Wakening Past Concepts without Past Data: 
Class-Incremental Learning from Online Placebos

Yaoyao Liu\textsuperscript{1,2}  \hspace{0.5cm}  Yingying Li\textsuperscript{3}  \hspace{0.5cm}  Bernt Schiele\textsuperscript{2}  \hspace{0.5cm}  Qianru Sun\textsuperscript{4}

\textsuperscript{1}Johns Hopkins University  \hspace{0.5cm}  \textsuperscript{2}Max Planck Institute for Informatics, Saarland Informatics Campus  \hspace{0.5cm}  \textsuperscript{3}University of Illinois Urbana-Champaign  \hspace{0.5cm}  \textsuperscript{4}Singapore Management University

Abstract

Not forgetting old class knowledge is a key challenge for class-incremental learning (CIL) when the model continuously adapts to new classes. A common technique to address this is knowledge distillation (KD), which penalizes prediction inconsistencies between old and new models. Such prediction is made with almost new class data, as old class data is extremely scarce due to the strict memory limitation in CIL. In this paper, we take a deep dive into KD losses and find that “using new class data for KD” not only hinders the model adaption (for learning new classes) but also results in low efficiency for preserving old class knowledge. We address this by “using the placebos of old classes for KD”, where the placebos are chosen from a free image stream, such as Google Images, in an automatical and economical fashion. To this end, we train an online placebo selection policy to quickly evaluate the quality of streaming images (good or bad placebos) and use only good ones for one-time feed-forward computation of KD. We formulate the policy training process as an online Markov Decision Process (MDP), and introduce an online learning algorithm to solve this MDP problem without causing much computation costs. In experiments, we show that our method 1) is surprisingly effective even when there is no class overlap between placebos and original old class data, 2) does not require any additional supervision or memory budget, and 3) significantly outperforms a number of top-performing CIL methods, in particular when using lower memory budgets for old class exemplars, e.g., five exemplars per class.\footnote{Code: https://github.com/yaoyao-liu/online-placebos}

1. Introduction

AI learning systems are expected to learn new concepts while maintaining the ability to recognize old ones. In many practical scenarios, they cannot access the old data due to the limitations such as storage or data privacy but are expected to be able to recognize all seen classes. A pioneer work [35] formulated this problem in the class-incremental learning (CIL) pipeline: training samples of different classes are loaded into the memory phase-by-phase, and the model keeps on re-training with new class data (while discarding old class data) and is evaluated on the testing data of both new and old classes. The key challenge is that re-training the model on the new class data tends to override the knowledge acquired from the old classes [15, 30, 31, 34], and the problem is called “catastrophic forgetting”. To alleviate this problem, most CIL methods [6, 9, 12–14, 23, 25, 27–29, 32, 35, 43, 52–55] are equipped with knowledge distillation (KD) losses that penalize any feature and/or prediction inconsistencies between the models in adjacent phases.

The ideal KD losses should be computed on old class data since the teacher model (i.e., the model in the last phase) was trained on them. This is, however, impossible in the CIL setting, where almost all old class data are inaccessible in the new phase. Existing methods have to use new class data as a substitute to compute KD losses. We argue that this 1) hampers the learning of new classes as it distracts the model from fitting the ground truth labels of new classes, and 2) can not achieve the ideal result of KD, as the model can not generate the same soft labels (or features) on new class data as on old class data. We justify this from an empirical perspective as shown in Figure 1 (a): the upper bound of KD is achieved when using “old class data”, and if compared to it, using “new class data” sees a clear performance drop for recognizing both old and new classes. In Figure 1 (b), we show the reason by diving into loss computation details: when using new class samples (as substitutes) to compute CE and KD losses simultaneously, these two losses actually weaken each other, which does not happen in the ideal case of using old class samples.

To solve the above issue, people tried to use unlabeled external data (called placebos in this paper) to compute KD losses (rather than using the new data) [17, 24]. First, this
Figure 1. (a) **Average accuracy when computing KD losses on different data** using iCaRL [35] on CIFAR-100. The KD losses (softmax KL divergence loss) are computed on new class data (dark blue), placebos (of old class data) selected by our method (light blue), and old class data (green), i.e., the ideal case. (b) **Conceptual illustrations of the loss problem if using new class data for KD.** The dark blue and orange numbers denote the predictions of old and new classes, respectively. It is clear in (i) that the objectives are different when using a new class sample for KD (the oracle case is to have both “ascent”), e.g., the ground truth label for the second old class is 0, while the “KD label” at this position is 0.8. This is not an issue when using the old class sample, e.g., in (ii), its ground truth label and “KD label” have consistent magnitudes at the same position (1 and 0.7, respectively). (c) **Our selected placebos for two old classes (“road” and “table”) and their activation maps** using GradCAM [38] on CIFAR-100. The free image stream is ImageNet-1k which does not have class overlap with CIFAR-100. They are selected because their partial image regions contain similar visual cues to old classes.

idea is practical because we don’t need to allocate a large memory budget for placebos. We can select a small number of placebos from a free image stream, e.g., Google Image, and delete them immediately after computing KD losses. Second, this idea is effective because computing the KD losses on placebos can help to recall the old class knowledge without weakening the learning of new class data. As shown in Figure 1 (a), compared to the conventional way of using “new class data” (for KD), using placebos achieves the same-level new class performance as using “old class data”, and better old class recognition performance.

However, there are two open questions that need to be addressed when using placebos. **Q1:** How to adapt the placebo selection process in the non-stationary CIL pipeline. The ideal selection method needs to handle the dynamics of increasing classes in CIL, e.g., in a later incremental phase, it is expected to handle a more complex evaluation on the placebos of more old classes. **Q2:** How to control the computational and memory-related costs during the selection and utilization of placebos. It is not intuitive how to process external data without encroaching on the memory allocated for new class data or breaking the strict assumption of memory budget in CIL. Existing works [17, 51] cannot solve the above issues as they use fixed rules for placebo selection and require a large amount of memory to store the placebos.

We solve these questions by proposing a new method called PlaceboCIL that can adjust the policy of selecting placebos for each new incremental phase, in an online and automatic fashion without needing extra memory. Specifically, to tackle **Q1**, we formulate PlaceboCIL as an online Markov Decision Process (MDP) and introduce a novel online learning algorithm to learn a dynamic policy. In each new phase, this policy produces a phase-specific function to evaluate the quality of incoming placebos. The policy itself gets updated before the next phase. For Q2, we propose a mini-batch-based memory reusing strategy for PlaceboCIL. Given a free data stream, we sample a batch of unlabeled data, evaluate their quality by using our phase-specific evaluation function (generated by the learned policy), and keep only the high-quality placebos to compute the KD losses. After this, we remove this batch totally from memory before loading a new batch. In our implementation, this batch can be very small, e.g., 200 images. We randomly remove the same size (e.g., 200) of new class data to keep the strict assumption of memory budget.

We evaluate PlaceboCIL by incorporating it into multiple strong baselines such as PODNet [9], LUCIR [12], AANets [25], and FOSTER [44], and conducting a careful ablation study. Our results on three popular CIL benchmarks show the clear and consistent superiority of PlaceboCIL, especially when using a low memory budget for old class exemplars. For example, our method boosts the last-phase accuracy by 6.9 percentage points on average when keeping only 5 exemplars per old class in the memory. In addition, it is worth mentioning that PlaceboCIL is surprisingly efficient even when there is no class overlap between placebos and original old class data. The reason is that PlaceboCIL can make use of the local visual cues in placebos, e.g., similar visual cues of “table” are found on the local regions of an “piano” (and “dishwasher”) image as shown in Figure 1 (c).
Our contributions are three-fold. 1) A generic PlaceboCIL method that selects placebo images from a free image stream to solve the KD issue in existing methods. 2) A novel online learning algorithm for training a placebo selection policy and a mini-batch-based memory reusing strategy to avoid extra memory usage. 3) Extensive comparisons and visualizations on three CIL benchmarks, taking top-performing CIL models as baselines and with the same strict assumption on memory.

2. Related Work

Class-incremental learning (CIL) methods can be divided into three categories. Distillation-based methods introduce different knowledge distillation (KD) losses to consolidate previous knowledge. The key idea is to enforce prediction logits [22, 35], feature maps [9], or other essential information [13, 40, 42, 44, 49] to be close to those of the pre-phase model. Memory-based methods use a small number of preserved old class data (called exemplars) [5, 7, 28, 29, 33, 35, 39, 45, 47] or augmented data [55] to recall the old class knowledge. Network-architecture-based methods [1, 23, 37, 46, 48, 50] design incremental network architectures by expanding the network capacity for new classes or freezing partial network parameters to keep the old class knowledge. Our method can be used to improve different Distillation-based CIL methods.

Some prior works used unlabeled external data for class-incremental learning. [17] proposed a confidence-based sampling method to select unlabeled external data to compute a specially designed global distillation loss. [51] randomly selected unlabeled samples and used them to compute KD losses for model consolidation. [24] used unlabeled data to maximize the classifier discrepancy when integrating an ensemble of auxiliary classifiers. Our method differs from theirs in two aspects. 1) Our method uses the unlabeled data in a more generic way and can be applied to improve different distillation-based methods [12, 35, 44], while the existing methods use unlabeled data to assist their specially-designed loss terms or components. 2) We train an online policy to select better-unlabeled data to adapt to the non-stationary CIL pipeline while existing methods select unlabeled data by applying fixed (i.e., non-adaptive) rules in all incremental phases.

Online learning observes a stream of samples and makes a prediction for each element in the stream. There are mainly two settings in online learning: full feedback and bandit feedback. Full feedback means that the full reward function is given at each stage. It can be solved by Best-Expert algorithms [10]. Bandit feedback means that only the reward of the implemented decision is revealed. If the rewards are independently drawn from a fixed and unknown distribution, we may use, e.g., Thompson sampling [2] and UCB [4] to solve it. If the rewards are generated in a non-stochastic version, we can solve it by, e.g., Exp3 [3]. Online MDP is an extension of online learning. Many studies [11, 18–21] aim to solve it by converting it to online learning. In our case, we formulate the CIL as an online MDP and convert it into a classic online learning problem. The rewards in our MDP are non-stochastic because the training and validation data change in each phase. Therefore, we design our algorithm based on Exp3 [3].

3. Methodology

CIL has multiple “training-testing” phases during which the number of classes gradually increases to the maximum. In the 0-th phase, data \( D_{1:c_0} = \{D_1, ..., D_{c_0}\} \), including the training samples of \( c_0 \) classes, are used to learn the model \( \Theta_0 \). After this phase, only a small subset of \( D_{1:c_0} \) (i.e., exemplars denoted as \( E_{1:c_0} = \{E_1, ..., E_{c_0}\} \)) can be stored in the memory and used as replay samples in later phases. In the \( i \)-th phase, we use \( c_i \) to denote the number of classes we have observed so far. We get new class data \( D_{c_{i-1}+1:c_i} = \{D_{c_{i-1}+1}, ..., D_{c_i}\} \) of \( (c_i - c_{i-1}) \) classes and load exemplars \( E_{1:c_{i-1}} \) from the memory. Then, we initialize \( \Theta_i \) with \( \Theta_{i-1} \) and train it using \( T_{1:c_i} = E_{1:c_{i-1}} \cup D_{c_{i-1}+1:c_i} \). The model \( \Theta_i \) will be evaluated with a testing set \( Q_{1:c_i} = \{Q_1, ..., Q_{c_i}\} \) for all classes seen so far. Please note that in any phase of PlaceboCIL, we assume we can access a free image stream, where we can load unlabeled images and select placebos.

PlaceboCIL formulates the CIL task as an online MDP. In each phase, we update a policy, for which we sample a class-balanced subset from training data as the testing set, and use the updated policy to produce a phase-specific evaluation function. During model training, we sample unlabeled images, use the evaluation function to quickly judge the image quality (good or bad placebos), and select the good ones to compute KD losses. In this section, we introduce the formulation of online MDP in Section 3.1, show how to apply the policy to select placebos and compute KD losses in Section 3.2, and provide an online learning algorithm to update the policy in Section 3.3. The pseudocode is given in Algorithms 1 and 2.

3.1. Online MDP Formulation for CIL

The placebo selection process in CIL should be online inherently: training data (and classes) get updated in each phase, so the placebo selection policy should be updated accordingly. Thus, it is intuitive to formulate the CIL as an online MDP [23]. In the following, we provide detailed formulations.

**Stages.** Each phase in the CIL task can be viewed as a stage in the online MDP.

**States.** The state should define the current situation of the agent. In CIL, we use the model \( \Theta_i \) as the state of the \( i \)-th phase (i.e., stage). We use \( S \) to denote the state space.
Actions. We define the action as \( a_i = (\beta_i, \gamma_i) \), consisting of the hyperparameters \( (\beta_i \text{ and } \gamma_i) \) used to create an evaluation function. As \( \beta_i \) and \( \gamma_i \) vary in a continuous range, we discretize them to define a finite action space.\(^2\) We will elaborate on how to take an action and deploy the hyperparameters in Section 3.2.

Policy \( \pi = \{p(a|\Theta_i)\}_{a \in \mathcal{A}} \) is a probability distribution over the action space \( \mathcal{A} \), given the current state \( \Theta_i \). We will elaborate on how to update the policy using our proposed online learning algorithm in Section 3.3.

Environments. We take the training and testing data in each phase as the environment. In the \( i \)-th phase, the environment is \( \mathcal{H}_i = (\mathcal{T}_{1:c_i}, \mathcal{Q}_{1:c_i}) \), where \( \mathcal{T}_{1:c_i} \) is the training data and \( \mathcal{Q}_{1:c_i} \) is the testing data. The environment is time-varying because we observe different training data (and classes) in each new phase.

Rewards. CIL aims to train a model that is efficient in recognizing all classes seen so far. Therefore, it is intuitive to use testing accuracy as the reward in each phase. We cannot observe any reward (i.e., testing accuracy) directly because the testing data is not accessible during training. We solve this by building a local testing set using a subset of training data (see details in Section 3.3). Our objective is to maximize a cumulative reward, i.e., \( R = \sum_{i=1}^{N} r_{\mathcal{H}_i}(\Theta_i, a_i) \), where \( r_{\mathcal{H}_i}(\Theta_i, a_i) \) denotes the \( i \)-th phase reward. The reward function \( r_{\mathcal{H}_i} \) changes with \( \mathcal{H}_i \), so it is time-varying.

### 3.2. Placebo Selection

In the following, we introduce how to build phasespecific evaluation functions using the policy, select high-quality placebos without breaking memory constraints, and compute KD losses with the selected placebos. The computation flow (in each phase) is illustrated in Figure 2.

**Computing prototypes.** Our placebo selection is based on the distance from the placebo to the class prototype, i.e., the mean feature of each class [41]. First, we compute the prototypes of all seen classes. We use exemplars to compute the prototypes of old classes, and use new class training data for new class prototypes, as follows,

\[
\text{Pro}(\mathcal{E}_n) = \frac{1}{|\mathcal{E}_n|} \sum_{z \in \mathcal{E}_n} \mathcal{F}_{\Theta_i}(z), \quad \text{Pro}(\mathcal{D}_l) = \frac{1}{|\mathcal{D}_l|} \sum_{z \in \mathcal{D}_l} \mathcal{F}_{\Theta_i}(z),
\]

where \( \mathcal{F}_{\Theta_i}(\cdot) \) denotes the encoder (i.e., the feature extractor) of \( \Theta_i \). Pro(\( \mathcal{E}_n \)) and Pro(\( \mathcal{D}_l \)) denote the prototypes of the \( n \)-th old class and the \( l \)-th new class, respectively.

**Building evaluation functions.** We argue that high-quality placebos for the \( m \)-th old class should meet two requirements: (1) being close to the prototype of the \( m \)-th class in the feature space because they will be used to activate the related neurons of the \( m \)-th old class in the model; and (2) being far from the prototypes of all the other classes in the feature space so that they will not cause the KD issue (as shown in Figure 1). To achieve these, we design the following evaluation function \( S_m(x) \) for the \( m \)-th old class in the \( i \)-th phase:

\[
S_m(x) = -\text{Sim} \left( \mathcal{F}_{\Theta_i}(x), \text{Pro}(\mathcal{E}_m) \right) + \beta_i \sum_{n=1 \atop n \neq m}^{c_i} \frac{\text{Sim} \left( \mathcal{F}_{\Theta_i}(x), \text{Pro}(\mathcal{E}_n) \right)}{c_i - c_{i-1}} + \gamma_i \sum_{l=c_{i-1}+1}^{c_i} \frac{\text{Sim} \left( \mathcal{F}_{\Theta_i}(x), \text{Pro}(\mathcal{D}_l) \right)}{c_i - c_{i-1}}.
\]
Calculating loss with placebos. We calculate the overall loss as follows,
\[
    \mathcal{L} = \mathcal{L}_{\text{CE}}(\Theta_i; d \cup e) + \lambda \mathcal{L}_{\text{KD}}(\Theta_{i-1}; \Theta_i; p \cup e),
\]
where \(\mathcal{L}_{\text{CE}}\) and \(\mathcal{L}_{\text{KD}}\) denote the CE loss and KD losses, respectively. \(\lambda\) is a hyperparameter to balance the two losses [35]. To control the memory usage, we delete \(p\) from \(\mathcal{P}\) immediately after calculating the loss. When \(\mathcal{P}\) is empty, we repeat the placebo selection operation.

3.3. Online Policy Learning Algorithm

A common approach to solving an online MDP is to approximate it as an online learning problem and solve it using online learning algorithms [2, 3, 10]. We also follow this idea in PlaceboCIL, and our approximation follows [11], which is theoretically proved to have the optimal regret. Specifically, Even-Dar et al. [11] relax the Markovian assumption of the MDP by decoupling the cumulative reward function and letting it be time-dependent so that they can solve online MDP by standard online learning algorithms.

However, we cannot directly apply the algorithms proposed in [11] to our problem. It is because they assume full feedback, i.e., the model can observe the rewards of all actions in every learning phase (which is also why its online learning problem can be solved by Best Expert algorithms [10]). While in CIL, we cannot observe any reward (i.e., the testing accuracies) because the testing data \(Q_{1:c_i}\) are not accessible in any phase \(i\). To address this problem, we split the training data we have in each phase into two

where \(x\) denotes an unlabeled input image, and \(\text{Sim}(\cdot, \cdot)\) denotes cosine similarity. \(\beta_i\) and \(\gamma_i\) are two hyperparameters from the action \(a_i = (\beta_i, \gamma_i)\), sampled by the policy \(\pi\).

Allocating mini-batch-based memory for placebos. We need to allocate a small amount of memory to store unlabeled images (before evaluating them). At the beginning of the \(i\)-th phase, we allocate memory buffers \(\mathcal{U}\) and \(\mathcal{P}\) respectively for the unlabeled image candidates and the selected placebos. In order to not exceed the memory budget, we randomly remove the same number, i.e., \(|\mathcal{U} + \mathcal{P}|\), of samples from the training data of new classes. Our empirical results show this “remove” does not degrade the model performance on new classes.

Selecting placebos. Whenever the placebo buffer \(\mathcal{P}\) is empty, we load a batch of unlabeled samples \(\mathcal{U}\) from the free image stream, and choose \(K\) placebos for each old class to add into \(\mathcal{P}\), as follows,

\[
    \mathcal{P} := \left\{ x_k \right\}_{k=1}^{c_{i-1} \times K} = \arg\max_{x_k \in \mathcal{U}} \sum_{m=1}^{c_{i-1}} \sum_{k=1}^{K} S_m(x_k). \tag{3}
\]

Calculating loss with placebos. After selecting placebos,

we sample a batch of new class data \(d \subset \mathcal{D}_{c_{i-1} + 1:c_i}\), a batch of old class exemplars \(e \subset \mathcal{E}_{1:c_i}\), and a batch of placebos \(p \subset \mathcal{P}\). We calculate the overall loss as follows,

\[
    \mathcal{L} = \mathcal{L}_{\text{CE}}(\Theta_i; d \cup e) + \lambda \mathcal{L}_{\text{KD}}(\Theta_{i-1}; \Theta_i; p \cup e),
\]

Algorithm 1: Our PlaceboCIL in Phase \(i (i \geq 1)\)

**Input:** Old model \(\Theta_{i-1}\), training data \(\mathcal{T}_{1:c_i}\), testing data \(Q_{1:c_i}\), learnable parameters \(w\), numbers of epochs \(M_1\) and \(M_2\).

**Output:** New model \(\Theta_i\), new exemplars \(\mathcal{E}_{0:i}\), learnable parameters \(w\).

// Policy learning

1. if \(i = 1\) then
2.  Initialize \(w = \{1, \ldots, 1\}\):
3. for \(t \) in \(i, \ldots, T\) do
4.  Randomly sample a class-balanced subset \(B_{1:c_i}\)
    from \(\mathcal{T}_{1:c_i}\);
5.  Create the local environment
    \(h_i = ((\mathcal{T}_{1:c_i}) \setminus B_{1:c_i}, B_{1:c_i});\)
6.  Set the policy \(\pi\) by \(w/\|w\|;\)
7.  Sample an action \(a_t \sim \pi;\)
8. for \(j \) in \(i, \ldots, i+n\) do
9.  Train \(\Theta_j\) for \(M_1\) epochs by Algorithm 2
    with inputs \(\Theta_{j-1}, a_t, h_i;\)
10. Collect the reward \(r_{h_i}(\Theta_j, a_t);\)
11. Compute the cumulative reward \(R(a_t, h_i)\) by
    Eq. 5;
12. Update \(w\) by Eq. 6;
// CIL training
13. Sample an action \(a_i \sim \pi;\)
14. Train \(\Theta_i\) for \(M_2\) epochs by Algorithm 2 with inputs
    \(\Theta_{i-1}, a_i, h_i = (\mathcal{T}_{1:c_i}, Q_{1:c_i});\)
15. Select new exemplars \(\mathcal{E}_{1:c_i}\) from \(\mathcal{T}_{1:c_i}\),

Algorithm 2: Training with placebos for action \(a\)

**Input:** Old model \(\Theta_{old}\), action \(a = \{\beta, \gamma\}\),
environment \(h\) = \{\(T, Q\)\}.

**Output:** New model \(\Theta\), reward \(r_h(\Theta, a)\) (i.e., the testing accuracy).

1. Initialize \(\Theta\) with \(\Theta_{old};\)
2. Create \(\{S_m(x)\}_{m=1}^{c_{i-1}}\) based on \(a = \{\beta, \gamma\}\) using
   Eq. 2;
3. for epochs do
4.  Set \(\mathcal{P} = \emptyset;\)
5. while \(\mathcal{P} == \emptyset) do
6.  Sample \(\mathcal{U}\) from the free image stream;
7.  Select placebos \(\mathcal{P} \subset \mathcal{U}\) using Eq. 3;
8. for iterations do
9.  Sample mini-batches \(p, d,\) and \(e;\)
10. Compute the loss \(\mathcal{L}\) by Eq. 4 and update \(\Theta;\)
11. Update placebo buffer \(\mathcal{P} := \mathcal{P} \setminus p;\)
12. Compute the reward \(r_h(\Theta, a)\) on \(Q\).
In the first phase, we sample an action \( w \) of \( \pi_a \) in \( \mathbb{w} \), which is a constant, and can be regarded as the learning rate. After updating \( w \), we get the policy \( \pi = \frac{w}{|w|} \). The pseudocode is available in Algorithms 1 and 2.

### 4. Experiments

We evaluate our method on three CIL benchmarks and achieve consistent improvements over multiple baseline methods. Below we introduce datasets and implementation details, followed by results and analyses, including the comparison to the state-of-the-art, an ablation study, and the visualization of our PlaceboCIL.

#### Datasets and free image streams

We use three datasets: CIFAR-100 [16], ImageNet-100 [35], and ImageNet-1k [36]. ImageNet-100, which contains 100 classes, is sampled from ImageNet-1k. We use exactly the same classes or orders as the related works [12, 35]. For CIFAR-100, we use ImageNet-1k as the free image stream. For ImageNet-100, we use a 900-class subset of ImageNet-1k, which is the complement of ImageNet-100 in ImageNet-1k. For ImageNet-1k, we use a 1,000-class subset of ImageNet-21k [8] without any overlapping class (different subclasses from those in ImageNet-1k).

#### Implementation details

Following [9, 12, 25, 26], we use a modified 32-layer ResNet for CIFAR-100 and an 18-layer ResNet for ImageNet datasets. The number of exemplars for each class is 20 in the default setting. The training batch size is 128. On CIFAR-100 (ImageNet-Subset/1k), we train it for 160 (90) epochs in each phase, and divide the learning rate by 10 after 80 (30) and then after 120 (60) epochs. If the baseline is POD-AANets [25], we fine-tune the model for 20 epochs using only exemplars. We apply different forms of distillation losses on different baselines: (1) if the
From the table, we make the following observations. 1) Us-

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-100</th>
<th></th>
<th>ImageNet-100</th>
<th></th>
<th>ImageNet-1k</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N=5</td>
<td>10</td>
<td>25</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>TPCIL [42]</td>
<td>65.34</td>
<td>63.58</td>
<td>–</td>
<td>76.27</td>
<td>74.81</td>
</tr>
<tr>
<td>GeoDL [40]</td>
<td>65.14</td>
<td>65.03</td>
<td>63.12</td>
<td>76.63</td>
<td>75.40</td>
</tr>
<tr>
<td>DER [48]</td>
<td>68.65</td>
<td>67.48</td>
<td>66.18</td>
<td>78.40</td>
<td>78.20</td>
</tr>
<tr>
<td>ELI [13]</td>
<td>68.78</td>
<td>66.62</td>
<td>64.72</td>
<td>73.54</td>
<td>71.82</td>
</tr>
<tr>
<td>GD+ext [17]</td>
<td>63.17±0.47</td>
<td>58.71±0.39</td>
<td>51.79±0.42</td>
<td>75.67±0.51</td>
<td>72.08±0.61</td>
</tr>
<tr>
<td>MUC-LwF [24]</td>
<td>59.03±0.33</td>
<td>53.27±0.47</td>
<td>49.06±0.40</td>
<td>72.31±0.53</td>
<td>68.92±0.60</td>
</tr>
<tr>
<td>POD-AANets [25]</td>
<td>66.12±0.41</td>
<td>64.11±0.32</td>
<td>62.12±0.51</td>
<td>76.63±0.47</td>
<td>75.40±0.36</td>
</tr>
<tr>
<td>w/ PlaceboCIL (ours)</td>
<td>67.65±0.45</td>
<td>65.78±0.40</td>
<td>64.95±0.46</td>
<td>78.24±0.52</td>
<td>77.14±0.47</td>
</tr>
<tr>
<td>FOSTER [44]</td>
<td>70.62±0.58</td>
<td>68.43±0.45</td>
<td>63.83±0.62</td>
<td>80.21±0.67</td>
<td>77.63±0.73</td>
</tr>
<tr>
<td>w/ PlaceboCIL (ours)</td>
<td>71.97±0.40</td>
<td>70.31±0.59</td>
<td>67.02±0.65</td>
<td>82.03±0.40</td>
<td>79.52±0.60</td>
</tr>
</tbody>
</table>

Table 2. Average accuracy (%) across all phases. The first block shows top-performing CIL methods. The second block shows CIL methods that use unlabeled data. The third block shows our method.

<table>
<thead>
<tr>
<th>No.</th>
<th>Setting</th>
<th>iCaRL</th>
<th>LUCIR-AANets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average</td>
<td>Last</td>
</tr>
<tr>
<td>1</td>
<td>Baseline</td>
<td>57.12</td>
<td>47.49</td>
</tr>
<tr>
<td>2</td>
<td>PlaceboCIL</td>
<td>61.01</td>
<td>51.45</td>
</tr>
<tr>
<td>3</td>
<td>Overlapping</td>
<td>62.15</td>
<td>52.62</td>
</tr>
<tr>
<td>4</td>
<td>Non-overlapping</td>
<td>61.52</td>
<td>51.70</td>
</tr>
<tr>
<td>5</td>
<td>New data</td>
<td>57.70</td>
<td>47.51</td>
</tr>
<tr>
<td>6</td>
<td>Old data (oracle)</td>
<td>66.64</td>
<td>58.03</td>
</tr>
<tr>
<td>7</td>
<td>w/o Online learning</td>
<td>60.27</td>
<td>50.57</td>
</tr>
<tr>
<td>8</td>
<td>Offline RL</td>
<td>61.09</td>
<td>50.81</td>
</tr>
<tr>
<td>9</td>
<td>Higher confidence</td>
<td>60.43</td>
<td>49.36</td>
</tr>
<tr>
<td>10</td>
<td>Random placebos</td>
<td>56.27</td>
<td>46.64</td>
</tr>
</tbody>
</table>

Table 3. Ablation results (%) on CIFAR-100, N=5. (1) **First block: baselines.** Row 1 shows the baselines. Row 2 shows our method. All other settings (Rows 3-10) are based on Row 2. (2) **Second block: different free data streams.** Rows 3-6 show the ablation results for the following free data streams. (3) **Third block: different policy learning methods.** Row 7 is for using fixed evaluation functions (β=γ=1). Row 8 uses the offline RL (the REINFORCE algorithm) to train the selection policy. (4) **Fourth block: different placebo selection strategies.** Row 9 uses unlabeled data with higher confidence. Row 10 uses them randomly.

baselines are LwF and iCaRL, we use the softmax KL divergence loss; (2) if the baselines are LUCIR and AANets, we use the cosine embedding loss [12]; and (3) if the baseline is POD-AANets, we use pooled outputs distillation loss [9]. For our PlaceboCIL, |U| and |P| are set as 1,000 and 200, respectively. All experiments of our PlaceboCIL use the “strict budget” setting, i.e., deleting |U + P| samples from training data to avoid exceeding the memory budget.

**Results on five baselines.** Table 1 shows the average and last-phase accuracy for five baselines (i.e., LwF [22], iCaRL [35], LUCIR [12], AANets [25], and FOSTER [44]). From the table, we make the following observations. 1) Us-

<table>
<thead>
<tr>
<th></th>
<th>2000</th>
<th>1000</th>
<th>500</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc. (%)</td>
<td>61.13</td>
<td>61.01</td>
<td>58.23</td>
<td>57.12</td>
</tr>
</tbody>
</table>

Table 4. Ablation results (%) for different memory buffer sizes |U| on CIFAR-100, N=5. The baseline is iCaRL [35].

Intriguingly, we find that our PlaceboCIL outperforms all previous methods. Interestingly, we find that we can surpass others more when the number of phases is larger—where there are more serious forgetting problems. For example, when N=5, we improve POD-AANets by 4.4% on the ImageNet-100, while this number is only 1.6% when N=5 (which is an easier setting with more saturated results).

**Comparisons to the CIL methods using unlabeled data.** Table 2 (Blocks 2&3) shows the results of our best model (taking PlaceboCIL as a plug-in module in the top method [44]) and some recent top-performing methods. We can see that using our PlaceboCIL outperforms all previous methods. Interestingly, we find that we can surpass others more when the number of phases is larger—where there are more serious forgetting problems. For example, when N=5, we improve POD-AANets by 4.4% on the ImageNet-100, while this number is only 1.6% when N=5 (which is an easier setting with more saturated results). This reflects the encouraging efficiency of our method for reducing the forgetting of old class knowledge in CIL models.
DMC [51], we didn’t find the public code. So, we compare ours with DMC using their paper’s setting: iCaRL w/ ours achieves 62.3%, while the result of DMC is 59.1% (CIFAR-100, 10 phases, 10 classes/phase).

Ablation study. Table 3 shows the ablation results.

1) First block. Rows 1 and 2 show the baseline and our method, respectively.

2) Second block: different free data streams. Rows 3-6 show the ablation results for the following free data streams. (1) “Overlapping” means including samples from the overlapping classes between CIFAR-100 and ImageNet. (2) “Non-overlapping” means using only the samples of non-overlapping classes between CIFAR-100 and ImageNet (more realistic than “Overlapping”). (3) “New data” means using only the current-phase new class data (i.e., without using any free data stream) as candidates to select placebos. (4) “Old data” means the original old class data are all accessible when computing KD losses (i.e., the upper bound of KD effect). Please note that in (1) and (2), two classes are considered “overlapping” if their classes or super-classes overlap. For example, “n02640242 - sturgeon” in ImageNet-1k is regarded as an overlapping class of the “fish” in CIFAR-100, because they overlap at the level of super-class (i.e., “fish”). When comparing Row 4 with Row 2, we can find that our method is robust to the change of data streams: even if all overlapping classes are removed, our method can still achieve the same-level performance. Comparing Row 5 with Row 2, we can get a clear sense that using additional unlabeled data is definitely helpful. Comparing Row 6 with Row 2, we see that our method can still achieve the same-level performance.

3) Third block: different policy learning methods. Row 7 is for using fixed evaluation functions ($\beta_i=\gamma_i=1$). Row 8 uses the offline RL (the REINFORCE algorithm [26]) to train the selection policy. Comparing Row 7 with Row 2 shows that using online learning successfully boosts the model performance. Comparing Row 8 with Row 2, we are happy to see that our online learning method achieves the same-level performance as the offline RL while the training time is much less. The training time of the baseline (without learning a policy) is 2.7 hours. It becomes around 650 hours if we solve the MDP by offline RL. In contrast, using our online method takes only 4.5 hours.

4) Fourth block: different placebo selection strategies. Row 9 uses unlabeled data with higher confidence following [17]. Row 10 uses them randomly following [51]. Comparing these results with Row 2 shows our superiority. The “mini-batch-based memory reusing strategy” is applied in Rows 9 and 10.

5) Different memory buffer sizes. Table 4 shows the ablation results when using different buffer sizes for $U$. We can observe that larger buffer sizes achieve better results. Interestingly, we can also see that using a relatively small buffer size (e.g., 500) can still improve the baseline.

Visualization results. Figure 3 (a) demonstrates the activation maps visualized by Grad-CAM for the selected placebos of two old classes on CIFAR-100 (“road” and “table”). ImageNet-1k is the free data stream. We can see that the selected placebos contain the parts of “road” and “table” even though their original labels (on ImageNet-1k) are totally different classes. While this is not always the case, our method seems to find sufficiently related images to old classes that activate the related neurons for old classes (“road” and “table”). To illustrate that, Figure 3 (b) shows t-SNE results for placebos, old class data (not visible during training), and new class data. We can see that the placebos are located near the old class data and far away from the new class data. This is why placebos can recall the old knowledge without harming the new class learning.

5. Conclusions

We proposed a novel method, PlaceboCIL, which selects high-quality placebo data from free data streams and uses them to improve the effect of KD in CIL. We designed an online learning method to make the selection of placebos more adaptive in different phases and a mini-batch-based memory-reusing strategy to control memory usage. Extensive experimental results show that our method is general and efficient.
References


