Adversarial Pseudo-replay for Exemplar-free Class-incremental Learning

Hiroto Honda
Independent Researcher

Abstract

Exemplar-free class-incremental learning (EFCIL) aims to retain old knowledge acquired in the previous task while learning new classes, without storing the previous images due to storage constraints or privacy concerns. In EFCIL, the plasticity-stability dilemma, learning new tasks versus catastrophic forgetting, is a significant challenge, primarily due to the unavailability of images from earlier tasks. In this paper, we introduce adversarial pseudo-replay (APR), a method that perturbs the images of the new task with adversarial attack, to synthesize the pseudo-replay images online without storing any replay samples. During the new task training, the adversarial attack is conducted on the new task images with augmented old class mean prototypes as targets, and the resulting images are used for knowledge distillation to prevent semantic drift. Moreover, we calibrate the covariance matrices to compensate for the semantic drift after each task, by learning a transfer matrix on the pseudo-replay samples. Our method reconciles stability and plasticity, achieving state-of-the-art on challenging cold-start settings of the standard EFCIL benchmarks. Code is available at `https://github.com/hirotomusiker/APR-EFCIL`.

1. Introduction

Recent rapid advances in deep learning rely on a one-time training using a static dataset. However, in dynamic environments, data often arrive in a non-stationary stream format. The ability to dynamically accumulate new knowledge is referred to as continual (incremental, life-long) learning [18, 24]. Class-incremental learning (CIL) [27] addresses the setting where a group of new class data is only available in the current task and the data from the previously seen classes are not accessible. The knowledge acquired in the previous tasks is overwritten by the new information, a phenomenon known as catastrophic forgetting [3]. Exemplar-free (Non-exemplar) class-incremental learning methods [6, 12, 13, 28, 29, 31] tackle the issue by storing old class prototypes — usually class-wise mean features — to maintain knowledge without storing raw images (exemplars) in order to avoid privacy and storage issues. However, even with the use of prototypes, the plasticity-stability dilemma — trade-off between acquiring knowledge from new tasks (plasticity) and avoiding catastrophic forgetting (stability) — remains unresolved.

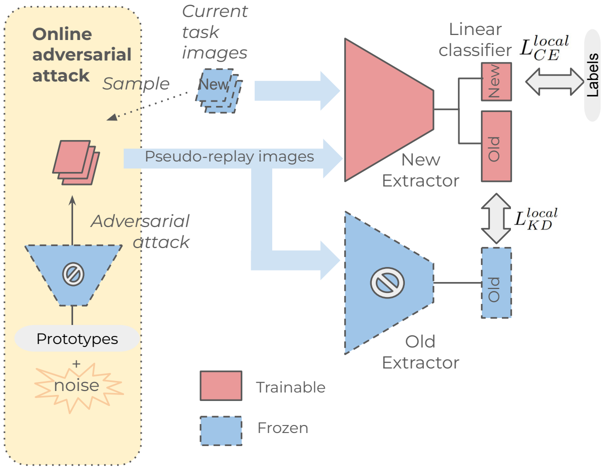

Figure 1. Adversarial Pseudo Replay. The images from the new task are transformed into old-task data via adversarial attack in an *online* manner. Local (logits-based) knowledge distillation using the pseudo-replay images and preserved (old) network prevents the new extractor from semantic drift.

The major cause of catastrophic forgetting is the change in the feature extractor, also known as *semantic drift* [26] occurring when learning a new task. This leads to inconsistency between the preserved prototypes and features of past tasks extracted by the updated extractor. There are two types of approaches for semantic drift: (a) stabilization of the feature extractor and (b) calibrating the prototypes.

LwF [13] addresses (a) by introducing knowledge distillation to ensure the network produces similar logits as the preserved old task network on the new task images. FeCAM [6] also addresses (a) by freezing the feature extractor and employing a network-free Mahalanobis classifier. While the method shows strong stability, it lacks the ability to learn new tasks (plasticity) and performs poorly under the cold-start (or small-start) settings [7, 14] where only a limited amount of data is available at the initial task. We also tackle (a) with knowledge distillation in a more effective way than [13], by transforming new task images into pseudo-replay samples.

Approaches addressing (b) include prototype calibration methods such as [12, 21, 26], which transfer prototypes or covariance matrices into the updated feature space. The gap between old and new feature spaces is typically estimated using the new task data and preserved feature extractor. ADC [7] generates synthetic samples through adversarial attack on the new data, for more accurate drift estimation.

In this paper, we address (a) by generating pseudo-replay samples via adversarial attack. These samples offer a more effective basis for knowledge distillation than the new task samples alone, in order to prevent the extractor's drift from the old task representation space. Synthetic data can be generated by adversarial attack [7] and stored at the beginning of the new task, or generated via an external generator network that synthesizes old task data during the new task training [15]. However, both approaches require additional storage space and are thus incompatible with the constraints of exemplar-free CIL. To address this, we propose generating pseudo-replay images in an *online* manner, by conducting the adversarial attack during the new task training. Before the new task training, we sample the *indices* of new task images and augmentation policies as candidate data, whose features are close to the prototype of each class. The recorded augmentation is applied deterministically throughout the new task training. The online adversarial attack is conducted with old-class target prototypes as targets. Inspired by PASS [29], we add Gaussian noise to the target prototypes to increase diversity of the resulting pseudo-replay samples. Both new-task and pseudo-replay images are passed through the current and old feature extractors, and knowledge distillation [13, 22] is performed to minimize the gap of the features. As a result, our APR paradigm effectively mitigates the extractor's semantic drift without requiring additional storage space.

We also tackle (b) by calibrating the prototypes and covariance matrices after each task using adversarially generated samples, inspired by [7, 21]. A transfer matrix trained for each class using the adversarially generated samples calibrates a covariance matrix via simple matrix multiplication.

Finally, we address an overlooked issue: non-negligible amount of storage required for covariance matrices. We demonstrate that low-rank approximation via singular value decomposition can reduce this overhead without degrading performance.

We summarize our contributions as follows:

- To the best of our knowledge, this is the first work to introduce the adversarial pseudo-replay, which perturbs new-task samples with adversarial attacks to synthesize the pseudo-replay images *online* to mitigate semantic drift, without storing any replay samples.
- We further leverage adversarially perturbed samples to calibrate the covariance matrices after each task, by learning transfer matrices to boost Mahalanobis distance-based EFCIL performance.
- Our proposed EFCIL pipeline, APR, achieves state-of-the-art results in challenging cold-start settings of CIFAR100, TinyImageNet and ImageNet-Subset, and competitive or best performance on warm-start CIFAR100 and TinyImageNet.

2. Related Work

To date, a plethora of class-incremental learning methods has been proposed under various data availability scenarios. In this paper, we focus on exemplar-free class-incremental learning (EFCIL) where only class mean features (prototypes) and an old model preserved in the previous task are available during the new task.

2.1. Semantic Drift Mitigation Methods

LwF [13] leverages knowledge distillation between new-model and old-model logits to mitigate semantic drift of the updated network. PASS [29] leverages multi-view samples generated by image rotation to learn robust representations in a self-supervised manner. Feature-level knowledge distillation is adopted to prevent semantic drift, and the linear classifier is stabilized with the Gaussian-augmented prototypes. The method is further improved by PASS++ [30] with hardness-aware prototype augmentation. IL2A [28] generates old features using covariance matrices to cope with the mismatch between the updated extractor and classifier.

Some approaches favor stability and freeze the feature extractor to avoid semantic drift. SSRE [31] proposes self-sustaining expansion and prototype selection strategy. SEED [20] employs the mixture-of-experts paradigm and selects the optimal expert based on the class distributions. FeTrIL [16] and FeCAM [6] show excellent performance by leveraging Mahalanobis distance with covariance matrices. These methods do not require incremental training of the feature extractor; however, they face challenges in assimilating new knowledge, especially in cold-start (or small-start) settings [7, 14] where the first task contains only a limited amount of data. On the other hand, we allow the feature extractor to be updated during the incremental tasks while maintaining the stability by preventing semantic drift with pseudo-replay.

As the feature extractor is updated in the new task, the preserved prototypes become incompatible with the new feature space. The incompatibility is mitigated by computing the feature space gap using the new task samples [26]. ADC [7] further perturbs the new task images by adversarial attacks to calculate the gap more accurately. FCS [12] introduces a transfer network to calibrate the prototype features for new feature spaces. AdaGauss [21] addresses dimensional collapse by introducing low-rank projection and anti-collapse loss, and calibrates the covariances with an adapter network after each task to compensate for semantic drift.

2.2. Generation-based Methods

Synthesizing the past image data (pseudo replay) is effective in mitigating semantic drift. For instance, DiffClass [15] fine-tunes diffusion models to synthesize previous-task data. However, a powerful generator requires storage and may remember (leak) the sensitive data [4]. In contrast, data-free methods [4, 17, 22] synthesize old-class samples using model inversion [25] to avoid the above issue. Adversarial attack on new-task images [7] or replay samples [10] is also effective to synthesize past-task data. Our method, APR, also leverages adversarial perturbation on the new task images, but in an online manner during the new task training, to mitigate semantic drift of the feature extractor without storing or fine-tuning exemplars or an external generator network.

3. Method

3.1. Adversarial Pseudo Replay : Overview

The training paradigm of our adversarial pseudo-replay is shown in Fig. 1 and Algorithm 1. At the initial task $t = 0$, the dataset D_0 that consists of the images belonging to the classes $c \in C_0$ is available. The feature extractor f^0 and the linear classifier g^0 are trained on D_0 with cross-entropy loss L_{CE}. The prototype μ_c with dimension of d and covariance Σ_c whose size is (d, d) are calculated for each class using the trained extractor f^0 and D_0.

At the subsequent task $t \in 1, 2, 3, ...T-1$, the new task dataset D_t for the new class group C_t is available and the old data are discarded. In EFCIL setting, the extractor f^{t-1} is preserved along with μ_c and Σ_c ($c \in C_{0:t-1}$). During the new task, the new data from D_t are perturbed by adversarial attack (Sec. 3.2, 3.3) with μ_c as a target in each iteration, to obtain the pseudo-replay samples of the previous classes $C_{0:t-1}$. f^t and g^t are trained with local cross-entropy loss L_{CE}^{local} and local knowledge distillation loss L_{KD}^{local} (Sec. 3.4) to reconcile new knowledge acquisition (plasticity) and semantic drift mitigation (stability).

After training on D_t, the prototypes μ_c and covariance matrices Σ_c of the old classes $c \in (C_0, ...C_{t-1})$ are calibrated to compensate for semantic drift of the extractor. The calibration is conducted with the adversarially perturbed images to accurately measure the difference between the features extracted by f_t and f_{t-1}. The transfer matrix W is trained as a single layer perceptron so that the old covariance matrices are calibrated to the new feature space with a simple matrix multiplication (Sec. 3.5).

Algorithm 1 *APR*: Adversarial Pseudo Replay

```
Initialize: Training data $(D_0, D_1, \ldots, D_{T-1})$ with class sets $C_t$ for each $D_t$, feature extractor $f^0$, linear classifier $g^0$
Train $f^0$ and $g^0$ on $D_0$ with $L_{CE}$
Calculate prototype $\mu_c$ and covariance $\Sigma_c$ for $c \in C_0$
for $t = 1, 2, 3, \ldots T-1$ do
    for $c \in (C_0, ...C_{t-1})$ do
        Sample $I_c \subseteq \{1, \ldots, |D_t|\}$ with $f^{t-1}$ and $\mu_c$
        Record policy $P_c$                                  ▷ (Sec. 3.2)
    end for
    Form $D_{APR}$ with $D_t$ and $\{I_c, P_c\}_{c \in (C_0, \ldots C_{t-1})}$
    for epoch $e = 1$ to $E$ do
        for batch $B_t \subset D_t$ and $B_{APR} \subset D_{APR}$ do
            Perturb $B_{APR} \rightarrow B^{\dagger}_{APR}$   ▷ APR (Sec. 3.3)
            Train $f^t$ and $g^t$ on $B_t, B^{\dagger}_{APR}$ with Eq. 7
        end for
    end for
    for $c \in (C_0, ...C_{t-1})$ do
        Obtain $D^{\dagger}_c$ with ADC
        Train $W$ with $f^{t-1}$, $f^t$ and $D^{\dagger}_c$        ▷ (Sec. 3.5)
        Calibrate $\mu_c$ and $\Sigma_c$                      ▷ (Eq. 8)
    end for
    Calculate $\mu_c$ and $\Sigma_c$ for $c \in C_t$
end for
```

At test time, three types of classifiers are available; i) linear classifier g^t, ii) nearest class mean (NCM) classifier using μ_c and iii) Mahalanobis classifier using μ_c and Σ_c .

3.2. Candidate Sampling

At the beginning of the task t, the new task data D_t, old network f^{t-1}, prototypes μ_c^{t-1} and Σ_c^{t-1} are available. We wish to synthesize the old task data with adversarial perturbation [7], without relying on an external generator network [15], and use the generated pseudo-replay data for knowledge distillation [22]. One option to achieve the scheme is to prepare pseudo-replay images before the main training loop of the task T , and keep them throughout the task to be loaded along with the new task dataset D_t. However, since one of the premises of EFCIL is limited storage space, keeping all the pseudo-replay images (e.g. $224 \times 224 \times 3$ for ImageNet-Subset) during the task is not acceptable.

Therefore, in APR we only keep *indices* and *augmentation parameters* of the candidate samples from the new task dataset D_t before the task. More specifically, for every old classes $c \in (C_0, ...C_{t-1})$, we load and apply augmentations on the images from D_t, extract the features with the old extractor f^{t-1} and calculate the Euclidean distance between the features and the prototype μ_c:

$$d_i = \|f^{t-1}(P_i(x_i)) - \mu_c\|_2, x_i \subseteq D_t, \tag{1}$$

where P_i is the augmentation policy with the parameters that are randomly set for each sample. The parameters include random crop coordinates, horizontal flip flag and auto-augment [1] policy parameters. The typical number of the policy parameters is less than 30, which we consider satisfies the EFCIL's storage limitation demand. More details can be found in the Supplementary Material.

Then we pick k indices and augmentation policy parameters of the samples from the smallest distance:

$$I_c = \text{argsort}(d)_{[:k]}. \tag{2}$$

We gather I_c, $P_c = \{P_i\}_{i \in I_c}$ and the pseudo-labels $Y_c = \{c\}_{i \in I_c}$ for the all old classes $c \in (C_0, ...C_{t-1})$ and form a pseudo-replay candidate dataset D_{APR} that reproduces the candidate images without storing actual image data.

3.3. Adversarial Attack

In each training iteration, a batch of data $B_t \subset D_t$ for task-t and another batch of pseudo-replay candidate data $B_{APR} \subset D_{APR}$ are loaded and augmented. Both B_t and B_{APR} consist of the new task images but the latter is loaded using the candidate indices I_c and deterministic augmentation policy P_c.

Subsequently, the images $x \in B_{APR}$ are perturbed into $x_{adv} \in B^{\dagger}_{APR}$ with adversarial attack. Following [7], we move the feature of x towards the corresponding prototype μ_c. For each $x \in B_{APR}$, the feature is extracted with the frozen old task network f_{t-1}. The loss function L calculates the distance between the feature and the prototype μ_c, which is back-propagated through f_{t-1} and the input image x is updated:

$$x_{adv} \leftarrow x - \alpha \frac{\nabla_x L(f_{t-1}(x), \mu_c)}{\|\nabla_x L(f_{t-1}(x), \mu_c)\|_2^2}, \tag{3}$$

where α is the attack magnitude. The attack is repeated N_{attack} times. We do not ensure the similarity between x_{adv} and x [5] or clip pixel values after the attack [7]. To encourage generalization of training, Gaussian noise $r\mathcal{N}(0,1)$ is applied to μ_c for every perturbation target. Similar to prototype augmentation [29] , the magnitude r is calculated from trace of covariances:

$$r = \sqrt{\sum_c \frac{\text{Tr}(\Sigma_c^{t-1})}{d}}, \tag{4}$$

where Σ_c^{t-1} is the covariance matrix calibrated before task t (Sec. 3.5) and d is the feature dimension. The resulting images $x_{adv} \in B^{\dagger}_{APR}$ are directly used for the following knowledge distillation without postprocessing.

3.4. Knowledge Distillation with Pseudo-Replay

We adopt the knowledge distillation paradigm proposed in [13], that imposes the local cross-entropy loss L_{CE}^{local} and local knowledge distillation loss L_{KD}^{local} on the batches of B_t and $B^{\dagger}_{APR}$.

$$L_{CE}^{local} = \text{CE}(g_n^t(f^t(x)), y^{\dagger}), x \in B_t, \tag{5}$$

$$L_{KD}^{local} = \text{KD}(g_o^t(f^t(x)), g_o^{t-1}(f^{t-1}(x))), x \in \{B_t, B^{\dagger}_{APR}\}, \tag{6}$$

where CE is the cross-entropy loss, KD is the knowledge distillation loss [13], g_n and g_o are the split linear classifiers corresponding to new-task and old-task classes, and $y^{\dagger}$ is the relative class index in the current task classes.

In summary:

- L_{CE}^{local} : applied on the new-class logits, using only new-task data,
- L_{KD}^{local} : applied on the old-class logits, using new-task data and pseudo-replay data.

The learning target at task $t(>0)$ is summarized below:

$$\mathcal{L} = \mathcal{L}_{CE}^{local} + \lambda_{kd}\mathcal{L}_{KD}^{local}, \tag{7}$$

where λ_{kd} is the loss weight parameter.

3.5. Prototype and Covariance Calibration

The prototypes calculated in the previous task are calibrated to mitigate the effect of semantic drift caused by the update of f^t. We leverage perturbed samples $D_c^{\dagger}$ corresponding to each class for calibration. To transfer the covariance matrix to the updated representation space, the (d,d) transfer matrix W is trained for each class. More specifically, the samples $x_c^{\dagger}$ from the dataset $D_c^{\dagger}$ are fed to the old and new feature extractors f^{t-1} and f^t, and W is trained so that the gap between $f^t(x_c^{\dagger})$ and $Wf^{t-1}(x_c^{\dagger})$ is minimized. The prototype μ_c and covariance Σ_c are calibrated as follows:

$$\mu_c^t = \mu_c^{t-1} + \Delta_c, \;\; \Sigma_c^t = W\Sigma_c^{t-1}W^T, \tag{8}$$

where Δ_c stands for mean feature differences between $f^t(x_c^{\dagger})$ and $f^{t-1}(x_c^{\dagger})$. This covariance calibration is simpler and faster than the existing work [21] which applies the transform on the data sampled from the multivariate Gaussian $(\mathcal{N}(\mu_c^{t-1}, \Sigma_c^{t-1}))$ to calculate Σ_c^t.

At test time, Mahalanobis distance between the test data feature and the multivariate distribution $\mathcal{N}(\mu_c^t, \Sigma_c^t)$ is calculated. Following [6], we shrink and normalize the covariance matrix before distance calculation:

$$\Sigma_s = \Sigma + \gamma_1 V_1 I + \gamma_2 V_2 I, \tag{9}$$

$$\Sigma^*(i,j) = \frac{\Sigma_s(i,j)}{\sqrt{\Sigma_s(i,i)\Sigma_s(j,j)}}, \tag{10}$$

where V_1 and V_2 are the average on-diagonal and off-diagonal variances of Σ respectively.

3.6. Covariance Decomposition

The covariance matrices have rich information from the past classes but require more storage than prototypes. The size of a covariance matrix (d, d) corresponds to d prototypes. [21] suggests that the feature representation incurs dimensional collapse throughout the tasks. The sparsity of the features indicates a possibility to compress the covariances into storage-efficient data. We leverage singular value decomposition (SVD) for obtaining the rank-k $(< d)$ representation;

$$\Sigma_k = U_k S_k V_k. \tag{11}$$

The decomposed matrices U_k, S_k, V_k have the total size of $2kd + k^2$, which is considerably smaller than the full covariance matrix.

4. Experiment

4.1. Benchmark Setup

Datasets and CIL Settings. The benchmarks are conducted on the following three datasets: CIFAR100 [9] contains 100 classes, each with 500 (train) and 100 (test) images of (32, 32) resolution. TinyImageNet [11] consists of 200 classes each of which has 500 train and 50 test images of (64, 64) resolution. ImageNet-Subset contains 100 classes [23] selected from ImageNet-1k [2], each of which has approx. 1300 train and 50 test images of size (224, 224). In typical T-task EFCIL, the model is trained on half of the total classes at the initial task, and $1/(T-1)$ of the rest are incremented at the following $T-1$ tasks. This setting is referred to as **warm-start setting**. In contrast, the more challenging and practical **cold-start setting** was introduced in [7], where only $1/T$ of the classes are available at the initial task. The stability-favoring methods such as [6, 16] perform well under warm-start settings but struggle in cold-start scenarios due to the limited knowledge acquired at the initial task. We aim to build a method that performs well in both settings.

Evaluation Metric. We employ *average incremental accuracy* (A_{inc}) and *final accuracy* (A_{last}) for evaluation. A_{inc} and A_{last} are standard metrics [6, 7, 19] to evaluate overall accuracy trends and final performance. The accuracy up to task t (A_t) and A_{inc} are defined as:

$$A_t = \frac{1}{t+1}\sum_{j=0}^{t} a_{t,j}\ ,\ \ A_{inc} = \frac{1}{T}\sum_{t=0}^{T-1} A_t, \tag{12}$$

where $a_{t,j}$ stands for accuracy of class group j at task t.

4.2. Implementation Details

For fair comparison, we fix the following components that affect the training outcome significantly:

Network: Following EFCIL literature [6, 7], we employ ResNet18 [8] for feature extractor and split cosine classifier. For ImageNet-Subset, we employ the first convolution with stride=1 and MaxPool down-sampling with stride=2 [6], which deals with larger feature maps than the 2-stride convolution counterpart. The output feature size is set to 512-dim, except for 64-dim in AdaGauss following [21].

Augmentation: For CIFAR100, TinyImageNet and ImageNet-Subset, we apply random crop that outputs (32, 32), (32, 32), and (224, 224) images respectively. We do not use a four-view self-supervision setting [29] or class augmentation [28]. In CIFAR100, AutoAugment [1] policies for CIFAR10 are applied after horizontal flipping.

Optimization: In the initial task, the model is trained with SGD, learning rate (lr) = 0.1, and weight decay (wd) = 5e-4 for 200 epochs; in incremental stages with lr = 0.01, wd = 2e-4 for 100 epochs (60 for ImageNet-Subset) both under cosine decay. For warm-start, the incremental lr is 0.001.

CE and KD Loss: We apply local CE loss (eq. 5) to the new-class part and local KD loss (eq. 6) to the old-class part of the logits following [7, 13]. The KD loss weight is set $\lambda_{kd} = 10$ as in [7, 13] for all the settings.

The settings regarding our method are as follows:

APR: The batch size for B_{APR} is set to 64, 64 and 32 for CIFAR100, Tiny-ImageNet and ImageNet-Subset respectively. 200 new task images and augmentation policies per class are sampled for pseudo-replay. For CIFAR100 and ImageNet-Subset, AutoAugment policies for CIFAR10 and ImageNet are applied after horizontal flip. The attack magnitude α is fixed to 64 and number of iteration N_{attack} is set to 4, 6 and 2 for CIFAR100, Tiny-ImageNet and ImageNet-Subset respectively.

Covariance Calibration: After the task, ADC [7] is performed for each class to generate perturbed samples $D^{\dagger}$. We train a transfer matrix W as a (d, d) linear layer with learning rate of 1e-4 for 64 epochs.

Covariance Shrinkage: We apply covariance shrinkage (eq. 9) before evaluation. To avoid overfitting to the test data, the shrinkage parameters γ_1 and γ_2 in eq. 9 are determined by splitting the validation dataset (N=50 per class) from the train dataset.

All the experiments are carried out with a single NVIDIA RTX4070 GPU. Other implementation details are provided in Supplementary Material.

4.3. Benchmark Results

First, we report performance comparison on the cold-start EFCIL setting in Table 1 and Fig. 2. The results of the existing methods above the double horizontal lines are from their papers, and the rest including Joint and APR are evaluated with the averages of experiments across three random seeds. The *Joint* stands for the upper-bound accuracy results, where all the old task classes are available at each task. APR surpasses all the methods with large margins

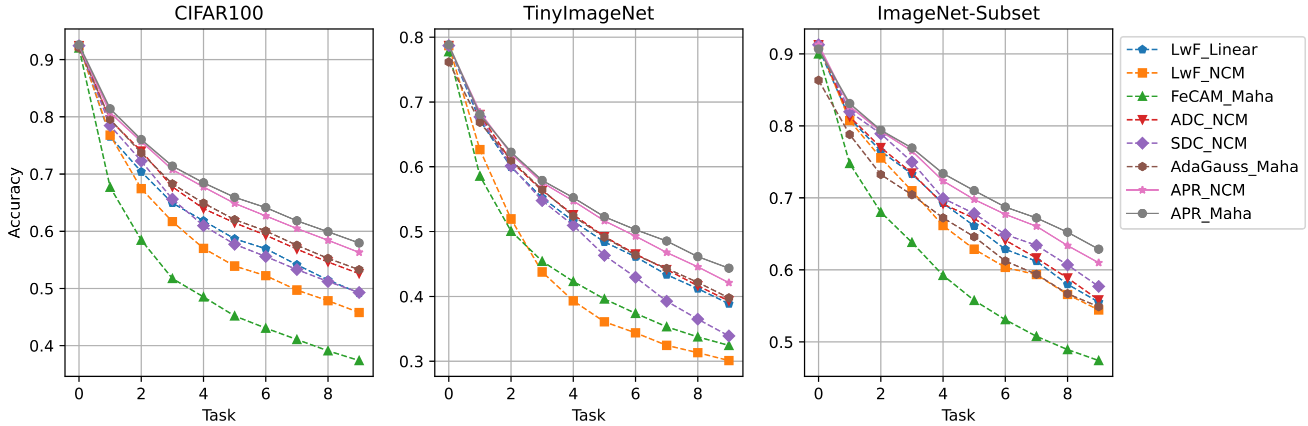

Figure 2. Accuracy transition across all tasks on the cold-start settings. All the results are averaged over three random seeds using our implementation. Best viewed in color.

		CIFAR-100				TinyImageNet				ImageNet-Subset	
		T=5		T=10		T=5		T=10		T=10	
Method	Classifier	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}
SDC [26]	NCM	64.82	54.94	58.02	41.36	50.82	40.05	40.46	27.15	65.83	43.72
ADC [7]	NCM	69.62	59.14	61.35	46.48	50.94	41.00	43.04	32.32	67.07	47.58
AdaGauss [21]	Maha	-	-	60.20	46.10	-	-	50.60	36.50	65.00	51.10
Joint	Linear	83.25	78.69	84.18	78.69	69.84	65.71	70.54	65.71	87.48	85.39
LwF [13]	Linear	71.20	59.39	63.66	49.19	59.67	48.66	53.06	38.87	69.47	55.53
LwF [13]	NCM	67.89	53.96	60.48	45.79	50.30	35.26	44.07	30.10	67.83	54.44
FeCAM [6]	Maha	62.90	48.08	52.61	37.82	53.51	40.95	46.01	33.48	61.02	47.83
SDC [26]	NCM	71.97	60.18	63.68	49.28	58.86	46.85	51.12	33.90	71.16	57.71
ADC [7]	NCM	72.53	61.41	66.24	52.59	59.00	47.69	53.75	39.30	69.98	55.83
AdaGauss [21]	Maha	72.95	61.74	66.66	53.27	58.52	47.20	53.53	39.68	67.96	55.07
APR	Linear	72.58	61.05	67.20	53.18	59.06	47.05	54.70	41.78	70.93	58.81
APR	NCM	*74.39*	*64.39*	*69.00*	*56.29*	*60.14*	*49.55*	*55.58*	*42.11*	*73.00*	*60.99*
APR	Maha	**74.99**	**65.55**	**69.96**	**57.94**	**60.74**	**50.77**	**56.39**	**44.35**	**73.86**	**62.88**

Table 1. Average and final incremental accuracy in *cold-start* EFCIL settings across three benchmarks. Only $1/T$ of the total classes are in the initial task.Experiments below the double lines are the averaged results over three runs with our implementation described in Sec. 4.1. 'Maha' stands for Mahalanobis classifier. The results of [26] are from [7]. For ImageNet-Subset, the first convolution and MaxPool strides are 1 and 2 following [6]. **Best** in bold red, *second best* in italic blue.

in A_{inc}, +3.3% for CIFAR-100, +2.9% for Tiny-ImageNet and +2.7% for ImageNet-Subset under $T = 10$ setting. The improvement in A_{last} is also considerable, +4.7% in CIFAR100, +4.5% in Tiny-ImageNet and +5.2% in ImageNet-Subset ($T = 10$), compared with the best value among existing methods. These results validate APR's ability to mitigate semantic drift while adapting to new tasks flexibly. FeCAM [6] and AdaGauss [21] store covariance matrices for old classes, with the dimension of 512 and 64 respectively. Our APR surpasses the methods even with NCM classifier without covariances (+11.8% over FeCAM and +5.7% over AdaGauss in A_{inc} of ImageNet-Subset).

Second, Table 2 shows the warm-start benchmarks, where half of the classes are available at the initial task. APR is competitive or the best on most settings. In CIFAR100, APR with Mahalanobis classifier shows the state-of-the-art performance in both $T = 6$ and $T = 11$ settings, even surpassing FeCAM that freezes the feature extractor during the incremental stages. In Tiny-ImageNet, APR is competitive or best on most settings. FeCAM is slightly better on A_{inc} / A_{last} ($T = 11$) and A_{last} ($T = 6$) than APR (Maha). However, unlike FeCAM APR does not discard the capability of learning the new task.

APR requires more training time (31.0 hours on Imagenet-Subset $T = 10$), compared with FeCAM (2.9h), ADC (13.5h), and AdaGauss (20.1h), due to online adversarial attack and pseudo-replay (see Supp. Material for details). Computational complexity for inference does not increase from ADC or FeCAM, since APR does not modify network architecture or attach additional components.

		$T = 6$		$T = 11$	
Method	Classifier	A_{inc}	A_{last}	A_{inc}	A_{last}
PASS [29]	Linear	63.84	55.67	59.87	49.03
PASS++ [30]	Linear	69.12	59.87	66.50	57.69
DCMI [17]	Linear	67.90	-	66.80	-
SEED [20]	Linear	70.9	-	69.3	-
Joint	Linear	81.43	78.69	81.43	78.69
LwF [13]	Linear	69.74	59.32	65.40	53.10
LwF [13]	NCM	71.82	61.96	67.72	54.75
FeCAM [6]	Maha	71.62	62.39	*71.52*	*62.39*
SDC [26]	NCM	72.77	64.26	69.56	58.43
ADC [7]	NCM	72.74	64.36	70.32	60.27
AdaGauss [21]	Maha	72.38	64.60	70.04	59.76
APR	Linear	71.83	62.90	67.73	56.64
APR	NCM	*73.92*	*66.14*	71.49	62.26
APR	Maha	**74.48**	**67.30**	**72.57**	**63.74**

(a) CIFAR100.

		$T = 6$		$T = 11$	
Method	Classifier	A_{inc}	A_{last}	A_{inc}	A_{last}
PASS [29]	Linear	49.53	41.58	47.15	39.28
PASS++ [30]	Linear	54.13	46.93	53.14	46.66
DCMI [17]	Linear	54.8	-	53.9	-
Joint	Linear	67.92	65.71	67.91	65.89
LwF [13]	Linear	60.61	52.09	55.73	42.02
LwF [13]	NCM	53.10	39.35	45.31	27.70
FeCAM [6]	Maha	*60.77*	**53.49**	**60.59**	**53.49**
SDC [26]	NCM	58.93	49.91	54.35	39.29
ADC [7]	NCM	58.91	50.54	55.18	44.38
AdaGauss [21]	Maha	58.38	50.33	56.08	46.46
APR	Linear	**60.90**	52.98	57.32	45.97
APR	NCM	59.81	52.06	56.67	47.30
APR	Maha	60.66	*53.24*	*57.69*	*48.56*

(b) Tiny-ImageNet.

Table 2. Average and final incremental accuracy in *warm-start* EFCIL in (a) CIFAR100 and (b) Tiny-ImageNet. *Half* of the classes are in the initial task. Experiments below the double lines are the averaged results over three runs with our implementation. Results of [29] are from [30]. **Best** in bold red, *second best* in italic blue.

Calibration	Pseudo Replay	Adv. attack	NCM A_{inc}	NCM A_{last}	Mahalanobis A_{inc}	Mahalanobis A_{last}	Train time
✓			71.48	57.88	72.30	60.47	0.48
✓	✓		71.07	56.59	72.03	58.85	0.66
	✓	✓	70.00	56.92	69.81	56.45	0.93
✓	✓	✓	**73.00**	**60.99**	**73.86**	**62.88**	1.00

Table 3. Ablation study for adversarial pseudo replay and calibration on ImageNet-Subset $T = 10$. First convolution and MaxPool strides are 1 and 2 [6].

4.4. Ablation Study

Adversarial Pseudo Replay. Table 3 compares pseudo-replay configurations on ImageNet-Subset cold-start $T = 10$ setting. All metrics reflect averages over three random seeds and train time is normalized by the full setting. Applying pseudo-replay without adversarial attack (row 2) increases training time but contributes to the notable accuracy gain (+1.9% A_{inc} in NCM and +1.8% in Mahalanobis), highlighting its essential role. Ablating pseudo replay entirely (row 1) shows slightly better performance than the second row (0.4% A_{inc} in NCM and 0.3% in Mahalanobis), showing that pseudo replay without adversarial refinement can be counterproductive. Thus, adversarial attack is the key driver of performance. Table 5 confirms that attack effects stabilize around the chosen perturbation magnitude α and repetition N_{attack}.

Determ. aug.	Prototype noise	NCM A_{inc}	NCM A_{last}	Mahalanobis A_{inc}	Mahalanobis A_{last}
	✓	72.91	61.46	73.65	63.29
✓		72.44	59.81	73.35	61.75
✓	✓	**73.00**	**60.99**	**73.86**	**62.88**

Table 4. Ablation study for deterministic augmentation and target prototype noise on ImageNet-Subset $T = 10$ setting. First convolution and MaxPool strides are 1 and 2 [6].

α	A_{inc}	A_{last}
8	68.88	55.35
16	69.52	57.04
32	70.00	57.40
64	**69.96**	**57.94**

loops	A_{inc}	A_{last}	Time (h)
1	69.56	56.74	3.9
2	70.01	57.51	4.3
4	**69.96**	**57.94**	**5.0**
6	69.89	58.05	5.8

Table 5. Sensitivity sweep on CIFAR100 ($T = 10$). Parameters used in Tab. 1 are in bold. 'Time' denotes total benchmark runtime.

SVD k	Size	γ_1, γ_2	A_{inc}	A_{last}
N/A	100%	56	73.37	61.75
64	26.6%	56	73.36	61.73
8	3.1%	96	73.23	61.53
NCM	-	-	72.50	59.91

Table 6. Effect of covariance decomposition on Mahalanobis-based performances on ImageNet-Subset $T = 10$. First convolution and MaxPool strides are 1 and 2 [6].

Component	FeCAM	AdaGauss	APR	$\text{APR}_{\text{SVD}k=8}$
Prototypes	0.18	0.02	0.18	0.18
Covariances	94.32	1.44	94.32	2.97
Candidate inds	0.00	0.00	0.14	0.14
Aug. Params	0.00	0.00	1.08	1.08
Sum	94.50	1.46	95.73	4.38

Table 7. Comparison of memory/storage usage (MB) at $t = 9$ on ImageNet-Subset $T = 10$, assuming 90 old classes, 200 candidates per class and 13k new-task images.

Prototype and covariance calibration. The third row in Table 3 omits prototype and covariance calibration after each task. Without this component, NCM and Mahalanobis classifiers experience significant performance degradation (-3.0% and -4.0%, respectively). In this setting prototypes and covariances remain fixed after they are initially created. Although APR mitigates semantic drift of the extractor, it is still crucial to align the prototypes and covariance matrices to the evolving feature space.

Deterministic augmentation and prototype noise. Table 4 investigates the impact of components for generat-

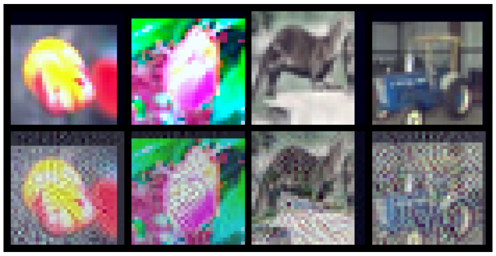

Figure 3. Example of pseudo-replay samples (top: before attack, bottom: after attack) from CIFAR100 [9].

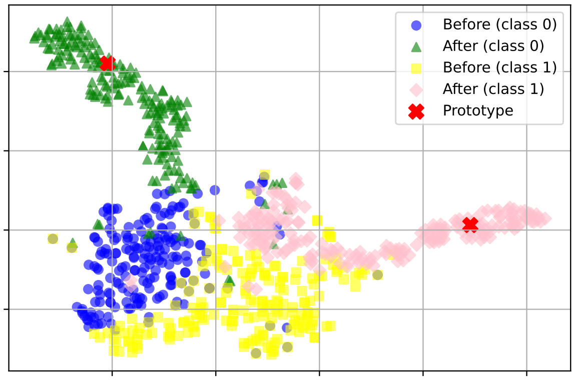

Figure 4. t-SNE analysis of online adversarial attacks during $t = 1$ training on CIFAR100. The features extracted by f^{t-1} and prototypes of two classes are shown. Best viewed in color.

ing pseudo-replay samples. Deterministic augmentation applies the same transforms used for candidate selection; without it (top row), random transforms weaken pseudo-replay, causing a small drop in A_{inc} (0.20%). Prototype noise [29], ablated in the second row, diversifies the pseudo-replay samples and ensures the adversarial attack generalization, similar to data augmentation. Removing it reduces both NCM and Mahalanobis accuracy significantly (0.56% A_{inc} for NCM and 0.51% for Mahalanobis), highlighting its importance for APR.

4.5. Analysis

Online adversarial attack. The effectiveness of adversarial pseudo-replay samples stems from their proximity to the class prototypes in the feature space. Fig. 3 illustrates how online adversarial attacks alter the appearance of input images. The perturbations manifest as visible noise patterns, which guide the extracted features toward the target prototypes. Fig. 4 visualizes feature distributions via t-SNE, and Fig. 5 shows the Euclidean distances to the target prototype before and after the attack. The results confirm that the features shift significantly closer to their respective prototypes after adversarial attacks, indicating the new task images are effectively transformed into pseudo-replay representations. Additional visualizations are in the Supp. Material.

Covariance Decomposition. Table 6 shows the effect of covariance decomposition (Sec. 3.6). The Mahalanobis accuracy does not deteriorate even at $k = 8$ corresponding to size reduction of 97%. As k gets smaller, more covariance shrinkage is necessary to maintain performance. However this result significantly mitigates the often-overlooked storage demand for covariance matrices.

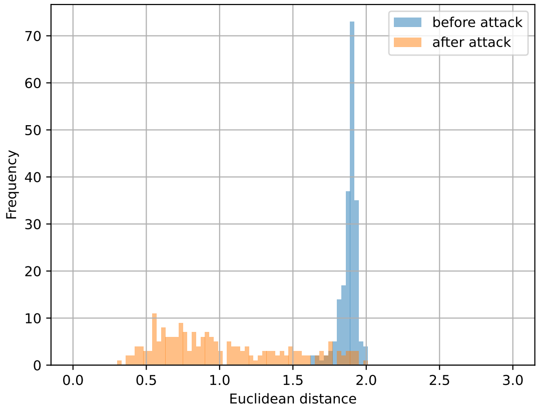

Figure 5. Euclidean distance distributions between target prototype and image features extracted by f^{t-1} before and after attack. Best viewed in color.

Storage Comparison. Required storage for each component at $t = 9$ is summarized in Table 7. Covariance matrices require the largest space but are mitigated with small dimension (AdaGauss) or decomposition (APR$_{\mathrm{SVD}k=8}$). The candidate indices and augmentation parameters are small, enabling storage-efficient pseudo-replay without storing the actual image data (e.g. >10 GB). See Supp. Material for more details.

5. Conclusion

In this paper we presented Adversarial Pseudo Replay (APR), a method designed to mitigate semantic drift problem of exemplar-free class-incremental learning. Pseudo-replay data are synthesized online during the new-task training using an adversarial attack, and are used for local knowledge distillation to mitigate semantic drift of the feature extractor. The effectiveness of APR is further enhanced by augmenting target prototypes during adversarial attack. Mahalanobis classifier shows the best performance with after-task calibration of the covariance matrices, using transfer matrices trained with perturbed samples. Our APR significantly improves the plasticity-stability trade-off in both cold-start and warm-start settings, without the need to store replay samples or an external generative network.

Limitations. APR is storage-efficient, but incurs increased training time due to the use of pseudo-replay data and adversarial attacks. Improving efficiency of adversarial attack can be a promising future research direction for efficient training.

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 4, 5

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[3] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[4] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022. 3

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 4

[6] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 4, 5, 6, 7

[7] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bart Twardowski, Joost van de Weijer, et al. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28525–28534, 2024. 1, 2, 3, 4, 5, 6, 7

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[9] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5, 8

[10] Lilly Kumari, Shengjie Wang, Tianyi Zhou, and Jeff A Bilmes. Retrospective adversarial replay for continual learning. *Advances in neural information processing systems*, 35: 28530–28544, 2022. 3

[11] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 5

[12] Qiwei Li, Yuxin Peng, and Jiahuan Zhou. Fcs: Feature calibration and separation for non-exemplar class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28495–28504, 2024. 1, 2, 3

[13] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 1, 2, 4, 5, 6, 7

[14] Simone Magistri, Tomaso Trinci, Albin Soutif, Joost van de Weijer, and Andrew D. Bagdanov. Elastic feature consolidation for cold start exemplar-free incremental learning. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2

[15] Zichong Meng, Jie Zhang, Changdi Yang, Zheng Zhan, Pu Zhao, and Yanzhi Wang. Diffclass: Diffusion-based class incremental learning. In *European Conference on Computer Vision*, pages 142–159. Springer, 2025. 2, 3

[16] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3911–3920, 2023. 2, 5

[17] Zihuan Qiu, Yi Xu, Fanman Meng, Hongliang Li, Linfeng Xu, and Qingbo Wu. Dual-consistency model inversion for non-exemplar class incremental learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24025–24035. IEEE Computer Society, 2024. 3, 7

[18] Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *IET Computer Vision*, 19(1):e70013, 2025. 1

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 5

[20] Grzegorz Rypeść, Sebastian Cygert, Valeriya Khan, Tomasz Trzciński, Bartosz Zieliński, and Bartłomiej Twardowski. Divide and not forget: Ensemble of selectively trained experts in continual learning. *arXiv preprint arXiv:2401.10191*, 2024. 2, 7

[21] Grzegorz Rypeść, Sebastian Cygert, Tomasz Trzcinski, and Bartłomiej Twardowski. Task-recency bias strikes back: Adapting covariances in exemplar-free class incremental learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3, 4, 5, 6, 7

[22] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9374–9384, 2021. 2, 3

[23] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pages 398–414. Springer, 2022. 5

[24] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1

[25] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8715–8724, 2020. 3

[26] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6982–6991, 2020. 1, 2, 6, 7

[27] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2024. 1

[28] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34:14306–14318, 2021. 1, 2, 5

[29] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021. 1, 2, 4, 5, 7, 8

[30] Fei Zhu, Xu-Yao Zhang, Zhen Cheng, and Cheng-Lin Liu. Pass++: A dual bias reduction framework for non-exemplar class-incremental learning. *arXiv preprint arXiv:2407.14029*, 2024. 2, 7

[31] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9296–9305, 2022. 1, 2