DOES AN EXAMPLE CONTRIBUTE TO THE PLASTICITY OR STABILITY IN LIFELONG LEARNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Lifelong Learning (LL) is the sequential transformation of Multi-Task Learning, which learns new tasks in order like human-beings. Traditionally, the primary goal of LL is to achieve the trade-off between the Stability (remembering past tasks) and Plasticity (adapting to new tasks). Rehearsal, seeking to remind the model by storing examples from old tasks in LL, is one of the most effective ways to get such trade-off. However, the Stability and Plasticity (SP) are only evaluated when a model is trained well, and it is still unknown what leads to the final SP in rehearsal-based LL. In this paper, we study the cause of SP from the perspective of example difference. First, we theoretically analyze the example-level SP via the influence function and deduce the influence of each example on the final SP. Moreover, to avoid the calculation burden of Hessian for each example, we propose a simple yet effective MetaSP algorithm to simulate the acquisition of example-level SP. Last but not least, we find that by adjusting the weights of each training example, a solution on the SP Pareto front can be obtained, resulting in a better SP trade-off for LL. Empirical results show that our algorithm significantly outperforms state-of-the-art methods on benchmark LL datasets.

1 INTRODUCTION

Machine learning paradigms have shown human-level performance and exceeded that in solving specific tasks (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Lyu et al., 2021). However, the traditional machine learning algorithms, training on a static environment, still have gap from being human-like learning. According to one of the most important skills, continually learning from changing or non-stationary data, is missing. In recent years, lifelong learning (LL, a.k.a. continual learning and incremental learning) becomes popular, owing to learning new tasks in sequence.

The major goal for a LL system is to approach the **Stability (S)-Plasticity (P) trade-off**. On the one hand, the Stability represents the ability of preventing performance drops of old tasks when the data distribution drifts (*i.e.*, catastrophic forgetting (French, 1999; Kirkpatrick et al., 2017)). On the other hand, the Plasticity measures if the new task can be learned rapidly and unimpededly based on the old knowledge. These two simultaneous constraints require model stability to alleviate catastrophic forgetting and plasticity to integrate novel information, which is known as **SP dilemma**.

The existing LL approaches (Kirkpatrick et al., 2017; Lyu et al., 2021; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; 2019; Aljundi et al., 2019a) show their effectiveness on harnessing SP trade-off. They evaluate SP on trained model in task-level, which is equivalent to the performance on the whole testing set after each task learned. *However, to the best of our knowledge, there exist few studies on what leads to S and P in nature.*

By observing the training and inference of a machine learning model, we consider the influence chain "`Data-Model-Performance`". We pay more attention on the training data, i.e., from examples to model, along with the whole LL training process. In this paper, we propose and prove that the task-level SP can be decomposed into the gather of example-level SP of every training datum. As shown in Fig. 1, in the training process of LL, the traditional task-level SP influence (evaluate after a trained task) can be decomposed into example-level influence from any training example to the testing data (old and new tasks).

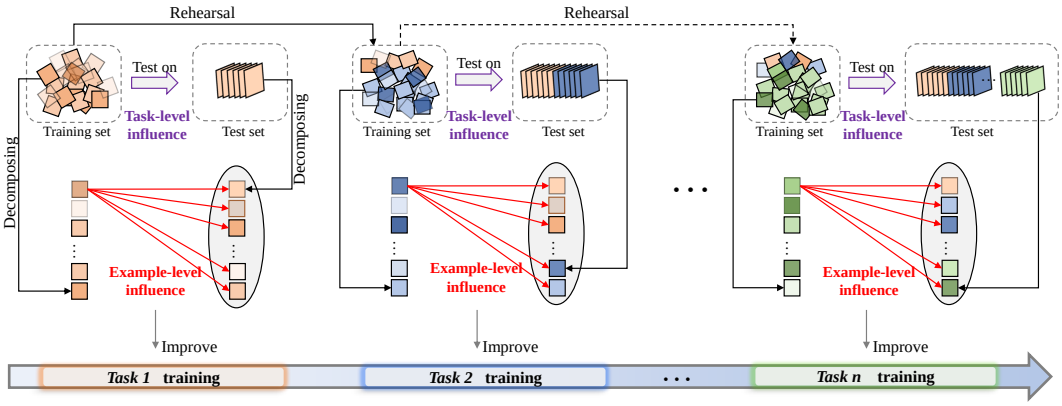


Figure 1: Influence decomposition. The traditional influence (including Stability and Plasticity) refer to the testing phase, where the training set influence the testing set via the trained model. By decomposing, the task-level influence can be expressed as the gather of example-level influence, which may guide the LL training itself.

The decomposition of SP indicates if we leverage the differences between each example, we may freely steer both the Stability and Plasticity and further improve the LL training. Under this assumption, this paper proposes to weight the training for each example on the basis of its contributions to SP in a rehearsal LL schema, which retrains a small stored subset of old tasks (saved in memory buffer) together with new task. At any iteration, the examples in the memory batch and the new task batch will be weighted for shared SP. Inspired by the Influence Function (IF) (Koh & Liang, 2017), we compute the example-level SP by measuring the loss sensitiveness of two testing sets w.r.t. the new and old tasks. We propose a novel meta-learning algorithm called MetaSP which simulate the above process and avoid the expensive computation of Hessian and its inverse in the IF. First, two pseudo updates are held for Stability-aware and Plasticity-aware models. Then, two validation sets sampled from seen data are used to measure the SP contributions and update the Stability-aware and Plasticity-aware weights. In this way, two weights on behalf of S and P are obtained.

At any time, an LL algorithm should provide a multi-task/class classifier (Lin et al., 2019; Sener & Koltun, 2018), where the objective is to find solutions not dominated by any others. Such solutions are said to be Pareto optimal (Deb & Gupta, 2005). To meet the SP trade-off goal, our MetaSP seeks to achieve SP Pareto optimal between old and new tasks. In this paper, we treat the two optimization on Stability-aware and Plasticity-aware weights as a multi-objective optimization problem, and refine the weights via the well-verified MGDA algorithm (Désidéri, 2012). This guarantee the proposed method has a mathematical and concrete SP Pareto optimal solution and yields SP trade-off. We evaluate the proposed MetaSP on three LL datasets. The experimental results show that by considering the example-level SP the whole SP within the training on old and new tasks can be improved significantly.

2 Related Work

2.1 Rehearsal-based Lifelong Learning

Rehearsal-based LL tackles the challenge of SP dilemma by retaining a subset of old tasks in a stored memory buffer with bounded resources. Back to 1990s, a simple but efficient technique named Experience Replay(ER) (Ratcliff, 1990) mingle the stored samples with current training batch to constrain the parameters update, which is also been demonstrated effective in LL (Chaudhry et al., 2019). Rehearsal-based LL methods suffer from two challenges. First, the stored memory with small size still has large difference from the original dataset, which also results in experience overfitting. Second, constrained by the joint training on old and new tasks, the task interference are inevitable and difficult to achieve SP trade-off. Thus, most of the recent works focus on solving these two problems in the way such as memory retrieval strategy, model and memory update strategy. GEM (Lopez-Paz & Ranzato, 2017) and AGEM (Chaudhry et al., 2018) build optimization constraints to ensure the loss for past tasks does not increase at every training step. MIR (Aljundi et al., 2019a) retrieve the samples which are most interfered by the foreseen parameters update.

GSS (Aljundi et al., 2019b) focuses on maximizing the diversity of samples in the replay buffer with parameters gradient as the feature. Besides, GDUMB (Prabhu et al., 2020) trains the model from scratch using the balanced memory buffer only which gives a strong baseline for rehearsal methods. Although the existing methods apply themselves to achieve better SP trade-off, they only consider the task-level influence without exploring what contributes to the Stability and Plasticity. In this work, we explore the problem in the perspective of example difference, where we argue that each example contributes differently to the SP.

2.2 Influence of example difference

Examples are different, even they belong to a same distribution. Because of such difference, the example contributions are different, and the learning can be more generalized. On the other hand, few adversarial examples can hijack and disable a model (Biggio et al., 2012; Yang et al., 2017). To obtain the different contributions, the training can be improved. In recent year, many methods are proposed to evaluate the contributions and leverage the difference. Influence Function (Koh & Liang, 2017) evaluates the contribution by setting a small pertubation to each specific examples and computing the gradient on the pertubation. Some studies propose similar idea and use the contributions to reweight the training (Ren et al., 2018; Zhong et al., 2021; Fan et al., 2020). Also, some studies propose to dropout examples with negative effect according to their contributions (Wang et al., 2018; Fan et al., 2017). In this paper, we inspire by the Influence Function and convert the task-level SP to example-level SP via their contributions, which can be used to control the training.

3 Demystifying Stability and Plasticity

3.1 Preliminary: rehearsal-based lifelong learning

Given T different tasks with respect to datasets $\{\mathcal{D}_1, \cdots, \mathcal{D}_T\}$, Lifelong learning (LL) seeks to learns 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 testing set $\mathcal{D}_t^{\text{tst}}$, where N_t is the number of examples. At any time, LL aims at learning a multi-task/multi-class predictor to predict tasks/classes that have been learned. To suppress the catastrophic forgetting, the Rehearsal-based LL (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Riemer et al., 2018; Chaudhry et al., 2018; Guo et al., 2019) 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 the training phase, the data in the whole memory $\mathcal{M} = \cup_{k<t}\mathcal{M}_k$ will be retrained together with the current tasks as

$$\min_{\boldsymbol{\theta}} \quad \ell(\mathcal{D}_t^{\text{trn}}, \boldsymbol{\theta}) + \ell(\mathcal{M}, \boldsymbol{\theta}), \tag{1}$$

where ℓ is the empirical loss and $\boldsymbol{\theta}$ is the trainable parameters across all tasks. Specifically, a mini-batch training step in rehearsal-based LL have the same goal to achieve the trade-off between the Stability and Plasticity, which is trained as

$$\min_{\boldsymbol{\theta}_t} \quad \ell(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t), \quad \mathcal{B}_{\text{old}} \subset \mathcal{M} \text{ and } \mathcal{B}_{\text{new}} \subset \mathcal{D}_t^{\text{trn}}. \tag{2}$$

3.2 Stability and Plasticity decomposition

However, to the best of our knowledge, few works explore what influences the SP in nature. To understand the SP, we summarize the recent definition on Stability and Plasticity (Lange et al., 2021) and give a probabilistic representation. Traditionally, the Stability of a task is evaluated by the performance difference on testing set after any later task trained, which is also known as Forgetting (Chaudhry et al., 2018). The Plasticity of a task is defined as the ability of integrating new knowledge, which is regarded as the testing performance of this task. Different from the routine, we give the mathmatical definition of Stability and Plasticity as follows.

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

$$S_t^k = p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_t) - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k), \quad k < t, \tag{3}$$

$$P_t = p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_t) - p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_{t-1}), \tag{4}$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ denotes the performance probability of $\mathcal{D}$ conditioned to the model $\boldsymbol{\theta}$.

Because of the SP-dilemma, the SP will inevitably interfere mutually. The existing LL solutions, for the sake of SP trade-off, is only reflected in the final performance till all tasks trained, which is unexplainable and ambiguous. As shown in Eq. (3) and Eq. (4), the Stability and Plasticity highly depend on the training data. Examples are different, previous studies have proven one example may influence the training (Koh & Liang, 2017; Wang et al., 2018; Ren et al., 2018). *This significantly motivates us to decompose the SP and explore the example-level contributions.* We naturally find that the Stability and the Plasticity can be decomposed into the integration of example-level influences

$$S_t^k \approx \prod_{x_k^{\text{tst}} \in \mathcal{D}_k^{\text{tst}}} \prod_{x_t^{\text{trn}} \in \mathcal{D}_t^{\text{trn}}} p(x_k^{\text{tst}}|x_t^{\text{trn}}), \quad P_t \approx \prod_{x_t^{\text{tst}} \in \mathcal{D}_k^{\text{tst}}} \prod_{x_t^{\text{trn}} \in \mathcal{D}_t^{\text{trn}}} p(x_t^{\text{tst}}|x_t^{\text{trn}}), \tag{5}$$

where we omit the constants $p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k)$ and $p(\mathcal{D}_t^{\text{tst}}|\boldsymbol{\theta}_{t-1})$. The detail can be seen in Appendix A.2.

By the decomposition, the Stability and Plasticity of the model can be significantly represented by each training example to each testing example. If we get the influence in advance, we can impel the training toward better SP trade-off. A simple way to get the influence is the Leave-One-Out (LOO) strategy, which trains model with and without a specific training example x^{trn} then evaluate the performance difference on a testing example x^{tst}. However, the LOO strategy suffers from massive computational cost especially for large datasets. Here, we introduce the Influence Function to denote this example-to-example influence on SP in the next section.

3.3 Influence function for SP

In this paper, we propose to convert the LOO strategy into setting a small weight to the example inspired by the Influence Function (IF) (Cook & Weisberg, 1982). Given a training batch $\mathcal{B}$, the normal model update is

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right). \tag{6}$$

A small weight perturbation ϵ is added to the training example $x \in \mathcal{B}$, then we have

$$\hat{\boldsymbol{\theta}}_{\epsilon,x} = \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right) + \epsilon\ell(x, \boldsymbol{\theta}), \quad x \in \mathcal{B}. \tag{7}$$

Easy to know, the influence $p(x^{\text{tst}}|x^{\text{trn}})$ that from a training example x^{trn} to a testing one x^{tst} is reflected in the derivative $\frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}}_{\epsilon,x})}{\partial \epsilon}\big|_{\epsilon=0}$. By the chain rule, the example-to-example influence can be computed by

$$p(x^{\text{tst}}|x^{\text{trn}}) \Longleftrightarrow I(x^{\text{trn}}, x^{\text{tst}}) = \frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\bigg|_{\epsilon=0} = -\nabla_{\boldsymbol{\theta}} \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}}) \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} \ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}}), \tag{8}$$

where $\mathbf{H} = \nabla_{\boldsymbol{\theta}}^2 \ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}})$ is a Hessian and is assumed as positive definite. The detail can be seen in Appendix A.1. Unfortunately, the inverse of Hessian requires the complexity $O(nq^2 + q^3)$ (n denotes the batch size and q is the number of parameters), which is a huge challenge for efficient training. By observing the chain rule, we find the left part $\frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}$ is the gradient of the testing loss, and the right part $\frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}$ is the gradient of the parameter to the perturbation. *Because each example contributes SP differently, if we can discriminate such difference, the rehearsal-based LL can be trained more effectively and may achieve better SP.* In the following, motivated by the chain rule, we will introduce a simple yet effective meta-based method named MetaSP to simulate the chain rule of Influence Function and show how to apply it to the LL training.

4 Meta Learning on Stability and Plasticity

4.1 Influence Function Simulation

Based on the Meta Learning paradigm, we transform the problem into solving a meta gradient descent one named MetaSP. For each training step in a rehearsal-based LL, we have two mini-batches data $\mathcal{B}_{\text{old}}$ and $\mathcal{B}_{\text{new}}$ in resepct to old and new tasks. Our goal is to obtain the influence

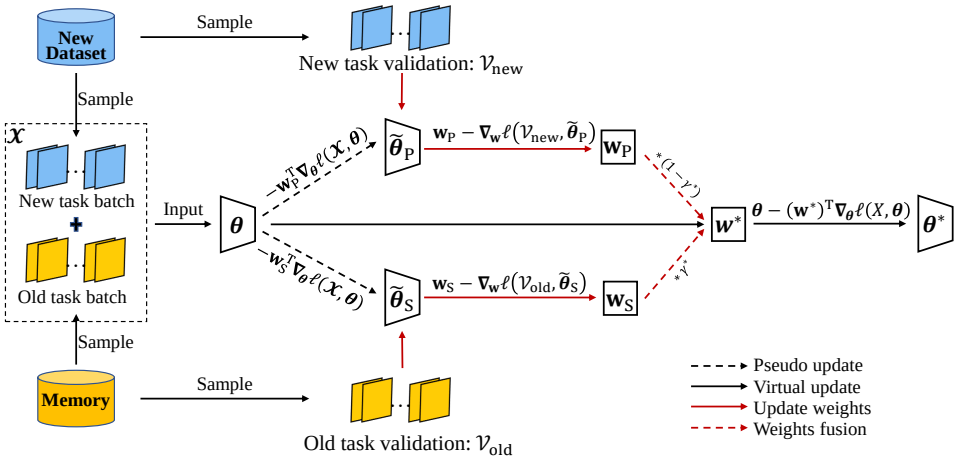


Figure 2: One-step update of MetaSP Algorithm. At each iteration in LL training, MetaSP initializes two weights for Stability and Plasticity and update in pseudo to obtain the Stability-aware and Plasticity-aware parameters. Then, two validation sets for old tasks ($\mathcal{V}_{\text{old}}$) sampled from memory buffer and for new task ($\mathcal{V}_{\text{new}}$) from the current training dataset are used to update the weights. Finally, the two weights will be combined via a SP Pareto optimal factors and support the virtual model update. Note we omit the step size in the update.

on Stability and Plasticity from every example in $\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}$, which yields two weights $\mathbf{w}_{\text{S}} \in \mathbb{R}^{(|\mathcal{B}_{\text{old}}|+|\mathcal{B}_{\text{new}}|)\times 1}$ and $\mathbf{w}_{\text{P}} \in \mathbb{R}^{(|\mathcal{B}_{\text{old}}|+|\mathcal{B}_{\text{new}}|)\times 1}$. Note that both Stability-aware and Plasticity-aware weights 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.

To simulate the IF $I(x^{\text{trn}}, x^{\text{tst}}) = \frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\Big|_{\epsilon=0}$, our MetaSP has two key steps: 1) Stability-aware and Plasticity-aware pseudo update to obtain $\hat{\boldsymbol{\theta}}$; 2) SP weights update to obtain influence for all training examples. In rehearsal-based LL, we have IF vector in a mini-batch training

$$\mathbf{I}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \mathcal{V}) = \frac{\partial \ell(\mathcal{V}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{w}, \mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}}}{\partial \mathbf{w}}\Big|_{\mathbf{w}=0}, \tag{9}$$

where $\{\mathcal{V}, \mathbf{w}\} = \{\mathcal{V}_{\text{old}}, \mathbf{w}_{\text{S}}\}$ or $\{\mathcal{V}_{\text{new}}, \mathbf{w}_{\text{P}}\}$. The detail can be seen in Appendix A.1. Noteworthily, because the testing set is unavailable at the training phase, we use two dynamic validation sets $\mathcal{V}_{\text{old}}$ and $\mathcal{V}_{\text{new}}$ to act as the alternative of the testing set $\mathcal{D}^{\text{tst}}$ in the LL 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 ($\mathcal{V}_{\text{new}}$). In detail, the two simulation steps are illustrated as follows.

Stability-aware and Plasticity-aware pseudo update. With the two initialized weights, we first update the neural network in pseudo as

$$\tilde{\boldsymbol{\theta}}_{\text{S}} = \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} \mathbf{w}_{\text{S}}^{\top} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}), \quad \tilde{\boldsymbol{\theta}}_{\text{P}} = \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} \mathbf{w}_{\text{P}}^{\top} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}), \tag{10}$$

where $\tilde{\boldsymbol{\theta}}_{\text{S}}$ denotes the Stability-aware update, and $\tilde{\boldsymbol{\theta}}_{\text{P}}$ represents the Plasticity-aware update. $\mathbf{L}$ denotes the loss vector for a mini-batch. Note that the two updates differ and highly depend on the initialization, and $\tilde{\boldsymbol{\theta}}_{\text{S}} = \tilde{\boldsymbol{\theta}}_{\text{P}}$ if the two weights are initialized equally.

SP weights update. Based on the two SP-aware models, we conduct updates on two SP weights via the two validation sets $\mathcal{V}_{\text{old}}$ and $\mathcal{V}_{\text{new}}$. The updates are computed as

$$\mathbf{w}_{\text{S}} = \mathbf{w}_{\text{S}} - \nabla_{\mathbf{w}_{\text{S}}} \ell(\mathcal{V}_{\text{old}}, \tilde{\boldsymbol{\theta}}_{\text{S}}), \quad \mathbf{w}_{\text{P}} = \mathbf{w}_{\text{P}} - \nabla_{\mathbf{w}_{\text{P}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}}), \tag{11}$$

where the gradients on two validation sets can be seen as the influence on Stability and Plasticity, which in essence simulates the second part of IF. If we have zero initialization for the two weights, then the influence is equal to the inverse of gradient.

Algorithm 1: MetaSP Algorithm (One-step update).

Input: $\mathcal{B}_{\text{old}}, \mathcal{B}_{\text{new}}, \mathcal{V}_{\text{old}}, \mathcal{V}_{\text{new}}, \mathbf{w}_{\text{S}}, \mathbf{w}_{\text{P}}, \boldsymbol{\theta}, \alpha$; // Training batches, Validation batches, Intialized weights, Model, Step size

Output: $\boldsymbol{\theta}^*$; // Updated model

/* 1) Stability-aware and Plasticity-aware pseudo update */

Pseudo update $\tilde{\boldsymbol{\theta}}_{\text{S}} = \boldsymbol{\theta} - \alpha \cdot \mathbf{w}_{\text{S}}^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta})$ for Stability;

Pseudo update $\tilde{\boldsymbol{\theta}}_{\text{P}} = \boldsymbol{\theta} - \alpha \cdot \mathbf{w}_{\text{P}}^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta})$ for Plasticity;

/* 2) SP weights update */

Update the Stability-aware weights $\mathbf{w}_{\text{S}} = -\nabla_{\mathbf{w}_{\text{S}}} \ell(\mathcal{V}_{\text{old}}, \tilde{\boldsymbol{\theta}}_{\text{S}})$;

Update the Plasticity-aware weights $\mathbf{w}_{\text{P}} = -\nabla_{\mathbf{w}_{\text{P}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}})$;

/* 3) Weights fusion for SP trade-off */

Get the optimal fusion hyper-parameter γ^* via Eq. (14);

Weights fusion $\mathbf{w}^* = \gamma^* \mathbf{w}_{\text{S}} + (1 - \gamma^*) \mathbf{w}_{\text{P}}$;

Model update $\boldsymbol{\theta}^* = \boldsymbol{\theta} - \alpha \cdot (\mathbf{w}^*)^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta})$.

4.2 Weights fusion for SP trade-off

We have obtained two weights $\mathbf{w}_{\text{S}}$ and $\mathbf{w}_{\text{P}}$ to represent the contributions of each training example to Stability and Plasticity. Given the parameter $\boldsymbol{\theta}$ of the previous step, MetaSP seeks to update $\boldsymbol{\theta}$ via fusing $\mathbf{w}_{\text{S}}$ and $\mathbf{w}_{\text{P}}$ into $\mathbf{w}^*$ towards SP trade-off parameter $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta} - \alpha \cdot (\mathbf{w}^*)^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}), \tag{12}$$

where α is the step size. To obtain the SP trade-off between stability and plasticity, we integrate the update of $\mathbf{w}_{\text{S}}$ and $\mathbf{w}_{\text{P}}$ into a multi-objective optimization (MOO) problem and joint training them following the multiple-gradient descent algorithm (MGDA) (Désidéri, 2012). Specifically, according to the Karush-Kuhn-Tucker conditions of ours (Fliege & Svaiter, 2000), we have

$$\gamma^* = \arg\min_{\gamma} \quad \left\| \gamma \nabla_{\mathbf{w}_{\text{old}}} \ell(\mathcal{V}_{\text{old}}, \tilde{\boldsymbol{\theta}}_{\text{S}}) + (1 - \gamma) \nabla_{\mathbf{w}_{\text{new}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}}) \right\|_2^2, \quad 0 \leq \gamma \leq 1. \tag{13}$$

It is easy to know that the optimal γ^* can be computed as

$$\gamma^* = \min \left(\max \left(\frac{\left(\nabla_{\mathbf{w}_{\text{new}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}}) - \nabla_{\mathbf{w}_{\text{old}}} \ell(\mathcal{V}_{\text{old}}, \tilde{\boldsymbol{\theta}}_{\text{S}}) \right)^T \nabla_{\mathbf{w}_{\text{new}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}})}{\| \nabla_{\mathbf{w}_{\text{new}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}}) - \nabla_{\mathbf{w}_{\text{old}}} \ell(\mathcal{V}_{\text{old}}, \tilde{\boldsymbol{\theta}}_{\text{S}}) \|_2^2}, 0 \right), 1 \right). \tag{14}$$

Thus, the final example-level weights can be computed by

$$\mathbf{w}^* = \gamma^* \mathbf{w}_{\text{S}} + (1 - \gamma^*) \mathbf{w}_{\text{P}}, \tag{15}$$

where we normalize $\mathbf{w}^*$ to make the weights positive and with sum 1.

The MetaSP algorithm can be seen in Fig. 2 and Alg. 1. MetaSP offers weighted update at every step for rehearsal-based LL, which leads the LL training to better SP trade-off but with only the complexity of $O(nq + vq)$ (v denotes the validation size) compared with that of IF, $O(nq^2 + q^3)$. Moreover, the appropriate factors for weights integration guarantee the update is SP Pareto Optimum, which yields a better SP trade-off. For the old tasks, though the stored memory may have large distribution difference from the original datasets, the Stability can be held well if we consider the example difference of training data and memory data by reducing the negative influence on Stability. For the new task, the old tasks will hinder less and some useless example will be restrained, which may improve the acquisition of new knowledge and gain better first accuracy.

5 Experiments

5.1 Datasets

We use three commonly used benchmarks for evaluation.

Table 1: Comparison on three LL benchmarks, averaged across 5 runs with fixed seeds. All compared methods are evaluated by First Accuracy A_1 (%), Finished Accuracy A_∞ (%) and Mean Average Accuracy A_m (%). Note that GDUMB implements with 3-fold epochs than other methods.

Method	**Split CIFAR-10**								
Fine-tune	19.39±0.03								
Joint	82.74±0.57								
Buffer Size	$\|\mathcal{M}\| = 300$			$\|\mathcal{M}\| = 500$			$\|\mathcal{M}\| = 1000$		
	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m
GDUMB	32.78±2.51	32.78±2.51	32.78±2.51	33.24±3.45	33.24±3.45	33.24±3.45	44.01±0.67	44.01±0.67	44.01±0.67
GEM	89.74±1.47	30.21±2.64	52.51±1.31	85.00±3.11	32.68±2.86	54.54±1.15	80.33±3.99	31.69±2.77	53.66±0.75
AGEM	93.71±0.30	20.16±0.81	44.89±0.61	93.71±0.21	20.12±0.43	45.20±0.75	93.81±0.31	20.14±0.59	44.98±0.54
HAL	88.34±0.82	27.81±1.43	51.60±1.69	89.37±1.28	33.62±3.19	54.38±1.21	89.46±0.76	36.41±2.91	57.82±1.25
GSS	94.04±0.49	32.13±1.22	53.54±1.08	94.25±0.24	34.81±1.35	55.84±0.88	93.55±0.49	41.13±3.00	59.64±0.67
MIR	94.07±0.19	36.05±2.42	56.45±1.28	94.02±0.15	39.08±1.94	59.37±1.21	94.00±0.31	46.24±2.65	62.49±0.82
ER	94.17±0.22	32.64±1.07	53.49±0.94	93.93±0.45	35.93±1.42	55.97±1.06	93.62±0.76	42.32±0.49	59.74±0.52
MetaSP (Ours)	**94.44±0.26**	**37.18±1.72**	**57.66±1.16**	**94.48±0.18**	**39.82±0.73**	**61.28±0.89**	**94.12±0.16**	**46.37±1.27**	**65.65±1.03**
Method	**Split CIFAR-100**								
Fine-tune	8.7±0.20								
Joint	51.05±0.94								
Buffer Size	$\|\mathcal{M}\| = 500$			$\|\mathcal{M}\| = 1000$			$\|\mathcal{M}\| = 2000$		
	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m
GDUMB	7.52±0.43	7.52±0.43	7.52±0.43	11.09±0.52	11.09±0.52	11.09±0.52	16.02±0.95	16.02±0.95	16.02±0.95
AGEM	77.76±0.33	8.654±0.11	21.33±0.22	77.76±0.30	8.74±0.09	21.29±0.24	78.04±0.23	8.66±0.02	21.38±0.22
HAL	65.81±1.85	8.92±0.93	21.59±0.56	67.67±1.59	11.89±1.10	23.55±0.71	67.15±2.06	14.09±1.56	26.86±0.70
GSS	77.84±0.42	11.70±0.50	25.05±0.46	77.59±0.32	13.78±0.32	27.23±0.26	77.07±0.12	16.80±0.72	30.11±0.47
MIR	78.56±0.75	11.69±0.31	25.09±0.33	78.45±0.70	13.90±0.17	27.35±0.26	78.03±0.65	17.58±0.72	30.54±0.66
ER	78.04±0.59	11.79±0.20	24.96±0.22	77.63±1.24	13.90±0.33	27.01±0.46	77.53±0.50	17.45±0.21	30.17±0.53
MetaSP (Ours)	**79.85±0.81**	**13.67±0.38**	**29.46±0.41**	**79.61±0.50**	**17.97±0.52**	**34.23±0.48**	**78.44±0.53**	**24.02±0.94**	**40.25±0.34**
Method	**Split Mini-Imagenet**								
Fine-tune	9.73±0.71								
Joint	37.74±1.28								
Buffer Size	$\|\mathcal{M}\| = 1000$			$\|\mathcal{M}\| = 2000$			$\|\mathcal{M}\| = 3000$		
	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m
GDUMB	8.06±0.50	8.06±0.50	8.06±0.50	10.62±0.42	10.62±0.42	10.62±0.42	12.32±0.34	12.32±0.34	12.32±0.34
AGEM	46.66±0.88	10.01±0.68	20.71±0.58	46.35±0.94	9.94±0.57	20.61±0.51	46.55±0.62	9.89±0.86	20.71±0.38
HAL	31.12±1.18	5.69±0.31	15.64±0.41	30.38±1.08	5.70±0.45	16.02±0.54	30.65±0.87	6.02±0.32	16.72±0.40
GSS	48.42±0.94	11.20±0.20	22.17±0.69	48.68±0.73	12.46±0.15	23.36±0.66	48.29±0.63	13.42±0.36	24.17±0.49
MIR	48.57±0.64	11.37±0.30	22.44±0.61	48.27±0.57	12.43±0.51	23.46±0.49	47.36±0.65	13.39±0.37	24.20±0.48
ER	48.57±0.71	11.46±0.46	22.24±0.58	48.17±0.59	12.73±0.20	23.18±0.43	47.74±1.03	13.31±0.27	23.89±0.75
MetaSP (Ours)	**51.62±0.65**	**13.49±0.24**	**25.03±0.53**	**50.56±0.44**	**15.93±0.51**	**26.94±0.49**	**49.95±0.60**	**17.65±0.78**	**28.47±0.39**

- **Split CIFAR-10** (Zenke et al., 2017) consists of 5 tasks, with 2 distinct classes each and 5000 exemplars per class, deriving from the CIFAR-10 dataset;
- **Split CIFAR-100** (Zenke et al., 2017) consists of splitting the original CIFAR-100 dataset into 20 disjoint subsets, each of which is considered as a separate task with 5 classes;
- **Split Mini-Imagenet** (Vinyals et al., 2016) is a subset of 100 classes from ImageNet (Deng et al., 2009), rescaled to size 32×32. Each class has 600 samples, randomly subdivided into training sets (80%) and test sets (20%). We splits Mini-Imagenet dataset into 5 disjoint tasks with 20 classes each.

Note that all datasets are built and fed to model without task identities, which is also known as class-incremental LL, the hardest scenarios in LL.

5.2 Evaluated Metrics

To better evaluate the SP trade-off, we propose and suggest to evaluate a LL algorithms with three metrics as follows. We use the sign function $\mathbf{1}(\cdot)$ to represent if the prediction of model is equal to the ground truth.

- **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 on learning new knowledge.
- **Finished 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 the Stability, *i.e.*, the capability on suppressing catastrophic forgetting.

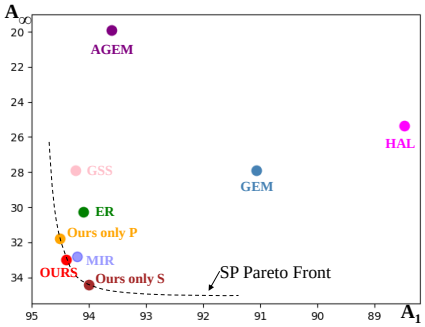


Figure 3: SP Pareto front.

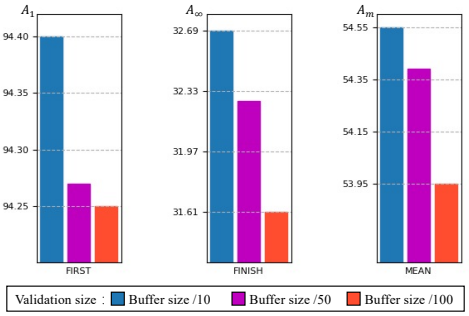


Figure 4: Validation size effect for Cifar-10 with $|\mathcal{M}| = 200$.

- **Mean Average Accuracy** ($A_{\text{m}} = \frac{1}{T}\sum_t \left(\frac{1}{t}\sum_{k\leq t}\sum_{x_i\in\mathcal{D}_k^{\text{tst}}} \mathbf{1}(y_i, \boldsymbol{\theta}_t(x_i))\right)$) This metric is computed along the LL process, indicating the SP trade-off after each task trained done.

5.3 Implementation Details

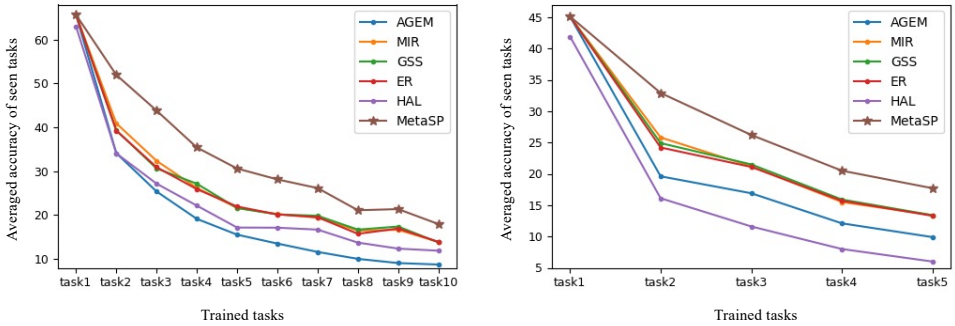
For all datasets, we employ ResNet18 (He et al., 2016) as a backbone. The Stochastic Gradient Descent (SGD) optimizer is with a learning rate of 0.01 for both pseudo and virtual update. The training has 5 epochs, among which the first three epochs are only for new data fine-tune, and the last two epochs are mixed training of new and old data. In all experiments, we keep the batch size 10 unchanged for both new data training and memory rehearsal in order to guarantee an equal number of updates. For rehearsal methods, we use another fixed batch size of 10 samples from memory buffer. At each step, the plasticity and stability validation sets are obtained by separately random sampling 10% of the seen data and 10% of memory buffer. All results are averaged over 5 fixed seeds from 1231 to 1235 for fairness.

5.4 Compared Methods

We compare our method against 7 rehearsal-based methods (GDUMB (Prabhu et al., 2020), GEM (Lopez-Paz & Ranzato, 2017), AGEM (Chaudhry et al., 2018), HAL (Chaudhry et al., 2021), GSS (Aljundi et al., 2019b), MIR (Aljundi et al., 2019a) and ER (Chaudhry et al., 2019)). 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). We test all compared methods combination using the same setting in their articles. All compared methods are evaluated by three aforementioned metrics, *i.e.*, First Accuracy A_1, Finished Accuracy A_∞ and Mean Average Accuracy A_{m}. The three metrics evaluate the capability to approach Plasticity, Stability and SP trade-off, respectively.

5.5 Main Comparison Results

In Table. 1, we show the main comparison among all compared and the proposed MetaSP. First of all, by controlling the example-level training through their Stability and Plasticity, the proposed MetaSP outperforms other methods on all metrics. With the memory buffer size growth, the all rehearsal-based LL gets better performance, while the advantages of MetaSP are more obvious. In terms of the First Accuracy A_1, indicating the ability of learning new tasks, MetaSP outperforms all other methods. This is because the new tasks always have larger gradient to dominant the update for all rehearsal-based LL, but MetaSP improves the example with positive effective and suppresses the negative-impact example. In terms of the Finished Accuracy A_∞, which is used to measure the forgetting, MetaSP has larger advantages than the compared methods because the examples may lead to forget are constrained. In terms of the Mean Average Accuracy A_{m}, which evaluates the SP trade-off, MetaSP shows its superiority because it can better balance the Stability and Plasticity on a SP Pareto front.



(a) Split Cifar-100 (b) Split Mini-Imagenet

Figure 5: Comparisons of LL training processes on two datasets.

5.6 Analysis on SP Pareto Optimum

In this paper, we propose to convert the Stability-aware and Plasticity-aware weights fusion as a MOO problem and leverage the MGDA to guarantee the final solution is a SP Pareto optimum. As shown in Fig. 3, we show the comparison of the First Accuracy and Finished Accuracy coordinate visualization for all compared methods. We block the weights fusion in MetaSP, and evaluate with only Stability-aware (Ours only S) and with only Plasticity-aware (Ours only P). Obviously, with only one weight, MetaSP can achieve even better Stability/Plasticity. However, the integration of SP yields a better SP trade-off compared to them. Moreover, compared with other methods, our proposed MetaSP outperforms other methods, and the existing methods cannot approach the SP Pareto front well as MetaSP.

5.7 Validation size effect and process visualization

In Fig. 4, 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 Stability, Plasticity and SP trade-off. We also visualize the LL training process on Split CIFAR-100 and Split Imagenet in Fig. 5. The first observation is that the proposed MetaSP outperforms other methods a lot, which means better SP trade-off along the LL training. Second, the forgetting cannot be eliminated even in the rehearsal-based LL. Even so, MetaSP offers an elegant add-in for the rehearsal-based LL methods and can further improve their performance.

6 Conclusion and Future Work

In this paper, we proposed to analyze the Stability-Plasticity (SP) dilemma in rehearsal-based Lifelong Learning in the aspect of example difference. That is, different examples contribute to the Stability and Plasticity differently. To achieve that, we first summarized and defined the SP in probability and decompose the task-level SP into example-to-example SP, *i.e.*, the influence. Inspired by the Influence Function, we evaluated the example-level influence via small weight perturbation instead of the computationally expensive Leave-one-out strategy. Furthermore, we proposed a simple yet effective MetaSP algorithm to avoid the computation of Hessian in Influence Function. At each iteration in LL training, MetaSP builds two weights for Stability and Plasticity. Two validation sets for old tasks sampled from memory buffer and for new tasks from the current training are used to updates the weights. Finally, the two weights will be combined via a SP Pareto optimal factors and support the regular model update. The experimental results on three popular LL datasets verified the effectiveness of the proposed MetaSP. In the future, we plan to promote the example-level SP to the general LL and online LL.

REFERENCES

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Proceedings of Advances in Neural Information Processing Systems*. 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Proceedings of Advances in Neural Information Processing Systems*, 2019b.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of International Conference on Learning Representations*, 2018.

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.

Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

R Dennis Cook and Sanford Weisberg. Residuals and influence in regression. 1982.

Kalyanmoy Deb and Himanshu Gupta. Searching for robust pareto-optimal solutions in multi-objective optimization. In *Proceedings of the International Conference on Evolutionary Multi-criterion Optimization*, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

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

Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. Learning what data to learn. *arXiv preprint arXiv:1702.08635*, 2017.

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.

Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical methods of operations research*, 2000.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Cision and Pattern Recognition*, 2016.

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*, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference on Machine Learning*, 2017.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Jia Xu, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, 2017.

Fan Lyu, Shuai Wang, Wei Feng, Zihan Ye, Fuyuan Hu, and Song Wang. Multi-domain multi-task rehearsal for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

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

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. 2017.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of International Conference on Machine Learning*, 2018.

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.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Proceedings of Advances in Neural Information Processing Systems*, 2018.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Proceedings of Advances in neural information processing systems*, 2016.

Tianyang Wang, Jun Huan, and Bo Li. Data dropout: Optimizing training data for convolutional neural networks. In *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*, 2018.

Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of International Conference on Machine Learning*, 2017.

Yongjian Zhong, Bo Du, and Chang Xu. Learning to reweight examples in multi-label classification. *Neural Networks*, 2021.

A Appendix

A.1 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.*, $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right)$. First, we set a small weight to a specific example x

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

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

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

Algorithm 2: LL training procedure with MetaSP.

Input: Training sets $\{\mathcal{D}_t^{\text{trn}}\}_{t=1}^{T}$; Initial parameters $\boldsymbol{\theta}_0$; Memory buffer $\mathcal{M}$; Step size α
Output: $\boldsymbol{\theta}_T$; `// The final trained model after` T `tasks`

for $t \leq T$ **do**
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1}$;
  foreach $\mathcal{B}_{\text{new}} \sim \mathcal{D}_t^{\text{trn}}$ **do**
    $\mathcal{M}_t \leftarrow$ reservoir sample from $\mathcal{B}_{\text{new}}$;
    $\mathcal{B}_{\text{old}} \sim \mathcal{M}$;
    if $t = 1$ **then**
      Model update $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \alpha \cdot \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t)$;
    else
      `/* Stability-aware and Plasticity-aware pseudo update */`
      Zero-initialize the S-aware and P-aware weights $\mathbf{w}_{\text{S}}$ and $\mathbf{w}_{\text{P}}$;
      Pseudo update $\tilde{\boldsymbol{\theta}}_{\text{S}} = \boldsymbol{\theta}_t - \alpha \cdot \mathbf{w}_{\text{S}}^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta})$ for Stability;
      Pseudo update $\tilde{\boldsymbol{\theta}}_{\text{P}} = \boldsymbol{\theta}_t - \alpha \cdot \mathbf{w}_{\text{P}}^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta})$ for Plasticity;
      `/* SP weights update */`
      Update the Stability-aware weights $\mathbf{w}_{\text{S}} = -\nabla_{\mathbf{w}_{\text{S}}} \ell(\mathcal{V}_{\text{old}}, \tilde{\boldsymbol{\theta}}_{\text{S}})$;
      Update the Plasticity-aware weights $\mathbf{w}_{\text{P}} = -\nabla_{\mathbf{w}_{\text{P}}} \ell(\mathcal{V}_{\text{new}}, \tilde{\boldsymbol{\theta}}_{\text{P}})$;
      `/* Weights fusion for SP trade-off */`
      Get the optimal fusion hyper-parameter γ^* via Eq. (14);
      Weights fusion $\mathbf{w}^* = \gamma^* \mathbf{w}_{\text{S}} + (1 - \gamma^*) \mathbf{w}_{\text{P}}$;
      Model update $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \alpha \cdot (\mathbf{w}^*)^{\top} \nabla_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}_{\text{old}} \cup \mathcal{B}_{\text{new}}, \boldsymbol{\theta}_t)$;
    end
  end
  $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_t$
end

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

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

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 $\mathbf{w} = [\epsilon_1, \epsilon_2, \cdots, \epsilon_n]^{\top}$ 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

$$\begin{aligned} \mathbf{I}(\mathcal{B}, \mathcal{V}) &= \frac{\partial l(\mathcal{V}, \hat{\boldsymbol{\theta}})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{0}} \\ &= \frac{\partial l(\mathcal{V}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{w},\mathcal{B}}}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{0}} \\ &= -\frac{1}{|\mathcal{V}|} \sum_{x_i \in \mathcal{V}} \nabla_{\boldsymbol{\theta}} l(x_i, \hat{\boldsymbol{\theta}}) \mathbf{H}^{-1} \nabla^{\top}_{\boldsymbol{\theta}} \mathbf{L}(\mathcal{B}, \hat{\boldsymbol{\theta}}), \end{aligned}$$

where

$$\mathbf{H} = \frac{1}{|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} \nabla^2_{\boldsymbol{\theta}} \ell(x_i, \hat{\boldsymbol{\theta}}_{\epsilon}),$$

Table 2: Time (s) comparison with SOTA methods.

Batch Size	ER	GSS	AGEM	HAL	MIR	GEM	MetaSP (val=10)	MetaSP (val=30)	MetaSP (val=50)
1	0.0131	0.0153	0.0293	0.0433	0.0775	0.2901	0.2438	0.2474	0.2507
5	0.0144	0.0172	0.0297	0.0459	0.0845	0.3445	0.2844	0.2950	0.3052
10	0.0160	0.0208	0.03216	0.0479	0.0958	0.3564	0.3335	0.3449	0.3527

Table 3: Notation related to the paper.

Symbol	Description
$\boldsymbol{\theta}_t$	The model trained after the t-th task
α	Optimization step size
$\mathcal{D}_t$	$\{(x_t^{(n)}, y_t^{(n)})\}_{n=1}^{N_t}$; The t-th dataset
$\mathcal{D}_t^{\text{trn}}$	The training set of the t-th dataset
$\mathcal{D}_t^{\text{tst}}$	The testing set of the t-th dataset
$\mathcal{M}_t$	The memory buffer sampled from the t-th training set
$\mathcal{M}$	$\cup_{k<t}\mathcal{M}_k$
$\mathcal{B}_{\text{old}}$	a mini-batch sampled from $\mathcal{M}$
$\mathcal{B}_{\text{new}}$	a mini-batch sampled from $\mathcal{D}_t^{\text{trn}}$
$p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_t)$	Probability of $\mathcal{D}_k^{\text{tst}}$ conditioned on $\boldsymbol{\theta}_t$
S_t^k	$p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_t) - p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_k), \quad k < t$ Stability
P_t	$p(\mathcal{D}_t^{\text{tst}}\|\boldsymbol{\theta}_t) - p(\mathcal{D}_t^{\text{tst}}\|\boldsymbol{\theta}_{t-1})$ Plasticity
$I(x^{\text{trn}}, x^{\text{tst}})$	$\frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\Big\|_{\epsilon=0}$
$\mathbf{H}$	$\nabla^2_{\boldsymbol{\theta}} \ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}})$; Hessian assumed positive define
$\hat{\theta}_{\text{S}}$	Stability-aware parameters
$\hat{\theta}_{\text{P}}$	Plasticity-aware parameters
$\mathbf{w}_{\text{S}}$	Stability-aware weights
$\mathbf{w}_{\text{P}}$	Plasticity-aware weights
γ	weight fusion factor
l	Loss for an exmaple or average loss for a mini-batch
$\mathbf{L}$	Loss vector for a mini-batch

$$\frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{w},\mathcal{B}}}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{0}} = -\mathbf{H}^{-1}\left[\frac{\partial \mathbf{L}}{\partial \boldsymbol{\theta}}\right]^{\top}.$$

A.2 Proof of SP decomposition

In this section, we prove the SP decomposition. First, we decompose the Stability as

$$\begin{aligned}
S_t^k &= p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_t) - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k) \\
&= \prod_{x_k^{\text{tst}} \in \mathcal{D}_k^{\text{tst}}} p(x_k^{\text{tst}}|\boldsymbol{\theta}_t) - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k) \\
&= \prod_{x_k^{\text{tst}} \in \mathcal{D}_k^{\text{tst}}} \prod_{x_t^{\text{trn}} \in \mathcal{D}_t^{\text{trn}}} p(x_k^{\text{tst}}|\boldsymbol{\theta}_t, x_t^{\text{trn}}) - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k) \\
&= \prod_{x_k^{\text{tst}} \in \mathcal{D}_k^{\text{tst}}} \prod_{x_t^{\text{trn}} \in \mathcal{D}_t^{\text{trn}}} p(x_k^{\text{tst}}|x_t^{\text{trn}}) \frac{p(\boldsymbol{\theta}_t|x_k^{\text{tst}}, x_t^{\text{trn}})}{p(\boldsymbol{\theta}_t|x_t^{\text{trn}})} - p(\mathcal{D}_k^{\text{tst}}|\boldsymbol{\theta}_k) \\
&\approxeq \prod_{x_k^{\text{tst}} \in \mathcal{D}_k^{\text{tst}}} \prod_{x_t^{\text{trn}} \in \mathcal{D}_t^{\text{trn}}} p(x_k^{\text{tst}}|x_t^{\text{trn}}) + \sigma_{\text{S}},
\end{aligned}$$

where $\sigma_{\mathrm{S}} = -p(\mathcal{D}_k^{\mathrm{tst}}|\boldsymbol{\theta}_k)$ is a constant. Similarly, the Plasticity can be decomposed into

$$P_t \approxeq \prod_{x_t^{\mathrm{tst}} \in \mathcal{D}_t^{\mathrm{tst}}} \prod_{x_t^{\mathrm{trn}} \in \mathcal{D}_t^{\mathrm{trn}}} p(x_t^{\mathrm{tst}}|x_t^{\mathrm{trn}}) + \sigma_{\mathrm{P}}.$$

where $\sigma_{\mathrm{P}} = -p(\mathcal{D}_t^{\mathrm{tst}}|\boldsymbol{\theta}_{t-1})$ is a constant.

The above two equations indicate that the Stability and Plasticity can be represented as the example-to-example influence. Such influence only depends on the training example and testing example.

A.3 Complete MetaSP Algorithm

In Alg. 2, we show the complete training algorithm for MetaSP. For T tasks, we train them in sequence with the initial parameters $\boldsymbol{\theta}$. The memory is sampled by the Reservoir sampling strategy. When $t = 1$, we update as a usual learning algorithm. When $t > 1$, the proposed MetaSP is used to enhance the rehearsal training in LL. The computed weights for SP trade-off are used to weight the training in example-level.

A.4 Time comparison with SOTA methods

We list the training time (s) of a mini-batch update overhead in Table 2 for both task-level methods and our proposed MetaSP in Cifar10 dataset. The proposed MetaSP will compute the backward on the weights on examples, thus the time highly depends to the batch size. With a large batch-size, integrating the pseudo update, backward on weights and weights fusion, our method takes more time than other methods except less than GEM. By reducing batch size, the proposed algorithm can be dramatically speed up. In detail, the time complexity to obtain minibatch samples influence on S is $O(nq + vq)$ (q is the number of parameter, n is the batch size and v is the validation size). In contrast, Influence Function requires $O(nq^2 + q^3)$ operations causing the inverse of Hessian.

A.5 Notation

We list the mentioned notation in the Table 3.