Online Prototype Learning for Online Continual Learning

Yujie Wei1  Jiaxin Ye1  Zhizhong Huang2  Junping Zhang2  Hongming Shan1,3,4*
1 Institute of Science and Technology for Brain-inspired Intelligence, Fudan University
2 Shanghai Key Lab of Intelligent Information Processing, School of Computer Science
Fudan University
3 MOE Frontiers Center for Brain Science, Fudan University
4 Shanghai Center for Brain Science and Brain-inspired Technology
{yjwei22, jxye22}@m.fudan.edu.cn, {zzhuang19, jpzhang, hmshan}@fudan.edu.cn

Abstract

Online continual learning (CL) studies the problem of learning continuously from a single-pass data stream while adapting to new data and mitigating catastrophic forgetting. Recently, by storing a small subset of old data, replay-based methods have shown promising performance. Unlike previous methods that focus on sample storage or knowledge distillation against catastrophic forgetting, this paper aims to understand why the online learning models fail to generalize well from a new perspective of shortcut learning. We identify shortcut learning as the key limiting factor for online CL, where the learned features may be biased, not generalizable to new tasks, and may have an adverse impact on knowledge distillation. To tackle this issue, we present the online prototype learning (OnPro) framework for online CL. First, we propose online prototype equilibrium to learn representative features against shortcut learning and discriminative features to avoid class confusion, ultimately achieving an equilibrium status that separates all seen classes well while learning new classes. Second, with the feedback of online prototypes, we devise a novel adaptive prototypical feedback mechanism to sense the classes that are easily misclassified and then enhance their boundaries. Extensive experimental results on widely-used benchmark datasets demonstrate the superior performance of OnPro over the state-of-the-art baseline methods. Source code is available at https://github.com/weillllllls/OnPro.

1. Introduction

Current artificial intelligence systems [30, 36, 52, 16] have shown excellent performance on the tasks at hand; however, they are prone to forget previously learned knowl-

dege while learning new tasks, known as catastrophic forgetting [20, 23, 9]. Continual learning (CL) [46, 44, 14, 19] aims to learn continuously from a non-stationary data stream while adapting to new data and mitigating catastrophic forgetting, offering a promising path to human-like artificial general intelligence. Early CL works consider the task-incremental learning (TIL) setting, where the model selects the task-specific component for classification with task identifiers [1, 41, 50, 14]. However, this setting lacks flexibility in real-world scenarios. In this paper, we focus on a more general and realistic setting—the class-incremental learning (CIL) in the online CL mode [42, 13, 27, 51]—where the model learns incrementally classes in a sequence of tasks from a single-pass data stream and cannot access task identifiers at inference.

Various online CL methods have been proposed to mitigate catastrophic forgetting [51, 43, 25, 28, 11, 5, 13]. Among them, replay-based methods [11, 43, 26, 2, 25] have shown promising performance by storing a subset of data from old classes as exemplars for experience replay. Unlike previous methods that focus on sample storage [51, 3], we are interested in how generalizable the learned features are to new classes, and aim to understand why the online learning models fail to generalize well from a new perspective of shortcut learning.
Intuitively, the neural network tends to “take shortcuts” [22] and focuses on simplistic features. This behavior of shortcut learning is especially serious in online CL, since the model may learn biased and inadequate features from the single-pass data stream. Specifically, the model may be more inclined to learn trivial solutions unrelated to objects, which are hard to generalize and easily forgotten. Take Fig. 1 as an example, when classifying two classes, saying airplanes in the sky and cat on the grass, the model may easily identify the shortcut clue between two classes—blue sky vs. green grass—unfortunately, the learned features are delicate and unrelated to the classes of interest. When new bird and deer classes come, which may also have sky or grass, the model has to be updated due to inapplicable previous knowledge, leading to poor generalization and catastrophic forgetting. Thus, learning representative features that best characterize the class is crucial to resist shortcut learning and catastrophic forgetting, especially in online CL.

In addition, the intuitive manifestation of catastrophic forgetting is the confusion between classes. To alleviate class confusion, many works [26, 41, 48, 4, 7, 55] employ self-distillation [17, 32] to preserve previous knowledge. However, the premise for knowledge distillation to succeed is that the model has learned sufficient discriminative features in old classes, and these features still remain discriminative when learning new classes. As mentioned above, the model may learn oversimplified features due to shortcut learning, significantly compromising the generalization to new classes. Thus, distilling these biased features may have an adverse impact on new classes. In contrast, we consider a more general paradigm to maintain discrimination among all seen classes, which can tackle the limitations of knowledge distillation.

In this paper, we aim to learn representative features of each class and discriminative features between classes, both crucial to mitigate catastrophic forgetting. Toward this end, we present the Online Prototype learning (OnPro) framework for online continual learning. The online prototype introduced is defined as “a representative embedding for a group of instances in a mini-batch.” There are two reasons for this design: (1) for new classes, the data arrives sequentially from a single-pass stream, and we cannot access all samples of one class at any time step (iteration); and (2) for old classes, computing the prototypes of all samples in the memory bank at each time step is computationally expensive, especially for the online scenario with limited resources. Thus, our online prototypes only utilize the data available at the current time step (i.e., data within a mini-batch), which is more suitable for online CL.

To resist shortcut learning in online CL and maintain discrimination among seen classes, we first propose Online Prototype Equilibrium (OPE) to learn representative and discriminative features for achieving an equilibrium status that separates all seen classes well while learning new classes. Second, instead of employing knowledge distillation that may distill unfaithful knowledge from previous models, we devise a novel Adaptive Prototypical Feedback (APF) that can leverage the feedback of online prototypes to first sense the classes—that are easily misclassified—and then adaptively enhance their decision boundaries.

The contributions are summarized as follows.

1) We identify shortcut learning as the key limiting factor for online CL, where the learned features may be biased, not generalizable to new tasks, and may have an adverse impact on knowledge distillation. To the best of our knowledge, this is the first time to identify the shortcut learning issues in online CL, offering new insights into why online learning models fail to generalize well.

2) We present the online prototype learning framework for online CL, in which the proposed online prototype equilibrium encourages learning representative and discriminative features while adaptive prototypical feedback leverages the feedback of online prototypes to sense easily misclassified classes and enhance their boundaries.

3) Extensive experimental results on widely-used benchmark datasets demonstrate the superior performance of our method over the state-of-the-art baseline methods.

2. Related Work

Continual learning. Continual learning methods can be roughly summarized into three categories: regularization-based, parameter-isolation-based, and replay-based methods. Regularization-based methods [9, 1, 39, 31] add extra regularization constraints on network parameters to mitigate forgetting. Parameter-isolation-based methods [49, 50, 38, 18] avoid forgetting by dynamically allocating parameters or modifying the architecture of the network. Replay-based methods [11, 2, 3, 10, 4, 47] maintain and update a memory bank (buffer) that stores exemplars of past tasks. Among them, replay-based methods are the most popular for their simplicity yet efficiency. Experience Replay [11] randomly samples from the buffer. MIR [2] retrieves buffer samples by comparing the interference of losses. Furthermore, in the online setting, ASER [51] introduces a buffer management theory based on the Shapley value. SCR [43] utilizes supervised contrastive loss [34] for training and the nearest-class-mean classifier for testing. OCM [26] prevents forgetting through mutual information maximization. Unlike these methods that focus on selecting which samples to store or learning features only by instances, our work rethinks the catastrophic forgetting from a new shortcut learning perspective, and proposes to learn representative and discriminative features through online prototypes.

Knowledge distillation in continual learning. Another solution to catastrophic forgetting is to preserve previ-
Figure 2. Illustration of the proposed OnPro framework. At time step (iteration) \( i \), the incoming data \( X \) and replay data \( X^b \) are augmented and fed to the model to learn features with OPE. Then, the proposed APF senses easily misclassified classes from all seen classes and enhances their decision boundaries. Concretely, APF adaptively selects more data for mixup according to the probability distribution \( P \).

### Prototypes in continual learning.

Some previous methods [48, 43, 55] attempt to utilize prototypes to mitigate catastrophic forgetting. As mentioned above, iCaRL and SCR employ class prototypes as classifiers, and PASS distills old prototypes to retain learned knowledge. Nevertheless, computing prototypes with all samples is extremely expensive for training. There are also some works considering the use of prototypes in the online scenario. CoPE [15] designs the prototypes with a high momentum-based update for each observed batch. A recent work [28] estimates class prototypes on all seen data using mean update criteria. However, regardless of momentum update or mean update, accumulating previous features as prototypes may be detrimental to future learning, since the features learned in old classes may not be discriminative when encountering new classes due to shortcut learning. In contrast, the proposed online prototypes only utilize the data visible at the current time step, which significantly decreases the computational cost and is more suitable for online CL.

### Contrastive learning.

Inspired by breakthroughs in self-supervised learning [45, 29, 12, 24, 6, 33], many studies [43, 5, 26, 7, 28] in CL use contrastive learning to learn generalized features. An early work [21] analyzes and reveals the impact of contrastive learning on online CL. Among them, the work most related to ours is PCL [40], which calculates infoNCE loss [45] between instance and prototype. The most significant difference is that the loss in OPE only considers online prototypes, and there is no involvement of instances. Please refer to the supplementary material for detailed comparisons between our OPE and PCL.

### 3. Method

Fig. 2 presents the illustration of the proposed OnPro. In this section, we start by providing the problem definition of online CIL. Then, we describe the definition of the online prototype, the proposed online prototype equilibrium, and the proposed adaptive prototypical feedback. Finally, we propose an online prototype learning framework.

#### 3.1. Problem Definition

Formally, online CIL considers a continuous sequence of tasks from a single-pass data stream \( D = \{D_1, \ldots, D_T\} \), where \( D_t = \{x_i, y_i\}_{i=1}^{N_t} \) is the dataset of task \( t \), and \( T \) is the total number of tasks. Dataset \( D_t \) contains \( N_t \) labeled samples, \( y_i \) is the class label of sample \( x_i \) and \( y_i \in C_t \), where \( C_t \) is the class set of task \( t \) and the class sets of different tasks are disjoint. For replay-based methods, a memory bank is used to store a small subset of seen data, and we also maintain a memory bank \( M \) in our method. At each time step of task \( t \), the model receives a mini-batch data \( X \cup X^b \) for
training, where $X$ and $X^b$ are drawn from the i.i.d distribution $D$, and the memory bank $M$, respectively. Moreover, we adopt the single-head evaluation setup [9], where a unified classifier must choose labels from all seen classes at inference due to unavailable task identifiers. The goal of online CIL is to train a unified model on data seen only once while predicting well on both new and old classes.

### 3.2. Online Prototype Definition

Prior to introducing the online prototypes, we first present the network architecture of our OnPro. Suppose that the model consists of three components: an encoder network $f$, a projection head $g$, and a classifier $\varphi$. Each sample $x$ in incoming data $X$ (a mini-batch data from new classes) is mapped to a projected vectorial embedding (representation) $z$ under encoder $f$ and projector $g$:

$$z = g(f(aug(x); \theta_f); \theta_g),$$

where $aug$ represents the data augmentation operation, $\theta_f$ and $\theta_g$ represent the parameters of $f$ and $g$, respectively, and $z$ is $l_2$-normalized. Similar to Eq. (1), we use $z^b$ to denote the representation of replay data $X^b$ (a mini-batch data from seen classes in the memory bank).

At each time step of task $t$, the online prototype of each class is defined as the mean representation in a mini-batch:

$$p_i = \frac{1}{n_i} \sum_j z_j \cdot 1\{y_j = i\},$$

where $n_i$ is the number of samples for class $i$ in a mini-batch, and $1$ is the indicator function. We can get a set of $K$ online prototypes in $X$, $P = \{p_i\}_{i=1}^K$, and a set of $K^b$ online prototypes in $X^b$, $P^b = \{p^b_i\}_{i=1}^K$. Note that $K = |P| \leq |C|$ and $K^b = |P^b| \leq \sum_{i=1}^K |C_i|$, where $|\cdot|$ denotes the cardinal number.

### 3.3. Online Prototype Equilibrium

The introduced online prototypes can provide representative features and avoid class-unrelated information. These characteristics are exactly the key to countering shortcut learning in online CL. Besides, maintaining the discrimination among seen classes is also essential to mitigate catastrophic forgetting. Based on these, we attempt to learn representative features of each class by pulling online prototypes $P$ and their augmented views $\tilde{P}$ closer in the embedding space, and learn discriminative features between classes by pushing online prototypes of different classes away, formally defined as a contrastive loss:

$$\ell(P, \tilde{P}) = \frac{1}{|P|} \sum_{i=1}^{|P|} \log \frac{\exp \left( \frac{P^T \tilde{P}_i}{\tau} \right)}{\sum_j \exp \left( \frac{P^T \tilde{P}_j}{\tau} \right) + \sum_{j \neq i} \exp \left( \frac{P^T \tilde{P}_j}{\tau} \right)},$$

where $\tau$ is the temperature hyper-parameter, $P$ and $\tilde{P}$ are $l_2$-normalized. To compute the contrastive loss across all positive pairs in both $(P, \tilde{P})$ and $(\tilde{P}, P)$, we define $\mathcal{L}_{pro}$ as the final contrastive loss over online prototypes:

$$\mathcal{L}_{pro}(P, \tilde{P}) = \frac{1}{2} \left[ \ell(P, \tilde{P}) + \ell(\tilde{P}, P) \right].$$

Considering the learning of new classes and the consolidation of learned knowledge simultaneously in online CL, we propose Online Prototype Equilibrium (OPE) to learn representative and discriminative features on both new and seen classes by employing $\mathcal{L}_{pro}$:

$$\mathcal{L}_{OPE} = \mathcal{L}_{pro}^{\text{new}}(P, \tilde{P}) + \mathcal{L}_{pro}^{\text{seen}}(P^b, \tilde{P}^b),$$

where $\mathcal{L}_{pro}^{\text{new}}$ focuses on learning knowledge from new classes, and $\mathcal{L}_{pro}^{\text{seen}}$ is dedicated to preserving learned knowledge of all seen classes. This process is similar to a zero-sum game, and OPE aims to achieve the equilibrium to play a win-win game. Concretely, as the model learns, the knowledge of new classes is gained and added to the prototypes over the memory bank $M$, causing $\mathcal{L}_{pro}^{\text{seen}}$ gradually changes to the equilibrium that separates all seen classes well, including new ones. This variation is crucial to mitigate forgetting and is consistent with the goal of CIL.

### 3.4. Adaptive Prototypical Feedback

Although OPE can bring an overall equilibrium, it tends to treat each class equally. In fact, the degree of confusion varies among classes, and the model should focus purposefully on confused classes to consolidate learned knowledge. To this end, we propose Adaptive Prototypical Feedback (APF) with the feedback of online prototypes to sense the classes that are prone to be misclassified and then enhance their decision boundaries.

For each class pair in the memory bank $M$, APF calculates the distances between online prototypes of all seen classes from the previous time step, showing the class confusion status by these distances. The closer the two prototypes are, the easier to be misclassified. Based on this analysis, our idea is to enhance the boundaries for those classes. Therefore, we convert the prototype distance matrix to a probability distribution $P$ over the classes via a symmetric Gaussian kernel, defined as follows:

$$P_{i,j} \propto \exp\left(-\|p_i^b - p_j^b\|_2^2\right),$$

where $i, j \in \{1, \ldots, |P^b|\}$ and $i \neq j$. Then, all probabilities are normalized to a probability mass function that sums to one. APF returns probabilities to $M$ for guiding the next sampling process and enhancing decision boundaries of easily misclassified classes.

Our adaptive prototypical feedback is implemented as a sampling-based mixup. Specifically, APF adaptively selects more samples from easily misclassified classes in $M$.
for mixup [54] according to the probability distribution $P$. Considering not over-penalizing the equilibrium of current online prototypes, we introduce a two-stage sampling strategy for replay data $X^b$ of size $m$. First, we select $n_{APF}$ samples with $P$, and a larger $P_{a,b}$ means more sampling from classes $a$ and $b$. Here, $n_{APF} = \alpha \cdot m$, and $\alpha$ is the ratio of APF. Second, the remaining $m - n_{APF}$ samples are uniformly randomly selected from the entire memory bank to avoid the model only focusing on easily misclassified classes and disrupting the established equilibrium.

3.5. Overall Framework of OnPro

The overall structure of OnPro is shown in Fig. 2. OnPro comprises two key components based on proposed online prototypes: Online Prototype Equilibrium (OPE) and Adaptive Prototypical Feedback (APF). With the two components, the model can learn representative features against shortcut learning, and all seen classes maintain discriminative when learning new classes. However, classes may not be compact, because the online prototypes cannot cover full instance-level information. To further achieve intra-class compactness, we employ supervised contrastive learning [34] to learn instance-wise representations:

$$L_{INS} = \sum_{i=1}^{2N} \frac{1}{|I_i|} \sum_{j \in I_i} \log \left( \frac{\exp \left( \frac{\text{sim}(z_i, z_j) - \alpha}{\tau'} \right)}{\sum_{k \neq i} \exp \left( \frac{\text{sim}(z_i, z_k) - \alpha}{\tau'} \right)} \right) + \frac{2m - 1}{|P_i|} \sum_{j \in P_i} \log \left( \frac{\exp \left( \frac{\text{sim}(z_i^b, z_j^b) - \alpha}{\tau'} \right)}{\sum_{k \neq i} \exp \left( \frac{\text{sim}(z_i^b, z_k^b) - \alpha}{\tau'} \right)} \right),$$

where $I_i = \{ j \in \{1, \ldots, 2N\} \mid j \neq i, y_j = y_i \}$ and $P_i = \{ j \in \{1, \ldots, 2m\} \mid j \neq i, y_i^b = y_j^b \}$ are the set of positive samples indexes to $z_i$ and $z_i^b$, respectively. $y_i^b$ is the class label of input $x_i^b$ from $X^b$. $N$ is the batch size of $X$. $\tau'$ is the temperature hyperparameter. The similarity function $\text{sim}$ is computed in the same way as Eq. (9) in OCM [26].

Thus, the total loss of our OnPro framework is given as:

$$L_{OnPro} = L_{OPE} + L_{INS} + L_{CE},$$

where $L_{CE} = CE(y^b, \varphi(f(\text{aug}(x^b))))$ is the cross-entropy loss; see the supplementary material for detailed training algorithms.

Following other replay-based methods [11, 43, 26], we update the memory bank in each time step by uniformly randomly selecting samples from $X$ to push into $M$ and, if $M$ is full, pulling an equal number of samples out of $M$.

4. Experiments

4.1. Experimental Setup

Datasets. We use three image classification benchmark datasets, including CIFAR-10 [35], CIFAR-100 [35], and TinyImageNet [37], to evaluate the performance of online CIL methods. Following [51, 43, 25], we split CIFAR-10 into 5 disjoint tasks, where each task has 2 disjoint classes, 10,000 samples for training, and 2,000 samples for testing, and split CIFAR-100 into 10 disjoint tasks, where each task has 10 disjoint classes, 5,000 samples for training, and 1,000 samples for testing. Following [26], we split TinyImageNet into 100 disjoint tasks, where each task has 2 disjoint classes, 1,000 samples for training, and 100 samples for testing. Note that the order of tasks is fixed in all experimental settings.

Baselines. We compare our OnPro with 13 baselines, including 10 replay-based online CL baselines: AGEM [10], MIR [2], GSS [3], ER [11], GDumb [47], ASER [51], SCR [43], CoPE [15], DVC [25], and OCM [26]; 3 offline CL baselines that use knowledge distillation by running them in one epoch: iCaRL [48], DER++ [4], and PASS [55]. Note that PASS is a non-exemplar method.

Evaluation metrics. We use Average Accuracy and Average Forgetting [51, 25] to measure the performance of our framework in online CIL. Average Accuracy evaluates the accuracy of the test sets from all seen tasks, defined as Average Accuracy = $\frac{1}{T} \sum_{j=1}^{T} a_{T,j}$, where $a_{i,j}$ is the accuracy on task $j$ after the model is trained from task $1$ to $i$. Average Forgetting represents how much the model forgets about each task after being trained on the final task, defined as Average Forgetting = $\frac{1}{T-1} \sum_{j=1}^{T-1} f_{T,j}$, where $f_{i,j} = \max_{k \in \{1, \ldots, i-1\}} a_{k,j} - a_{i,j}$.

Implementation details. We use ResNet18 [30] as the backbone $f$ and a linear layer as the projection head $g$ like [43, 26, 7]; the hidden dim in $g$ is set to 128 as [12]. We also employ a linear layer as the classifier $\varphi$. We train the model from scratch with Adam optimizer and an initial learning rate of $5 \times 10^{-4}$ for all datasets. The weight decay is set to $1.0 \times 10^{-4}$. Following [51, 25], we set the batch size $N$ as 10, and following [26] the replay batch size $m$ is set to 64. For CIFAR-10, we set the ratio of APF $\alpha = 0.25$. For CIFAR-100 and TinyImageNet, $\alpha$ is set to 0.1. The temperature $\tau = 0.5$ and $\tau' = 0.07$. For baselines, we also use ResNet18 as their backbone and set the same batch size and replay batch size for fair comparisons. We reproduce all baselines in the same environment with their source code and default settings; see the supplementary material for implementation details about all baselines. We report the average results across 15 runs for all experiments.

Data augmentation. Similar to data augmentations used in SimCLR [12], we use resized-crop, horizontal-flip, and gray-scale as our data augmentations. For all baselines, we
all classes

Airplane

Error

task 1

task 2

task 1

task 2

(a) ER

(b) OnPro

Table 1. Average Accuracy (higher is better) on three benchmark datasets with different memory bank sizes $M$. All results are the average and standard deviation of 15 runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10 (M = 0.1k)</th>
<th>CIFAR-10 (M = 0.2k)</th>
<th>CIFAR-10 (M = 0.5k)</th>
<th>CIFAR-100 (M = 0.1k)</th>
<th>CIFAR-100 (M = 0.2k)</th>
<th>CIFAR-100 (M = 1k)</th>
<th>CIFAR-100 (M = 2k)</th>
<th>TinyImageNet (M = 1k)</th>
<th>TinyImageNet (M = 2k)</th>
<th>TinyImageNet (M = 4k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>iCaRL [48]</td>
<td>31.0±1.2</td>
<td>33.9±0.9</td>
<td>42.0±0.9</td>
<td>12.8±0.4</td>
<td>16.5±0.4</td>
<td>17.6±0.5</td>
<td>5.0±0.3</td>
<td>6.6±0.4</td>
<td>7.8±0.4</td>
<td></td>
</tr>
<tr>
<td>DER++ [4]</td>
<td>31.5±2.9</td>
<td>39.7±2.7</td>
<td>50.9±1.8</td>
<td>16.0±0.6</td>
<td>21.4±0.9</td>
<td>23.9±1.0</td>
<td>3.7±0.4</td>
<td>5.1±0.8</td>
<td>6.8±0.6</td>
<td></td>
</tr>
<tr>
<td>PASS [55]</td>
<td>33.7±2.2</td>
<td>33.7±2.2</td>
<td>33.7±2.2</td>
<td>7.5±0.7</td>
<td>7.5±0.7</td>
<td>7.5±0.7</td>
<td>0.5±1.1</td>
<td>0.5±1.0</td>
<td>0.5±1.0</td>
<td></td>
</tr>
<tr>
<td>AGEM [10]</td>
<td>17.7±0.3</td>
<td>17.5±0.3</td>
<td>17.5±0.2</td>
<td>5.8±0.1</td>
<td>5.9±0.1</td>
<td>5.8±0.1</td>
<td>0.8±0.1</td>
<td>0.8±0.1</td>
<td>0.8±0.1</td>
<td></td>
</tr>
<tr>
<td>GSS [3]</td>
<td>18.4±0.2</td>
<td>19.4±0.7</td>
<td>25.2±0.9</td>
<td>8.1±0.2</td>
<td>9.4±0.5</td>
<td>10.1±0.8</td>
<td>1.1±0.1</td>
<td>1.5±0.1</td>
<td>2.4±0.4</td>
<td></td>
</tr>
<tr>
<td>ER [11]</td>
<td>19.4±0.6</td>
<td>20.9±0.9</td>
<td>26.0±1.2</td>
<td>8.7±0.3</td>
<td>9.9±0.5</td>
<td>10.7±0.8</td>
<td>1.2±0.1</td>
<td>1.5±0.2</td>
<td>2.0±0.2</td>
<td></td>
</tr>
<tr>
<td>MIR [2]</td>
<td>20.7±0.7</td>
<td>23.5±0.8</td>
<td>29.9±1.2</td>
<td>9.7±0.3</td>
<td>11.2±0.4</td>
<td>13.0±0.7</td>
<td>1.4±0.1</td>
<td>1.9±0.2</td>
<td>2.9±0.3</td>
<td></td>
</tr>
<tr>
<td>GDumb [47]</td>
<td>23.3±1.3</td>
<td>27.1±0.7</td>
<td>34.0±0.8</td>
<td>8.2±0.2</td>
<td>11.0±0.4</td>
<td>15.3±0.3</td>
<td>4.6±0.3</td>
<td>6.6±0.2</td>
<td>10.0±0.3</td>
<td></td>
</tr>
<tr>
<td>ASER [51]</td>
<td>20.0±1.0</td>
<td>22.8±0.6</td>
<td>31.6±1.1</td>
<td>11.0±0.3</td>
<td>13.5±0.3</td>
<td>17.6±0.4</td>
<td>2.2±0.1</td>
<td>4.2±0.6</td>
<td>8.4±0.7</td>
<td></td>
</tr>
<tr>
<td>SCR [43]</td>
<td>40.2±1.3</td>
<td>48.5±1.5</td>
<td>59.1±1.3</td>
<td>19.3±0.6</td>
<td>26.5±0.5</td>
<td>32.7±0.3</td>
<td>8.9±0.3</td>
<td>14.7±0.3</td>
<td>19.5±0.3</td>
<td></td>
</tr>
<tr>
<td>CoPE [15]</td>
<td>33.5±3.2</td>
<td>37.3±2.2</td>
<td>42.9±3.5</td>
<td>11.6±0.7</td>
<td>14.6±1.3</td>
<td>16.8±0.9</td>
<td>2.1±0.3</td>
<td>2.3±0.4</td>
<td>2.5±0.3</td>
<td></td>
</tr>
<tr>
<td>DVC [25]</td>
<td>35.2±1.7</td>
<td>41.6±2.7</td>
<td>53.8±2.2</td>
<td>15.4±0.7</td>
<td>20.3±1.0</td>
<td>25.2±1.6</td>
<td>4.9±0.6</td>
<td>7.5±0.5</td>
<td>10.9±1.1</td>
<td></td>
</tr>
<tr>
<td>OCM [26]</td>
<td>47.5±1.7</td>
<td>59.6±0.4</td>
<td>70.1±1.5</td>
<td>19.7±0.5</td>
<td>27.4±0.3</td>
<td>34.4±0.5</td>
<td>10.8±0.4</td>
<td>15.4±0.4</td>
<td>20.9±0.7</td>
<td></td>
</tr>
<tr>
<td>OnPro (ours)</td>
<td>57.8±1.1</td>
<td>65.5±1.0</td>
<td>72.6±0.8</td>
<td>22.7±0.7</td>
<td>30.0±0.4</td>
<td>35.9±0.6</td>
<td>11.9±0.3</td>
<td>16.9±0.4</td>
<td>22.1±0.4</td>
<td></td>
</tr>
</tbody>
</table>

4.2. Motivation Justification

Shortcut learning in online CL. Shortcut learning is severe in online CL since the model cannot learn sufficient representative features due to the single-pass data stream. To intuitively demonstrate this issue, we conduct Grad-CAM++ [8] on the training set of CIFAR-10 (\(M = 0.2k\)) after the model is trained incrementally, as shown in Fig. 1. Each row in Fig. 1 represents a task with two classes. We can observe that although ER and DVC predict the correct class, the models actually take shortcuts and focus on some object-unrelated features. An interesting phenomenon is that ER tends to take shortcuts in each task. For example, ER learns the sky on both the airplane class in task 1 (the first row) and the bird class in task 2 (the second row) . Thus, ER forgets almost all the knowledge of the old classes. DVC maximizes the mutual information between instances like contrastive learning [12, 29], which only partially alleviates shortcut learning in online CL. In contrast, OnPro focuses on the representative features of the objects themselves. The results confirm that learning representative features is crucial against shortcut learning; see the supplementary material for more visual explanations.

Class confusion in online CL. Fig. 3 provides the t-SNE [53] visualization results for ER and OnPro on the test set of CIFAR-10 (\(M = 0.2k\)). We can draw intuitive observations as follows. (1) There is serious class confusion in ER. When the new task (task 2) arrives, features learned in task 1 are not discriminative for task 2, leading to class confusion and decreased performance in old classes. (2) Shortcut learning may cause class confusion. For example, the
performance of ER decreases more on airplanes compared to automobiles, probably because birds in the new task have more similar backgrounds to airplanes, as shown in Fig. 1. (3) OnPro achieves better discrimination both on task 1 and task 2. The results demonstrate that OnPro can maintain discrimination of all seen classes and significantly mitigate forgetting by combining the proposed OPE and APF.

4.3. Results and Analysis

Performance of average accuracy. Table 1 presents the results of average accuracy with different memory bank sizes ($M$) on three benchmark datasets. Our OnPro consistently outperforms all baselines on three datasets. Remarkably, the performance improvement of OnPro is more significant when the memory bank size is relatively small; this is critical for online CL with limited resources. For example, compared to the second-best method OCM, OnPro achieves about 10% and 6% improvement on CIFAR-10 when $M$ is 100 and 200, respectively. The results show that our OnPro can learn more representative and discriminative features with a limited memory bank. Compared to baselines that use knowledge distillation (iCaRL, DER++, PASS, OCM), our OnPro achieves better performance by leveraging the feedback of online prototypes. Besides, OnPro significantly outperforms PASS and CoPE that also use prototypes, showing that online prototypes are more suitable for online CL.

We find that the performance improvement tends to be gentle when $M$ increases. The reason is that as $M$ increases, the samples in the memory bank become more diverse, and the model can extract sufficient information from massive samples to distinguish seen classes. In addition, many baselines perform poorly on CIFAR-100 and TinyImageNet due to a dramatic increase in the number of tasks. In contrast, OnPro still performs well and improves accuracy over the second best.

Performance of average forgetting. We report the Average Forgetting results of our OnPro and all baselines on three benchmark datasets in Table 2. The results confirm that OnPro can effectively mitigate catastrophic forgetting. For CIFAR-10 and CIFAR-100, OnPro achieves the lowest average forgetting compared to all replay-based baselines. For TinyImageNet, our result is a little higher than iCaRL.
Table 3. Ablation studies on CIFAR-10 ($M = 0.1k$) and CIFAR-100 ($M = 0.5k$). “baseline” means $L_{\text{INS}} + L_{\text{CE}}$. “$L_{\text{pro}}$ w/o $L_{\text{new}}$” means $L_{\text{pro}}$ do not consider new classes in current task.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc ↑(Forget ↓)</td>
<td>Acc ↑(Forget ↓)</td>
</tr>
<tr>
<td>baseline</td>
<td>46.4 ± 1.2(36.0 ± 2.1)</td>
<td>18.8 ± 0.8(18.5 ± 0.7)</td>
</tr>
<tr>
<td>w/o OPE</td>
<td>53.1 ± 1.4(24.7 ± 2.0)</td>
<td>19.3 ± 0.7(15.9 ± 0.9)</td>
</tr>
<tr>
<td>w/o APF</td>
<td>52.0 ± 1.5(34.6 ± 2.4)</td>
<td>21.5 ± 0.5(16.3 ± 0.8)</td>
</tr>
<tr>
<td>$L_{\text{pro}}$</td>
<td>54.8 ± 1.2(22.1 ± 3.0)</td>
<td>19.6 ± 0.8(19.9 ± 0.7)</td>
</tr>
<tr>
<td>$L_{\text{pro}}$ w/o $L_{\text{new}}$</td>
<td>55.7 ± 1.4(25.5 ± 1.5)</td>
<td>20.1 ± 0.4(16.2 ± 0.6)</td>
</tr>
<tr>
<td>$L_{\text{seen}}$ w/o $L_{\text{new}}$</td>
<td>56.2 ± 1.2(26.4 ± 2.3)</td>
<td>20.8 ± 0.6(17.9 ± 0.7)</td>
</tr>
<tr>
<td>OnPro (ours)</td>
<td>57.8 ± 1.1(23.2 ± 1.3)</td>
<td>22.7 ± 0.7(15.0 ± 0.8)</td>
</tr>
</tbody>
</table>

Table 4. Comparison of Random Mixup and APF on CIFAR-10.

<table>
<thead>
<tr>
<th>Method</th>
<th>$M = 0.1k$</th>
<th>$M = 0.2k$</th>
<th>$M = 0.5k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>53.5 ± 2.7</td>
<td>62.9 ± 2.5</td>
<td>70.8 ± 2.2</td>
</tr>
<tr>
<td>APF (ours)</td>
<td>57.8 ± 1.1</td>
<td>65.5 ± 1.0</td>
<td>72.6 ± 0.8</td>
</tr>
</tbody>
</table>

Figure 5. The cosine similarity between online prototypes and prototypes of the entire memory bank.

APF boosts OPE by about 6% improvements on CIFAR-10 ($M = 0.1k$), and the performance of APF is improved by about 3% on CIFAR-100 ($M = 0.5k$) by combining OPE.

Equilibrium in OPE. When learning new classes, the data of new classes is involved in both $L_{\text{pro}}$ and $L_{\text{pro}}$ of OPE, where $L_{\text{pro}}$ only focuses on learning new knowledge while $L_{\text{pro}}$ tends to alleviate forgetting on seen classes. To explore the best way of learning new classes, we consider three scenarios for OPE in Table 3. The results show that only learning new knowledge (w/o $L_{\text{pro}}$) or only consolidating the previous knowledge (w/o $L_{\text{pro}}$) can significantly degrade the performance, which indicates that both are indispensable for online CL. Furthermore, when $L_{\text{pro}}$ only considers old classes and ignores new classes ($L_{\text{pro}}$ w/o $L_{\text{pro}}$), the performance also decreases. These results show that the equilibrium of all seen classes (OPE) can achieve the best performance and is crucial for online CL.

Effects of APF. To verify the advantage of APF, we compare it with the completely random mixup in Table 4. APF outperforms random mixup in all three scenarios. Notably, APF works significantly when the memory bank size is small, which shows that the feedback can prevent class confusion due to a restricted memory bank; see the supplementary material for extra ablation studies.

4.5. Validation of Online Prototypes

Fig. 5 shows the cosine similarity between online prototypes and global prototypes (prototypes of the entire memory bank) at each time step. For the first mini-batch of each task, online prototypes are equal to global prototypes.
(similarity is 1, omitted in Fig. 5). In the first task, online and global prototypes are updated synchronously with the model updates, resulting in high similarity. In subsequent tasks, the model initially learns inadequate features of new classes, causing online prototypes to be inconsistent with global prototypes and low similarity, which shows that accumulating early features as prototypes may be harmful to new tasks. However, the similarity will improve as the model learns, because the model gradually learns representative features of new classes. Furthermore, the similarity on old classes is only slightly lower, showing that online prototypes are resistant to forgetting.

5. Conclusion

This paper identifies shortcut learning as the key limiting factor for online CL, where the learned features are biased and not generalizable to new tasks. It also sheds light on why the online learning models fail to generalize well. Based on these, we present a novel online prototype learning (OnPro) framework to address shortcut learning and mitigate catastrophic forgetting. Specifically, by taking full advantage of introduced online prototypes, the proposed OPE aims to learn representative features of each class and discriminative features between classes for achieving an equilibrium status that separates all seen classes well when learning new classes, while the proposed APF is able to sense easily misclassified classes and enhance their decision boundaries with the feedback of online prototypes. Extensive experimental results on widely-used benchmark datasets validate the effectiveness of the proposed OnPro as well as its components. In the future, we will try more efficient alternatives, such as designing a margin loss to ensure discrimination between classes further.

Acknowledgement This work was supported in part by STIT2030-Major Projects (No. 2021ZD02000204), National Natural Science Foundation of China (Nos. 62101136 and 62176059), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01), ZJ Lab, Shanghai Municipal of Science and Technology Project (No. 20JC1419500), and Shanghai Center for Brain Science and Brain-inspired Technology.

References


