Inverse Hessian-vector product at the t-th step
$\mathcal{I}(z)$	Influence of z on the test loss
$\mathcal{I}_{\epsilon,z}(z')$	Influence of z' after z is upweighted by ϵ
$\mathcal{I}^{(2)}(z, z')$	Second-order influence of z and z'
$\Delta I(z')$	Total interference on the influence of z'
$\mathcal{R}(\cdot)$	Our proposed regularizer

Table 5. Notation in the main paper.

Algorithm 1 Learning Procedure for Task T

1: **Input:** Dataset $\mathcal{Z}$ of task T, coreset $\mathcal{C}_{t-1}$ from the last round of selection, the number of epochs $e_{\max}$, model parameter θ, learning rate η.
2: **for** $e = 1$ **to** $e_{\max}$ **do**
3:     **for** each batch $\mathcal{Z}_t \in \mathcal{Z}$ **do**
4:         Sample a replay batch $\mathcal{B}_\mathcal{C} \in \mathcal{C}_{t-1}$
5:         $\theta \leftarrow \theta - \eta \nabla_\theta \sum_{z \in \mathcal{Z}_t \cup \mathcal{B}_\mathcal{C}} L(z, \theta)$
6:         **if** $e = e_{\max}$ **then**
7:             Update coreset $\mathcal{C}_t \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t$ by Sec. 3.6
8:             $t \leftarrow t + 1$

---

loss can be computed by the chain rule:

$$\begin{aligned}\mathcal{I}(z) &= \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \frac{dL(z_i, \hat{\theta}_{\epsilon,z})}{d\epsilon}\bigg|_{\epsilon=0} \\ &= \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z_i, \hat{\theta}_t)^\top \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\bigg|_{\epsilon=0} \\ &= -\sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z_i, \hat{\theta}_t)^\top H_{\hat{\theta}_t}^{-1} \nabla_\theta L(z, \hat{\theta}_t).\end{aligned} \quad (22)$$

D. Derivation of the second-order influence

This section explains the derivation of the second-order effects $\mathcal{I}^{(2)}(z, z')$ in Eq. (8) of Sec. 3.3. The derivation applies to Eq. (6) as well, since they share a similar form.

In that case, the influence score of a subsequent sample z' after the previous z is upweighted by ϵ is as follows:

$$\mathcal{I}_{\epsilon,z}(z') = -\left(\sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1}) + \epsilon \nabla_\theta L(z, \hat{\theta}_{t+1}) \right)^\top \\ \left(H_{\hat{\theta}_{t+1}} + \epsilon H_{\hat{\theta}_{t+1},z} \right)^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}). \quad (23)$$

Figure 5. Illustration of two consecutive selection steps based on influence functions. The latter selection turns out to be non-ideal, as evidenced by its decision boundary (between high and low density regions indicated by color intensity) being rotated under the interference of the previous step on gradient information.

The inverse matrix therein can be effectively approximated with a Neumann series as $\epsilon \to 0$:

$$
\begin{aligned}
(A + \epsilon B)^{-1} &= A^{-1}(I + \epsilon BA^{-1})^{-1} \\
&= A^{-1}\sum_{k=0}^{\infty}(-\epsilon BA^{-1})^k \\
&= A^{-1} - \epsilon A^{-1}BA^{-1} + o(\epsilon).
\end{aligned}
\tag{24}
$$

Take $A = H_{\hat{\theta}_{t+1}}$ and $B = H_{\hat{\theta}_{t+1},z}$ and substitute into Eq. (23), then we get:

$$
\begin{aligned}
\mathcal{I}_{\epsilon,z}(z') = -&\left(\sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1}) + \epsilon \nabla_\theta L(z, \hat{\theta}_{t+1})\right)^\top \\
&\left(H_{\hat{\theta}_{t+1}}^{-1} - \epsilon H_{\hat{\theta}_{t+1}}^{-1} H_{\hat{\theta}_{t+1},z} H_{\hat{\theta}_{t+1}}^{-1} + o(\epsilon)\right)\nabla_\theta L(z', \hat{\theta}_{t+1}),
\end{aligned}
\tag{25}
$$

which can be further rearranged into:

$$
\begin{aligned}
\mathcal{I}_{\epsilon,z}(z') = -&\sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1})^\top H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}) \\
+&\epsilon \sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1})^\top H_{\hat{\theta}_{t+1}}^{-1} H_{\hat{\theta}_{t+1},z} H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}) \\
-&\epsilon \nabla_\theta L(z, \hat{\theta}_{t+1})^\top H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}) \\
+&o(\epsilon).
\end{aligned}
\tag{26}
$$

With notation $s_{t+1} = H_{\hat{\theta}_{t+1}}^{-1} \sum_{z_i \in \mathcal{C}_t \cup \mathcal{Z}_{t+1}} \nabla_\theta L(z_i, \hat{\theta}_{t+1})$, its derivative w.r.t. ϵ can be written as:

$$
\begin{aligned}
\mathcal{I}^{(2)}(z, z') &= \frac{d\mathcal{I}_{\epsilon,z}(z')}{d\epsilon}\bigg|_{\epsilon=0} \\
&= -(\nabla_\theta L(z, \hat{\theta}_{t+1}) - H_{\hat{\theta}_{t+1},z} s_{t+1})^\top H_{\hat{\theta}_{t+1}}^{-1} \nabla_\theta L(z', \hat{\theta}_{t+1}).
\end{aligned}
\tag{27}
$$

E. Intuition behind the deviration

To illustrate the physical meaning behind the equations, this section presents Figure 5 as an intuitive example of the second-order effects on sample selection.

It is depicted that after two rounds of selection, the samples are more concentrated in the upper right corner. On a closer look, the prior selection alters the overall gradient, thereby distorting the next selection boundary which is inherently orthogonal to the gradient (by the inner product defined in Sec. 3.3. The final result is thus biased and less diversified.

The illustrated example, which focuses on the drift of decision boundary due to the deviation in coreset gradient, is characterized by our first case of second-order influences in Eq. (6). Complementarily, the disturbance to Hessian-related information is tackled in the second case of Eq. (8).

F. Comparison with group influences

Our second-order influences have a different origin from the group influences proposed by Basu *et al*. [6]. The group effects [6,29] in their work arise from the interaction within a group of reweighted datapoints on the inner objective, so they are limited to jointly optimized samples. Our second-order terms, derived from separate analyses of inner and outer objectives, in contrast, have no such restrictions and apply to sequentially incoming data.

G. Connection to diversity

This section presents an algebraic view of the connection between our regularizer and gradient diversity, as a complement to the geometric perspective in Sec. 3.5.

Let $\mathcal{R}^o(\mathcal{C}_t)$ and $\mathcal{R}^i(\mathcal{C}_t)$ denote the regularizers under the $\mu = 0$ and identical Hessian settings, respectively. They are expressed as:

$$
\begin{aligned}
\mathcal{R}^o(\mathcal{C}_t) &= \left\|\sum_{z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z, \hat{\theta}_t) - \sum_{z \in \mathcal{C}_t} \nabla_\theta L(z, \hat{\theta}_t)\right\|, \\
\mathcal{R}^i(\mathcal{C}_t) &= \left\|(1 - \alpha\mu)\sum_{z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z, \hat{\theta}_t) - \sum_{z \in \mathcal{C}_t} \nabla_\theta L(z, \hat{\theta}_t)\right\|.
\end{aligned}
\tag{28}
$$

where α is a coefficient related only to the coreset size. The comparison of the two regularizers yields:

$$
\begin{aligned}
\mathcal{R}^i(\mathcal{C}_t)^2 - \mathcal{R}^o(\mathcal{C}_t)^2 = \underbrace{(-2\alpha\mu + \alpha^2\mu^2)\left\|\sum_{z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z, \hat{\theta}_t)\right\|^2}_{\text{constant}} \\
+ \underbrace{2\alpha\mu\left(\sum_{z \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \nabla_\theta L(z, \hat{\theta}_t)\right)^\top\left(\sum_{z \in \mathcal{C}_t} \nabla_\theta L(z, \hat{\theta}_t)\right)}_{\text{diversity}},
\end{aligned}
\tag{29}
$$

in which the latter term enforces the coreset gradient to be less aligned with the main gradient. Thus, the regularizer $\mathcal{R}^i(\mathcal{C}_t)$ additionally encourages the inclusion of gradients in other directions and promotes gradient diversity.

Method	Class-incremental	Task-incremental
Grad matching	39.56±1.52 ●	88.98±0.95 ●
Grad diversity	43.94±2.03 ●	87.82±1.38 ●
Vanilla IF	47.09±0.85 ●	90.78±1.21
Ours	**52.81±1.26**	**92.43±1.11**

Table 6. Comparison with only gradient regularization, in terms of ACC (%) on Split CIFAR-10 with $m = 500$. ● indicates significant improvement with p-value less than 0.05 in paired t-tests.

H. Taylor expansion of the regularizer

To optimize the new equivalent form of our regularizer in Eq. (15), we perform a first-order Taylor expansion near the initial weight $w_{t,i}^o$:

$$\mathcal{R}(w_t) \approx \mathcal{R}(w_t^o) - \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \beta^T (\nabla_\theta L(z_i, \hat{\theta}_t) - \mu H_{\hat{\theta}_t, z_i} s_t)(w_{t,i} - w_{t,i}^o),$$
(30)

where β is a vector independent of $w_{t,i}$:

$$\beta = \sum_{z_i \in \mathcal{C}_{t-1} \cup \mathcal{Z}_t} \frac{(1 - w_{t,i}^o)(\nabla_\theta L(z_i, \hat{\theta}_t) - \mu H_{\hat{\theta}_t, z_i} s_t)}{\mathcal{R}(w_t^o)}. \quad (31)$$

The result is a linear combination of $w_{t,i}$, and thus can be minimized with greedy heuristics, *i.e.*, by iteratively setting the $w_{t,i}$ with the largest coefficient to zero.

I. Additional results

Time cost with Hessian-vector product. The overhead in evaluating the Hessian-vector product is 0.014±0.001 seconds per step on Split CIFAR-10. This is fairly small compared to the base cost of 0.368±0.029 seconds per step for computing first-order influence functions.

Comparison with only gradient regularization. Combination of memory replay with gradient regularization based approaches can partly bypass the interference issue. However, it lacks efficiency in buffering the most critical samples for performance preservation. We verify this point through the comparisons in Table 6, which empirically justifies the motivation of our proposed influence-based scheme.

Comparison with multi-epoch competitors. Additional comparisons with the classical multi-epoch methods iCaRL [41] and BiC [54] are given in Table 7, which confirm the edge of our method in 50-epoch learning. Results are presented with standard deviations.

In combination with ER-ACE. Table 7 further tests our strategy on the more advanced replay framework ER-ACE [11] instead of the previously adopted ER [14]. It is

Method	Class-incremental	Task-incremental
iCaRL [41]	47.87±0.47 ●	90.35±1.13
BiC [54]	51.49±1.37	90.99±0.78
Ours	**52.81±1.26**	**92.43±1.11**
ER-ACE [11]	56.86±0.64 ●	89.59±3.23
ER-ACE + Ours	**60.57±0.93**	**91.84±0.71**

Table 7. Comparison with multi-epoch methods and ER variant in 50-epoch learning. Detailed settings follow Table 6.

Method	Split CIFAR-100		Multiple Datasets	
	ACC (%)	BWT	ACC (%)	BWT
iCaRL [41]	60.3	-0.04	-	-
EWC [28]	49.5	-0.48	42.7	-0.28
A-GEM [13]	50.7	-0.19	-	-
ER [14]	46.9	-0.21	-	-
GSS [3]	59.7	-0.04	60.2	-0.07
ER-MIR [1]	60.2	-0.04	56.9	-0.11
Stable SGD [36]	57.4	-0.07	53.4	-0.16
Bilevel [7]	60.1	-0.04	58.1	-0.08
OCS [55]	60.5	-0.04	61.5	**-0.03**
GCR [49]	60.9	-	-	-
Vanilla IF	60.0	-0.05	59.7	-0.07
Ours	**61.2**	**-0.04**	**61.6**	-0.05

Table 8. Comparison with another group of baseline methods in task-incremental evaluations. The results of most methods come from the summary in OCS [55], while the result of GCR [49] is provided in its supplementary material.

observed that the proposed method combines well with ER-ACE and yields a 3.71% gain in class-incremental learning.

Additional comparison. To compare with other replay-based competitors such as OCS [55], GCR [55] and Bilevel [7], as well as some regularization-based methods such as Stable SGD [36] and EWC [28], we reimplement our approach using the codebase of OCS. Its framework differs in mainly two aspects: (1) Methods are evaluated on two task-incremental benchmarks, including 20-split CIFAR-100 and a mixture of five datasets from different domains. (2) Each learning stage features much fewer training epochs, so the resulting ACC will be lower than before, while the forgetting metric BWT will be much better.

As shown in Tab. 8, our approach continues to deliver considerable improvement over the base strategy Vanilla IF. Like many replay-based methods, we outperform regularization-based methods by a large margin. Furthermore, our method surpasses the top two competitors OCS and GCR in terms of ACC on both benchmarks. These results again demonstrate the effectiveness of our approach.