eTag: Class-Incremental Learning via Embedding Distillation and Task-Oriented Generation

Libo Huang[1], **Yan Zeng**[2], **Chuanguang Yang**[1], **Zhulin An**[1*], **Boyu Diao**[1], **Yongjun Xu**[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] School of Mathematics and Statistics, Beijing Technology and Business University
{www.huanglibo, yanazeng013}@gmail.com, {yangchuanguang, anzhulin, diaoboyu2012, xyj}@ict.ac.cn

Abstract

Class incremental learning (CIL) aims to solve the notorious forgetting problem, which refers to the fact that once the network is updated on a new task, its performance on previously-learned tasks degenerates catastrophically. Most successful CIL methods store exemplars (samples of learned tasks) to train a feature extractor incrementally, or store prototypes (features of learned tasks) to estimate the incremental feature distribution. However, the stored exemplars would violate the data privacy concerns, while the fixed prototypes might not reasonably be consistent with the incremental feature distribution, hindering the exploration of real-world CIL applications. In this paper, we propose a data-free CIL method with ***e***mbedding distillation and ***Ta***sk-oriented ***g***eneration (***eTag***), which requires neither exemplar nor prototype. Embedding distillation prevents the feature extractor from forgetting by distilling the outputs from the networks' intermediate blocks. Task-oriented generation enables a lightweight generator to produce dynamic features, fitting the needs of the top incremental classifier. Experimental results confirm that the proposed eTag considerably outperforms state-of-the-art methods on several benchmark datasets.

Introduction

Dynamic scenarios require the deployed model can handle sequential arriving tasks (Parisi et al. 2019; Xu et al. 2021). Models often incur the *catastrophic forgetting* problem in these scenarios, which implies that its performance dramatically degenerates on the previously-learned tasks when the model learns a new task (McCloskey and Cohen 1989).

One of the most general solutions to the forgetting problem is Class-Incremental Learning (CIL) (Aljundi, Chakravarty, and Tuytelaars 2017; Masana et al. 2022). The oracle CIL method, *joint training*, assumes that the data of previously-learned tasks are always available, and combines such data with the current-task one to jointly train the CIL model, whereas this oracle violates the CIL setting where previous data are not allowed (Chen and Liu 2018; Caruana 1997). Early CIL methods intuitively attempted to rebuild the training or inference environment of joint training to achieve comparable performance. For instance[1], deep generative replay (DGR), while training a solver (S), trains a generator (G) to produce past data when needed (Shin et al. 2017; Wu et al. 2018; Ramapuram, Gregorova, and Kalousis 2020), as illustrated in Fig.1(a). However, generative replay is hindered in generating complex images (van de Ven, Siegelmann, and Tolias 2020). Recently, CIL research has shifted from rebuilding the joint training environment to the knowledge distillation (KD) (Hinton et al. 2015). It transfers the probabilistic knowledge between prediction layers from the old frozen network to a current trainable one. The solver network is usually divided into two parts, a feature extractor and a classifier, corresponding to the extractor-aimed and prototype-aimed methods, respectively.

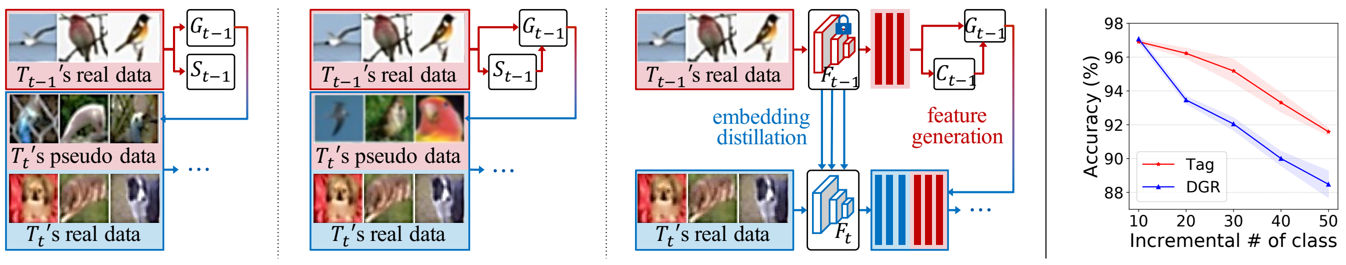
On the one hand, Rebuffi et al. (2017) first introduced extractor-aimed methods that take the cross-entropy criterion to differentiate the learned classes and get a discriminative feature extractor. Instead of the cross-entropy criterion, various metric criteria have been proposed. Yu et al. (2020) enforced that the feature distance between relevant and irrelevant samples is larger than a fixed margin. Cha, Lee, and Shin (2021) constrained the contrastive relations between samples to be constant (Yang et al. 2022). In particular, Lucir established a cosine similarity between the extracted features and the parameters in the classifier (Hou et al. 2019). Benefiting from such a similarity, Lucir obtained a discriminative feature extractor and rectified the inter-class separations when confusion arose in CIL. However, all the above extractor-aimed methods require storing a few exemplars for each learned class (Castro et al. 2018; Douillard et al. 2020).

For data privacy and security concerns, it is generally not allowed to store exemplars of learned tasks (Yu et al. 2020; Liu et al. 2020). Therefore, on the other hand, prototype-aimed methods emerge. They store the feature mean (i.e., prototype) and keep the classifier behind the feature extractor to avoid the mismatch between the stored prototype and the feature extractor. For instance, PASS incrementally expanded the Gaussian distribution dominated by each stored

[*]Zhulin An is the corresponding author.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Methods of rebuilding inferences environment of joint training are as follows. Parameter isolation retains either a well-trained network (Rusu et al. 2016) or a set of parameters (Li and Hoiem 2017) for each learned task. The regularization-based method prevents crucial parameters of the learned tasks from adapting to the current one (Kirkpatrick et al. 2017).



(a) DGR (b) Tag (c) eTag (d) Preliminary results

Figure 1: Comparison of DGR and our proposed Tag and eTag. (a) DGR trains the generator G_{t-1} on the task T_{t-1} to generate the pseudo-samples for learning task T_t. (b) Tag trains G_{t-1} on T_{t-1} under the guidance of the trained and frozen solver S_{t-1}. (c) eTag splits S into a feature extractor F and a classifier C, which incrementally trains F with embedding distillation and trains C with Tag. (d) Preliminary results on 5 permutation MNIST CIL tasks illustrate Tag is superior to DGR.

prototype (Zhu et al. 2021). Liu et al. (2020) argued that prototypes cannot reasonably cover the actual feature distribution of classes. They hence introduced a generator to incrementally produce the learned feature and suggested Generative Feature Replay (GFR) for CIL.

Compared with the classifier that commonly consists of a few layers, the feature extractor is very large, and the conventional KD of the final predictions may not be enough for a suitable extractor. On the contrary, richer knowledge embedded in the intermediate layers is lost during the CIL process. Besides, the purpose of the generative network, i.e., generating real instances, is different from that of the classifier, i.e., classifying instances accurately. Since humans cannot sketch (generate) an actual dollar bill, but can easily spot (classify) a fake bill from the real one, resolving this paradox between reality and discriminability may matter for CIL in a similar way (Epstein 2016). In this paper, we propose a novel CIL method based on a discriminative feature extractor and a well-designed feature generative network. Specifically, we train the feature extractor incrementally in a block-wise embedding distillation manner, which distills the embeddings of the network's intermediate block to maintain more discriminative knowledge. Further, as shown in Fig.1(b), we devise a *Ta*sk-oriented *g*eneration (Tag) strategy. Tag trains the solver network S first, then the generator G guided by the frozen S, enabling a generator to produce proper features that preferably fit the needs of the classifier.

In summary, our contributions are as follows: (1) We propose an embedding distillation model for CIL to transfer richer knowledge embedded in the network's intermediate blocks from the old frozen feature extractor to the current trainable one. To our best knowledge, this is the first study to use embedding distillation with neither exemplars nor prototypes in CIL. (2) We devise a task-oriented generation strategy to make the best of the trained classifier, and empirically validate its merits in improving the performances of CIL tasks. (3) We propose a data-free CIL method with *e*mbedding distillation and *Ta*sk-oriented *g*eneration (eTag as shown in Fig.1(c)). It significantly outperforms SOTA methods on several benchmark datasets. Our code and Supplementary Materials (**SM**) are available[2].

Related Work

Generative Replay in CIL

Generative Sample Replay. The core idea is to additionally train an incremental generator to produce samples of the learned task. Shin et al. (2017) first adopted the generative network to CIL and proposed deep generative replay (DGR). As shown in Fig.1(a), DGR, while training a solver (S) for inference, trains a generator (G) to produce previous pseudo-samples when needed. The generator could be either Generative Adversarial Networks (GAN) (Shin et al. 2017) or Variational Autoencoder (VAE) (Ramapuram, Gregorova, and Kalousis 2020). Instead of replaying the generated data, Wu et al. (2018) implemented a memory replay mechanism by constraining the previous and current GANs with the same input-output pairs on the learned tasks. Huang et al. (2022) developed a similar mechanism by extending VAE's intrinsic sample reconstruction property to knowledge reconstruction. BI-R (van de Ven, Siegelmann, and Tolias 2020) improved the training efficiency using VAE's symmetrical structure. In Fig.1(b), we propose Tag, which makes the best of the solver to guide the training of the generator. Preliminary CIL experiments on the permutation MNIST (van de Ven, Siegelmann, and Tolias 2020) shown in Fig.1(d) verified its benefits compared with DGR.

Generative Feature Replay. To circumvent the limitations of DGR on generating the complex image, generative feature replay (GFR) focuses on estimating the feature distribution rather than the original image distribution. Note that the solver S in GFR is split into a feature extractor F and a classifier C, i.e., $S = F + C$. Xiang et al. (2019) employed a frozen feature extractor that is pre-trained on a large dataset and cannot be updated incrementally. Liu et al. (2020) distilled the final prediction from the stored feature extractor into the current trainable one, while updating the classifier with GFR. As shown in Figs. 1(b) and 1(c), we explore CIL through a task-oriented generation strategy.

[2]https://github.com/libo-huang/eTag

Embedding Distillation

Embedding distillation (Romero et al. 2015) extends the vanilla knowledge distillation (Hinton et al. 2015) from mimicking only the final prediction to the intermediate embeddings between the frozen network (i.e., teacher network) and the trainable one (i.e., student network). The main difficulty is determining what intermediate knowledge must be distilled and how to distill it. Various efforts have been made to address these two issues concurrently. Ahn et al. (2019) reduced the knowledge gaps in intermediate layers between the student and teacher networks by maximizing the mutual information while Romero et al. (2015) aligned the intermediate feature maps using the mean square error. Similarly, Heo et al. (2019) employed the mean square error but for aligning the activation boundaries derived from the hidden neurons. Recent works have introduced comparative information between samples to explore richer intermediate knowledge, e.g., Peng et al. (2019), Tian, Krishnan, and Isola (2019), Tung and Mori (2019), and Park et al. (2019), etc. Beyond that, building additional self-supervised (SS) tasks to train the intermediate layer achieves superior performance (Yang et al. 2021). PodNet (Douillard et al. 2020) confirmed the advantages of considering intermediate feature maps for CIL, but it aligned the features by a strong pixel-to-pixel constriction and stored exemplars. We relax that by embedding knowledge distillation with a self-supervised task to train the feature extractor and achieve a data-free CIL method with task-oriented generation.

Proposed Method

The proposed CIL method, eTag, owns two main modules: a feature generator, G, and a solver, S, where S has a feature extractor, F, and a classifier, C. eTag incrementally trains F and C via embedding distillation (Sec.) and task-oriented generation (Sec.), respectively. Before going into depth, we provide the problem definition and framework overview.

Problem Definition. In CIL, a solver, S, needs to sequentially learn T tasks, where each task is characterized by a dataset, $\mathcal{D}_t = (\mathcal{X}_t, \mathcal{Y}_t) = \{(\boldsymbol{x}_t^i, y_t^i)\}_{i=1}^{n_t}, t = 0, ..., T-1$. Here, $\boldsymbol{x}_t^i \in \mathcal{X}_t$ is the i-th image of label $y_t^i \in \mathcal{Y}_t = \{\mathcal{Y}_t^1, ..., \mathcal{Y}_t^{m_t}\}$, and n_t and m_t are the number of images and the number of classes, respectively. The datasets for different tasks are disjoint, i.e., $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \emptyset$ for all $t \neq t'$. During training S_t on task t, we can access the stored previous solver, S_{t-1}, that trained on task $(t-1)$ and the current-task dataset, $\mathcal{D}_t$. Ultimately, we expect that the solver, S_t, performs well on all previously-learned tasks $t' \leq t$, even if the task ID of each test sample is unavailable.

Framework Overview. As illustrated in Fig.2, we use the modern blocked convolutional network, ResNet-18 (He et al. 2016), as our solver, and VAE as our feature generator[3]. Assuming the feature extractor contains L blocks, we append an auxiliary classifier behind each block (except the final one), yielding $(L-1)$ auxiliary classifiers, $\{C^l(\cdot)\}_{l=1}^{L-1}$. $C^l(\cdot)$ consists of block-wise convolutional blocks, a global average pooling layer, and a linear connected layer (Yang et al. 2021), while the final classifier C consists of a linear connected layer $\boldsymbol{w}$ and a non-linear activation layer σ. For the feature $\boldsymbol{f}$ of a given sample, $C(\boldsymbol{f}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{f}) \in \mathbb{R}^{1 \times m_t}$. We train the feature extractor, F_t, and the classifier, C_t, on task t with embedding distillation and feature generative replay. After that, we freeze F_t and C_t, and train the feature generator, G_t, with guidance from the classifier, C_t, i.e., task-oriented generation, as shown in Fig.2(b).

[3]It is straightforward to generalize the solver to other modern convolutional neural networks, VGG (Simonyan and Zisserman 2015), ViT (Wu et al. 2022), etc. Compared with GAN, VAE has a more stable training mechanism (Huang et al. 2022). Appendix-C.2 in **SM**[2] experimentally confirms the architecture generalization.

Embedding Distillation

Embedding distillation comprises two components: knowledge construction, where the feature extractor learns the current task, and knowledge distillation, which enabling the feature extractor to retain previously-learned tasks.

Embedding Knowledge Construction. For the L-blocked feature extractor, we separate the embedded knowledge into a final output feature, $\boldsymbol{f}^L$, and $(L-1)$ embeddings, $\{\boldsymbol{f}^l\}_{l=1}^{L-1}$.

On the one hand, to enable the feature extractor, F_{t-1}, to extract the discriminative final feature, $\boldsymbol{f}_{t-1}^L$, on task $(t-1)$, we train the solver, $S_{t-1} = F_{t-1} + C_{t-1}$, with the following conventional cross-entropy loss,

$$\mathcal{L}_{CE}^{final}(F_{t-1}, C_{t-1}) = \mathbb{E}_{\mathcal{X}_{t-1}, \mathcal{Y}_{t-1}} \mathcal{L}_{CE}\left(C_{t-1}\left(\boldsymbol{f}_{t-1}^L\right), y\right),$$

where $\mathbb{E}$ stands for the expectation, $\boldsymbol{f}_{t-1}^L = F_{t-1}(\boldsymbol{x})$ is the final feature extracted from task $(t-1)$ data by F_{t-1}.

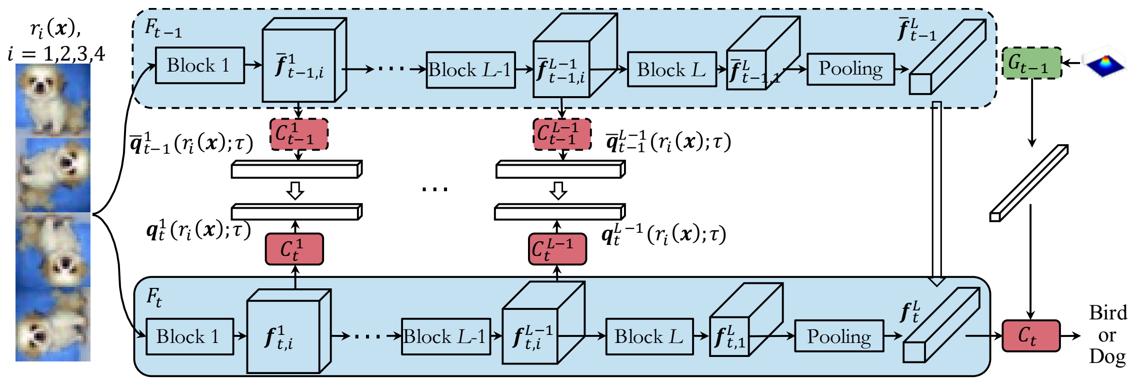
On the other hand, to obtain $(L-1)$ embeddings, $\{\boldsymbol{f}_{t-1}^l\}_{l=1}^{L-1}$, by F_{t-1}, we train the auxiliary classifiers $\{C_{t-1}^l(\cdot)\}_{l=1}^{L-1}$ on the constructed additional self-supervised (SS) task. Specifically, given the image $\boldsymbol{x} \in \mathcal{X}_{t-1}$ and label $y \in \mathcal{Y}_{t-1}$, we employ 4 rotations (i.e., $0°, 90°, 180°, 270°$) on $\boldsymbol{x}$, i.e., $r_i(\boldsymbol{x}), i = 1, 2, 3, 4$, and 4 augmentation labels on y, i.e., y_{r_i}, as our additional SS task, where $r_1(\boldsymbol{x}) = \boldsymbol{x}$, $y_{r_1} = y$. We train the $(L-1)$ auxiliary classifiers with,

$$\mathcal{L}_{CE}^{inter}(F_{t-1}, C_{t-1}^l) = \frac{1}{4} \sum_{i=1}^{4} \sum_{l=1}^{L-1} \mathcal{L}_{CE}\left(C_{t-1}^l(\boldsymbol{f}_{t-1,i}^l),\ y_{r_i}\right),$$

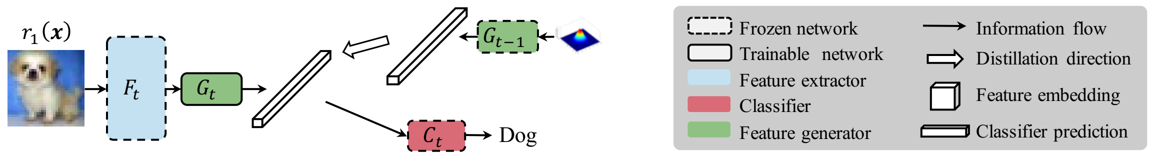
where $\boldsymbol{f}_{t-1,i}^l = F_{t-1}(r_i(\boldsymbol{x}))$ is l-th intermediate embedding of the rotated image, $r_i(\boldsymbol{x})$. $C_{t-1}^l(\boldsymbol{f}_{t-1,i}^l) \in \mathbb{R}^{1 \times 4 \cdot m_{t-1}}$ is the final prediction produced by C_{t-1}^l.

Eventually, $\boldsymbol{f}_{t-1}^L$ and $\{\boldsymbol{f}_{t-1}^l\}_{l=1}^{L-1}$ can embed more knowledge from data $\mathcal{D}_{t-1}$ compared with PodNet's vanilla pixel-to-pixel alignment of $\boldsymbol{f}_{t-1}^L$ or PASS's SS task only. We experimentally verified this point on the CIL's initial task (i.e., Task 0). As shown in Fig.3, PASS uses SS to reduce the number of outlier points compared with PodNet. eTag improves the compaction of feature clusters by employing the proposed embedding knowledge construction, and these feature clusters are kept separate from each other during CIL.

Embedding Knowledge Distillation. When the new task t arrives, we employ the embedding knowledge distillation to retain the knowledge in the frozen feature extractor, F_{t-1}.


$r_i(\boldsymbol{x})$, $i=1,2,3,4$
F_{t-1}
Block 1
Block L-1
Block L
Pooling
$\bar{\boldsymbol{f}}^L_{t-1}$
G_{t-1}
C^1_{t-1}
C^{L-1}_{t-1}
$\bar{\boldsymbol{q}}^1_{t-1}(r_i(\boldsymbol{x});\tau)$
$\bar{\boldsymbol{q}}^{L-1}_{t-1}(r_i(\boldsymbol{x});\tau)$
$\boldsymbol{q}^1_t(r_i(\boldsymbol{x});\tau)$
$\boldsymbol{q}^{L-1}_t(r_i(\boldsymbol{x});\tau)$
C^1_t
C^{L-1}_t
F_t
$\boldsymbol{f}^L_t$
C_t
Bird or Dog

(a) Train $S_t = F_t + C_t$ with embedding distillation



(b) Train G_t with task-oriented generation

Figure 2: Framework of the proposed eTag, including (a) incrementally training the solver $S_t = F_t + C_t$ with the help of the stored feature extractor F_{t-1}, generator G_{t-1}; (b) incrementally training the generator G_t with the guidance from C_t.

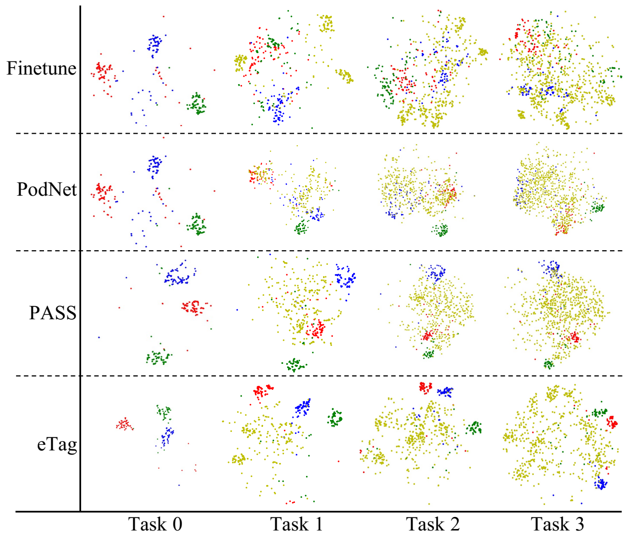


Figure 3: t-SNE features of 4 equally divided CIFAR-100 tasks. Random 3 of 25 classes are provided for a good view.

Firstly, we distill the final feature distribution from F_{t-1} to F_t using $\mathcal{L}_2$ loss (Liu et al. 2020),

$$\mathcal{L}^{final}_2(F_t) = \mathbb{E}_{\mathcal{X}_t}\left[\|\boldsymbol{f}^L_t - \bar{\boldsymbol{f}}^L_{t-1}\|_2\right],$$

where $\bar{\boldsymbol{f}}^L_{t-1} = F_{t-1}(\boldsymbol{x})$ is the final feature extracted from task t data by F_{t-1}. Note that we use a symbol bar to denote the feature of the current task data extracted with the old model. Secondly, we distill $(L-1)$ intermediate embeddings from F_{t-1} to F_t with a block-wise Kullback–Leibler divergence loss,

$$\mathcal{L}^{inter}_{KL}(F_t, C^l_t) \tag{1}$$
$$= \frac{1}{4}\sum_{i=1}^{4}\sum_{l=1}^{L-1}\tau^2\mathcal{L}_{KL}\left(\sigma\left(C^l_{t-1}(\bar{\boldsymbol{f}}^l_{t-1,i})/\tau\right)\middle\|\sigma\left(C^l_t(\boldsymbol{f}^l_{t,i})/\tau\right)\right),$$

where $\bar{\boldsymbol{f}}^l_{t-1,i} = F_{t-1}(r_i(\boldsymbol{x}))$ and $\boldsymbol{f}^l_{t,i} = F_t(r_i(\boldsymbol{x}))$ are the l-th intermediate embedding of the rotated image $r_i(\boldsymbol{x})$ extracted by the frozen F_{t-1} and the trainable F_t, respectively. σ is the softmax operation, τ is a hyper-parameter to scale the smoothness of distribution, and τ^2 is utilized to keep the gradient contributions unchanged (Hinton et al. 2015). At last, we train the solver, $S_t = F_t + C_t$, with $\mathcal{L}^{final}_{CE}$ on $\mathcal{D}_t$, rendering F_t plastic to learn the incremental task, and fit the generated features, $\hat{\boldsymbol{f}}^L_{:t-1}$, to its corresponding label $y \in \mathcal{Y}_{:t-1}$ with $\mathcal{L}_{CE}$, enabling the classifier C_t stably retain the learned knowledge,

$$\mathcal{L}^{new}_{CE}(F_t, C_t) = \mathcal{L}^{final}_{CE}(F_t, C_t) \tag{2}$$
$$+ \lambda_{CE}\cdot\mathbb{E}_{\hat{\boldsymbol{f}}^L_{:t-1},\mathcal{Y}_{:t-1}}\mathcal{L}_{CE}\left(\sigma\left(\boldsymbol{w}_{:t-1}\cdot\hat{\boldsymbol{f}}^L_{:t-1}\right), y\right),$$

where $\hat{\square}$ indicates the generated feature, $\mathcal{Y}_{:t-1}$ represents the label set of $(t-1)$ learned tasks, and λ_{CE} is automatically set to $m_{:t-1}/m_t$, i.e., the ratio of classes in the previously learned task to the current task, for simplifying the training process (Liu et al. 2020).

We validated the superiority of embedding knowledge distillation on the incremental task (i.e., Task $t > 0$). As shown in Fig.3, PodNet aligns the feature maps at the pixel level in the incremental tasks 1-3, which does maintain the feature distribution of the initial task compared with the finetune strategy, but restricts the model from learning the new features of incremental tasks. In contrast, embedding knowledge distillation not only maintains the feature distribution with stability for the initial task but also learns new feature distributions with plasticity.

Task-Oriented Generation

We incrementally train the feature generator G_t to estimate the conditional distribution of features $p(\hat{\boldsymbol{f}}_{:t}^L|y_{:t}) = G_t(y_{:t}, \boldsymbol{z})$, preventing the incremental classifier C_t from forgetting, where $\boldsymbol{z}$ is a Gaussian noise vector, and $y_{:t}$ is the label of a sample from the tasks learned so far.

For the current task t, we specifically propose a task-oriented generation strategy, which allows G_t to produce features that better fit the needs of the classifier, C_t. We realize that by adopting a conditional VAE network with the following modified VAE loss,

$$\begin{aligned}\mathcal{L}_{VAE}^{new}(G_t) = &\mathcal{L}_{KL}\left(p_{\phi_t}\left(\boldsymbol{z} \mid \boldsymbol{f}_t^L, y\right) \| p(\boldsymbol{z})\right) \\ &- \mathbb{E}_{p_{\phi_t}(\boldsymbol{z}|\boldsymbol{f}_t^L, y), \mathcal{X}_t, \mathcal{Y}_t}\left[\log p_{\theta_t}\left(\hat{\boldsymbol{f}}_t^L | y, \boldsymbol{z}\right)\right] \\ &+ \mathbb{E}_{p_{\theta_t}(\hat{\boldsymbol{f}}_t^L|y,\boldsymbol{z}), \mathcal{Y}_t, p(\boldsymbol{z})}\mathcal{L}_{CE}\left(\sigma\left(\boldsymbol{w}_t \cdot \hat{\boldsymbol{f}}_t^L\right), y\right),\end{aligned} \tag{3}$$

where ϕ_t and θ_t indicate the parameters of encoder and decoder in G_t, respectively; and $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$, i.e., the standard Gaussian distribution. By minimizing the first two terms, the encoder promotes $\boldsymbol{f}_t^L$ and y to be the prior Gaussian distribution, while the decoder reconstructs the surrogate $\hat{\boldsymbol{f}}_t^L$ of the original sample $\boldsymbol{f}_t^L$ from y and $\boldsymbol{z}$. The last term encourages the generator G_t to produce specific features such that the classified label of C_t is correct, improving the coherence of the feature from p_{θ_t} with the label y.

To prevent G_t from forgetting, we use the knowledge reconstruction strategy (Huang et al. 2022) that enables G_t to reconstruct the historical knowledge retained in G_{t-1},

$$\begin{aligned}&\mathcal{L}_{VAE}^{old}(G_t) \\ = &-\mathbb{E}_{p_{\theta_{t-1}}(\hat{\boldsymbol{f}}_{:t-1}^L|y,\boldsymbol{z}), U\{\mathcal{Y}_{:t-1}\}, p(\boldsymbol{z})}\left[\log p_{\theta_t}\left(\hat{\boldsymbol{f}}_{:t-1}^L \mid y, \boldsymbol{z}\right)\right],\end{aligned} \tag{4}$$

where $U\{\cdot\}$ is the discrete uniform distribution; and y is uniformly sampled from the labels of learned tasks. As illustrated in Fig.3, compared with PASS, which fixes the prototypes from moving, eTag learns a more plastic feature distribution that appropriately adjusts the old features and accommodates new ones. Notably, Fig.1(d) shows that task-oriented generation exhibits superior incremental results compared with DGR, which uses the generator directly.

Integrated Objective

eTag addresses the forgetting problem in CIL. When learning the initial task 0, there is no previous knowledge to retain. Hence we train the initial generator with $\mathcal{L}_{VAE}^{new}(G_0)$ as Eq.(3), and train the solver with the following unified loss,

$$\mathcal{L}\left(F_0, C_0, C_0^l\right) = \mathcal{L}_{CE}^{final}(F_0, C_0) + \mathcal{L}_{CE}^{inter}\left(F_0, C_0^l\right). \tag{5}$$

When learning the incremental tasks $t \geq 1$, we train the solver and generator incrementally with the weighted losses,

$$\begin{aligned}&\mathcal{L}\left(F_t,\, C_t,\, C_t^l\right) \\ =&\mathcal{L}_{CE}^{new}(F_t,\, C_t) + \lambda \cdot \left(\mathcal{L}_2^{final}(F_t) + \mathcal{L}_{KL}^{inter}\left(F_t, C_t^l\right)\right),\end{aligned} \tag{6}$$

Algorithm 1: CIL with eTag

```
Input: A sequence of $T$ task: $\mathcal{D}_0, ..., \mathcal{D}_t, ..., \mathcal{D}_{T-1}$,
       where $\mathcal{D}_t = (\mathcal{X}_t, \mathcal{Y}_t)$
Output: Incremental solver $S_{T-1} = (F_{T-1}, C_{T-1})$
function eTag($\mathcal{D}_0, ..., \mathcal{D}_{T-1}$)
    Train $F_0$, $C_0$, and $C_0^l$ on $\mathcal{D}_0$ using Eq.(5)
    Train $G_0$ on $\mathcal{D}_0$ using Eq.(3)
    Evaluate $F_0$ and $C_0$ on the initial task $0$
    for $t = 1, ..., T-1$ do
        Train $F_t$, $C_t$, and $C_t^l$ on $\mathcal{D}_t$ using Eq.(6)
        Train $G_t$ on $\mathcal{D}_t$ using Eq.(7)
        Evaluate $F_t$ and $C_t$ on the learned task $1:t$
    end
    return $F_{T-1}$ and $C_{T-1}$
end
```

$$\mathcal{L}(G_t) = \mathcal{L}_{VAE}^{new}(G_t) + \lambda_{VAE} \cdot \mathcal{L}_{VAE}^{old}(G_t), \tag{7}$$

where parameters λ and λ_{VAE} are also automatically set to the same value as λ_{CE}, i.e., $\lambda = \lambda_{VAE} = m_{:t-1}/m_t$. We summarize the whole training process in Algorithm 1.

Experiment

We conduct experiments on CIFAR-100 (Liu et al. 2020) and ImageNet-100 (Wang et al. 2022); and employ three metrics: *average incremental accuracy* (A), *average forgetting measure* (F), and classification *accuracies* (ACC) for evaluation. Please see details of datasets, evaluation metrics, comparisons, and implementation in Appendix-A of **SM**[2].

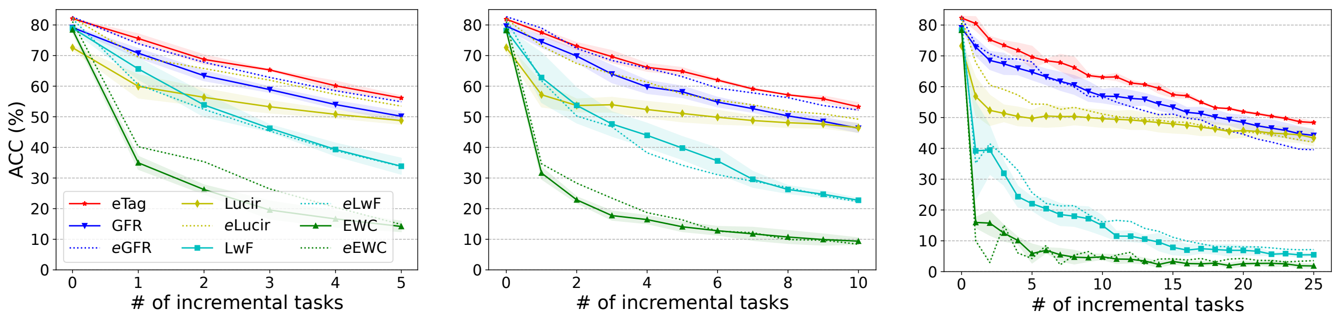
Overall Evaluation

Referring to (Zhu et al. 2021; Liu et al. 2020; Hou et al. 2019), we first train a model on half of the classes of CIFAR-100 or ImageNet-100, and then incrementally train on the remaining classes with equally separated 5, 10 and 25 tasks. We compare two reference methods, fine-tuning (Fine) and joint training (Joint) (Chen and Liu 2018), as well as several state-of-the-art (SOTA) methods, including two parameter isolation methods, LwF (Li and Hoiem 2017) and LwM (Dhar et al. 2019), two regularization-based methods, EWC (Kirkpatrick et al. 2017) and MAS (Aljundi et al. 2018), two extractor-aimed methods, IL2M (Belouadah and Popescu 2019) and Lucir (Hou et al. 2019), and two prototype-aimed methods, GFR (Liu et al. 2020) and PASS (Zhu et al. 2021). In addition, we also modify each of these SOTA methods by introducing embedding distillation, resulting in modified methods denoted by eXX, where XX is LwF, LwM, MAS, IL2M, Lucir, GFR, or PASS.

As shown in Tab.1, eTag outperforms all compared methods in incremental accuracy by a large margin. Specifically, on CIFAR-100 and ImageNet-100, eTag improves average incremental accuracy by more than 5% and 7%, respectively, compared to GFR. It also exhibits less forgetting, showcasing its competitive performance. Interestingly, introducing embedding distillation directly into other SOTA methods does not consistently enhance learning incremental tasks.

Dataset	CIFAR-100						ImageNet-100					
	5 tasks		10 tasks		25 tasks		5 tasks		10 tasks		25 tasks	
Metric	$A(\uparrow)$	$F(\downarrow)$	$A(\uparrow)$	$F(\downarrow)$	$A(\uparrow)$	$F(\downarrow)$	$A(\uparrow)$	$F(\downarrow)$	$A(\uparrow)$	$F(\downarrow)$	$A(\uparrow)$	$F(\downarrow)$
Joint	75.21	3.79	75.26	4.09	75.11(±1.13)	6.60(±1.29)	78.92	3.84	79.53	6.27	80.12(±0.25)	8.74(±0.20)
Fine	22.59	65.49	12.20	72.59	4.66(±0.06)	76.64(±0.73)	19.53	75.20	9.58	79.40	4.60(±0.03)	80.13(±0.62)
EWC	31.66	56.10	21.41	62.51	7.93(±0.56)	73.18(±0.65)	30.53	59.18	21.00	66.84	10.87(±0.12)	73.62(±0.51)
*e*EWC	36.39	53.67	23.21	60.02	7.86(±0.52)	75.67(±1.63)	35.10	59.52	21.50	69.41	10.68(±0.41)	10.68(±0.41)
MAS	25.95	63.78	17.43	67.93	7.87(±1.69)	74.45(±1.06)	29.20	63.39	17.93	70.66	8.13(±0.81)	76.25(±0.38)
*e*MAS	31.64	59.17	20.69	62.43	7.13(±0.97)	76.89(±2.11)	32.53	62.43	19.76	71.11	8.56(±0.66)	78.51(±1.17)
LwF	52.98	31.18	42.27	39.51	16.92(±1.21)	63.89(±1.32)	54.59	33.11	44.60	40.76	26.41(±0.47)	57.61(±0.42)
*e*LwF	52.16	35.77	40.58	44.93	19.45(±0.84)	64.88(±1.83)	47.43	45.35	44.26	44.67	23.37(±1.18)	63.64(±1.41)
LwM	56.95	26.60	44.85	38.08	25.40(±1.36)	55.47(±1.32)	52.90	34.92	42.86	42.44	26.41(±0.39)	57.28(±0.98)
*e*LwM	57.94	28.83	45.86	40.34	24.71(±0.52)	59.39(±1.24)	50.60	41.18	41.74	47.39	25.03(±0.16)	62.08(±0.15)
IL2M	53.83	30.78	53.19	29.00	46.45(±0.39)	33.83(±2.06)	55.89	31.12	56.63	26.46	52.85(±0.59)	29.71(±0.94)
*e*IL2M	58.63	27.88	55.80	31.35	47.66(±1.84)	35.58(±2.87)	54.80	36.39	53.16	35.21	50.17(±1.59)	35.50(±1.75)
Lucir	56.95	<u>18.74</u>	52.87	<u>21.71</u>	49.35(±2.66)	24.83(±1.83)	68.23	**13.69**	66.90	**14.83**	64.83(±0.29)	19.90(±0.82)
*e*Lucir	65.06	20.77	60.60	26.93	51.69(±2.16)	32.00(±3.45)	70.49	18.27	67.61	19.94	61.21(±3.83)	25.05(±3.16)
PASS	61.65	23.42	58.85	27.96	51.75(±1.82)	29.90(±0.99)	69.69	20.46	62.92	26.05	59.16(±0.37)	25.07(±0.50)
*e*PASS	64.51	20.14	63.13	24.15	<u>58.90</u>(±0.49)	23.43(±1.03)	67.64	19.82	64.31	25.93	60.72(±0.69)	24.13(±0.32)
GFR	62.74	19.69	59.85	21.74	56.76(±1.62)	<u>23.21</u>(±1.62)	69.91	18.24	67.30	19.52	63.70(±0.61)	22.23(±0.48)
*e*GFR	<u>66.76</u>	19.03	<u>64.62</u>	23.95	55.20(±0.94)	28.53(±1.58)	<u>76.14</u>	14.76	<u>74.24</u>	15.28	<u>70.33</u>(±0.17)	<u>18.37</u>(±0.22)
eTag	**67.99**	**16.89**	**65.50**	**18.01**	**61.63**(±0.79)	**21.95**(±0.74)	**76.79**	<u>14.50</u>	**75.17**	<u>15.04</u>	**71.77**(±0.12)	**17.86**(±0.27)

Table 1: Average incremental accuracy (A) and forgetting measure (F) results on CIFAR-100 and ImageNet-100 after the model incrementally learns # tasks. eXX stands for the modified method that introduces the proposed *embedding distillation* in XX. Bold and underline indicate the best and second best results, respectively.

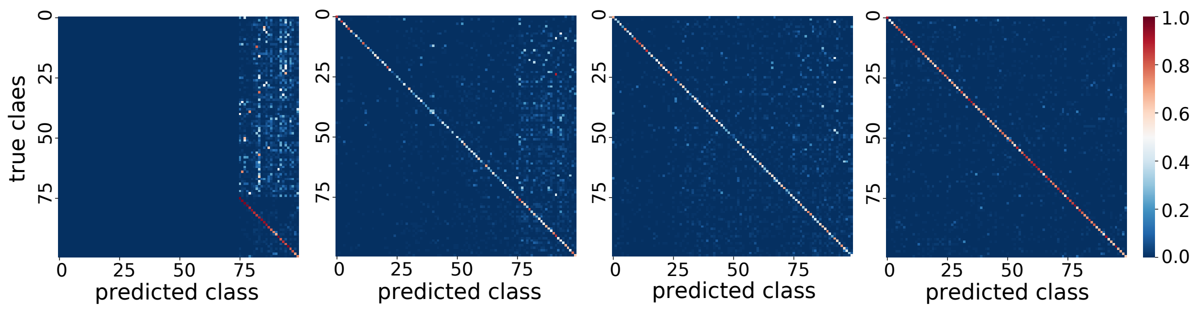


(a) 5 incremental tasks (b) 10 incremental tasks (c) 25 incremental tasks

Figure 4: Detailed ACCs on CIFAR-100 with various CIL tasks. Results on ImageNet-100 are given in Appendix-B.1 of SM^2.

For instance, on 10 tasks of CIFAR-100, eGFR improves GFR by 4.77% in average incremental accuracy but suffers more severe forgetting, increasing from 21.74% to 23.95%. This may be due to confusion between incremental features and the incremental classifier. In contrast, eTag achieves superior accuracy with less forgetting. This is mainly attributed to additional self-supervised tasks for distilling embedding knowledge and Tag for generating specific features to train the incremental classifier. We provide the detailed ACCs and standard deviations of four methods, EWC, LwM, Lucir, and GFR as they show better incremental accuracy in each comparison category. The confidence intervals of them are over 95%. As shown in Fig.4, EWC and LwM do not perform well, especially in long incremental tasks. Lucir gets reasonable ACCs with stored exemplars, but its cosine criterion affects the performance of the initial task. Conversely, GFR uses the cross-entropy criterion to extract features and achieves comparable results to Lucir, even without any exemplar. Moreover, the results show that directly incorporating embedding distillation in these methods does not constantly improve performance for every incremental task. For instance, on 25 incremental tasks, eGFR's ACCs decrease compared with the original GFR. By unifying Tag and embedding distillation, eTag consistently attains exceeded performances in all settings compared with SOTA methods by a large margin.

Confusion Matrix. We analyze the final classification results of the incremental model and visualize the final confusion matrices of Fine, GFR, eTag, and Joint by conducting experiments on four equally divided tasks of CIFAR-100.



(a) Fine (b) GFR (c) eTag (d) Joint

Figure 5: Confusion matrix of (a) Fine, (b) GFR, (c) eTag, and (d) Joint.

Baseline	B0	B1	B2	B3	B4	eTag
GE	T	N	T	N	N	T
KD	E	N	N	E	E	E
SS	-	-	-	✓	-	✓
$A(\uparrow)$	58.38	57.30	59.82	61.18	56.89	64.10

Table 2: Ablation results. Generation strategies (GE) include naïve (N) / task-oriented (T) feature generation, Knowledge distillation (KD) strategies include naïve (N) / embedding (E) KD, and whether to use SS task.

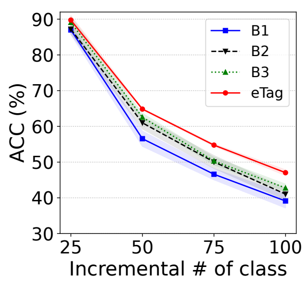


Figure 6: Component effect.

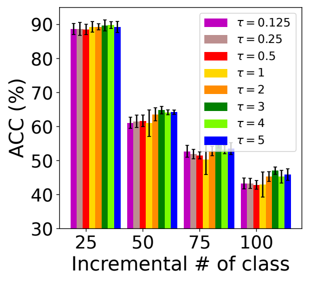


Figure 7: Parameter effect.

As shown in Fig.5, eTag primarily improves performance by addressing the class imbalance problem in GFR, where samples are inclined to be inferred as the recent task. Fine suffers significantly from this inclining problem, while Joint can avoid it. We attribute the success of eTag mainly to embedding distillation and task-oriented generation compared with the naive distillation and naive generation in GFR[4].

Ablation Studies

We conduct ablation experiments on four equally divided tasks of CIFAR-100. There are five baselines, B0, B1, B2, B3, and B4. B0 is similar to eTag only without SS, B1 uses naïve feature generation and naïve knowledge distillation (KD). B2 replaces B1's naïve feature generation with the task-oriented generation, while B4 replaces B1's naïve KD with embedding KD. B3 improves B4 by using the SS task.

Effects of Each Component. As shown in Tab.2, overall, the comparison pair "B1 versus B2" reflects the effectiveness of task-oriented feature generation; the pair "B1 versus B3" reflects the effectiveness of embedding distillation; and the pairs "B3 versus B4" and "B0 versus eTag" along with the above analyses reflects the effectiveness of SS. A more detailed analysis is given in Appendix-C.1 of **SM**[2].

Parameter Analysis. We study the sensitivity of the only manually set parameter, τ, in eTag (Eq.(1)) with ACCs. As shown in Fig.7, the results indicate that eTag is generally insensitive to the values of τ. Concretely, the maximum accuracies (average accuracy plus standard deviation) for different values of τ on each incremental task are nearly the same, although the standard deviation is relatively large when $\tau = 1$. This suggests that maintaining identical embeddings before and after CIL makes the model less plastic when learning new information[5].

Conclusion

To embrace the merits of both incrementally training a feature extractor and estimating the feature distribution, we develop a class-incremental learning method with *e*mbedding distillation and *Ta*sk-oriented *g*eneration (eTag). eTag incrementally distills the embeddings from intermediate block outputs to gain more knowledge for the feature extractor, and endows the generator to produce features that match the classifier. Extensive experiments on several benchmark datasets verified the effectiveness of our eTag in tackling the forgetting problem.

Albeit satisfactory performances, CIL is far from being solved. Particularly, eTag performs lower than the upper bound of joint training. We hope that the ideas and the success of eTag will inspire more effective data-free CIL strategies to reduce the remaining performance gap. In the future, it is desirable to improve the training efficiency of eTag and engage the pre-trained big backbone model for CIL.

[4]Additional evaluations, including the experiments on ImageNet-100 / ImageNet-1000 / Tiny-ImageNet compared with the recent exemplar-free / sample-generative CIL methods, are provided in Appendix-B of **SM**[2].

[5]We provide extensive ablation studies about SS, architecture generalization, Tag's effectiveness, time consumption, and memory budget in Appendix-C of **SM**[2].

References

Ahn, S.; Hu, S. X.; Damianou, A.; Lawrence, N. D.; and Dai, Z. 2019. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9163–9171.

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 139–154.

Aljundi, R.; Chakravarty, P.; and Tuytelaars, T. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3366–3375.

Belouadah, E.; and Popescu, A. 2019. IL2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 583–592.

Caruana, R. 1997. Multitask learning. *Machine Learning*, 28(1): 41–75.

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C.; and Alahari, K. 2018. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, 233–248.

Cha, H.; Lee, J.; and Shin, J. 2021. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 9516–9525.

Chen, Z.; and Liu, B. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3): 1–207.

Dhar, P.; Singh, R. V.; Peng, K.-C.; Wu, Z.; and Chellappa, R. 2019. Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5138–5146.

Douillard, A.; Cord, M.; Ollion, C.; Robert, T.; and Valle, E. 2020. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of European Conference on Computer Vision (ECCV)*, 86–102. Springer.

Epstein, R. 2016. The empty brain. https://aeon.co/essays/your-brain-does-not-process-information-and-it-is-not-a-computer. Accessed: 2023-03-01.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Heo, B.; Lee, M.; Yun, S.; and Choi, J. Y. 2019. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, 3779–3787.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 831–839.

Huang, L.; An, Z.; Zhi, X.; and Xu, Y. 2022. Lifelong generative learning via knowledge reconstruction. *arXiv preprint arXiv:2201.06418*.

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

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12): 2935–2947.

Liu, X.; Wu, C.; Menta, M.; Herranz, L.; Raducanu, B.; Bagdanov, A. D.; Jui, S.; and de Weijer, J. v. 2020. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 226–227.

Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A. D.; and Van De Weijer, J. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5): 5513–5533.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, 109–165. Elsevier.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113: 54–71.

Park, W.; Kim, D.; Lu, Y.; and Cho, M. 2019. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3967–3976.

Peng, B.; Jin, X.; Liu, J.; Li, D.; Wu, Y.; Liu, Y.; Zhou, S.; and Zhang, Z. 2019. Correlation congruence for knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 5007–5016.

Ramapuram, J.; Gregorova, M.; and Kalousis, A. 2020. Lifelong generative modeling. *Neurocomputing*, 404: 381–400.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001–2010.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. *Advances in Neural Information Processing Systems (NIPS)*, 30.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tian, Y.; Krishnan, D.; and Isola, P. 2019. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*.

Tung, F.; and Mori, G. 2019. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1365–1374.

van de Ven, G. M.; Siegelmann, H. T.; and Tolias, A. S. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1): 1–14.

Wang, F.-Y.; Zhou, D.-W.; Ye, H.-J.; and Zhan, D.-C. 2022. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision (ECCV)*, 398–414. Springer.

Wu, C.; Herranz, L.; Liu, X.; van de Weijer, J.; Raducanu, B.; et al. 2018. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems (NIPS)*, 31.

Wu, K.; Zhang, J.; Peng, H.; Liu, M.; Xiao, B.; Fu, J.; and Yuan, L. 2022. Tinyvit: Fast pretraining distillation for small vision transformers. In *European Conference on Computer Vision (ECCV)*, 68–85. Springer.

Xiang, Y.; Fu, Y.; Ji, P.; and Huang, H. 2019. Incremental learning using conditional adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 6619–6628.

Xu, Y.; Liu, X.; Cao, X.; Huang, C.; Liu, E.; Qian, S.; Liu, X.; Wu, Y.; Dong, F.; Qiu, C.-W.; et al. 2021. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4).

Yang, C.; An, Z.; Cai, L.; and Xu, Y. 2021. Hierarchical self-supervised augmented knowledge distillation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1217–1223.

Yang, C.; Zhou, H.; An, Z.; Jiang, X.; Xu, Y.; and Zhang, Q. 2022. Cross-image relational knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12319–12328.

Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S.; and Weijer, J. v. d. 2020. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6982–6991.

Zhu, F.; Zhang, X.-Y.; Wang, C.; Yin, F.; and Liu, C.-L. 2021. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5871–5880.