Bi-Objective Continual Learning: Learning 'New' While Consolidating 'Known'

Xiaoyu Tao,[1] **Xiaopeng Hong,**[1,3*] **Xinyuan Chang,**[2] **Yihong Gong**[2]
[1]Faculty of Electronic and Information Engineering, Xi'an Jiaotong University
[2]School of Software Engineering, Xi'an Jiaotong University
[3]Research Center for Artificial Intelligence, Peng Cheng Laboratory
{txy666793, cxy19960919}@stu.xjtu.edu.cn, {hongxiaopeng, ygong}@mail.xjtu.edu.cn

Abstract

In this paper, we propose a novel single-task continual learning framework named Bi-Objective Continual Learning (BOCL). BOCL aims at both consolidating historical knowledge and learning from new data. On one hand, we propose to preserve the old knowledge using a small set of *pillars*, and develop the *pillar consolidation* (PLC) loss to preserve the old knowledge and to alleviate the *catastrophic forgetting* problem. On the other hand, we develop the *contrastive pillar* (CPL) loss term to improve the classification performance, and examine several data sampling strategies for efficient onsite learning from 'new' with a reasonable amount of computational resources. Comprehensive experiments on CIFAR10/100, CORe50 and a subset of ImageNet validate the BOCL framework. We also reveal the performance accuracy of different sampling strategies when used to finetune a given CNN model. The code will be released.

Introduction

In recent years, Convolutional Neural Networks (CNNs) have achieved superior performances in a broad range of computer vision tasks (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2015; Wang et al. 2010; Ren et al. 2015; Redmon et al. 2015; Cheng et al. 2016; Shun et al. 2016; Long, Shelhamer, and Darrell 2015; Chen et al. 2017; Liu et al. 2017; Deng et al. 2018; Tusng-Yu, Aruni, and Subhransu 2015; Wei et al. 2018a; 2018b). Nonetheless, when a CNN model is put into a continuous use, it will inevitably encounter new data it has never seen before, and may produce erroneous recognition results. For a model to stand the test of time, it is crucial to learn from new data and evolve to adapt to the changes continuously. This ability is referred to as *continual learning* (CL) (Parisi et al. 2019) in the literature.

Recent CL works (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017; Li and Hoiem 2018; Lee et al. 2017; Lopez-Paz and others 2017; Chaudhry et al. 2018) usually conduct their research under a *multi-task* (MT) setting, where the model is required to learn a sequence of independent tasks. Each task is assigned with a specific classification layer for prediction during the test phase. Such setting requires task identity for selecting the task-specific classifier, which is hard to be satisfied in practical where task information is typically unknown. Another series of work (Rebuffi et al. 2017; Castro et al. 2018; Wu et al. 2019) carries out research under a *single-task*, *class-incremental-learning* (ST-CIL) setting, where the model meets a sequence of new class training samples and is required to incrementally learn a unified classification layer for all encountered classes. This CIL setting is based on the assumption that new training data comes from new classes.

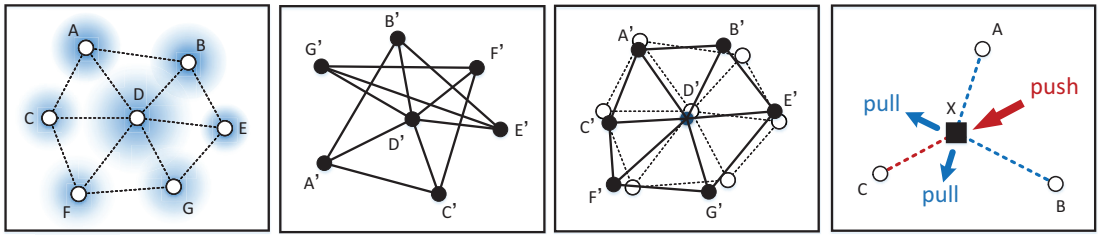
Table 1: A comparison of the CL problem settings

Setting	task-free	classifiers	NC	NI
MT	no	task-specific	perhaps	perhaps
ST-CIL	yes	unified	yes	no
ST-NIL	yes	unique	no	yes

In practical use, the deployed models are often exposed to new instances of existing classes with new patterns. It is crucial for the model to learn from them to improve the recognition performance continuously. This scenario is common in real-world applications, but is seldom explored by current CL research. In this paper, we organize our study based on this scenario. We adopt a *single-task*, *new-instance learning* (ST-NIL) setting that aims to improve the recognition performance on existing classes by continuously learning on new instances. ST-NIL is originally proposed in (Maltoni and Lomonaco 2018) with a naïve baseline that simply finetunes the model on new instances. Table 1 compares the differences of the three settings mentioned above.

To accomplish CL, a straightforward approach is to retrain the model on both old and new data. It is inefficient or even prohibitive in practical, especially for embedded systems where resources are too limited to perform retraining. Alternatively, we can finetune the model on new data. However, this is prone to *catastrophic forgetting* (French 1999) that the performance on old data deteriorates drastically. Hence current CL works mainly focus on mitigating forgetting. Some works attempt to preserve old tasks' knowledge by imposing regularization to the network weights (Kirk-

*Corresponding author



(a) (b) (c) (d)

Figure 1: Conceptual visualization of the PLC and CPL loss terms. (a) The pillars $\{A, B, \cdots, G\}$ maintain the key memories of the historical knowledge in the feature space. The edge between two pillars indicates they are topologically adjacent. (b) Finetuning on new data breaks the topology and causes catastrophic forgetting. (c) The PLC term fastens the pillar points to stabilize the memories of old knowledge and alleviate forgetting. (d) The CPL term pushes the new training example to its nearest matching pillar with the same class label (C), while pulling it away from those with different labels (A and B).

patrick et al. 2017; Zenke, Poole, and Ganguli 2017). These methods are typically based on certain naïve assumptions about the weights' posterior distribution, which may not hold in complex scenarios. Another series of work introduces the idea of *knowledge distillation* (Hinton, Vinyals, and Dean 2015) into CL, where a distillation loss is used for maintaining the old knowledge contained in the output logits. As a supplement, they usually adopt the *rehearsal* (Lopez-Paz and others 2017) technique that stores or generates a small set of samples representative of old data for mitigating forgetting.

In this paper, we focus on ST-NIL setting and propose a novel CL framework named Bi-Objective Continual Learning (BOCL), which aims at both *consolidating historical knowledge* and *learning from new data*, as shown in Figure 1. On one hand, to mitigate forgetting, we propose to preserve the old knowledge using *pillars*, which are linked to the representative feature points of the historical data, and develop the *Pillar Consolidation* (PLC) loss term to penalize the shift of pillars in the feature space during the subsequent learning sessions. On the other hand, to efficiently learn new patterns from new data, we propose the *Contrastive Pillar* (CPL) loss term that pushes a new training example to its nearest pillar with the same labels, while pulling it away from those with different labels. We also examine several data sampling strategies to tackle the endless new data amassed over time. In practical applications such as 'smart album' on mobile devices, there is no bound on the amount of new data as time goes by, and thus their size may go far beyond the system limits in memory and computational power. Therefore, an effective sampling scheme must be provided to finetune the model using reasonable amount of memory and computational resources.

We perform comprehensive experiments on CIFAR10/100, CORe50 and a subset of ImageNet, and demonstrate the effectiveness of the proposed BOCL framework for the ST-NIL setting. We also reveal the performance of different sampling strategies when used for finetuning a given CNN model. To summarize, the main contributions of the paper include:

- We focus on the challenging, practical ST-NIL setting and propose a bi-objective continual learning (BOCL) framework to reconcile both *learning from new data* and *consolidating historical knowledge*.
- We propose to preserve the old knowledge using a small set of *pillars*, and develop the PLC term in the loss function to alleviate the *catastrophic forgetting* problem.
- We develop the CPL loss term to improve the classification performance of the onsite 'learning from new' phase using the pillars. Moreover, we examine several data sampling strategies for learning with reasonable amount of computational resources.

Related Work

Multi-task (MT) methods. A series of works adopts the MT setting for CL. Most of them focus on overcoming *catastrophic forgetting*. An earlier work LwF (Li and Hoiem 2018) firstly introduces the idea of *knowledge distillation* (Hinton, Vinyals, and Dean 2015) into multi-task CL to mitigate forgetting. It uses the distillation loss to stabilize the old tasks' output during finetuning on new task. EWC (Kirkpatrick et al. 2017) and its variant, SI (Zenke, Poole, and Ganguli 2017) attempt to alleviate forgetting by imposing constraint to the network weights related to old tasks. They approximate the posterior distribution of the weights as Gaussian distribution to measure their importance. IMM (Lee et al. 2017) incrementally matches the moment of the posterior distribution of the networks trained on the old and new tasks, respectively. It requires to average two networks for prediction. GEM and its accelerated version, A-GEM, use an external memory to store representative examples of historical tasks. The pre-stored examples are used for computing and constraining old tasks' losses.

In contrast to these MT methods, we focus on the single-task (ST) setting, which treats the whole learning process as one task and use a unique classification layer for all encountered data. For comparison purpose, we adapt some representative MT methods to the ST-NIL setting, including EWC, SI, IMM and A-GEM.

Single-task class-incremental learning (ST-CIL) methods. The ST-CIL methods incrementally learn a unified clas-

sification layer to recognize all encountered classes. Many of them adopt the distillation technique to alleviate forgetting on old classes, where the distillation loss is applied to the output logits/probabilities corresponding to old classes (Rebuffi et al. 2017; Castro et al. 2018; Wu et al. 2019).

These distillation-based ST-CIL methods can not be directly used for the ST-NIL setting, as new training data has the same classes as the old one and the number of output logits remains unchanged during the whole training process. For comparison, we adapt the representative EEIL (Castro et al. 2018) method to ST-NIL setting by using the examples of old data for computing the distillation loss.

Problem Description

We define the *single-task, new-instance learning* (ST-NIL) problem setting as follows. The entire set of training data is divided into N independent training sessions $\mathcal{S} = \{S_1, S_2, \cdots, S_N\}$, where $S_i = \{(\mathbf{X}_j, y_j)\}_{j=1}^{|S_i|}$, $\mathbf{X}_j$ and y_j are the training image and label, respectively. Each session contains new training patterns of the same classes from a predefined, common label space $L = \{1, \cdots, C\}$, where C is the number of classes. The model is learned on all training sessions in $\mathcal{S}$ in a sequential manner with a unique classification layer, and gradually improve the recognition performance on the common test set $\mathcal{T}$.

Bi-Objective Continual Learning Framework

Given an input $\mathbf{X}$, a CNN can be seen as a composition of a feature extractor $f(\mathbf{X};\theta)$ and a cascaded classification layer that predicts the probabilities over classes. To accomplish the ST-NIL setting, we start by training the network on S_1 from scratch with the cross-entropy loss. Then, we finetune the model on the subsequent sessions $\{S_2, \cdots, S_n\}$ one by one. Directly finetuning on $S_i, i > 1$ would degrade the performance on previous sessions $\{S_1, \cdots, S_{i-1}\}$, due to catastrophic forgetting. To address this problem, we try to preserve the old knowledge using a small set of *pillars* $\mathcal{P} = \{\mathbf{p}_k\}_{k=1}^{|\mathcal{P}|}$. A *pillar* is a representative point in the feature space defined by $f(\cdot;\theta)$, which can be seen as a key memory of the knowledge. It takes the form of a 4-tuple: $\mathbf{p}_k = (\mathbf{a}_k, \boldsymbol{\omega}_k, \mathbf{I}_k, b_k)$, where k denotes the index, $\mathbf{a}_k \in \mathbb{R}^n$ is the n-dim feature vector, $\boldsymbol{\omega}_k \in \mathbb{R}^n$ is a vector that stores important statistics for $\mathbf{a}_k$. We additionally store a sample image $\mathbf{I}_k$ and its corresponding label b_k to compute the active values of $\mathbf{a}_k$, which is $f(\mathbf{I}_k;\theta)$.

By using the *pillars* to represent the knowledge, we formulate ST-NIL as the bi-objective optimization problem. The overall loss function at the i-th session is defined as:

$$\ell(S_i, \mathcal{P}_{i-1}; \theta_i) = \ell_{lfn}(S_i, \mathcal{P}_{i-1}; \theta_i) + \lambda \ell_{plc}(\mathcal{P}_{i-1}; \theta_i). \quad (1)$$

In the above equation, ℓ_{lfn} is the loss term for learning new knowledge from new data, and ℓ_{plc} is the *Pillar Consolidation* (PLC) term for consolidating the old knowledge. We use θ_i to denote the active values of θ at session i. The hyper-parameter $\lambda > 0$ balances the strength of ℓ_{lfn} and ℓ_{plc}. We name the overall framework as the *Bi-Objective Continual Learning* (BOCL) framework. In the following subsections, we will elaborate the components of BOCL in detail.

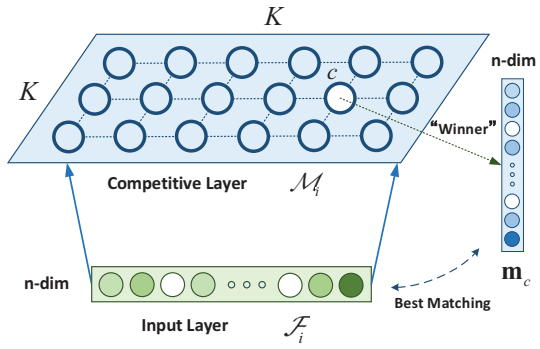


Figure 2: Illustration of the *self-organizing map* (SOM).

Pillar Set Generation

After training on session S_i, we extract the set of feature vectors $\mathcal{F}_i = \{f(\mathbf{X};\theta_i) | \mathbf{X} \in S_i\}$, and generate the *pillar* set $\mathcal{P}_i$ on $\mathcal{F}_i$. To achieve this purpose, we try to map high-dimensional $\mathcal{F}_i$ to low-dimensional $\mathcal{P}_i$ using a *self-organizing map* (SOM) (Kohonen 2013). SOM is a neural network that produces a low-dimensional (typically 2D), discretized representation of the input space of the training data. As shown in Figure 2, it consists of an input layer, which is fed with the feature vectors in $\mathcal{F}_i$, and a competitive layer, which contains $K \times K$ nodes organized in a regular 2D grid $\mathcal{M}_i = \{\mathbf{m}_k^{(i)}\}_{k=1}^{K^2}$, where $\mathbf{m}_k^{(i)}$ denotes the centroid vector of node k. SOM maintains the neighbourhood relationship of the centroid vectors in topology space: similar centroids are located closer while less similar ones gradually farther away. This property makes SOM efficient for searching nearest neighbours. Given an input vector $\mathbf{f} \in \mathcal{F}_i$, SOM finds the 'winner' node c hit by $\mathbf{f}$, by picking the centroid vector $\mathbf{m}_c$ closest to $\mathbf{f}$:

$$c = \arg\min_k \|\mathbf{f} - \mathbf{m}_k\|_2^2. \quad (2)$$

When training SOM on $\mathcal{F}_i$, $\mathbf{m}_c$ is updated by averaging all feature vectors that hit node c using the batching-computing technique in (Kohonen 2013). Therefore, each centroid can represent the set of the feature vectors closest to it, which reflects the distribution of the neighbourhood feature space. Moreover, we can adapt SOM to new data easily and efficiently by updating the centroids. All these properties make SOM a good choice for generating *pillars*.

We train a SOM of size $K \times K$ to generate K^2 *pillars*, where each SOM node corresponds to a *pillar* point. For the k-th *pillar* $\mathbf{p}_k = (\mathbf{a}_k, \boldsymbol{\omega}_k, \mathbf{I}_k, b_k)$ generated by the k-th SOM node, we define $\boldsymbol{\omega}_k$ as the weighting vector that measures the importance of each dimension of $\mathbf{a}_k$. Some dimensions of $\mathbf{a}_k$ may encode semantic attributes of high intra-class diversity (e.g. color), and we should allow free adaptations of these dimensions in order to learn new patterns better. This is achieved by assigning a smaller weight to these dimensions when consolidating them for mitigating forgetting. To get $\boldsymbol{\omega}_k$, we compute the variance for each dimension of the feature vectors that hit SOM node k, which we have:

$$\boldsymbol{\omega}_k = (\frac{1}{\upsilon_k^1}, \frac{1}{\upsilon_k^2}, \cdots, \frac{1}{\upsilon_k^n}), \quad (3)$$

Algorithm 1 Generating the *pillar* set.

Input: S_i, $\mathcal{F}_i$, $\mathcal{M}_{i-1}$, $\mathcal{P}_{i-1}$, K. $\mathbf{c}_{i-1} = \{c_k^{(i-1)}\}_{k=1}^{K^2}$ is for counting the hits of each SOM node by the input.
Output: $\mathcal{M}_i$, $\mathcal{P}_i$, $\mathbf{c}_i$

Initialize an SOM of $K \times K$ nodes $\mathcal{M}_i$ using the technique in (Kohonen 2013).
Train $\mathcal{M}_i$ on $\mathcal{F}_i$.
for $k = 1$ *to* K^2 **do**
  Count the hits of node k and store it in $c_k^{(i)}$.
  Compute $\boldsymbol{\omega}_k$ using Eq. (3).
end for
// merge $\mathcal{M}_{i-1}$ into $\mathcal{M}_i$.
for $k = 1$ *to* K^2 **do**
  $\mathbf{m}_k^{(i)} \leftarrow (c_k^{(i-1)}\mathbf{m}_k^{(i-1)} + c_k^{(i)}\mathbf{m}_k^{(i)})/(c_k^{(i-1)} + c_k^{(i)})$.
  $c_k^{(i)} \leftarrow (c_k^{(i-1)} + c_k^{(i)})$.
end for
// mix old *pillars*' feature vectors and $\mathcal{F}_i$, and re-sample *pillars* via SOM.
$\mathcal{D} \leftarrow \{\mathbf{a} | \ \forall(\mathbf{a}, \boldsymbol{\omega}, \mathbf{I}, b) \in \mathcal{P}_{i-1}\} \cup \mathcal{F}_i$.
for $k = 1$ *to* K^2 **do**
  Find the index of the nearest sample: $j = \arg\min_s \|\mathbf{f}_s - \mathbf{m}_k^{(i)}\|_2^2, \ \forall \mathbf{f}_s \in \mathcal{D}$.
  $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{(\mathbf{f}_j, \boldsymbol{\omega}_j, \mathbf{X}_j, y_j)\}$.
  $\mathcal{D} \leftarrow \mathcal{D} \setminus \{\mathbf{f}_j\}$.
end for

where υ_k^j denotes the variance at the j-th feature vector dimension. The *pillar* set generation algorithm is shown in Algorithm 1. It takes the previously generated *pillars* $\mathcal{P}_{i-1}$ and SOM $\mathcal{M}_{i-1}$ as the input, and outputs the newly generated *pillars* $\mathcal{P}_i$ and the updated SOM $\mathcal{M}_i$.

Pillar Consolidation

Given the *pillar* set $\mathcal{P}_{i-1} = \{\mathbf{p}_k\}_{k=1}^{|\mathcal{P}_{i-1}|}$ constructed from the previous sessions, PLC accomplishes the objective of preserving the old knowledge by constraining the *pillars* to remain as stable as possible in the feature space. Expressing this in math form, we have:

$$\ell_{plc}(\mathcal{P}_{i-1}; \theta_i) = \sum_{k=1}^{|\mathcal{P}_{i-1}|} \|\boldsymbol{\omega}_k \odot f(\mathbf{I}_k; \theta_i) - \boldsymbol{\omega}_k \odot \mathbf{a}_k\|_2^2, \quad (4)$$

where $\odot$ denotes the element-wise multiplication between the two vectors of the same dimension. We re-weight each dimension of the *pillar*'s feature vectors $f(\mathbf{I}_k; \theta_i)$ (active) and $\mathbf{a}_k$ (pre-stored) using $\boldsymbol{\omega}_k$ computed in Eq. (3), allowing those dimensions of high diversity freely adapt to new training patterns contained in new instances.

Figure 1(a-c) illustrates the motivation of the proposed PLC term. A small set of *pillar* points represents the topological structure of the historical data distribution in the feature space, and can be considered as the key memories of the historical knowledge. The conventional way of finetuning the model on new data breaks this topological structure, causing the catastrophic forgetting problem. PLC strives to alleviate catastrophic forgetting by maintaining this topological structure in the feature space.

Onsite Learning with Contrastive Pillar Term

Given S_i and the *pillar* set $\mathcal{P}_{i-1}$, we enforce the learned feature of a new training instance $(\mathbf{X}_j, y_j) \in S_i$ to be as close as possible to its nearest matching *pillar* with the same class label $b_n = y_j$, and as far away as possible to other pillars with different labels. Figure 1(d) illustrates the motivation of the proposed CPL term. Cooperating CPL term with the cross-entropy loss, we formulate ℓ_{lfn} as:

$$\ell_{lfn}(S_i, \mathcal{P}_{i-1}; \theta_i) = \sum_{j=1}^{|S_i|}(-\log \hat{p}_{y_j}(\mathbf{X}_j; \theta_i) + \gamma \frac{\|f(\mathbf{X}_j; \theta_i) - \mathbf{a}_m\|_2^2}{\sum_{l \neq y_j} \sum_{n \in \mathcal{N}_{j,l}} \|f(\mathbf{X}_j; \theta_i) - \mathbf{a}_n\|_2^2}). \quad (5)$$

The first term on the right-hand side of Eq. (5) is the cross-entropy loss, where $\hat{p}_{y_j}(\mathbf{X}_j; \theta_i)$ is the estimated probability of the label y_j. The second term is the proposed CPL term, where $\mathbf{a}_m$ is the feature vector of the nearest matching *pillar* of $f(\mathbf{X}_j; \theta_i)$ with the label $b_m = y_j$, and $\mathcal{N}_{j,l}$ is the index set of the K_p nearest neighbours of the feature $f(\mathbf{X}_j; \theta_i)$ subject to $\forall t \in \mathcal{N}_{j,l}, \ \mathbf{a}_t \in \mathcal{P}_{i-1}$ and $b_t = l$. As discussed above, we use SOM to select nearest neighbours efficiently. γ is used for controlling the strength of the CPL term.

Sampling Strategies for Onsite Learning

Generally speaking, a large volume of training data are desirable to guarantee a high-quality model training. Thus existing methods usually use all the new data for 'present' session. Nonetheless, it is problematic to learn from unlimited data stream in mobile devices with limited computational power and energy consumption. Therefore, to provide more computationally efficient solutions, we sample a subset $\tilde{S}_i$ from S_i using some strategy $\mathcal{R}$ and finetune the network θ_i only on $\tilde{S}_i$, where $\tilde{S}_i = \mathcal{R}(S_i)$. As the data distribution of $\tilde{S}_i$ may vary from that of S_i, the recognition performance of the trained model is affected by different sampling strategies. Therefore, we explore several sampling strategies $\mathcal{R}$ for on-site learning and compare their performance for CL.

Random sampling ($\mathcal{R}_{rnd}$): We finetune the model using K_s examples randomly sampled from S_i as the baseline sampling strategy.

Sampling hard examples ($\mathcal{R}_{hard}$): We finetune the model only by hard examples, which are mined by computing the classification loss for each $(\mathbf{X}, \mathbf{y}) \in S_i$ using the network θ_{i-1} trained on S_{i-1}, and then selecting K_s examples with the highest losses.

Sampling misclassified examples ($\mathcal{R}_{err}$): A straightforward way to boost the performance of an existing model is to learn from misclassified examples (Dalal and Triggs 2005). We randomly pick a set of K_s examples $\{(\mathbf{X}_j, y_j)\}_{j=1}^{K_s} \subset S_i$ that are misclassified by the network θ_{i-1} trained on S_{i-1}, and finetune the model only using these K_s examples.

Experiments

Experimental Setups

We conduct experimental evaluations using the ST-NIL problem setting. We use CIFAR10/100 (Krizhevsky and

Hinton 2009), CORe50 (Lomonaco and Maltoni 2017) and a 1000-class subset of ImageNet (Deng et al. 2009) as the benchmark datasets.[1]

CIFAR10/100 dataset. Both datasets contain 60,000 natural RGB images of the size 32×32, including 50,000 training and 10,000 test images. CIFAR10 has 10 classes, while CIFAR100 has 100 classes. We further split the training images into 5 sessions, and apply different illumination and saturation for each subsequent session as new training patterns.

CORe50 dataset. It consists of 164,866 images of 50 domestic objects, which are split into eleven sessions. Three sessions (#3, #7 and #10) are selected for test and the remaining ones for training. Each session contains about 15,000 RGB-D images of the size 128×128.

SubImageNet. We select a 1000-class subset from the original ImageNet-1k dataset as the *subImageNet*. It contains 250,000 training images that are split into 5 training sessions. Each image is resized and cropped to 224×224. We use the original 50,000 validation images as the test set.

We use the popular ResNet18 (He et al. 2015) (and the thumbnail version for CIFAR) as the baseline CNN model. For CIFAR10/100, the models are trained on each session using SGD with a mini-batch size of 100. When training on S_1, we set the initial learning rate to 0.01, and decrease it to 0.001 after 15 epochs. We stop training when the cross-entropy loss becomes stable, which takes about 20 epochs in total. Then, we use a constant learning rate of 0.001 and finetune each subsequent session for 20 epochs. For CORe50, we adopt the training setting in (Lomonaco and Maltoni 2017). While for subImageNet, we train the models on each session for 60 epochs with a larger initial learning rate 0.1. After training on S_i, we evaluate the model θ_i on the test set $\mathcal{T}$ and report the classification accuracy. All reported results are averaged over 5 runs.

We use SOMs of sizes 15×15, 32×32, 25×25, and 64×64 for CIFAR10/100, CORe50 and subImageNet, respectively. We extract feature vectors from CNN's penultimate fc layer as f, for Eq. (4) and (5). We set $\lambda = 10$ in Eq. (1), $\gamma = 10$ and $K_p = 1$ (for simplicity, we pick 1 nearest *pillar* for $\mathcal{N}_{j,l}$) in Eq. (5).

Using each of the sampling strategies described in Section , we sample $K_s = 1,000$ examples for CIFAR10/100 and CORe50, and $K_s = 10000$ for subImageNet, from S_i to form the training set $\tilde{S}_i$. We use the abbreviation 'Our-PLC' for finetuning with PLC term, and 'Our-PLC-CPL' for using both PLC and CPL terms. As very few methods are designed for ST-NIL setting, for comparison purpose, we adapt the state-of-the-art regularization and rehearsal methods to this setting, including EWC (Kirkpatrick et al. 2017), SI (Zenke, Poole, and Ganguli 2017), IMM (Lee et al. 2017), A-GEM (Chaudhry et al. 2018), and EEIL (Castro et al. 2018). We also compare with the 'naïve' solution in (Maltoni and Lomonaco 2018) that originally proposes the *new instances* setting, and the upper bound 'cumulative' method that jointly trains all encountered data. For fair comparison, we implement all methods using the same network (ResNet18) and measure each method under the same size of the memory that stores *pillars* or image exemplars.

[1]The code is released at https://github.com/xyutao/bocl.

Comparison Results

We compare the proposed BOCL with the state of the arts on CIFAR10, CIFAR100, CORe50 and subImageNet. The results are shown in Figure 3. All the methods are finetuned on the full and the $\mathcal{R}_{rnd}$ sampled set for each training session, respectively. We choose $\mathcal{R}_{rnd}$ as our final sampling strategy, because it outperforms other strategies. In all the figures, the green line corresponds to the naïve finetuning method. The deep blue line represents the upper-bound cumulative method that jointly trains all encountered data. We use the red and orange lines for Our-PLC and Our-PLC-CPL, respectively. The experimental results in Figure 3 are summarized as follows:

- With both datasets, and for training on both the full and the sampled set, our proposed BOCL outperforms other state-of-the-art methods on each encountered session, and its accuracy curves is the closest to that of the upper bound cumulative method.

- By comparing each pair of the red and green lines in Figure 3, we can easily observe that the BOCL with PLC alone outperforms the naïve method by up to **9.22%**. Moreover, using both PLC and CPL terms further achieves up to **1.53%** accuracy gain than using PLC alone. It is thus safe to conclude that both two terms contribute to the models's performance improvement.

- With CIFAR10, after learning all the sessions, BOCL achieves accuracy of **77.43%** and **72.10%** when finetuning on the full and randomly sampled training set, respectively. In comparison, the second best one namely A-GEM* achieves the accuracy of 75.16% and 70.97%, correspondingly. BOCL outperforms the best A-GEM* method by up to **2.27%**.

- With CIFAR100, BOCL achieves the final accuracy of **53.76%** and **41.87%** when finetuning on the full and randomly sampled training set, respectively, while the second best one A-GEM* achieves the accuracy of 51.24% and 40.89%, correspondingly. BOCL outperforms A-GEM* by up to **2.52%**.

- With CORe50, BOCL achieves the accuracy of **74.31%** and **67.83%** when finetuning on the full and randomly sampled training set, respectively, while the corresponding accuracy achieved by A-GEM* are 71.54% and 65.23%, respectively. BOCL outperforms A-GEM* by up to **2.77%**.

- With subImageNet, when finetuning on the full training set, BOCL achieves the top-5 accuracy of **50.32%**, exceeding the second best IMM* (48.52%) by **1.80%**; while for randomly sampled training set, BOCL achieves the top-5 accuracy of **37.77%**, outperforming the second best A-GEM* (34.53%) by **3.24%**. It demonstrate the effectiveness of BOCL on the dataset with a large number of classes.

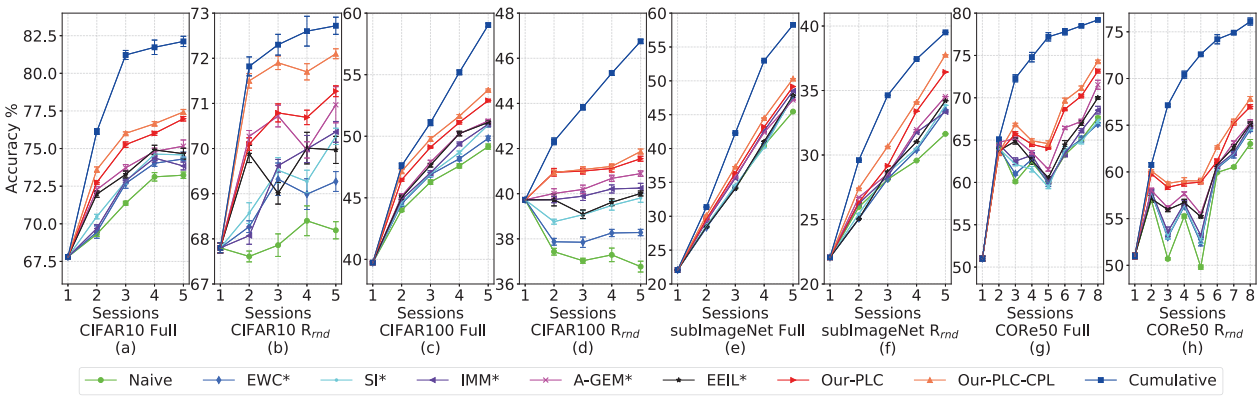


Figure 3: Comparison results on CIFAR10, CIFAR100, subImageNet and CORe50. All methods are finetuned on the full and $\mathcal{R}_{rnd}$ sampled set of the training sessions, respectively. The results are averaged over 5 runs.

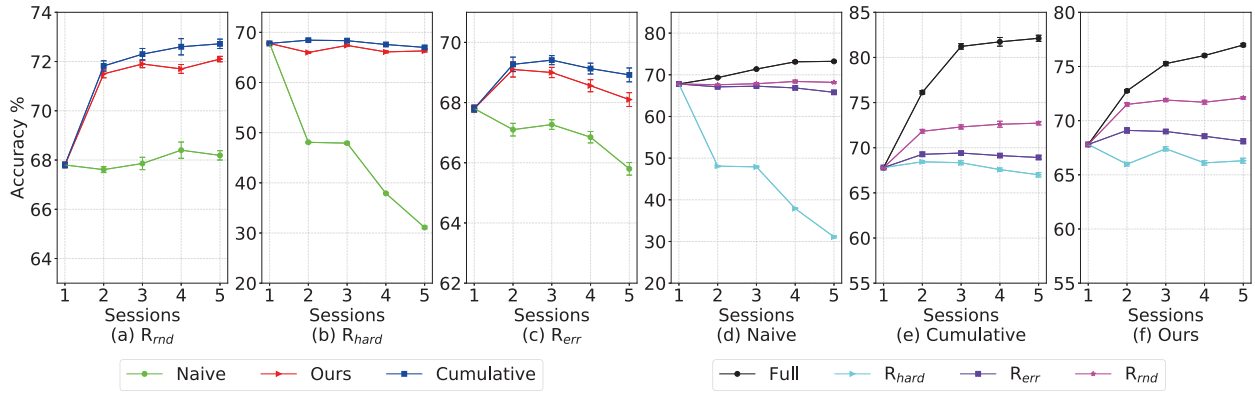


Figure 4: Comparison results of different sampling strategies with different methods. The left three figures compare the performance of naïve, cumulative and our BOCL under different sampling strategies. The right three figures compare the performance of different sampling strategies implemented using different methods. The results are averaged over 5 runs.

Ablation Study

To investigate the impact of each component of BOCL, we perform comprehensive ablation studies on CIFAR10 dataset using full set of new data. In our BOCL, we train a SOM to generate the *pillar* set that represents the knowledge in the feature space, which is used by PLC term for mitigating forgetting and CPL term for improving feature learning on new instances. Alternatively, we can use random sampling as in (Castro et al. 2018) to select representative examples, and then impose quadratic constraint to their features or compute distillation loss to mitigate forgetting.

Table 2 compares the test accuracy at the last session achieved by different terms with different strategy for preserving the knowledge. PLC$^-$ denotes the quadratic feature regularization term without the re-weighting vector $\boldsymbol{\omega}_k$. We can learn that using *pillars* and feature re-weighting technique achieves the best accuracy, as it provides more comprehensive statistics of the feature space (i.e., $\boldsymbol{\omega}$) to facilitate the consolidation of old knowledge.

Table 2: Comparison of the test accuracy at the last session.

Method	random selection	SOM *pillars*
PLC$^-$ (w/o $\boldsymbol{\omega}$)	75.72	75.93
PLC	75.72	**76.98**
PLC-CPL	76.62	**77.43**

Study of Sampling Strategies

To examine the three sampling strategies, we compare the naïve, cumulative , and our BOCL on the CIFAR10 training sessions sampled with $\mathcal{R}_{rnd}$, $\mathcal{R}_{hard}$, and $\mathcal{R}_{err}$, respectively, as illustrated in Figure 4 (a - c). For the random sampling strategy $\mathcal{R}_{rnd}$ in Figure 4 (a), the BOCL method (the red solid lines) has a steady-state growth in accuracy and are close to the cumulative method (the blue dashed lines). For the other two strategies $\mathcal{R}_{hard}$ and $\mathcal{R}_{err}$ in Figure 4 (b) and (c), the performance of the naïve method (the green dashed lines) drops significantly for subsequent sessions which indicates the occurrence of catastrophic forgetting. BOCL method can effectively alleviate catastrophic

Table 3: Classification accuracy w.r.t. different λ after learning all sessions, with $\gamma = 10$.

λ	0.1	1	5	10	50
Acc. (%)	69.77	70.14	71.61	72.10	72.06

Table 4: Classification accuracy w.r.t. different γ after learning all sessions, with $\lambda = 10$.

γ	0.1	1	5	10	50
Acc. (%)	71.31	71.85	72.07	72.10	71.92

forgetting and maintain the recognition performance.

Figure 4 (d - f) further compare different sampling strategies implemented with naïve, cumulative and BOCL, respectively. We can learn that regardless of the upper bound strategy that uses all data for training, the random sampling strategy $\mathcal{R}_{rnd}$ significantly outperforms the other two strategies $\mathcal{R}_{hard}$ and $\mathcal{R}_{err}$, while $\mathcal{R}_{hard}$ is the worst.

We analyze the reason why the $\mathcal{R}_{hard}$ and $\mathcal{R}_{err}$ sampling strategies achieve inferior performance as follows. Hard examples are usually those difficult to be separated by the decision boundary. However, it is prone to catastrophic forgetting when finetuning only on hard ones, which may overfit to noise. A similar case occurs if we only finetune on misclassified examples. In real applications, the distribution of the incoming new data is uncertain and dynamic, and we typically have no knowledge about it. As a consequence, a specifically designed sampling technique (e.g. $\mathcal{R}_{hard}$) may fail. Not surprisingly, the simple random sampling becomes the most versatile and effective strategy since it makes no assumptions about the distribution of the entire dataset.

Sensitivity Study of Hyper-parameters

We perform the sensitivity study of λ (Eq. (1)) and γ (Eq. (5)) about how the performance accuracy varies with different settings of hyper-parameters. Tables 3 and 4 report the classification accuracy on CIFAR10 test set after learning all sessions with the sampling strategy $\mathcal{R}_{rnd}$, w.r.t. different settings of λ and γ, respectively. We can see that a large value of λ and γ can strengthen the effect of the PLC and CPL loss term, respectively. However, too large values may destabilize the convergence of the training phase and weaken the contribution of the cross-entropy loss and the classification accuracy. To determine their values, we split a temporary validation set out of the training sessions, train the networks with different hyper-parameter values, and then pick the ones with the best accuracy on the validation set, which are $\lambda = 10$ and $\gamma = 10$.

Analysis of the Number of Pillars

The proposed BOCL employs the *pillars* generated by an SOM of a specific size to represent the old knowledge. Here we conduct an analysis to reveal how different number of *pillars* affects the performance. Table 5 shows the accuracy on the CIFAR10 test set after learning all the sessions with the sampling strategy $\mathcal{R}_{rnd}$ w.r.t. different numbers of *pillars*. We observed that using a moderate number of *pillars* helps to achieve a better recognition performance, while too many *pillars* will lead to the performance saturation and higher computation cost. We choose the number of *pillars* based on the scale of the dataset. Heuristically, we set the number of *pillars* to about $\sqrt{NC/2}$, where N is the number of training samples of each session, and C is the number of classes.

Table 5: Classification accuracy w.r.t. different numbers of *pillars* after learning all sessions

Num. of *pillars*	10	25	100	225	400
Acc. (%)	69.43	69.82	71.64	72.10	72.08

Discussion

The proposed BOCL method avoids catastrophic forgetting using the PLC term that penalizes the shift of the *pillars* in the feature space f (Eq. (4)). In comparison, other approaches typically impose constraints to the network's parameter space θ, such as EWC (Kirkpatrick et al. 2017), or to the network's classification outputs, such as EEIL (Castro et al. 2018). On the contrary, BOCL constrains the *pillars* in the feature space, avoiding the pillars from shifting, while allowing the feature space adapting for new data freely. The feature space has higher dimensionality than the output space and contains richer information for representing the learned knowledge. Thus constraining the feature space is more effective for learning 'new' while consolidating 'known', which is demonstrated by the experimental results shown in Figure 3.

We also investigated three different sampling strategies for onsite learning: random sampling, hard-example sampling and misclassified-example sampling. Experimental evaluations have revealed that random sampling is the simplest yet most effective sample strategy for finetuning the model on new data.

Conclusion

We focus on the *single-task new-instance learning* (ST-NIL) problem and propose a bi-objective continual learning framework. For alleviating forgetting, we propose the PLC loss term to penalize the shift of *pillars* in the feature space during finetuning. For efficient on-site learning from new data, we propose the CPL loss term to enhance the classification performance. By using pillars as a carrier of the past knowledge, the PCL term provides a seamless mechanism to link the target of 'learning from new' to those of 'consolidating old'. We also investigate different strategies for sampling new instances. Experimental results show that the proposed BOCL remarkably outperforms the state-of-the-art methods. BOCL is simple and efficient, with less computational and storage overhead when cooperating with deep CNNs. Next we will generalize BOCL to more applications and put it into practical use.

Acknowledgements

This work is supported by National Basic Research Program of China (Grant No.2015CB351705) and National Major Project (Grant No.2017YFC0803905).

References

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C.; and Alahari, K. 2018. End-to-end incremental learning. In *ECCV*, 233–248.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

Chen, L.-C.; Papandreou, G.; Schroff, F.; and Adam, H. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Cheng, D.; Gong, Y.; Zhou, S.; Wang, J.; and Zheng, N. 2016. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. *CVPR*.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, 886–893. IEEE Computer Society.

Deng, J.; Dong, W.; Socher, R.; Li, L. J.; Li, K.; and Li, F. F. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2018. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* 3(4):128–135.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *Computer Science* 14(7):38–39.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114(13):3521–3526.

Kohonen, T. 2013. Essentials of the self-organizing map. *Neural Networks* 37(1):52–65.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Lee, S.-W.; Kim, J.-H.; Jun, J.; Ha, J.-W.; and Zhang, B.-T. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*, 4652–4662.

Li, Z., and Hoiem, D. 2018. Learning without forgetting. *T-PAMI* 40(12):2935–2947.

Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; and Song, L. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 212–220.

Lomonaco, V., and Maltoni, D. 2017. Core50: a new dataset and benchmark for continuous object recognition. *arXiv preprint arXiv:1705.03550*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 3431–3440.

Lopez-Paz, D., et al. 2017. Gradient episodic memory for continual learning. In *NIPS*, 6467–6476.

Maltoni, D., and Lomonaco, V. 2018. Continuous learning in single-incremental-task scenarios. *arXiv preprint arXiv:1806.08568*.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *CVPR*, 2001–2010.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2015. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 91–99.

Shun, Z.; Yihong, G.; Jia-Bin, H.; Jongwoo, L.; Jinjun, W.; Narendra, A.; and Ming-Hsuan, Y. 2016. Tracking persons-of-interest via adaptive discriminative features. In *ECCV*, 415–433.

Tusng-Yu, L.; Aruni, R.; and Subhransu, M. 2015. Bilinear cnn models for fine-grained visual recognition. In *ICCV*.

Wang, J.; Yang, J.; Kai, Y.; Lv, F.; Huang, T. S.; and Gong, Y. 2010. Locality-constrained linear coding for image classification. In *CVPR*.

Wei, X.; Zhang, Y.; Gong, Y.; Zhang, J.; and Zheng, N. 2018a. Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In *ECCV*.

Wei, X.; Zhang, Y.; Gong, Y.; and Zheng, N. 2018b. Kernelized subspace pooling for deep local descriptors. In *CVPR*.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; and Fu, Y. 2019. Large scale incremental learning. *arXiv preprint arXiv:1905.13260*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *ICML*, 3987–3995. JMLR. org.