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

Yaoyao Liu[1,2] Yingying Li[3] Bernt Schiele[2] Qianru Sun[4]

[1]Johns Hopkins University [2]Max Planck Institute for Informatics, Saarland Informatics Campus
[3]University of Illinois Urbana-Champaign [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.[1]

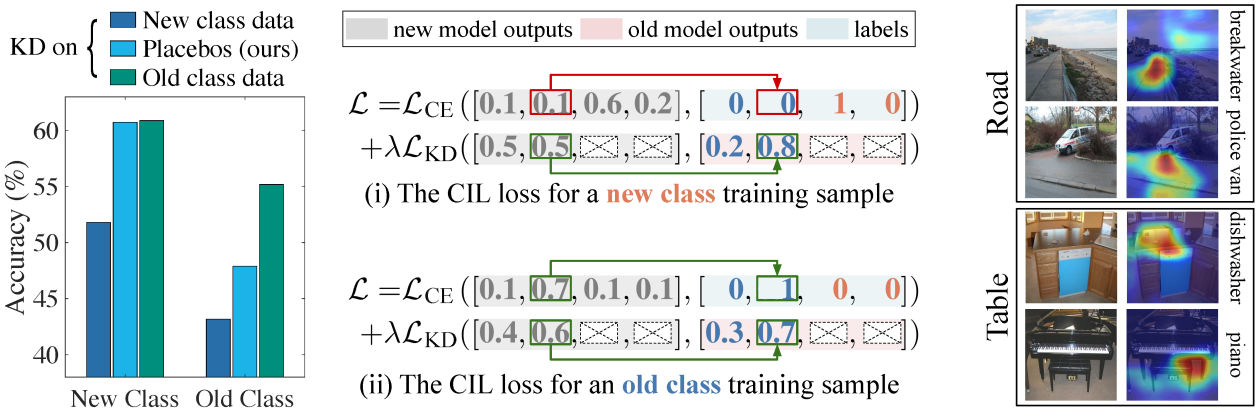
1. Introduction

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

[1]Code: https://github.com/yaoyao-liu/online-placebos



(a) Average accuracy (b) Conceptual illustrations of the CIL losses (c) Selected placebos

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 class data 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 $\mathcal{D}_{1:c_0}$=$\{\mathcal{D}_1, ..., \mathcal{D}_{c_0}\}$, including the training samples of c_0 classes, are used to learn the model Θ_0. After this phase, only a small subset of $\mathcal{D}_{1:c_0}$ (i.e., exemplars denoted as $\mathcal{E}_{1:c_0}$=$\{\mathcal{E}_1, ..., \mathcal{E}_{c_0}\}$) 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 $\mathcal{D}_{c_{i-1}+1:c_i}$=$\{\mathcal{D}_{c_{i-1}+1}, ..., \mathcal{D}_{c_i}\}$ of $(c_i - c_{i-1})$ classes and load exemplars $\mathcal{E}_{1:c_{i-1}}$ from the memory. Then, we initialize Θ_i with Θ_{i-1}, and train it using $\mathcal{T}_{1:c_i}$=$\mathcal{E}_{1:c_{i-1}} \cup \mathcal{D}_{c_{i-1}+1:c_i}$. The model Θ_i will be evaluated with a testing set $\mathcal{Q}_{1:c_i}$=$\{\mathcal{Q}_1, ..., \mathcal{Q}_{c_i}\}$ 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 Θ_i as the state of the i-th phase (i.e., stage). We use $\mathbb{S}$ to denote the state space.

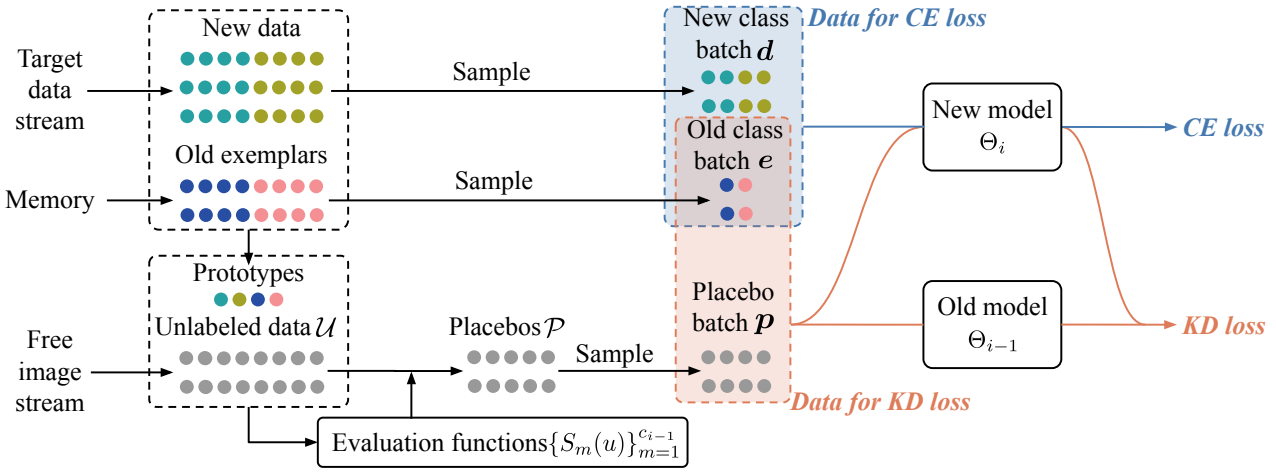


Figure 2. Our PlaceboCIL in the i-th phase. At the beginning of this phase, we build phase-specific evaluation functions $\{S_m(u)\}_{m=1}^{c_{i-1}}$. During training, we select placebos as follows. 1) We load an unlabeled data batch $\mathcal{U}$ from the free image stream. 2) We compute scores using $\{S_m(u)\}_{m=1}^{c_{i-1}}$ for all samples in $\mathcal{U}$. 3) For each old class m, we select K placebos with the highest scores and add them to $\mathcal{P}$. 4) We delete used placebos from $\mathcal{P}$ after computing the loss. 5) When we use up the selected placebos in $\mathcal{P}$, we repeat the selection steps.

Actions. We define the action as $\mathbf{a}_i{=}(\beta_i, \gamma_i)$, consisting of the hyperparameters (β_i and γ_i) used to create an evaluation function. As β_i and γ_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 $\pi{=}\{p(\mathbf{a}|\Theta_i)\}_{\mathbf{a}\in\mathbb{A}}$ is a probability distribution over the action space $\mathbb{A}$, given the current state Θ_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., $R = \sum_{i=1}^{N} r_{\mathcal{H}_i}(\Theta_i, \mathbf{a}_i)$, where $r_{\mathcal{H}_i}(\Theta_i, \mathbf{a}_i)$ 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 phase-specific 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,

$$\mathrm{Pro}(\mathcal{E}_n) = \frac{1}{|\mathcal{E}_n|}\sum_{z\in\mathcal{E}_n}\mathcal{F}_{\Theta_i}(z), \quad \mathrm{Pro}(\mathcal{D}_l) = \frac{1}{|\mathcal{D}_l|}\sum_{z\in\mathcal{D}_l}\mathcal{F}_{\Theta_i}(z), \tag{1}$$

where $\mathcal{F}_{\Theta_i}(\cdot)$ denotes the encoder (i.e., the feature extractor) of Θ_i. $\mathrm{Pro}(\mathcal{E}_n)$ and $\mathrm{Pro}(\mathcal{D}_l)$ 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 $\mathcal{S}_m(x)$ for the m-th old class in the i-th phase:

$$\begin{aligned}\mathcal{S}_m(x) = &-\mathrm{Sim}\left(\mathcal{F}_{\Theta_i}(x), \mathrm{Pro}(\mathcal{E}_m)\right)\\ &+\beta_i\sum_{\substack{n=1\\n\neq m}}^{c_{i-1}}\frac{\mathrm{Sim}\left(\mathcal{F}_{\Theta_i}(x), \mathrm{Pro}(\mathcal{E}_n)\right)}{c_{i-1}-1}\\ &+\gamma_i\sum_{l=c_{i-1}+1}^{c_i}\frac{\mathrm{Sim}\left(\mathcal{F}_{\Theta_i}(x), \mathrm{Pro}(\mathcal{D}_l)\right)}{c_i-c_{i-1}},\end{aligned} \tag{2}$$

[2] Though discretization suffers the curse of dimensionality, our experiments show that with a coarse grid, we already have significant improvements over pre-fixed hyperparameters.

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

Input : Old model Θ_{i-1}, training data $\mathcal{T}_{1:c_i}$, testing data $\mathcal{Q}_{1:c_i}$, learnable parameters $\mathbf{w}$, numbers of epochs M_1 and M_2.
Output: New model Θ_i, new exemplars $\mathcal{E}_{0:i}$, learnable parameters $\mathbf{w}$.

// Policy learning
if i=1 **then**
  Initialize $\mathbf{w} = \{1, \ldots, 1\}$;
for t in $i, ..., T$ **do**
  Randomly sample a class-balanced subset $\mathcal{B}_{1:c_i}$ from $\mathcal{T}_{1:c_i}$,;
  Create the local environment $h_i = ((\mathcal{T}_{1:c_i}) \setminus \mathcal{B}_{1:c_i}, \mathcal{B}_{1:c_i})$;
  Set the policy $\pi = \mathbf{w}/||\mathbf{w}||$;
  Sample an action $\mathbf{a}_t \sim \pi$;
  for j in $i, ..., i+n$ **do**
    Train Θ_j for M_1 epochs by **Algorithm 2** with inputs Θ_{j-1}, $\mathbf{a}_t$, h_i;
    Collect the reward $r_{h_i}(\Theta_j, \mathbf{a}_t)$;
  Compute the cumulative reward $R(\mathbf{a}_t, h_i)$ by Eq. 5;
  Update $\mathbf{w}$ by Eq. 6;
// CIL training
Sample an action $\mathbf{a}_i \sim \pi$;
Train Θ_i for M_2 epochs by **Algorithm 2** with inputs Θ_{i-1}, $\mathbf{a}_i$, $\mathcal{H}_i = (\mathcal{T}_{1:c_i}, \mathcal{Q}_{1:c_i})$;
Select new exemplars $\mathcal{E}_{1:c_i}$ from $\mathcal{T}_{1:c_i}$.

Algorithm 2: Training with placebos for action $\mathbf{a}$

Input : Old model Θ_{old}, action $\mathbf{a} = \{\beta, \gamma\}$, environment $h = \{\mathcal{T}, \mathcal{Q}\}$.
Output: New model Θ, reward $r_h(\Theta, \mathbf{a})$ (i.e., the testing accuracy).

Initialize Θ with Θ_{old};
Create $\{S_m(x)\}_{m=1}^{c_{i-1}}$ based on $\mathbf{a} = \{\beta, \gamma\}$ using Eq. 2;
for *epochs* **do**
  Set $\mathcal{P} = \varnothing$;
  while $\mathcal{P} == \varnothing$ **do**
    Sample $\mathcal{U}$ from the free image stream;
    Select placebos $\mathcal{P} \subset \mathcal{U}$ using Eq. 3;
    for *iterations* **do**
      Sample mini-batches $\boldsymbol{p}$, $\boldsymbol{d}$, and $\boldsymbol{e}$;
      Compute the loss $\mathcal{L}$ by Eq. 4 and update Θ;
      Update placbo buffer $\mathcal{P} := \mathcal{P} \setminus \boldsymbol{p}$;
Compute the reward $r_h(\Theta, \mathbf{a})$ on $\mathcal{Q}$.

where x denotes an unlabeled input image, and $\text{Sim}(\cdot, \cdot)$ denotes cosine similarity. β_i and γ_i are two hyperparameters from the action $\mathbf{a}_i = (\beta_i, \gamma_i)$, sampled by the policy π.

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,

$$\mathcal{P} := \{x_k\}_{k=1}^{c_{i-1} \times K} = \text{argmax}_{x_k \in \mathcal{U}} \sum_{m=1}^{c_{i-1}} \sum_{k=1}^{K} \mathcal{S}_m(x_k). \quad (3)$$

Calculating loss with placebos. After selecting placebos, we sample a batch of new class data $\boldsymbol{d} \subset \mathcal{D}_{c_{i-1}+1:c_i}$, a batch of old class exemplars $\boldsymbol{e} \subset \mathcal{E}_{1:c_0}$, and a batch of placebos $\boldsymbol{p} \subset \mathcal{P}$. We calculate the overall loss as follows,

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(\Theta_i; \boldsymbol{d} \cup \boldsymbol{e}) + \lambda \mathcal{L}_{\text{KD}}(\Theta_{i-1}, \Theta_i; \boldsymbol{p} \cup \boldsymbol{e}), \quad (4)$$

where $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{KD}}$ denote the CE loss and KD losses, respectively. λ is a hyperparameter to balance the two losses [35]. To control the memory usage, we delete $\boldsymbol{p}$ 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

Method	20 exemplars/class		10 exemplars/class		5 exemplars/class	
	Average	Last	Average	Last	Average	Last
LwF [22]	53.19	43.18	45.96	34.10	35.41	24.91
w/ ours	59.08 +5.89	49.15 +5.97	53.61 +7.65	38.36 +4.26	41.55 +6.14	28.68 +3.77
iCaRL [35]	57.12	47.49	53.43	41.49	43.73	34.33
w/ ours	61.24 +4.12	51.47 +3.98	59.11 +5.68	46.42 +4.93	51.55 +7.82	39.35 +5.02
LUCIR [12]	63.17	53.71	60.50	49.08	51.36	39.37
w/ ours	65.28 +2.11	56.23 +2.52	64.79 +4.29	55.44 +6.36	62.74 +11.35	53.25 +13.88
LUCIR-AANets [25]	66.72	55.77	61.12	48.83	53.81	42.93
w/ ours	67.16 +0.44	59.14 +3.37	64.30 +3.18	52.92 +4.09	60.27 +6.46	48.45 +5.52
FOSTER [44]	70.62	62.97	62.03	52.23	56.80	43.11
w/ ours	71.97 +1.35	64.43 +1.46	65.12 +3.09	54.81 +2.48	62.78 +5.98	50.72 +7.61

Table 1. Evaluation results (%) on CIFAR-100 (N=5) using different baselines *w/* and *w/o* our PlaceboCIL. "Average" denotes the average accuracy over all phases. "Last" denotes the last phase (5-th phase) accuracy.

subsets: one for training and another for validation. Once we have a validation set, we can solve our online learning problem based on Exp3 [3,23]—a simple and effective bandit algorithm. In the following, we elaborate on how we do this data splitting in each local dataset (i.e., the entire data we have in each training phase of CIL), compute the decoupled cumulative reward, and learn the policy π with Exp3.

Rebuilding local datasets. To compute reward, we sample a class-balanced subset $\mathcal{B}_{1:c_i}$ from the training data $\mathcal{T}_{1:c_1}$. $\mathcal{B}_{1:c_i}$ contains the same number of samples for both the old and new classes. In this way, we rebuild the local training and validate sets, and update the environment from the oracle $\mathcal{H}_i$=$(\mathcal{T}_{1:c_i}, \mathcal{Q}_{1:c_i})$ (which is unavailable in CIL) to the local environment h_i=$(\mathcal{T}_{1:c_i} \setminus \mathcal{B}_{1:c_i}, \mathcal{B}_{1:c_i})$.

Decoupled cumulative reward. We create the decoupled cumulative reward function R based on the original cumulative reward function $\sum_{j=1}^{N} r_{\mathcal{H}_j}(\Theta_j, \mathbf{a}_j)$. In the i-th phase, we compute R as follows,

$$R(\mathbf{a}_i, h_i) = \sum_{j=i}^{i+n} r_{h_i}(\Theta_j, \mathbf{a}_i) + \text{constant}, \tag{5}$$

where the "constant" denotes the historical rewards from the 1-st phase to the (i-1)-th phase. It doesn't influence policy optimization. $R(\mathbf{a}_i, h_i)$ is the long-term reward of a time-invariant local MDP based on the local environment h_i. We use $R(\mathbf{a}_i, h_i)$ as an estimation of the final cumulative reward, following [11]. Because we don't know the total number of phases N during training, we assume there will be n phases in the future. Furthermore, we fix the action $\mathbf{a}_i$ to simplify the training process. $R(\mathbf{a}_i, h_i)$ is a function of $\mathbf{a}_i$ and h_i.

Training policy with Exp3. Exp3 [3] introduces an auxiliary variable $\mathbf{w} = \{w(\mathbf{a})\}_{\mathbf{a}\in\mathbb{A}}$. It is updated as follows. In the 1-st phase, we initialize $\mathbf{w}$ as $\{1, \ldots, 1\}$. In each phase i (i≥1), we update $\mathbf{w}$ for T iterations. In the t-th iteration, we sample an action $\mathbf{a}_t$∼π, apply the action $\mathbf{a}_t$ to the CIL system, and compute $R(\mathbf{a}_t, h_i)$ using Eq. 5. After that, we update $w(\mathbf{a}_t)$ in $\mathbf{w}$ as,

$$w(\mathbf{a}_t) \leftarrow w(\mathbf{a}_t) \exp(\xi R(\mathbf{a}_t, h_i)/p(\mathbf{a}_t|\Theta_i)), \tag{6}$$

where ξ is a constant, which can be regarded as the learning rate. After updating $\mathbf{w}$, we get the policy π=$\mathbf{w}/||\mathbf{w}||$. *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 placebos.

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 superclasses 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

Method	*CIFAR-100*			*ImageNet-100*			*ImageNet-1k*	
	N=5	10	25	5	10	25	5	10
TPCIL [42]	65.34	63.58	–	76.27	74.81	–	64.89	62.88
GeoDL [40]	65.14	65.03	63.12	76.63	75.40	71.43	65.23	64.46
DER [48]	68.65	67.48	66.18	78.40	78.20	75.40	68.13	65.97
ELI [13]	68.78	66.62	64.72	73.54	71.82	70.32	–	–
GD+ext [17]	63.17$_{\pm 0.47}$	58.71$_{\pm 0.39}$	51.79$_{\pm 0.42}$	75.67$_{\pm 0.51}$	72.08$_{\pm 0.61}$	65.13$_{\pm 0.56}$	–	–
MUC-LwF [24]	59.03$_{\pm 0.35}$	53.27$_{\pm 0.47}$	49.06$_{\pm 0.49}$	72.31$_{\pm 0.53}$	68.92$_{\pm 0.60}$	62.93$_{\pm 0.62}$	–	–
POD-AANets [25]	66.12$_{\pm 0.41}$	64.11$_{\pm 0.32}$	62.12$_{\pm 0.51}$	76.63$_{\pm 0.47}$	75.40$_{\pm 0.36}$	71.43$_{\pm 0.32}$	67.60$_{\pm 0.39}$	64.79$_{\pm 0.42}$
w/ PlaceboCIL (ours)	67.65$_{\pm 0.45}$	65.78$_{\pm 0.40}$	64.95$_{\pm 0.46}$	78.24$_{\pm 0.52}$	77.14$_{\pm 0.47}$	**75.85**$_{\pm 0.42}$	68.55$_{\pm 0.34}$	65.49$_{\pm 0.38}$
FOSTER [44]	70.62$_{\pm 0.58}$	68.43$_{\pm 0.45}$	63.83$_{\pm 0.62}$	80.21$_{\pm 0.67}$	77.63$_{\pm 0.73}$	69.27$_{\pm 0.50}$	69.32$_{\pm 0.47}$	66.07$_{\pm 0.61}$
w/ PlaceboCIL (ours)	**71.97**$_{\pm 0.49}$	**70.31**$_{\pm 0.59}$	**67.02**$_{\pm 0.65}$	**82.03**$_{\pm 0.49}$	**79.52**$_{\pm 0.60}$	72.79$_{\pm 0.45}$	**71.02**$_{\pm 0.39}$	**68.82**$_{\pm 0.54}$

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.

No.	Setting	iCaRL		LUCIR-AANets	
		Average	Last	Average	Last
1	Baseline	57.12	47.49	66.72	57.77
2	PlaceboCIL	61.01	51.45	67.16	59.14
3	Overlapping	62.15	52.62	67.48	59.06
4	Non-overlapping	61.52	51.70	67.01	58.53
5	New data	57.70	47.51	66.69	57.33
6	Old data (oracle)	66.64	58.03	68.82	61.52
7	*w/o* Online learning	60.27	50.57	66.91	58.88
8	Offline RL	61.09	50.81	67.31	59.26
9	Higher confidence	60.43	49.36	66.97	58.12
10	Random placebos	56.27	46.64	66.23	57.22

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_i=γ_i=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, $|\mathcal{U}|$ and $|\mathcal{P}|$ are set as $1,000$ and 200, respectively. All experiments of our PlaceboCIL use the "strict budget" setting, i.e., deleting $|\mathcal{U}+\mathcal{P}|$ samples from training data to avoid exceeding the memory budget.

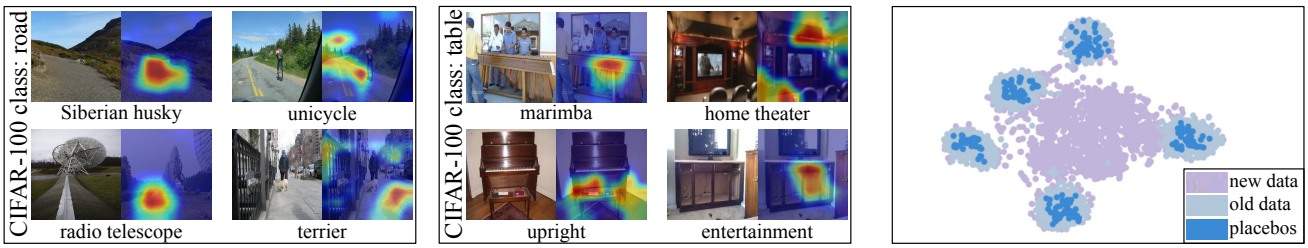
Results on five baselines. Table 1 shows the average and last-phase accuracy for five baselines (i.e., LwF [22], iCaRL [35], LUCIR [12], AANets [25], and FOSTER [44]). From the table, we make the following observations. 1) Using our PlaceboCIL boosts the performance of the baselines clearly and consistently in all settings, indicating that our method is generic and efficient. 2) When the number of exemplars decreases, the improvement brought by our method becomes more significant. For example, the last-phase accuracy improvement of LUCIR increases from 2.52 to 13.88 percentage points when the number of exemplars per class decreases from 20 to 5. This reveals that the superiority of our method is more obvious when the forgetting problem is more serious (with fewer exemplars) due to a tighter memory budget in CIL. 3) Our PlaceboCIL can boost the performance of all KD terms, i.e., not only for logits-based KD [35] but also for feature-based KD [9, 12].

$\lvert\mathcal{U}\rvert$	2000	1000	500	0
Acc. (%)	61.13	61.01	58.23	57.12

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

Comparisons to the state-of-the-art. Table 2 (Blocks 1&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. Intriguingly, 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=25, 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.

Comparisons to the CIL methods using unlabeled data. Table 2 (Blocks 2&3) shows the results of our best model and CIL methods using unlabeled data (GD+ext [17] and MUC-LwF [24]). We can see that our method consistently performs better than others. For another related work,



(a) Selected placebos (b) t-SNE visualization

Figure 3. (a) Selected placebos for two CIFAR-100 classes and their GradCAM activation maps. The free image stream is non-matching ImageNet-1k. (b) The t-SNE results on CIFAR-100 (N=5). For clear visualization, we randomly pick five new classes and five old classes. The **purple**, **light blue**, and **dark blue** points denote the new data, old data, and selected placebos, respectively.

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 achieves comparable results to the upper bound.

3) Third block: different policy learning methods. Row 7 is for using fixed evaluation functions (β_i=γ_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 $\mathcal{U}$. 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 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

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *CVPR*, pages 3931–3940, 2020. 3

[2] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, volume 23, pages 39.1–39.26, 2012. 3, 5

[3] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002. 3, 5, 6

[4] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010. 3

[5] Jihwan Bang, Heesu Kim, Youngjoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, pages 8218–8227, 2021. 3

[6] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, pages 241–257, 2018. 1

[7] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Dual-teacher class-incremental learning with data-free generative replay. In *CVPR*, pages 3543–3552, 2021. 3

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[9] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102, 2020. 1, 2, 3, 6, 7

[10] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Experts in a markov decision process. In *NeurIPS*, pages 401–408, 2005. 3, 5

[11] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009. 3, 5, 6

[12] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 1, 2, 3, 6, 7

[13] KJ Joseph, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Vineeth N Balasubramanian. Energy-based latent aligner for incremental learning. In *CVPR*, pages 7452–7461, 2022. 1, 3, 7

[14] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*, pages 16071–16080, 2022. 1

[15] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017. 1

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[17] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *ICCV*, pages 312–321, 2019. 1, 2, 3, 7, 8

[18] Yingying Li, Subhro Das, and Na Li. Online optimal control with affine constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8527–8537, 2021. 3

[19] Yingying Li and Na Li. Online learning for markov decision processes in nonstationary environments: A dynamic regret analysis. In *2019 American Control Conference (ACC)*, pages 1232–1237. IEEE, 2019. 3

[20] Yingying Li, James A Preiss, Na Li, Yiheng Lin, Adam Wierman, and Jeff S Shamma. Online switching control with stability and regret guarantees. In *Learning for Dynamics and Control Conference*, pages 1138–1151. PMLR, 2023. 3

[21] Yingying Li, Aoxiao Zhong, Guannan Qu, and Na Li. Online markov decision processes with time-varying transition probabilities and rewards. In *ICML workshop on Real-world Sequential Decision Making*, 2019. 3

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, pages 614–629, 2016. 3, 6, 7

[23] Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Online hyperparameter optimization for class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8906–8913, 2023. 1, 3, 6

[24] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *ECCV*, pages 699–716, 2020. 1, 3, 7

[25] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *CVPR*, pages 2544–2553, 2021. 1, 2, 6, 7

[26] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. In *NeurIPS*, pages 3478–3490, 2021. 6, 8

[27] Yaoyao Liu, Bernt Schiele, Andrea Vedaldi, and Christian Rupprecht. Continual detection transformer for incremental object detection. In *CVPR*, pages 23799–23808, 2023. 1

[28] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, pages 12245–12254, 2020. 1, 3

[29] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *CVPR*, pages 11371–11380, 2023. 1, 3

[30] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[31] K. McRae and P. Hetherington. Catastrophic interference is eliminated in pre-trained networks. In *CogSci*, 1993. 1

[32] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *CVPR*, pages 3513–3522, 2021. 1

[33] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, pages 524–540, 2020. 3

[34] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97:285–308, 1990. 1

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542, 2017. 1, 2, 3, 5, 6, 7

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[37] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv*, 1606.04671, 2016. 3

[38] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *CVPR*, pages 618–626, 2017. 2

[39] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, pages 2990–2999, 2017. 3

[40] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *CVPR*, pages 1591–1600, 2021. 3, 7

[41] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087, 2017. 4

[42] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *ECCV*, pages 254–270, 2020. 3, 7

[43] Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Yatao Bian, Han-Jia Ye, De-Chuan Zhan, and Peilin Zhao. 3ef: Class-incremental learning via efficient energy-based expansion and fusion. In *ICLR*, 2023. 1

[44] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 2022. 2, 3, 6, 7

[45] Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *NeurIPS*, 2018. 3

[46] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *NeurIPS*, pages 899–908, 2018. 3

[47] Shipeng Yan, Lanqing Hong, Hang Xu, Jianhua Han, Tinne Tuytelaars, Zhenguo Li, and Xuming He. Generative negative text replay for continual vision-language pretraining. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *ECCV*, pages 22–38, 2022. 3

[48] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 3, 7

[49] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, pages 6982–6991, 2020. 3

[50] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *CVPR*, pages 12455–12464, 2021. 3

[51] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *WACV*, pages 1131–1140, 2020. 2, 3, 8

[52] Yixiao Zhang, Xinyi Li, Huimiao Chen, Alan L. Yuille, Yaoyao Liu, and Zongwei Zhou. Continual learning for abdominal multi-organ and tumor segmentation. In *MICCAI*, volume 14221 of *Lecture Notes in Computer Science*, pages 35–45, 2023. 1

[53] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020. 1

[54] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023. 1

[55] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *NeurIPS*, pages 14306–14318, 2021. 1, 3