Exploring Example Influence in Continual Learning

Qing Sun*, **Fan Lyu***, **Fanhua Shang**, **Wei Feng**, **Liang Wan**†
College of Intelligence and Computing, Tianjin University
{sssunqing, fanlyu, fhshang, wfeng, lwan}@tju.edu.cn
https://github.com/SSSunQing/Example_Influence_CL

Abstract

Continual Learning (CL) sequentially learns new tasks like human beings, with the goal to achieve better Stability (S, remembering past tasks) and Plasticity (P, adapting to new tasks). Due to the fact that past training data is not available, it is valuable to explore the influence difference on S and P among training examples, which may improve the learning pattern towards better SP. Inspired by Influence Function (IF), we first study example influence via adding perturbation to example weight and computing the influence derivation. To avoid the storage and calculation burden of Hessian inverse in neural networks, we propose a simple yet effective MetaSP algorithm to simulate the two key steps in the computation of IF and obtain the S- and P-aware example influence. Moreover, we propose to fuse two kinds of example influence by solving a dual-objective optimization problem, and obtain a fused influence towards SP Pareto optimality. The fused influence can be used to control the update of model and optimize the storage of rehearsal. Empirical results show that our algorithm significantly outperforms state-of-the-art methods on both task- and class-incremental benchmark CL datasets.

1 Introduction

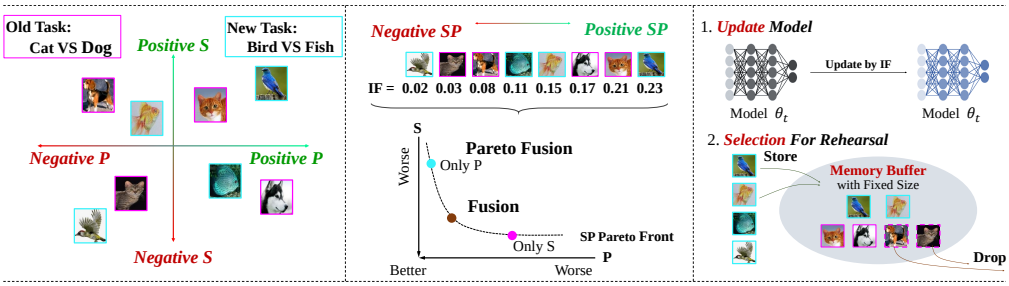
By mimicking human-like learning, Continual Learning (CL) aims to enable a model to continuously learn from novel knowledge (new tasks, new classes, etc.) in a sequential order. The major challenge in CL is to harness catastrophic forgetting and knowledge transition, namely the Stability-Plasticity dilemma, with Stability (S) showing the ability to prevent performance drops for old tasks and Plasticity (P) referring if the new task can be learned rapidly and unimpededly. Intuitively speaking, a robust CL system should achieve outstanding S and P through sequential learning.

The sequential paradigm means CL does not access past training data. Comparing to traditional machine learning, the training data in CL is thus more precious. It is valuable to explore the influence difference on S and P among training examples. Following the accredited influence chain "`Data-Model-Performance`", exploring this difference is equivalent to tracing from performance back to example difference. With appropriate control, this may improve the learning pattern towards better SP. On top of this, *the goal of this paper is to explore the reasonable influence from each training example to SP, and apply the example influence to CL training*.

To understand example influence, one classic successful technique is the Influence Function (IF) [20], which leverages the derivation chain rule from a test objective to training examples. However, directly applying the chain rule leads to computing the inverse of Hessian with the complexity of $O(nq^2 + q^3)$ (n is the number of examples and q is parameter size), which is computationally intensive and may run out-of-memory in neural networks. In this paper, we propose a novel meta-learning algorithm, called MetaSP, to compute example influence via simulating IF. We design based on the rehearsal-based

*Co-first authors.
†Corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) Influence Acquirement (b) Influence Fusion (c) Influence Utilization

Figure 1: Training examples have different influences on Stability and Plasticity. Given an old task with classes "cat" and "dog" and a new task with classes "Bird" and "Fish", we compute the influence on S and P for each example. Then, we fuse the two kinds of influence towards SP Pareto front. We also show that example influence can be used to adjust model update and optimize rehearsal selection.

CL framework, which avoids forgetting via retraining a part of old data. First, a pseudo update is held with example-level perturbations. Then, two validation sets sampled from seen data are used to compute the gradients on example perturbations. The gradients are regarded as the example influence on S and P. As shown in Fig. 1(a), examples can be distinguished by the value of influence on S and P.

To leverage the two independent kinds of influence in CL, we need to take full account of the influence on both S and P. However, the influence on S and P may interfere with each other, which leads us to make a trade-off. This can be seen as a Dual-Objective Optimization (DOO) problem, which aims to find solutions not dominated (no other better solution) by any other one, i.e. Pareto optimal solutions [8]. We say the solutions as the example influence on SP. Following the gradient-based MGDA algorithm [12], we obtain the fused example influence on SP by meeting the Karush-Kuhn-Tucker (KKT) condition, as illustrated in Figure 1(b).

Finally, we show that the fused influence can be used to control the update of model and optimize the storage of rehearsal in Figure 1(c). On one hand, the fused influence can be directly used to control the magnitude of training loss for each example. On the other hand, under a fixed memory budget, the fused influence can be used to select appropriate examples storing and dropping, which keeps the rehearsal memory always have larger positive influence on SP.

In summary, our contributions are four-fold: 1) Inspired by the influence function, we study CL from the perspective of example difference and propose MetaSP to compute the example influence on S and P. 2) We propose to trade off S and P influence via solving a DOO problem and fuse them towards SP Pareto optimal. 3) We leverage the fused influence to control model update and optimize the storage of rehearsal. 4) The verification contribution: by considering the example influence, in our experiments on both task- and class-incremental CL, better S and more stable P can be observed.

2 Related Work

Continual Learning. Due to many researchers' efforts, lots of methods for CL have been proposed, which can be classified into three categories. The regularization-based methods [19, 11, 10] are based on regularizing the parameters corresponding to the old tasks and penalizing the feature drift. The parameter isolation based methods [14, 26] generate task-specific parameter expansion or sub-branch. Rehearsal-based methods [29, 7, 22, 6, 1, 2, 28, 3, 25] tackle the challenge of SP dilemma by retaining a subset of old tasks in a stored memory buffer with bounded resources. Although the existing methods apply themselves to achieve better SP, they fail to explore what contributes to the Stability and Plasticity inside the training data. In this work, we explore the problem in the perspective of example difference, where we argue that each example contributes differently to the SP. We focus our work on the rehearsal-based CL framework in order to omit the divergence between models, while evaluating the old data's influence simultaneously.

Example Influence. In recent years, as the impressive Interpretable Machine Learning (IML) [27] develops, people realize the importance of exploring the nature of data-driven machine learning. Examples are different, even they belong to the same distribution. Because of such difference, the example contributes differently to the learning pattern. In other words, the influence acquired in advance from different training examples can significantly improve the CL training. Some studies

propose a similar idea and use the influences to reweight or dropout the training data [31, 13, 36]. In contrast to complicated model design, a model-agnostic algorithm estimates the training example influence via computing the derivation from a test loss to a training data weight. One typical example method is the Influence Function [20], which leverages a pure second-order derivation (Hessian) with the chain rule. In this paper, to avoid the expensive computation of Hessian inverse, we design a meta learning [18] based method, which can be used to control the training.

3 Demystifying Example Influence on SP

3.1 Preliminary: Rehearsal-based CL

Given T different tasks w.r.t. datasets $\{\mathcal{D}_1, \cdots, \mathcal{D}_T\}$, Continual Learning (CL) seeks to learn them in sequence. For the t-th dataset (task), $\mathcal{D}_t = \{(x_t^{(n)}, y_t^{(n)})\}_{n=1}^{N_t}$ is split into a training set $\mathcal{D}_t^{\text{trn}}$ and a test set $\mathcal{D}_t^{\text{tst}}$, where N_t is the number of examples. At any time, CL aims at learning a multi-task/multi-class predictor to predict tasks/classes that have been learned (say task-incremental and class-incremental CL). To suppress the catastrophic forgetting, the rehearsal-based CL [30, 22, 32, 6, 16] builds a small size memory buffer $\mathcal{M}_t$ sampled from $\mathcal{D}_t^{\text{trn}}$ for each task (*i.e.*, $|\mathcal{M}_t| \ll |\mathcal{D}_t^{\text{trn}}|$). At training phase, the data in the whole memory $\mathcal{M} = \cup_{k<t}\mathcal{M}_k$ will be retrained together with the current tasks. Accordingly, a mini-batch training step of task t in rehearsal-based CL is denoted as

$$\min_{\boldsymbol{\theta}_t} \quad \ell(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t), \quad \mathcal{B}_{\text{old}} \subset \mathcal{M} \text{ and } \mathcal{B}_{\text{new}} \subset \mathcal{D}_t^{\text{trn}}, \tag{1}$$

where ℓ is the empirical loss. $\boldsymbol{\theta}_t$ is the trainable parameters at task t and is updated from scratch.

3.2 Example Influence on Stability and Plasticity

Definition 1 (Stability and Plasticity) *Suppose the parameter of a model is initialized to $\boldsymbol{\theta}_0$. At the training on the t-th task, given test sets of an old task $\mathcal{D}_k^{\text{tst}}(k < t)$ and the current task $\mathcal{D}_t^{\text{tst}}$, the Stability S_t^k and Plasticity P_t can be evaluated by:*

$$S_t^k = p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k), \quad P_t = p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_{t-1}),$$

where $p(\mathcal{D}_1|\boldsymbol{\theta}, \mathcal{D}_2)$ represents the performance (accuracy in classification) of $\mathcal{D}_1$ conditioned to the model $\boldsymbol{\theta}$ training on $\mathcal{D}_2$. $p(\mathcal{D}|\boldsymbol{\theta})$ denotes the performance of $\mathcal{D}$ tested on the model $\boldsymbol{\theta}$.

The S of a task is evaluated by the performance difference on the test set after training on any later tasks, which is also known as Forgetting [6]. The P of a task is defined as the ability to integrate new knowledge, which is regarded as the test performance of this task. As many existing CL methods demonstrate, the SP inevitably interferes mutually.

Definition 2 (Example Influence on SP) *At the training on the t-th task, with a sampled example $x^{\text{trn}} \in \mathcal{D}_t^{\text{trn}}$, the example influence from x^{trn} to Stability S_t^k and Plasticity P_t for $k < t$ can be evaluated by the gap from deleting it then retraining the model:*

$$I_S(\mathcal{D}_k^{\text{tst}}, x^{\text{trn}}) = p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}/x^{\text{trn}}),$$
$$I_P(\mathcal{D}_t^{\text{tst}}, x^{\text{trn}}) = p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}/x^{\text{trn}}),$$

where $\mathcal{D}_t^{\text{trn}}/x^{\text{trn}}$ denotes the dataset $\mathcal{D}_t^{\text{trn}}$ without the training example x^{trn}.

However, deleting every example to compute full influences is impractical due to the highly computational cost. Instead, the performance change can be indicated by the loss change, which leads to a derivable way to approximate the influence:

$$I_S(\mathcal{D}_k^{\text{tst}}, x^{\text{trn}}) \stackrel{\text{def}}{=} \frac{\partial \ell(\mathcal{D}_k^{\text{tst}})}{\partial \epsilon}, \quad I_P(\mathcal{D}_t^{\text{tst}}, x^{\text{trn}}) \stackrel{\text{def}}{=} \frac{\partial \ell(\mathcal{D}_t^{\text{tst}})}{\partial \epsilon}, \tag{2}$$

where ϵ is the weight perturbation to the training example and $\stackrel{\text{def}}{=}$ means define. This influence can be computed by the Influence Function [20] that will be introduced in the next section.

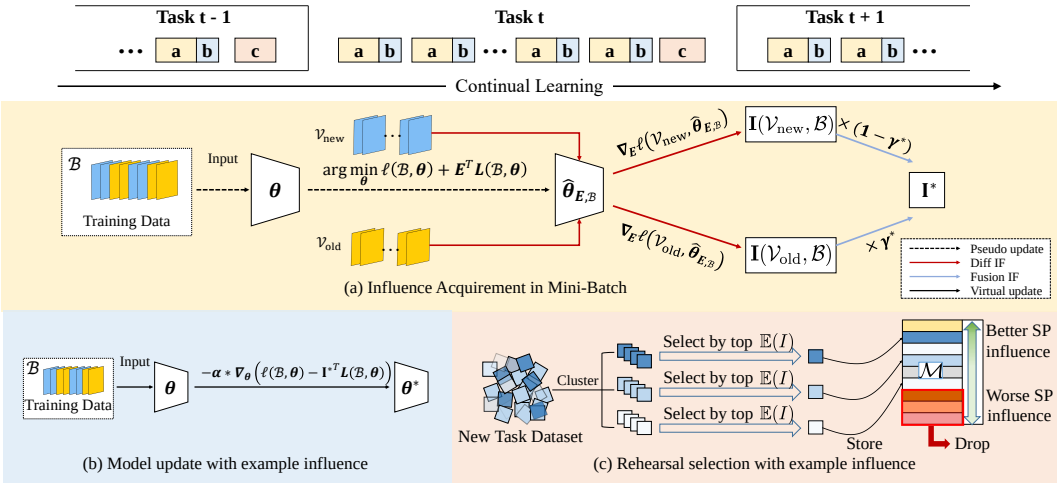


Figure 2: Evaluating and making use of example influence in mini-batch Continual Learning. (a) At each iteration in CL training, MetaSP updates in pseudo and use two validation sets representing old tasks and new task to obtain the example influence on S and P. The two kinds of influence are fused towards a Pareto optimal. (b) The computed influence can be directly used to update CL model and (c) select examples for rehearsal storing and dropping.

4 Meta Learning on Stability and Plasticity

4.1 Influence Function for SP

A mini-batch, $\mathcal{B}$, from the training data is sampled, and the normal model update is

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right). \tag{3}$$

In Influence Function (IF) [20], a small weight perturbation ϵ is added to the training example $x^{\text{trn}} \in \mathcal{B}$

$$\hat{\boldsymbol{\theta}}_{\epsilon,x} = \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right) + \epsilon\ell(x^{\text{trn}}, \boldsymbol{\theta}), \quad x^{\text{trn}} \in \mathcal{B}. \tag{4}$$

We can easily promote this to the mini-batch

$$\hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}} = \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right) + \mathbf{E}^{\top}\mathbf{L}(\mathcal{B}, \boldsymbol{\theta}), \tag{5}$$

where $\mathbf{L}$ denotes the loss vector for a mini-batch and $\mathbf{E} \in \mathbb{R}^{|\mathcal{B}|\times 1}$ denotes the perturbation on each example in it. It is easy to know that the example influence $I(\mathcal{D}^{\text{tst}}, \mathcal{B})$ is reflected in the derivative $\nabla_{\mathbf{E}}\ell(\mathcal{D}^{\text{tst}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},x})\big|_{\mathbf{E}=\mathbf{0}}$. By the chain rule, the example influence in IF can be computed by

$$\mathbf{I}(\mathcal{D}^{\text{tst}}, \mathcal{B}) \stackrel{\text{def}}{=} \nabla_{\mathbf{E}}\ell(\mathcal{D}^{\text{tst}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},x})\big|_{\mathbf{E}=\mathbf{0}} = -\nabla_{\boldsymbol{\theta}}\ell(\mathcal{D}^{\text{tst}}, \hat{\boldsymbol{\theta}})\mathbf{H}^{-1}\nabla_{\boldsymbol{\theta}}^{\top}\mathbf{L}(\mathcal{B}, \hat{\boldsymbol{\theta}}), \tag{6}$$

where $\mathbf{H} = \nabla_{\boldsymbol{\theta}}^2\ell(\mathcal{B}, \hat{\boldsymbol{\theta}})$ is a Hessian. Unfortunately, the inverse of Hessian requires the complexity $O(|\mathcal{B}|q^2 + q^3)$ and huge storage for neural networks (maybe out-of-memory), which is challenging for efficient training.

In Eq. (6), we have $\mathbf{I}(\mathcal{D}^{\text{tst}}, \mathcal{B}) = [I(\mathcal{D}^{\text{tst}}, x^{\text{trn}})|x^{\text{trn}} \in \mathcal{B}]$ and find the loss will get larger if $I(\mathcal{D}^{\text{tst}}, x^{\text{trn}}) > 0$, which means the negative influence on the test set $\mathcal{D}^{\text{tst}}$. Similarly, $I(\mathcal{D}^{\text{tst}}, x^{\text{trn}}) < 0$ means the positive influence on the test set $\mathcal{D}^{\text{tst}}$. *Fortunately, the second-order derivation in IF is not necessary under the popular meta learning paradigm such as [18], instead we can easily get the derivation like IF through a one-step pseudo update.* In the following, we will introduce a simple yet effective meta-based method, named MetaSP, to simulate IF at each step with a two-level optimization to avoid computing Hessian inverse.

4.2 Simulating IF for SP

Based on the meta learning paradigm, we transform the example influence computation into solving a meta gradient descent problem, named **MetaSP**. For each training step in a rehearsal-based CL,

Algorithm 1: Computation of Example Influence (**MetaSP**)

Input: $\mathcal{B}_{\text{old}}, \mathcal{B}_{\text{new}}, \mathcal{V}_{\text{old}}, \mathcal{V}_{\text{new}}$; `// Training batches, Validation batches`
Output: $\mathbf{I}^*$; `// Pareto example influence on SP`

$\hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}} = \arg\min_{\boldsymbol{\theta}} \ \ell(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}) + \mathbf{E}^\top \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta})$; `// Pseudo update`
$\mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B}) = \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})$; `// Gradient from old val loss`
$\mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B}) = \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})$; `// Gradient from new val loss`
$\gamma^* \leftarrow$ Eq. (11); `// Optimal fusion hyper-parameter`
$\mathbf{I}^* = \gamma^* \cdot \mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B}) + (1-\gamma^*) \cdot \mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B})$; `// Influence fusion`

we have two mini-batches data $\mathcal{B}_{\text{old}}$ and $\mathcal{B}_{\text{new}}$ in respect to old and new tasks. Our goal is to obtain the influence on S and P from every example in $\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}$. Note that both S-aware and P-aware influence are applied to every example regardless of old or new tasks. That is, the contribution of an example is not deterministic. Data of old tasks may also affect the new task in positive, and vice-versa. In rehearsal-based CL, we turn to computing the derivations $\nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}})|_{\mathbf{E}=0}$ for example influence.

To compute the derivation, as shown in Fig. 2(a), our MetaSP has two key steps:

(1) Pseudo update. This step is to simulate Eq. (5) in IF via a pseudo update

$$\hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}} = \arg\min_{\boldsymbol{\theta}} \quad \ell(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}) + \mathbf{E}^\top \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}), \tag{7}$$

where $\mathbf{L}$ denotes the loss vector for a mini-batch combining both old and new tasks.

(2) Compute example influence. This step computes example influence on S and P for all training examples as simulating Eq. (6). Based on the pseudo updated model in Eq. (7), we compute S- and P-aware example influence via two validation sets $\mathcal{V}_{\text{old}}$ and $\mathcal{V}_{\text{new}}$. Noteworthily, because the test set $\mathcal{D}^{\text{tst}}$ is unavailable at training phase, we use two dynamic validation sets $\mathcal{V}_{\text{old}}$ and $\mathcal{V}_{\text{new}}$ to act as the alternative in the CL training process. One is sampled from the memory buffer ($\mathcal{V}_{\text{old}}$) representing the old tasks, and the other is from the seen training data representing the new task ($\mathcal{V}_{\text{new}}$). With $\mathbf{E}$ initialized to $\mathbf{0}$, the two kinds of example influence are computed as

$$\mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B}) = \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}), \quad \mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B}) = \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}). \tag{8}$$

Generally, each elements in two influence vectors $\mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B})$ and $\mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B})$ represents the example influence on S and P. Similar to IF, elements with positive value mean negative influence while elements with negative value mean positive influence.

5 Using Influence for Continual Learning

5.1 Before Using: Influence for SP Pareto Optimality

As shown in Eq. (8), the example influence is equal to the derivation from validation loss of old and new tasks to the perturbations $\mathbf{E}$. However, the two kinds of influence are independent and interfere with each other. That is, using only one of them may fail the other performance. We prefer to find a solution that makes a trade-off between the influence on both S and P. Thus, we integrate the two influence $\mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B})$ and $\mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B})$ into a DOO problem with two gradients from different objectives.

$$\min_{\mathbf{E}} \quad \left\{ \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}), \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}) \right\}. \tag{9}$$

The goal of Problem (9) is to obtain a fused way that satisfies the SP Pareto optimality.

Definition 3 (SP Pareto Optimality)

*1. (**Pareto Dominate**) Let $\mathbf{E}_a$, $\mathbf{E}_b$ be two solutions for Problem* (9)*, $\mathbf{E}_a$ is said to dominate $\mathbf{E}_b$ ($\mathbf{E}_a \prec \mathbf{E}_b$) if and only if $\ell(\mathcal{V}, \hat{\boldsymbol{\theta}}_{\mathbf{E}_a,\mathcal{B}}) \leq \ell(\mathcal{V}, \hat{\boldsymbol{\theta}}_{\mathbf{E}_b,\mathcal{B}})$, $\forall \mathcal{V} \in \{\mathcal{V}_{\text{old}}, \mathcal{V}_{\text{new}}\}$, and $\ell(\mathcal{V}, \hat{\boldsymbol{\theta}}_{\mathbf{E}_a,\mathcal{B}}) < \ell(\mathcal{V}, \hat{\boldsymbol{\theta}}_{\mathbf{E}_b,\mathcal{B}})$, $\exists \mathcal{V} \in \{\mathcal{V}_{\text{old}}, \mathcal{V}_{\text{new}}\}$.*
*2. (**SP Pareto Optimal**) $\mathbf{E}$ is called SP Pareto optimal if no other solution can have better values in $\ell(\mathcal{V}_{old}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})$ and $\ell(\mathcal{V}_{new}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})$.*

Algorithm 2: Using Example Influence in Rehearsal-based Continual Learning.

Input: Initialized $\boldsymbol{\theta}_0$, Learning rate α, Training set $\{\mathcal{D}_1^{\text{trn}}, \cdots, \mathcal{D}_T^{\text{trn}}\}$, Memory $\mathcal{M}$
Output: $\boldsymbol{\theta}_T$; `// Final model`

for *task* $t = 1 : T$ **do**
  $\boldsymbol{\theta}_t$= `TrainNewTask`($\boldsymbol{\theta}_{t-1}$, $\mathcal{D}_t^{\text{trn}}$, $\mathcal{M}$) (Alg. 3)
  $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{\frac{|\mathcal{M}|}{t}} \leftarrow$ K-Means($\mathcal{D}_t^{\text{trn}}$);
  Rank $\mathcal{C}_i$ with $\mathbb{E}(I^*(x)), x \in \mathcal{C}_i$;
  Rank $\mathcal{M}$ with $\mathbb{E}(I^*(x)), x \in \mathcal{M}$;
  for $i = 1 : \frac{|\mathcal{M}|}{t}$ **do**
    Pop the bottom of $\mathcal{M}$;
    Push the top of $\mathcal{C}_i$ to $\mathcal{M}$;
  end
end

Inspired by the Multiple-Gradient Descent Algorithm (MGDA) [12], we transform Problem (9) to a min-norm problem. Specifically, according to the KKT conditions [15], we have

$$\gamma^* = \arg\min_{\gamma} \left\| \gamma \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}) + (1-\gamma) \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}) \right\|_2^2, \quad s.t., 0 \leq \gamma \leq 1. \tag{10}$$

Referring to the study from Sener *et al.* [34], the optimal γ^* is easily computed as

$$\gamma^* = \min\left(\max\left(\frac{(\nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}) - \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}))^\top \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})}{\|\nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{new}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}) - \nabla_{\mathbf{E}} \ell(\mathcal{V}_{\text{old}}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})\|_2^2}, 0\right), 1\right). \tag{11}$$

Thus, the SP Pareto influence of the training batch can be computed by

$$\mathbf{I}^* = \gamma^* \cdot \mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B}) + (1-\gamma^*) \cdot \mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B}). \tag{12}$$

This process can be seen in Fig. 2(a). Different from the S-aware and P-aware influence, the integrated influence consider the Pareto optimum to both S and P, *i.e.*, reducing the negative influence on S or P and keeping the positive influence on both S and P. Then we will introduce how to leverage example influence in CL training, our algorithm can be seen in Alg. 1.

5.2 Model Update Using Example Influence

With the computed example influence in each mini-batch, we can easily control the model update of this mini-batch to adjust the training towards an ensemble positive direction. Given parameter $\boldsymbol{\theta}$ from the previous iteration the step size α, the model can be updated in traditional SGD as $\boldsymbol{\theta}^* = \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} (\ell(\mathcal{B}, \boldsymbol{\theta}))$, where $\mathcal{B} = \mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}$. By regularizing the update with the example influence $\mathbf{I}^*$, we have

$$\boldsymbol{\theta}^* = \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} \left(\ell(\mathcal{B}, \boldsymbol{\theta}) + (-\mathbf{I}^*)^\top \mathbf{L}(\mathcal{B}, \boldsymbol{\theta}) \right). \tag{13}$$

MetaSP offers regularized updates at every step for rehearsal-based CL, which leads the CL training to better SP but with only the complexity of $O(|\mathcal{B}|q + vq)$ (v denotes the validation size) compared with that of IF, $O(|\mathcal{B}|q^2 + q^3)$.

Algorithm 3: Training New Task

Input: Initialized $\boldsymbol{\theta}_t$, Training set $\mathcal{D}_t^{\text{trn}}$,
  Memory $\mathcal{M}$, Learning rate α
Output: Trained $\boldsymbol{\theta}_t$

for $i = 1$:*ITER_NUM* **do**
  $\mathcal{B}_{\text{new}} \sim \mathcal{D}_t^{\text{trn}}$;
  if $t = 1$ **then**
    $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \alpha \cdot \nabla_{\boldsymbol{\theta}} \ell(\mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t)$;
  else
    $\mathcal{B}_{\text{old}} \sim \mathcal{M}$, $\mathcal{V}_{\text{old}} \sim \mathcal{M}$, $V_{\text{new}} \sim \mathcal{D}_t^{\text{trn}}$;
    $\mathbf{I}^* \leftarrow$METASP($\mathcal{B}_{\text{old}}$, $\mathcal{B}_{\text{new}}$, $\mathcal{V}_{\text{old}}$, $\mathcal{V}_{\text{new}}$);
    $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \alpha \cdot \nabla_{\boldsymbol{\theta}} (\ell(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t)$
      $+ (-\mathbf{I}^*)^\top \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t))$;
  end
end

We show this application in Fig. 2(b). By updating like the above equation, we can make use of the influence of each example to a large extent. In this way, some useless examples are restrained and some positive examples are emphasized, which may improve the acquisition of new knowledge and the maintenance of old knowledge simultaneously.

5.3 Rehearsal Selection Using Example Influence

Rehearsal in fixed budget needs to consider *storing* and *dropping* to keep the memory $\mathcal{M}$ having the core set of all old tasks. In tradition, storing and dropping are both based on randomly example

selection, which ignores the influence difference on SP from each example. Given influence $I^*(x)$ representing contributions from example x to SP, we further design to use it to improve the rehearsal strategy under fixed memory budget. The above example influence on S and P is computed in mini-batch level, we can promote it to the whole dataset according to the law of large numbers, and the influence value for the example x is the value of expectation over batches, *i.e.*, $\mathbb{E}(I^*(x))$.

The fixed-size memory is divided averagely by the seen task number. After task t finishes its training, we conduct our influence-aware rehearsal selection strategy as shown in Fig. 2(c). For storing, we first cluster all training data into $\frac{|\mathcal{M}|}{t}$ groups using K-means to diversify the store data. Each group is ranked by its SP influence value, and the most positive influence on both SP will be selected to store. For dropping, we rank again on the memory buffer via their influence value, and drop the most negative $\frac{|\mathcal{M}|}{t}$ example. In this way, $\mathcal{M}$ always stores diverse examples with positive SP influence.

6 Experiments

6.1 Datasets and implementation details

We use three commonly used benchmarks for evaluation: 1) **Split CIFAR-10** [37] consists of 5 tasks, with 2 distinct classes each and 5000 exemplars per class, deriving from the CIFAR-10 dataset; 2) **Split CIFAR-100** [37] splits the original CIFAR-100 dataset into 10 disjoint subsets, each of which is considered as a separate task with 10 classes; 3) **Split Mini-Imagenet** [35] is a subset of 100 classes from ImageNet [9], rescaled to 32×32. Each class has 600 samples, randomly subdivided into training (80%) and test sets (20%). Mini-Imagenet dataset is equally divided into 5 disjoint tasks.

We employ ResNet-18 [17] as the backbone which is trained *from scratch*. We use Stochastic Gradient Descent (SGD) optimizer and set the batch size 32 unchanged in order to guarantee an equal number of updates. Also, the rehearsal batch sampled from memory buffer is set to 32. We construct the SP validation sets in MetaSP by randomly sampling 10% of the seen data and 10% of the memory buffer at each training step. We set other hyper-settings following ER tricks [4], including 50 total epochs and hyper-parameters. All results are averaged over 5 fixed seeds for fairness.

To better evaluate the CL process, we suggest evaluating SP with four metrics as follows. We use the sign function $\mathbf{1}(\cdot)$ to represent if the prediction of model is equal to the ground truth. 1) **First Accuracy** ($A_1 = \frac{1}{T}\sum_t \sum_{x_i \in \mathcal{D}_t^{\text{tst}}} \mathbf{1}(y_i, \boldsymbol{\theta}_t(x_i))$): For each task, when it is first trained done, we evaluate its testing performance immediately, which indicates the Plasticity, *i.e.*, the capability of learning new knowledge. 2) **Final Accuracy** ($A_\infty = \frac{1}{T}\sum_t \sum_{x_i \in \mathcal{D}_t^{\text{tst}}} \mathbf{1}(y_i, \boldsymbol{\theta}_T(x_i))$): This metric is the final performance for each task, which indicates Stability, *i.e.*, the capability of suppressing catastrophic forgetting. 3) **Mean Average Accuracy** ($A_{\text{m}} = \frac{1}{T}\sum_t \left(\frac{1}{t}\sum_{k \le t} \sum_{x_i \in \mathcal{D}_k^{\text{tst}}} \mathbf{1}(y_i, \boldsymbol{\theta}_t(x_i))\right)$): This metric computes along CL process, indicating the SP performance after each task trained done. 4) **Backward Transfer** ($BWT = \frac{1}{T-1}\sum_{t=1}^{T-1} \sum_{(x,y) \in \mathcal{D}_t^{\text{tst}}} (\mathbf{1}(y, \theta_T(x)) - \mathbf{1}(y, \theta_t(x))) = \frac{T}{T-1}(A_\infty - A_1)$): This metric is the performance drop from first to final accuracy of each task.

6.2 Main Comparison Results

We compare our method against 8 rehearsal-based methods (including GDUMB [28], GEM [22], AGEM [6], HAL [5], GSS [2], MIR [1], GMED [33] and ER [7]). What's more, we also provide a lower bound that train new data directly without any forgetting avoidance strategy (Fine-tune) and an upper bound that is given by all task data through joint training (Joint).

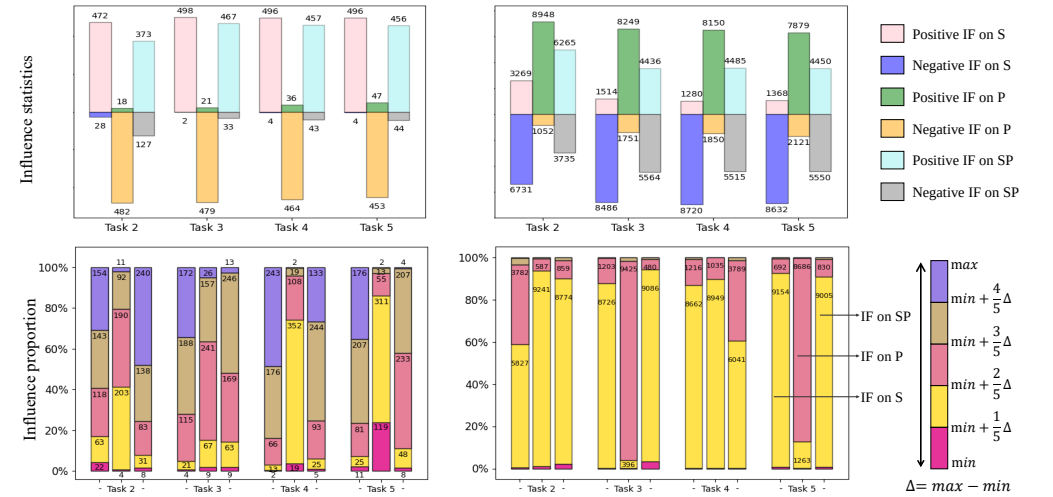
In Table 1, we show the quantitative results of all compared methods and the proposed MetaSP in class-incremental and task-incremental settings. First of all, by controlling the training according to the influence on SP, the proposed MetaSP outperforms other methods on all metrics. With the memory buffer size growth, all the rehearsal-based CL get better performance, while the advantages of MetaSP are more obvious. In terms of the First Accuracy A_1, indicating the ability to learn new tasks, our method outperforms most of the other methods with a little numerical advantage. In terms of the Final Accuracy A_∞, which is used to measure the forgetting, we have an obvious improvement of an average of 3.17 for class-incremental setting and averagely 1.77 for task-incremental setting w.r.t. the second best result. This shows MetaSP can significantly keep stable learning of the new task while suppressing the catastrophic forgetting. It is because although the new tasks may have

Table 1: Comparisons on three datasets, averaged across 5 runs (See std. in the Appendix). Red and blue values mean the best in our methods and the compared methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	CIFAR10 (Class increment)								CIFAR10 (Task increment)							
	buffer size 300				buffer size 500				buffer size 300				buffer size 500			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
	Finetune		19.66		Joint		91.79		Finetune		65.27		*Joint*		98.16	
GDUMB [28]	36.92				44.27				73.22				78.06			
GEM [22]	93.90•	37.51•	55.43•	−70.48	92.76•	36.95•	57.36•	−69.76•	96.62•	89.34•	92.49•	−9.09•	96.73•	90.42	92.93•	−7.88
AGEM [6]	96.57	20.02•	45.57•	−95.68•	96.56•	20.01•	46.52•	−95.69•	96.78•	85.52•	90.16•	−14.07•	96.71•	86.45•	90.90•	−12.83•
HAL [5]	91.30•	24.45•	46.34•	−83.56•	91.96•	27.94•	49.05•	−80.01•	91.41•	79.90•	83.78•	−14.39•	92.03•	81.84•	84.19•	−12.73•
MIR [1]	96.70•	38.53•	56.96•	−72.72•	96.65	42.65•	59.99•	−67.50•	96.76•	88.50•	90.87•	−10.33•	96.73•	90.63•	91.99•	−7.62
GSS [2]	96.53	35.89•	54.33•	−75.80•	96.55	41.96•	58.16•	−68.24•	96.56•	88.05•	90.60•	−10.63•	96.57•	90.38	92.19•	−7.73
GMED [33]	96.65	38.12•	58.92•	−73.16•	96.65	43.68•	62.56•	−66.21•	96.73•	88.91•	91.20•	−9.76•	96.72•	89.72•	92.10•	−8.75
ER [7]	96.73	34.19•	53.72•	−78.18•	96.74	40.45•	57.69•	−70.36•	96.93•	88.97•	91.12•	−9.95•	96.79•	90.60	92.28•	−7.74
Ours	**96.87**	42.42	63.52	−68.05	**96.82**	49.16	67.88	−59.57	97.10	89.40	92.54	−9.62	**97.31**	90.91	**93.38**	−7.99
Ours+RehSel	96.85	**43.76**	**63.69**	**-66.36**	96.81	**50.10**	**68.28**	**-58.38**	**97.11**	**89.91**	**92.66**	**-8.99**	97.30	**91.41**	93.28	**-7.36**

Method	CIFAR100 (Class increment)								CIFAR100 (Task increment)							
	buffer size 500				buffer size 1000				buffer size 500				buffer size 1000			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
	Finetune		9.14		Joint		71.25		Finetune		33.85		Joint		91.63	
GDUMB [28]	11.11				15.75				36.40				43.25			
GEM [22]	85.28•	15.91•	29.38•	−77.07	84.28•	22.79•	34.09•	−68.32	85.53•	68.68•	68.49•	−18.72	85.24•	73.71•	72.59•	−12.81
AGEM [6]	85.97	9.31•	24.60•	−85.18•	85.66	9.27•	24.67•	−84.88•	85.97•	55.28•	58.23•	−34.10•	85.66•	55.95•	59.96•	−33.01•
HAL [5]	67.33•	8.20•	22.72•	−65.70•	68.06•	10.59•	24.74•	−63.86•	67.64•	44.98•	50.79•	−25.17•	68.62•	50.07•	54.01•	−20.61•
MIR [1]	87.38	13.49•	28.88•	−82.09•	87.39	17.56•	32.48•	−77.59•	87.42	66.18•	67.43•	−23.60•	87.50	71.20•	71.42•	−18.10•
GSS [2]	86.03	14.01•	28.00•	−80.02•	86.31	17.87•	31.82•	−76.04•	86.10•	66.80•	66.55•	−21.44	86.44•	71.98•	71.00•	−16.06
GMED [33]	87.18	14.56•	33.41•	−80.68•	87.29	18.67•	38.69•	−76.23•	87.30•	68.82•	72.66	−20.53	87.49	73.91•	76.36	−15.10
ER [7]	87.23	13.75•	28.88•	−81.64•	87.33	17.56•	32.45•	−77.52•	87.29	66.82•	67.56•	−22.73	87.40	71.74•	71.60•	−17.40•
Ours	**88.13**	18.96	38.62	−76.85	**87.58**	24.78	45.20	−69.76	**88.94**	70.03	74.07	−21.01	88.94	75.32	78.09	−15.14
Ours+RehSel	87.81	**19.28**	**39.23**	**-76.13**	87.55	**25.72**	**45.48**	**-68.69**	88.58	**70.81**	**74.24**	**-19.74**	**89.03**	**76.14**	**78.27**	**-14.32**

Method	Mini-Imagenet (Class increment)								Mini-Imagenet (Task increment)							
	buffer size 500				buffer size 1000				buffer size 500				buffer size 1000			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
	Finetune		11.12		Joint		44.39		Finetune		23.46		Joint		62.30	
GDUMB [28]	6.22				7.15				16.37				17.69			
AGEM [6]	50.06•	10.69•	22.29•	−49.22	50.03•	10.69•	22.28•	−49.16•	50.06•	18.34•	28.05•	−39.65•	50.03•	18.78•	28.12•	−39.05•
MIR [1]	51.44	11.07•	23.65•	−50.46•	51.25•	11.32•	24.09•	−49.92•	51.47•	29.10•	35.20•	−27.95•	51.31•	31.39•	37.24•	−24.89•
GSS [2]	51.63	11.09•	23.62•	−50.66•	51.35•	11.42•	24.05•	−49.91•	51.64•	28.67•	35.22•	−28.71•	51.40•	31.75•	37.23•	−24.56•
GMED [33]	51.21	11.03•	24.47•	−50.23•	50.87	11.73•	25.50•	−48.93•	51.29•	30.47•	37.64•	−26.02•	51.00•	32.85•	39.66•	−22.69•
ER [7]	51.68	11.00•	23.71•	−50.84•	51.41	11.35•	24.08•	−50.08•	51.70	28.97•	35.30•	−28.40•	51.55•	31.59•	37.36•	−24.95•
Ours	51.76	12.48	**26.50**	−49.10	50.91	14.43	**28.47**	−45.59	**52.44**	32.59	39.38	−24.82	**52.27**	36.25	41.59	−20.02
Ours+RehSel	**51.81**	**12.74**	26.43	**-48.84**	50.96	**14.54**	28.44	**-45.52**	51.73	**34.36**	**40.48**	**-21.70**	51.47	**37.20**	**42.19**	**-17.83**



(a) Total 500 memory data (b) Total 10000 new data per task

Figure 3: Top: Statistics of examples with positive and negative influence on S, P, and SP. Bottom: We divide all example influences equally into 5 groups, and count the number in each range.

larger gradient to dominant the update for all rehearsal-based CL, our method improves the example with positive effective and restrain the negative-impact example. In terms of the Mean Average Accuracy A_m, which evaluates the SP throughout the whole CL process, our method shows its significant superiority with an average improvement of over 4.44 and 1.24 w.r.t the second best results in class-incremental and task-incremental settings. The complete results with std. can be viewed in the Appendix. Moreover, with the proposed rehearsal selection strategy (Ours+RehSel), we have our A_∞ improved, which means the selected example according to their influence has a clear ability for reducing catastrophic forgetting. With our Rehearsal Selection (RehSel) strategy, we have an improvement of 0.77 on A_∞, but A_1 and A_m have uncertain performance. This means better memory may bring in worse task conflict.

6.3 Analysis of Dataset Influence on SP

In Fig. 3, we count the example with positive/negative influences on old task (S), new task (P), and total SP in Split-CIFAR-10. At each task after task 2, we have 500 fixed-size memory and 10,000 new task data. We first find that most data of old tasks has a positive influence on S and a negative influence on P, while most data of new tasks has a positive influence on P and a negative influence on S. Even so, some data in both new and old tasks has the opposite influence. Then, for the total SP influence, most of memory data has positive influence. In contrast, examples of new tasks have near equal number of positive and negative SP influence. Thus, by clustering and storing examples via higher influence to rehearsal memory buffer, the old knowledge can be kept. By dividing all example influences equally into 5 groups from the minimum to the maximum, we find that most examples have mid-level influence, and server as the main part of the dataset. Also, the numbers of examples with large positive and negative influence are small, which means unique examples are few in the dataset. The observations suggest the example difference should be used to improve model training.

6.4 Analysis on SP Pareto Optimum

In this paper, we propose to convert the S-aware and P-aware influence fusion into a DOO problem and use the MGDA to guarantee the fused solution is an SP Pareto optimum. As shown in Fig. 4, we show the comparison of the First Accuracy and Final Accuracy coordinate visualization for all compared methods. We also evaluate with only stability-aware (Ours only S) and with only Plasticity-aware (Ours only P) influence. Obviously, with only one kind of influence, our method can already get better SP than other methods. The integration of two kinds of influence yield an even more balanced SP. On the other hand, the existing methods cannot approach the SP Pareto front well.

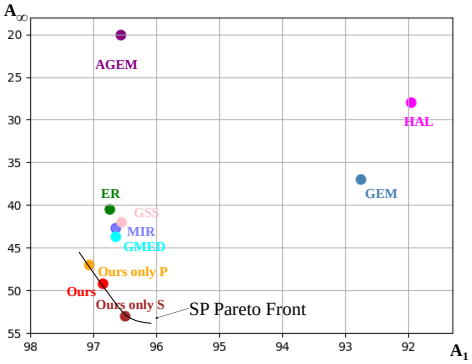


Figure 4: SP Pareto front.

6.5 Training Time

We list the training time of one-step update and total update overhead for all compared methods for Split CIFAR-10 dataset. In one-step update, we evaluate all methods with a batch on one update. Our method takes more time than other methods except for GEM, because of the pseudo update, backward on perturbation and influence fusion. To guarantee the efficiency, we utilize our proposed method only in the last 5 epochs among the total, and the rest are naive fine-tuning (See details in the Appendix). The results show the strategy is as fast as other light-weight methods but achieve huge improvement on SP. We also use this setting for the comparison in Table 1.

Table 2: Comparison of training time [s] on CIFAR-10.

Method	ER	GSS	AGEM	HAL	MIR	GMED	GEM	MetaSP
One-Step	0.013	0.015	0.029	0.043	0.077	0.093	0.290	0.250
Total	2685	2672	3812	5029	7223	8565	24768	5898

7 Conclusion

In this paper, we proposed to explore the example influence on Stability-Plasticity (SP) dilemma in rehearsal-based continual learning. To achieve that, we evaluated the example influence via small perturbation instead of the computationally expensive Hessian-like influence function and proposed a simple yet effective MetaSP algorithm. At each iteration in CL training, MetaSP builds a pseudo update and obtains the S- and P-aware example influence in batch level. Then, the two kinds of influence are combined via an SP Pareto optimal factor and can support the regular model update. Moreover, the example influence can be used to optimize rehearsal selection. The experimental results on three popular CL datasets verified the effectiveness of the proposed method. We list the limitation of the proposed method. (1) The proposed method relies on rehearsal selection, which may affect privacy and extra storage is needed. (2) The proposed method is not fast enough for online continual learning. In most situations, however, we can leverage our training tricks to reduce the time. (3) Our method is limited in the extremely small memory size. Large memory size means better remembering and an accurate validation set. The proposed method does not perform well when the memory size is extremely small.

Acknowledgement

This work is financially supported in part by the National Key Research and Development Program of China under Grant (No. 2019YFC1520904) and the Natural Science Foundation of China (Nos. 62072334, 62276182, 61876220). The authors would like to thank constructive and valuable suggestions for this paper from the experienced reviewers and AE.

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*, 2019.

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *NeurIPS*, 2019.

[3] Benedikt Bagus and Alexander Gepperth. An investigation of replay-based approaches for continual learning. In *IJCNN*, 2021.

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

[5] Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI*, 2021.

[6] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2018.

[7] 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. *arXiv preprint arXiv:1902.10486*, 2019.

[8] Kalyanmoy Deb and Himanshu Gupta. Searching for robust pareto-optimal solutions in multi-objective optimization. In *ICEMO*, 2005.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] Kaile Du, Linyan Li, Fan Lyu, Fuyuan Hu, Zhenping Xia, and Fenglei Xu. Class-incremental lifelong learning in multi-label classification. *arXiv preprint arXiv:2207.07840*, 2022.

[11] Kaile Du, Fan Lyu, Fuyuan Hu, Linyan Li, Wei Feng, Fenglei Xu, and Qiming Fu. Agcn: Augmented graph convolutional network for lifelong multi-label image recognition. In *ICME*, 2022.

[12] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 2012.

[13] Yang Fan, Yingce Xia, Lijun Wu, Shufang Xie, Weiqing Liu, Jiang Bian, Tao Qin, and Xiang-Yang Li. Learning to reweight with deep interactions. *arXiv preprint arXiv:2007.04649*, 2020.

[14] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[15] Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 2000.

[16] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Learning with long-term remembering: Following the lead of mixed stochastic gradient. *arXiv preprint arXiv:1909.11763*, 2019.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[19] 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. *PNAS*, 2017.

[20] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[21] Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*. IEEE, 1996.

[22] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

[23] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020.

[24] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *ICML*, 2016.

[25] Fan Lyu, Shuai Wang, Wei Feng, Zihan Ye, Fuyuan Hu, and Song Wang. Multi-domain multi-task rehearsal for lifelong learning. In *AAAI*, 2021.

[26] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[27] Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.

[28] Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

[29] Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological review*, 1990.

[30] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[31] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[32] 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. *arXiv preprint arXiv:1810.11910*, 2018.

[33] Liu Risheng, Liu Yaohua, Zeng Shangzhi, and Zhang Jin. Gradient-based editing of memory examples for online task-free continual learning. In *NeurIPS*, 2021.

[34] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

[35] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NeurIPS*, 2016.

[36] Tianyang Wang, Jun Huan, and Bo Li. Data dropout: Optimizing training data for convolutional neural networks. In *ICTAI*, 2018.

[37] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A] We use public dataset and open-sourced code.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Exploring Example Influence in Continual Learning (Appendix)

A Notation

We list the mentioned notation in Table 3 for quickly looking up.

Table 3: Notation related to the paper.

Symbol	Description	Symbol	Description
$\boldsymbol{\theta}_t$	The model trained after the t-th task	P_t	$p(\mathcal{D}_t^{\text{tst}}\|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_t^{\text{tst}}\|\boldsymbol{\theta}_{t-1})$, Plasticity
α	Optimization step size	$\mathbf{H}$	$\nabla^2_{\boldsymbol{\theta}}\ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}})$; Hessian assumed positive define
$\mathcal{D}_t$	$\{(x_t^{(n)}, y_t^{(n)})\}_{n=1}^{N_t}$; The t-th dataset	ϵ	Example-level perturbation
$\mathcal{D}_t^{\text{trn}}$	The training set of the t-th dataset	$\mathbf{E}$	Batch-level perturbation
$\mathcal{D}_t^{\text{tst}}$	The testing set of the t-th dataset	$\hat{\theta}_{\epsilon,\mathcal{B}}$	Pseudo updated model with example-level perturbation ϵ
$\mathcal{M}$	The memory buffer sampled from training set	$\hat{\theta}_{\mathbf{E},\mathcal{B}}$	Pseudo updated model with batch-level perturbation $\mathbf{E}$
$\mathcal{B}_{\text{old}}$	A mini-batch sampled from $\mathcal{M}$	$\mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B})$	Influence on S from mini-batch $\mathcal{B}$
$\mathcal{B}_{\text{new}}$	A mini-batch sampled from $\mathcal{D}_t^{\text{trn}}$	$\mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B})$	Influence on P from mini-batch $\mathcal{B}$
$\mathcal{V}_{\text{old}}$	Validation set of old tasks	γ	Weight fusion factor
$\mathcal{V}_{\text{new}}$	Validation set of new task	$\mathbf{I}^*$	Fusion Influence on SP from mini-batch $\mathcal{B}$
S_t^k	$p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_k), \quad k < t$ Stability	l	Loss for an example or average loss for a mini-batch
$\mathbf{L}$	Loss vector for a mini-batch		

B Influence Function

In this section, we illustrate how to get the Influence Function. Given a parameter $\boldsymbol{\theta}$ up to update, $\hat{\boldsymbol{\theta}}$ is the updated parameter using the training data, *i.e.*, $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell(\mathcal{B}, \boldsymbol{\theta})$. First, we set a small weight to a specific example x

$$\hat{\boldsymbol{\theta}}_{\epsilon,x} \coloneqq \arg\min_{\boldsymbol{\theta}} \ell(\mathcal{B}, \boldsymbol{\theta}) + \epsilon\ell(x, \boldsymbol{\theta}), \quad x \in \mathcal{B},$$

where ϵ is a small weight. Then, the variation of parameter can be shown as the IF on parameters

$$I(x, \hat{\boldsymbol{\theta}}) = \left.\frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\right|_{\epsilon=0} = -\mathbf{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\boldsymbol{\theta}} \ell(x, \hat{\boldsymbol{\theta}}).$$

Last, the influence from one training example to a testing example can be obtained by the chain rule

$$I(x^{\text{trn}}, x^{\text{tst}}) = \frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \left.\frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\right|_{\epsilon=0} = -\nabla_{\boldsymbol{\theta}} \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}}) \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} \ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}}),$$

where $\mathbf{H} = \nabla^2_{\boldsymbol{\theta}} \ell(\mathcal{B}, \hat{\boldsymbol{\theta}})$ is a Hessian and is assumed as positive definite.

Based on the rehearsal method, we consider the derivative of the loss of a validation set $\mathcal{V}$ of a mini-batch $\mathcal{B} = [x_1, x_2, \cdots, x_n]$ to weight vector $\mathbf{E} = [\epsilon_1, \epsilon_2, \cdots, \epsilon_n]^\top$ as $\mathbf{L} = \begin{bmatrix}\ell(x_1, \hat{\boldsymbol{\theta}}) & \ell(x_2, \hat{\boldsymbol{\theta}}) & \cdots & \ell(x_n, \hat{\boldsymbol{\theta}})\end{bmatrix}^\top$. With the Taylor expansion, we have

$$\frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \ell(x_i, \hat{\boldsymbol{\theta}}_\epsilon) + \mathbf{w}^\top \frac{\partial \mathbf{L}}{\partial \boldsymbol{\theta}} = \mathbf{0}.$$

The batch-level influence **vector** can be computed by

$$\mathbf{I}(\mathcal{V}, \mathcal{B}) = \left.\frac{\partial l(\mathcal{V}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})}{\partial \mathbf{E}}\right|_{\mathbf{E}=\mathbf{0}} = \frac{\partial l(\mathcal{V}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \left.\frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}}{\partial \mathbf{E}}\right|_{\mathbf{E}=\mathbf{0}} = -\nabla_{\boldsymbol{\theta}} l(\mathcal{V}, \hat{\boldsymbol{\theta}}) \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}}^\top \mathbf{L}(\mathcal{B}, \hat{\boldsymbol{\theta}}),$$

where

$$\mathbf{H} = \frac{1}{|\mathcal{B}|}\sum_{x_i \in \mathcal{B}} \nabla^2_{\boldsymbol{\theta}} \ell(x_i, \hat{\boldsymbol{\theta}}_{\epsilon_i}), \quad \left.\frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}}{\partial \mathbf{E}}\right|_{\mathbf{E}=\mathbf{0}} = -\mathbf{H}^{-1}\left[\frac{\partial \mathbf{L}}{\partial \boldsymbol{\theta}}\right]^\top.$$

Table 4: Comparisons on Split CIFAR-10, averaged across 5 runs. Red and blue values mean the best in our methods and the compared methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	Split CIFAR-10 (Class-Increment)							
Fine-tune	19.66±0.04							
Joint	91.79±0.68							
Buffer size	$\lvert\mathcal{M}\rvert = 300$				$\lvert\mathcal{M}\rvert = 500$			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
GDUMB	-	36.92±1.86	-	-	-	44.27±0.41	-	-
GEM	93.90±0.55	37.51±2.06	55.43±0.31	-70.48	92.76±1.13	36.95±2.34	57.36±1.02	-69.76
AGEM	96.57±0.40	20.02±0.23	45.57±0.40	-95.69	96.56±0.11	20.01±0.16	46.52±1.04	-95.70
HAL	91.30±1.95	24.45±2.09	46.34±1.77	-83.57	91.96±0.64	27.94±2.15	49.05±1.00	-80.02
MIR	96.70±0.12	38.53±1.72	56.96±1.16	-72.72	96.65±0.10	42.65±1.46	59.99±0.69	-67.50
GSS	96.53±0.20	35.89±2.46	54.33±1.35	-75.80	96.55±0.27	41.96±1.08	58.16±0.46	-68.25
GMED	96.65±0.24	38.12±0.99	58.92±0.67	-73.17	96.65±0.26	43.68±1.74	62.56±0.56	-66.22
ER	96.73±0.35	34.19±1.35	53.72±0.39	-78.18	96.74±0.08	40.45±2.14	57.69±1.41	-70.36
Ours	**96.87±0.09**	42.42±1.94	63.52±0.70	-68.07	**96.82±0.21**	49.16±1.48	67.88±0.82	-59.58
Ours+RehSel	96.85±0.09	**43.76±0.52**	**63.69±0.62**	**-66.37**	96.81±0.19	**50.10±1.32**	**68.28±0.88**	**-58.39**

Method	Split CIFAR-10 (Task-Increment)							
Fine-tune	65.27±2.28							
Joint	98.16±0.09							
Buffer size	$\lvert\mathcal{M}\rvert = 300$				$\lvert\mathcal{M}\rvert = 500$			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
GDUMB	-	73.22±0.67	-	-	-	78.06±1.41	-	-
GEM	96.62±0.05	89.34±0.99	92.49±0.36	-9.10	96.73±0.25	90.42±1.23	92.93±0.27	-7.89
AGEM	96.78±0.29	85.52±1.01	90.16±0.17	-14.07	96.71±0.10	86.45±1.06	90.90±0.70	-12.83
HAL	91.41±1.86	79.90±2.25	83.78±1.62	-14.39	92.03±0.64	81.84±2.13	84.19±1.46	-12.73
MIR	96.76±0.09	88.50±1.30	90.87±0.50	-10.33	96.73±0.08	90.63±0.63	91.99±0.46	-7.62
GSS	96.56±0.18	88.05±1.52	90.60±0.82	-10.63	96.57±0.27	90.38±0.87	92.19±0.59	-7.74
GMED	96.73±0.24	88.91±1.16	91.20±0.60	-9.77	96.72±0.22	89.72±1.25	92.10±0.65	-8.75
ER	96.93±0.07	88.97±0.67	91.12±0.79	-9.95	96.79±0.08	90.60±0.74	92.28±0.32	-7.75
Ours	97.10±0.12	89.40±0.93	92.54±0.38	-9.63	**97.31±0.18**	90.91±0.60	**93.38±0.31**	-7.99
Ours+RehSel	**97.11±0.11**	**89.91±0.55**	**92.66±0.23**	**-9.00**	97.30±0.04	**91.41±0.60**	93.28±0.32	**-7.36**

C Comparison Results with std.

In Table 4, 5, and 6, we show more comparison results about the proposed MetaSP with other SOTA methods. For each dataset, we evaluate all methods with task-incremental and class-incremental learning, where the task-incremental setting will offer the task id at the interference while the class-incremental will not. Every method will be evaluated with two different buffer sizes. All the experiments are implemented with 5 fixed seeds from 1231 to 1235 to keep the fair comparison as other CL methods.

D Validation Sets Size

In Fig. 5, we show the validation size effect in MetaSP. For three metrics, when the validation size grows, the value gets larger, which means that a large validation set will improve SP.

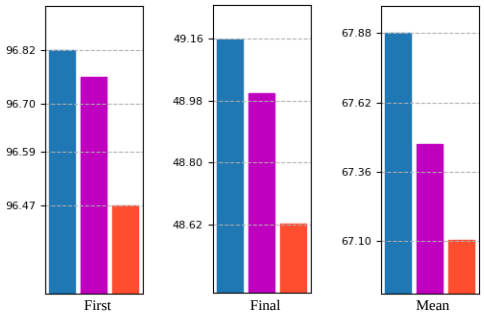


Figure 5: Performance with different validation sets size.

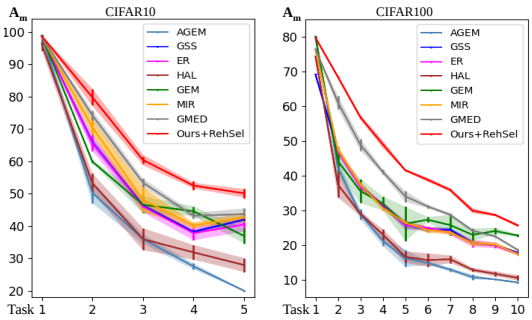


Figure 6: Learning processes on Split-CIFAR-10 and Split-CIFAR-100.

Table 5: Comparisons on Split CIFAR-100, averaged across 5 runs. Red and blue values mean the best in our methods and the compared methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	Split CIFAR-100 (Class-Increment)							
Fine-tune	9.14±0.18							
Joint	71.25±0.12							
Buffer size	$\lvert\mathcal{M}\rvert = 500$				$\lvert\mathcal{M}\rvert = 1000$			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
GDUMB	-	11.11±0.60	-	-	-	15.75±0.21	-	-
GEM	85.28±2.26	15.91±0.42	29.38±1.67	-77.07	84.28±2.34	22.79±0.31	34.09±1.75	-68.32
AGEM	85.97±1.27	9.31±0.13	24.60±0.90	-85.18	85.66±1.84	9.27±0.12	24.67±1.07	-84.88
HAL	67.33±1.89	8.20±0.84	22.72±0.71	-65.70	68.06±2.95	10.59±0.78	24.74±1.26	-63.86
MIR	87.38±1.30	13.49±0.18	28.88±1.57	-82.10	87.39±1.24	17.56±0.56	32.48±1.50	-77.60
GSS	86.03±1.91	14.01±0.50	28.00±2.00	-80.03	86.31±1.84	17.87±0.29	31.82±1.86	-76.04
GMED	87.18±1.45	14.56±0.24	33.41±1.37	-80.69	87.29±1.63	18.67±0.30	38.69±1.63	-76.24
ER	87.23±1.65	13.75±0.39	28.88±1.71	-81.65	87.33±1.51	17.56±0.35	32.45±1.78	-77.52
Ours	**88.13±0.80**	18.96±0.40	38.62±0.88	-76.85	**87.58±0.75**	24.78±0.68	45.20±0.97	-69.77
Ours+RehSel	87.81±0.87	**19.28±0.54**	**39.23±0.62**	**-76.14**	87.55±0.68	**25.72±0.48**	**45.48±0.76**	**-68.70**

Method	Split CIFAR-100 (Task-Increment)							
Fine-tune	33.89±3.14							
Joint	91.63±0.06							
Buffer size	$\lvert\mathcal{M}\rvert = 500$				$\lvert\mathcal{M}\rvert = 1000$			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
GDUMB	-	36.40±0.97	-	-	-	43.25±0.35	-	-
GEM	85.53±2.30	68.68±0.99	68.49±1.58	-18.72	85.24±2.28	73.71±0.42	72.59±2.12	-12.81
AGEM	85.97±1.27	55.28±1.04	58.23±1.19	-34.10	85.66±1.84	55.95±1.99	59.96±1.58	-33.02
HAL	67.64±1.94	44.98±1.86	50.79±1.40	-25.17	68.62±2.93	50.07±2.34	54.01±2.47	-20.62
MIR	87.42±1.30	66.18±1.25	67.43±1.90	-23.60	87.50±1.23	71.20±0.60	71.42±1.46	-18.11
GSS	86.10±1.89	66.80±0.54	66.55±1.89	-21.45	86.44±1.85	71.98±0.72	71.00±1.81	-16.07
GMED	87.30±1.41	68.82±0.80	72.66±1.86	-20.53	87.49±1.64	73.91±0.35	76.36±1.82	-15.10
ER	87.29±1.65	66.82±1.04	67.56±1.68	-22.74	87.40±1.50	71.74±0.55	71.60±1.90	-17.40
Ours	**88.94±0.80**	70.03±0.57	74.07±0.94	-21.01	88.94±0.73	75.32±0.43	78.09±0.97	-15.14
Ours+RehSel	88.58±0.82	**70.81±0.76**	**74.24±0.90**	**-19.75**	**89.03±0.65**	**76.14±0.88**	**78.27±0.89**	**-14.33**

E Training Process

We also visualize the CL training process on Split CIFAR-10 and Split CIFAR-100 in Fig. 6. The first observation is that the proposed MetaSP outperforms other methods a lot, which means better SP throughout the CL training. Second, the forgetting cannot be eliminated even in the rehearsal-based CL. Even so, MetaSP offers an elegant add-in for the rehearsal-based CL methods and can further improve performance.

F Time Analysis

Implemented epochs. In our implementation, we set 50 epochs in total for training each task, in which the first 45 epochs are naive fine-tuning and the last 5 epochs are with the proposed methods. As shown in Fig. 7(a), our method is implemented from the last 1 to 8 epochs. As the extra experiments use naive fine-tuning, where the new task is trained without interference. We prefer to evaluate if new tasks will be affected by memory. Easy to see, from last 1 to last 7, the performances of new task A1 get improved without a break, and from last 5, the performance barely grows anymore. This indicates that interference from old task to new task becomes hard from the last 1 to the last 5, and gets balance from the last 5 to 8. However, more epochs mean more computation costs. Thus, in our experiment, we leverage our methods in the last 5 epochs as a trade-off between efficiency and accuracy.

Iteration number for pseudo update. To compute example influence, we need to compute an argmin problem

$$\hat{\boldsymbol{\theta}}_{\mathbf{E},x} := \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right) + \mathbf{E}^\top \mathbf{L}(x, \boldsymbol{\theta}), \quad x \in \mathcal{B}.$$

In Fig. 7(b), we show the results with different iterations to solve the problem. Experiments are implemented under the last 5 epochs. Easy to observe, a large number of iterations is equal to better performance but increases computational time linearly. Thus, we set only one iteration to keep the efficient training in compared with other CL methods' time.

Table 6: Comparisons on Split Mini-Imagenet, averaged across 5 runs. Red and blue values mean the best in our methods and the previous methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	Split Mini-Imagenet (Class-Increment)							
Fine-tune	11.12±0.23							
Joint	44.39±0.74							
Buffer size	$\lvert\mathcal{M}\rvert = 500$				$\lvert\mathcal{M}\rvert = 1000$			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
GDUMB	-	6.22±0.27	-	-	-	7.15±1.96	-	-
AGEM	50.06±0.42	10.69±0.07	22.29±0.23	-49.22	50.03±0.31	10.69±0.22	22.28±0.09	-49.17
MIR	51.44±0.40	11.07±0.22	23.65±0.16	-50.46	51.25±0.37	11.32±0.18	24.09±0.18	-49.92
GSS	51.63±0.51	11.09±0.12	23.62±0.24	-50.67	51.35±.36	11.42±0.22	24.05±0.19	-49.91
GMED	51.21±0.69	11.03±0.18	24.47±0.29	-50.42	50.87±0.28	11.73±0.23	25.50±0.15	-49.93
ER	51.68±0.53	11.00±0.24	23.71±0.21	-50.84	51.41±0.60	11.35±0.35	24.08±0.31	-50.08
Ours	51.76±0.12	12.48±0.12	**26.50±0.15**	-49.10	50.91±0.41	14.43±0.31	**28.47±0.18**	-45.59
Ours+RehSel	**51.81±0.39**	**12.74±0.17**	26.43±0.11	**-48.85**	50.96±0.14	**14.54±0.22**	28.44±0.20	**-45.52**
Method	Split Mini-Imagenet (Task-Increment)							
Fine-tune	23.46±1.17							
Joint	62.3±0.48							
Buffer size	$\lvert\mathcal{M}\rvert = 500$				$\lvert\mathcal{M}\rvert = 1000$			
	A_1	A_∞	A_m	BWT	A_1	A_∞	A_m	BWT
GDUMB	-	16.37±0.37	-	-	-	17.69±3.24	-	-
AGEM	50.06±0.42	18.34±0.56	28.05±0.37	-39.66	50.03±0.31	18.78±0.54	28.12±0.30	-39.06
MIR	51.47±0.40	29.10±0.52	35.20±0.46	-27.96	51.31±0.37	31.39±0.44	37.24±0.52	-24.90
GSS	51.64±0.51	28.67±0.66	35.22±0.60	-28.72	51.40±0.35	31.75±0.71	37.23±0.48	-24.56
GMED	51.29±0.68	30.47±0.39	37.64±0.59	-26.03	51.00±0332	32.85±0.27	39.66±0.36	-22.69
ER	51.70±0.52	28.97±0.36	35.30±0.43	-28.41	51.55±0.57	31.59±0.78	37.36±0.57	-24.95
Ours	**52.44±0.20**	32.59±1.09	39.38±0.37	-24.82	**52.27±0.40**	36.25±0.27	41.59±0.33	-20.03
Ours+RehSel	51.73±0.41	**34.36±0.28**	**40.48±0.42**	**-21.71**	51.47±0.59	**37.20±0.73**	**42.19±0.51**	**-17.83**

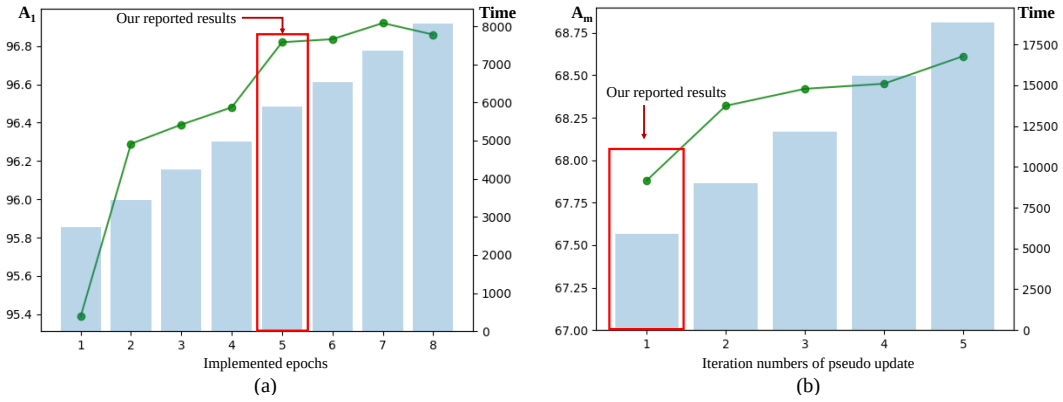


Figure 7: (a) Comparisons on different numbers of implemented epochs using only one-step pseudo update, from the last 1 to 8 epochs. (b) Comparisons on different numbers of pseudo update using 5 implemented epochs, from 1 to 5 iterations. The two subfigures are with the same legend, where the green lines represent the accuracy while blue bars mean the training time.

G Comparisons with Other Influence-based Rehearsal Methods

In this paper, we proposed an influence-based rehearsal selection method under fixed budget memory $\mathcal{M}$. After task t finishes its training, for storing, we first cluster all training data into $\frac{|\mathcal{M}|}{t}$ groups using K-means to diversify the store data. Each group is ranked by their SP influence expection, *i.e.*, $\mathbb{E}(I^*(x))$, and the most positive influence on both SP will be selected to store. For dropping, we rank again on memory buffer via their influence values, and drop the most negative $\frac{|\mathcal{M}|}{t}$ example.

We also show if we do not cluster but select the top $\frac{|\mathcal{M}|}{t}$ examples only via their influence on SP (w/o clustering). The results are shown in Table 7. Using examples with only larger influence makes the selection concentrate on some saliency examples and lack the example diversity. Moreover, we evaluate if we only cluster and select the nearest examples instead of the example influence (w/o

Table 7: Rehearsal comparisons.

Method	Split CIFAR-10						Split CIFAR-100					
	Class-Increment			Task-Increment			Class-Increment			Task-Increment		
	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m
Random	96.82	49.16	67.88	97.30	90.91	93.38	88.13	18.96	38.62	88.94	70.03	74.07
RehSel (w/o cluster)	96.60	38.50	62.38	97.27	85.55	91.22	87.54	16.40	35.83	88.35	65.83	70.67
RehSel (w/o influence)	96.82	47.55	67.49	97.15	90.33	93.07	87.96	19.04	38.90	88.78	70.32	74.18
RehSel	96.81	50.10	68.28	97.30	91.41	93.29	87.81	19.28	39.23	88.58	70.81	74.24

Table 8: Influence approximation comparisons.

Method	Total examples	True Positive	True Negative	False Positive	False Negative
Larsen [21]	1000	552	304	26	118
Luketina [24]	1000	533	313	17	137
Neumann Series [23]	1000	642	282	48	28
Ours	1000	650	315	15	20

influence). Finally, with both influence ranking and cluster, the selected examples make the training forget less.

H Details of Influence Statistics in Fig. 3 of This Paper

As shown in Fig. 3 in the paper, we count the example with positive and negative influence on old task (S), new task (P) and total SP and show the distribution in Split-CIFAR-10. Here, we give the details of these experiments. The experiments are with 50 implemented epochs and 5 pseudo-update iterations. We do not use the update and rehearsal selection strategy. For each task from task 2, we have 500 fixed-size memory and 10,000 new task data. We divide all example influences equally into 5 groups from the minimum to the maximum.

I MGDA, KKT Conditions and The Solution

In Eq. (10) of this paper, we introduce the dual objective problem and the Multiple-Gradient Descent Algorithm (MGDA) [12]. To obtain the objective, we first introduce the Steepest Gradient Method (SGM) [15] in dual-objective optimization. Given two tasks 1 and 2, the objective of SGM is

$$\mathbf{d}^*, \alpha^* = \arg\min_{\mathbf{d},\alpha} \quad \alpha + \frac{1}{2}||\mathbf{d}||^2, \quad \text{s.t.} \quad \mathbf{g}_1^\top \mathbf{d} \leq \alpha \text{ and } \mathbf{g}_2^\top \mathbf{d} \leq \alpha,$$

where $\mathbf{g}_1$ and $\mathbf{g}_2$ are the gradients for tasks 1 and 2 specifically. The two constraints can be seen as the difference between task gradients and the optimal gradients.

Considering the Lagrange multipliers λ_1 and λ_2 for the two constraints, we have the dual problem of the above problem as

$$\lambda_1^*, \lambda_2^* = -\max_{\lambda_1,\lambda_2} ||\lambda_1 \mathbf{g}_1 + \lambda_2 \mathbf{g}_2||^2, \quad \text{s.t.} \quad \lambda_1 + \lambda_2 = 1 \text{ and } \lambda_1 \geq 0, \lambda_2 \geq 0.$$

This is the objective of Eq. (10) of this paper, *i.e.*, MGDA [12]. In SGM, the KKT conditions can be written as

$$\begin{aligned} \lambda_1^*(\mathbf{g}_1^\top \mathbf{d}^* - \alpha^*) &= 0, \\ \lambda_2^*(\mathbf{g}_2^\top \mathbf{d}^* - \alpha^*) &= 0, \\ \lambda_1^* \geq 0, \lambda_2^* &\geq 0, \\ \lambda_1^* + \lambda_2^* &= 1, \\ \lambda_1^* \mathbf{g}_1 + \lambda_2^* \mathbf{g}_2 &= \mathbf{d}^*. \end{aligned}$$

The solution to the dual problem is

$$\gamma^* = \min\left(\max\left(\frac{(\mathbf{g}_2 - \mathbf{g}_1)^\top \mathbf{g}_2}{\|\mathbf{g}_2 - \mathbf{g}_1\|_2^2}, 0\right), 1\right).$$

This is the solution in Eq. (11) of this paper.

J Comparisons on Influence Function Approximation

To evaluate the example influence approximation from our meta method to Hessian influence function, we build a toy experiment compared with three extra baselines. The three baselines [21, 24, 23] use different ways to approximate inverse Hessian. We use 1000 FMNIST training data and 500 test data. We design a simple fc network with a single hidden layer. The results are the influence from the 1000 training data to 500 test data. We have the following observations: (1) Most examples (965/1000) have the true influence property compared with Hessian influence function; (2) The results show the proposed method has a better approximation rate compared with other inverse Hessian approximation methods.