Bring Evanescent Representations to Life in Lifelong Class Incremental Learning

Marco Toldo [1,2*] Mete Ozay[1]
[1]Samsung Research UK [2]University of Padova
marco.toldo@dei.unipd.it m.ozay@samsung.com

Abstract

In Class Incremental Learning (CIL), a classification model is progressively trained at each incremental step on an evolving dataset of new classes, while at the same time, it is required to preserve knowledge of all the classes observed so far. Prototypical representations can be leveraged to model feature distribution for the past data and inject information of former classes in later incremental steps without resorting to stored exemplars. However, if not updated, those representations become increasingly outdated as the incremental learning progresses with new classes. To address the aforementioned problems, we propose a framework which aims to (i) model the semantic drift by learning the relationship between representations of past and novel classes among incremental steps, and (ii) estimate the feature drift, defined as the evolution of the representations learned by models at each incremental step. Semantic and feature drifts are then jointly exploited to infer up-to-date representations of past classes (evanescent representations), and thereby infuse past knowledge into incremental training. We experimentally evaluate our framework achieving exemplar-free SotA results on multiple benchmarks. In the ablation study, we investigate nontrivial relationships between evanescent representations and models.

1. Introduction

Continual learning (also called *lifelong learning*) refers to the ability to continuously learn and adapt to new environments, exploiting knowledge gained from the past for solving novel tasks. Though being a common human trait, lifelong learning methods are hardly deployed in practical systems. As a matter of fact, learning models are usually constrained to well-defined and narrow tasks, where they can achieve remarkable performance. Nonetheless, when training a model on a continuous stream of tasks, the *catastrophic forgetting* arises; new information acquired by the model tends to erase what has been experienced so far.

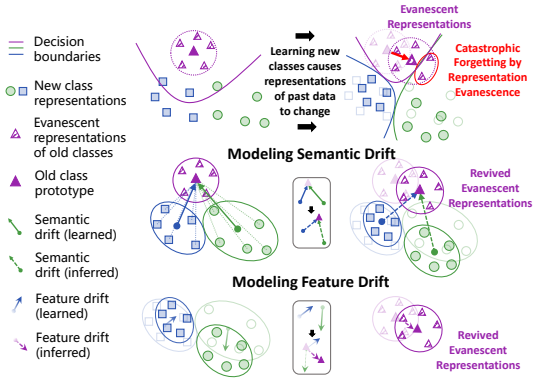

Figure 1. In CIL, training models on new classes causes representations of past categories to constantly change. Yet, unavailability of data of former classes prevents from tracking their evolution in feature spaces, leading to *evanescence* of their representations and, in turn, to catastrophic forgetting. We propose to model representation drift on a semantic level (*i.e.*, relationship among novel and past classes) and on a feature level (*i.e.*, the combined evolution of features learned by a classification model), and exploit it to infer up-to-date representations of past classes. By injecting old-class knowledge into the learning process, we counteract forgetting.

Continual learning has been extensively studied in a class incremental fashion [6, 22, 23]. In class incremental learning (CIL), a model is employed with sequential tasks, where classes to be learned progressively change (Fig. 1). For each incremental training task and step t, the training set is composed of images belonging to the current class set $\mathcal{C}_t$, whereas past semantic categories $\mathcal{C}_{old} \triangleq \{\mathcal{C}_{t'}\}_{t'=1}^{t-1}$ lack any training sample. The goal of the model is to maximise the generalisation (classification) accuracy on all the classes observed up to the current step. Yet, the change of distribution of training data $\mathcal{D}_t$ in the form of semantic drift (*i.e.*, due to change of experienced class set $\mathcal{C}_t$) leads to forgetting, where bias towards new data causes past information to be gradually erased and learned representations to be constantly updated (*i.e.*, feature drift) focusing on new tasks.

Contributions of this work to overcome the aforementioned limitations can be summarised as follows:

*Researched during internship at Samsung Research UK

- To expound the forgetting phenomena in CIL, we explore dynamics of incrementally learned classifiers using a probabilistic approach. Our investigation (Fig. 1) suggests that a source is the evanescence of representations $\mathcal{F}_{old}$ learned using old classes $\mathcal{C}_{old}$ and unavailability of their distribution $p(\mathcal{F}_{old})$ in incremental steps (Sec. 3).
- To revive the evanescent representations (ERs), we devise a framework which enables to model different types of representation drifts modularly. In the framework (Fig. 1, 2 and 3), we first define the change of feature representations by feature drift (*i.e.*, due to constantly evolving feature representations of different patterns learned from data in CIL) and propose an effective method to model it (Sec. 4.1). Next, we define the change of representations of classes by semantic drift (*i.e.*, due to the change of semantic categories learned at different incremental steps) and propose an effective method to model it (Sec. 4.2).
- We propose to train semantic and feature drift models together with feature learning and classification models. The proposed method integrates learning and inference in training: it is used to estimate distributions $p(\mathcal{F}_{old})$ to be exploited for preserving knowledge of $\mathcal{C}_{old}$ while learning new representations for $\mathcal{C}_t$, $\forall t$ (Sec. 5).
- In the experimental analyses, our proposed methods outperform SotA exemplar-free competitors on various benchmarks. We also provide a detailed ablation study of geometric and statistical properties of drift models (Sec. 6). Our experimental results explicate the nontrivial relationships between accuracy of models and distribution of evanescent and revived representations in CIL (Sect. 6.2).

2. Related Work

Most of the successful CIL methods use exemplars of old classes $\mathcal{C}_{old}$ to rehearse past knowledge [1, 3, 4, 7, 8, 12, 18, 20, 25, 27, 32, 36]. However, storing samples belonging to all classes might be impractical due to limited resource availability or privacy requirements. To address CIL without storing exemplars, regularisation methods have been proposed [2, 5, 16, 39, 41]; the common goal is to identify key model parameters to solve old tasks, and prevent their change when learning a new task. Alternatively, knowledge distillation [11, 19] has been introduced, where representations of new classes are forced to only slightly deviate from their original version computed at the beginning of an incremental step for learning the current task. Yet, those methods usually underperform state-of-the-art (SotA). Generation of pseudo-exemplars of past classes has been proposed in [15, 26, 28, 34, 37]. Nonetheless, these works resort to complex generative frameworks, which still must address an additional generative auxiliary task. We instead operate in feature spaces, where we can effectively leverage prototypes [29] and use lightweight models. Causal networks have been exploited for CIL [13], and provided SotA accuracy if jointly applied on top of exemplar-based methods. An exemplar-free approach using self-supervised learning was devised in [35], yet with focus on embedding networks.

In [43], class prototypes were used to inject past knowledge. Although showing promising results, this method fails to capture the representation drift that is present while incrementally training models. This is because representations $\mathcal{F}_{old} \triangleq \{\mathcal{F}_{t'}\}_{t'=1}^{t}$ of $\mathcal{C}_{old}$ (computed when corresponding data is available and then kept fixed for the rest of the training) are getting constantly staler and more outdated as the learning progresses and models are updated. A different work [40] proposes to estimate the change of prototypes $\Pi_{old} \triangleq \{\Pi_{t'}\}_{t'=1}^{t}$ of $\mathcal{C}_{old}$ while learning $\mathcal{C}_t$. However, they (i) do not directly capture the relationship between semantic representations of old (Π_{old}) and new classes ($\Pi_{t'}$), rather focusing on estimating the change of drifts across $\mathcal{C}_{old}$ and $\mathcal{C}_t$, (ii) neglect what we call feature drift, that is the joint evolution of model features during single task training (*i.e.*, they treat feature representation drift independently for each feature channel), (iii) limit their scope to embedding learning and (iv) devise a non-learnable module to estimate prototype shift, where we show that our framework learning representation drift provides models with higher capacity, and leads to overall improved classification accuracy.

3. Learning Representations in CIL

At each step $t \in [T] = \{0, 1, \dots, T\}$ of CIL, we are given a dataset $\mathcal{D}_t = (\mathcal{X}_t, \mathcal{Y}_t)$, where $\mathcal{X}_t = \{x_{t,j}\}_{j=1}^{N_t}$ is the set of samples, $\mathcal{Y}_t = \{y_{t,j} \in \mathcal{C}_t\}_{j=1}^{N_t}$ is the set of their labels. $\mathcal{C}_t$ is the set of class labels observed at this step, and $\mathcal{C}_t \cap \mathcal{C}_{t'} = \emptyset, \forall t \neq t'$. Popularly employed CIL models [43] are composed of a feature extraction model $f_\theta \in \mathcal{F}$ and a classifier $h_\phi \in \mathcal{H}$ with parameters $\theta \in \Theta$ and $\phi \in \Phi$. At each t^{th} step, the models are trained by solving

$$\underset{\theta_t, \phi_t, \epsilon}{\arg\min}\, \mathcal{L}_{cc}^{t} + \sum_{t'=0}^{t-1} \epsilon_{t'}, \tag{1}$$

where $\mathcal{L}_{cc}^{t} \triangleq \mathcal{L}_{\mu_t}(\mathcal{D}_t; \theta_t, \phi_t)$ is the expected loss of the model $g_t = h_{\phi_t} \circ f_{\theta_t}$ on $\mathcal{D}_t$ sampled from a distribution μ_t at step t, while $\epsilon_{t'}$ controls *forgetting* of representations of old classes [43]. Generative classifiers implementing $h_\phi \in \mathcal{H}$ optimise (1) to model $p(C, F; \mathcal{P})$ where $\mathcal{P} = \Theta \cup \Phi$. Discriminative classifiers, such as softmax classifier, optimise (1) to model $p(C|F; \mathcal{P})$ defining the loss by a function (*e.g.*, cross-entropy) of $p(C|F; \mathcal{P})$. CIL methods aim to model $p(C, F|\mathcal{P}_t)$ without using $\{\mathcal{D}_{t'}\}_{t'=0}^{t}$ at t, where

- $\mathcal{P}_t = \Theta_t \cup \Phi_t$, $\Theta_t = \{\theta_{t'}\}_{t'=0}^{t}$ and $\Phi_t = \{\phi_{t'}\}_{t'=0}^{t}$[1],
- $F \in \mathcal{F} = \mathcal{F}_t \cup \mathcal{F}_{old}$ is the random variable taking values from the set of feature representations $\mathcal{F}_t$ learned at step t on $\mathcal{D}_t$ and from $\mathcal{F}_{old} = \{\mathcal{F}_{t'}\}_{t'=0}^{t-1}$, and

[1] In some CIL methods, models trained at earlier steps $t' < t$ are frozen and re-used at consecutive steps, while the other methods incrementally update the models at each step. In the latter case, $\mathcal{P}_t = \{\theta_t\} \cup \{\phi_t\}$.

Term	Distribution	Term	Distribution
$\mathbf{P}_C$	$p(\mathcal{F}_{old}, \mathcal{F}_t)$	$\mathbf{P}_{12}$	$p(\mathcal{C}_t, \mathcal{F}_t, \mathcal{F}_{old})$
$\mathbf{P}_1$	$p(\mathcal{C}_t, \mathcal{F}_t)$	$\mathbf{P}_{13}$	$p(\mathcal{C}_t, \mathcal{C}_{old}, \mathcal{F}_t)$
$\mathbf{P}_2$	$p(\mathcal{C}_t, \mathcal{F}_{old})$	$\mathbf{P}_{14}$	$p(\mathcal{C}_{old}, \mathcal{F}_t, \mathcal{F}_{old})$
$\mathbf{P}_3$	$p(\mathcal{C}_{old}, \mathcal{F}_t)$	$\mathbf{P}_{24}$	$p(\mathcal{C}_t, \mathcal{C}_{old}, \mathcal{F}_{old})$
$\mathbf{P}_4$	$p(\mathcal{C}_{old}, \mathcal{F}_{old})$	$\mathbf{P}_{34}$	$p(\mathcal{C}_{old}, \mathcal{F}_{old}, \mathcal{F}_t)$

Table 1. Factors of the classification probablity $p(\mathcal{C}|\mathcal{F})$ modeled by a discriminative classifier in CIL.

- $C \in \mathcal{C} = \mathcal{C}_t \cup \mathcal{C}_{old}$ is a random variable of semantic (class) representations, where $\mathcal{C}_{old} = \{\mathcal{C}_{t'}\}_{t'=0}^{t-1}$.

To elucidate the dynamics of models used for CIL in this setting, we factorize the probability $p(C \in \mathcal{C}|F \in \mathcal{F})$ by[2]

$$p(C \in \mathcal{C}|F \in \mathcal{F}) \propto \frac{\mathbf{P}_A - \mathbf{P}_B}{\mathbf{P}_C}, \tag{2}$$

where $\mathbf{P}_C, \mathbf{P}_A := \mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3 + \mathbf{P}_4$, and $\mathbf{P}_B := \mathbf{P}_{12} + \mathbf{P}_{13} + \mathbf{P}_{14} + \mathbf{P}_{24} + \mathbf{P}_{34}$ are expressed in Table 1. More details are provided in supplemental material.

Deep learning models have been employed to model these probabilities implicitly by learning representations of old and new classes, and make predictions for new classes: (i) **Class posterior probabilities** for old ($p(C \in \mathcal{C}_{old}|\bullet)$) and new classes ($p(C \in \mathcal{C}_t|\bullet)$) are computed using classifiers $h_{\Phi_{t'}}, \forall t' \in [t]$. (ii) **Feature representations** $\mathcal{F}_t$ of new classes $\mathcal{C}_t$ are learned by updating $f_{\Theta_{t'}}, \forall t' \in [t]$.

Evanescent Representations (ERs): At the t^{th} step, we do not have access to old samples $\mathcal{D}_{old} = \{\mathcal{D}_{t'}\}_{t'=0}^{t-1}$. Thus, features $\mathcal{F}_{old}$ cannot be extracted from $\mathcal{D}_{old}$. Therefore, methods aiming to learn representations via (2) treat $\mathbf{P}_C$, which denotes the distribution of features learned using old classes, as a normalising partition function, and ignore it to compute $p(C \in \mathcal{C}|F \in \mathcal{F}) \propto \mathbf{P}_A - \mathbf{P}_B$.

To address this problem and bring the *evanescent representations* $\mathcal{F}_{old}$ to life, we exploit class prototypes[3] $\pi \in \Pi_{old}$. We leverage prototypes at the beginning of an incremental step to model their distribution $p(F \in \mathcal{F}_{old}|\pi_c \in \Pi_{old})$ (denoted by $\mathcal{F}_{old} \sim \Pi_{old}$ in Fig. 2 and Fig. 3). Then, we update $p(F \in \mathcal{F}_{old})$ throughout the incremental step by modeling the representation drift to *revive* evanescent representations. In the next section, we will provide a detailed description of the proposed approach.

4. Modeling Representation Drift

Let $\mathcal{F}_t^0$ denote the set of features extracted using a feature extractor f_{θ_t} at the beginning of the incremental step t, and $\mathcal{F}_t^n$ denote the set of features extracted using f_{θ_t} updated with $n > 0$ optimisation stages from the dataset $\mathcal{D}_t$. Since only $\mathcal{D}_t$ is available at the step t, $\mathcal{F}_t^0$ and $\mathcal{F}_t^n$ contain only representations of new classes $\mathcal{C}_t$. Similarly, $\Pi_{old}^{t,0}$ and $\Pi_{old}^{t,n}$ are the set of semantic representations (prototypes) of old classes $\mathcal{C}_{old}$, respectively available at the beginning of the step t and updated at the n^{th} stage ($n > 0$) of the same step. Moreover, we remark that storing prototypes is very memory efficient and compliant to privacy requirements [43], since a very limited amount of processed data has to be stored (*i.e.*, comparable to the size of a classifier).

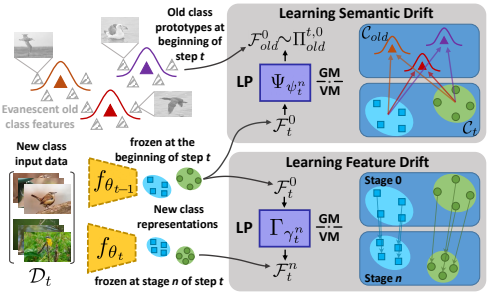

Figure 2. Illustration of the learning phase (LP) of semantic and feature drift models. SD (top): we model the relationship between representations of new and old classes at the beginning of the incremental step. FD (bottom): we capture the evolution of representations of novel classes within the incremental step.

We propose modeling the drift of feature and semantic representations using two models: (i) Γ_γ parameterized by γ, and (ii) Ψ_ψ parameterized by ψ in two phases:

- In the **Learning Phase (LP);** the parameters γ and ψ are optimized to estimate the relationship among representations available at an incremental step t (Fig. 2).
- In the **Inference Phase (IP);** the learned models Γ_γ and Ψ_ψ are used to infer $\mathcal{F}_{old}^n$ which are *evanescent representations revived at the step* t (Fig. 3).

The following subsections present the methods proposed for modeling drifts. To identify and train Γ_γ and Ψ_ψ, we propose employment of Gaussian (GM) and Variational (VM)[4] models parameterized by DNNs. We introduce model training algorithms in Sec. 5. The analyses given in Sec. 6 show that GMs provide more robust accuracy since they do not suffer from pathologies of VMs [21, 38]. On the other hand, VMs can provide higher accuracy for small $\mathcal{D}_t$.

4.1. Modeling Feature Drift

4.1.1 Learning Phase

We aim at modeling the feature drift (FD) on $\mathcal{D}_t$ as representations revive and evolve throughout the step $t > 0$ as depicted in Fig. 2. At stage $n > 0$, we extract $\mathcal{F}_t^n$ and train $\Gamma_{\gamma_t^n}$ to learn the relationship between $\mathcal{F}_t^0$ and $\mathcal{F}_t^n$.

GM: To employ GMs, we first identify $\Gamma_{\gamma_t^n} : \mathcal{F}_t^0 \rightarrow \mathcal{F}_t^n$ by a DNN, *e.g.*, a multilayer perceptron (MLP). Then, $\Gamma_{\gamma_t^n}$ is trained to track and model the evolution of revived evanescent representations (RERs) from stage 0 to n.

VM: We consider $\Gamma_{\gamma_t^n}$ as a stochastic map and identify it by a variational model such as a variational auto-encoder (VAE). The VM is trained by maximising the likelihood

[2] We drop $\mathcal{P}_t$ when dependency on it is trivial for the sake of simplicity.

[3] We obtain prototypes by computing class-wise mean of features [43].

[4] Although Gaussian processes can be used for VMs [14], we consider GMs and VM individually to explicate variational structure of VMs.

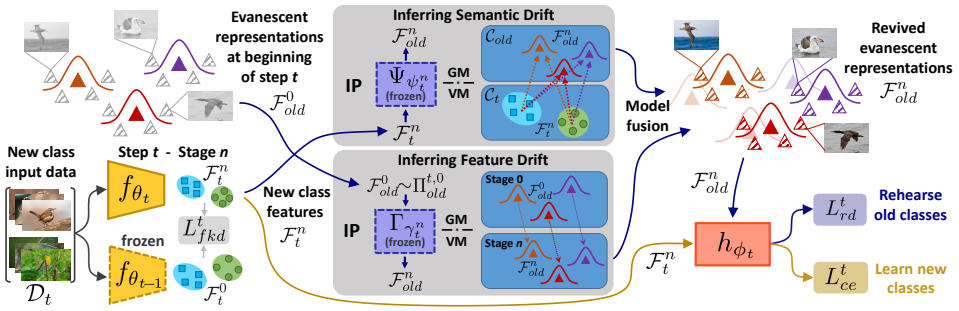

Figure 3. An outline of our CIL framework augmented with the proposed representation drift models. A classifier is trained on the dataset $\mathcal{D}_t$ at step t (L^t_{ce}). Feature knowledge distillation (L^t_{fkd}) is used to reduce feature drift. We exploit the drift models learned in LP (Fig. 2) to infer revived evanescent representations (RERs), which are leveraged by L^t_{rd} to inject past knowledge into the current training procedure.

$p(F \in \mathcal{F}^n_t | F \in \mathcal{F}^0_t; \gamma^n_t)$. Thereby, we can statistically model the FD across different stages $[n]$ at a given step t.

4.1.2 Inference Phase

In the LP, the model $\Gamma_{\gamma^n_t}$ is trained to learn the drift experienced by features of $\mathcal{C}_t$. In the IP (Fig. 3), we exploit the trained $\Gamma_{\gamma^n_t}$ to infer the FD undergone by features of $\mathcal{C}_{old}$, and distribution $p(F \in \mathcal{F}^n_{old})$ of RERs.

GM: We use $\Gamma_{\gamma^n_t}$ to infer representations of old class prototypes $\Pi^{t,n}_{old}$ at stage $n > 0$ under the FD. That is, $\Gamma_{\gamma^n_t}$ enables us to directly track the trajectory of Π^t_{old} from stage 0 to n, and thus map $\Pi^{t,0}_{old}$ to $\Pi^{t,n}_{old,f}$. If $n = 0$, then $\Gamma_{\gamma^0_t}$ is an identity map. Finally, we approximate the distribution $p(F \in \mathcal{F}^n_{old})$ of RERs at stage n by a Gaussian distribution $p(F \in \mathcal{F}^n_{old}; \pi_c \in \Pi^{t,n}_{old,f}) \sim \mathcal{N}(\pi_c, \sigma_c)$. For both GM and VM, the standard deviation σ_c is estimated at step t' when $c \in \mathcal{C}_{t'}$, and kept fixed at every step $t'' > t'$.

VM: The trained model $\Gamma_{\gamma^n_t}$ provides an approximation of $p(F \in \mathcal{F}^n_{old} | F \in \mathcal{F}^0_{old})$. At stage $n = 0$, we resort to $p(F \in \mathcal{F}^0_{old})$, as no feature drift has to be estimated, and we model the distribution of RERs by $p(F \in \mathcal{F}^0_{old}) \propto p(F \in \mathcal{F}^0_{old}; \pi_c \in \Pi^{t,0}_{old}) \sim \mathcal{N}(\pi_c, \sigma_c)$. At $n > 0$, training features are sampled from $p(F \in \mathcal{F}^n_{old} | F \in \mathcal{F}^0_{old}; \gamma^n_t) \cdot p(F \in \mathcal{F}^0_{old})$.

4.2. Modeling Semantic Drift

4.2.1 Learning Phase

We aim at capturing the semantic drift (SD) experienced by representations at each incremental step. To this end, at the beginning of each step, we extract $\mathcal{F}^0_t$ and train a network $\Psi_{\psi^0_t}$ with parameters ψ^0_t to model the relationship between $\mathcal{F}^0_t$ and $\mathcal{F}^0_{old}$. We employ prototypes $\pi \in \Pi^{t,0}_{old}$ to model $p(F \in \mathcal{F}^0_{old}) \propto p(F \in \mathcal{F}^0_{old}; \pi_c \in \Pi^{t,0}_{old}) \sim \mathcal{N}(\pi_c, \sigma_c)$. As opposed to Γ_γ, Ψ_ψ captures the SD observed at each new step. Therefore, an individual model Ψ_ψ is optimised at the start, and fixed for the rest of the step (*i.e.*, $\psi^n_t = \psi^0_t$). When model fusion is adopted, instead, Ψ_ψ is re-trained once per stage , to account for the drift estimated by Γ_γ (Sec. 5.2).

GM: We first identify $\Psi_{\psi^n_t} : \mathcal{F}^0_t \rightarrow \Pi^{t,0}_{old}$ by a DNN. (*e.g.*, MLP). Then, $\Psi_{\psi^n_t}$ is trained to model the SD between representations for classes available at the current step ($\mathcal{C}_t$) and those experienced in the past ($\mathcal{C}_{old}$).

VM: We first approximate the distribution $p(F \in \mathcal{F}^0_{old})$ of evanescent representations revived at stage $n = 0$ by $p(F \in \mathcal{F}^0_{old}; \pi_c \in \Pi^{t,0}_{old}) \sim \mathcal{N}(\pi_c, \sigma_c)$. Then, a conditional VM (*e.g.*, VAE) is trained by maximising the likelihood $p(F \in \mathcal{F}^0_{old} | F \in \mathcal{F}^0_t; \psi^n_t)$ to learn the representations shared among old and new classes.

4.2.2 Inference Phase

In the LP, $\Psi_{\psi^n_t}$ is trained to learn the SD experienced while moving from $\mathcal{C}_{old}$ to $\mathcal{C}_t$. The drift is captured at the beginning of the current step t, when up-to-date representations of both sets are available. We now exploit the trained $\Psi_{\psi^n_t}$ to infer the SD undergone at every stage $n > 0$, and estimate the distribution $p(F \in \mathcal{F}^n_{old})$ of RERs.

GM: We use the trained $\Psi_{\psi^n_t}$ to infer representations of old class prototypes $\Pi^{t,n}_{old}, \forall n \geq 0$ under SD. That is, $\Psi_{\psi^n_t} : \mathcal{F}^n_t \rightarrow \Pi^{t,n}_{old,s}$ is trained to estimate the relationship between feature and prototypical representations at stage n. We then approximate $p(F \in \mathcal{F}^n_{old})$ by $p(F \in \mathcal{F}^n_{old}; \pi_c \in \Pi^{t,n}_{old,s}) \sim \mathcal{N}(\pi_c, \sigma_c)$.

VM: The trained model $\Psi_{\psi^n_t}$ provides an approximation of the distribution $p(F \in \mathcal{F}^n_{old} | F \in \mathcal{F}^n_t), \forall n \geq 0$. To generate training feature samples, we perform inference using $p(F \in \mathcal{F}^n_{old} | F \in \mathcal{F}^n_t; \psi^n_t) \cdot p(F \in \mathcal{F}^n_t)$, where $\mathcal{F}^n_t$ is provided by f_{θ_t} trained for n optimisation stages and applied on samples from $\mathcal{D}_t$.

5. Training Models of Representation Drift

5.1. Training Classification Models

At each step t, we train $g_t = h_{\phi_t} \circ f_{\theta_t}$ on $\mathcal{D}_t$ using a cross-entropy loss $L^t_{ce} \triangleq L_{ce}(\mathcal{D}_t)$.

When $t > 0$, to mitigate forgetting of previous tasks, we generate features of $\mathcal{C}_{old}$ by modeling the drift and estimating the distribution $p(F \in \mathcal{F}_{old})$. Thus, we compute

Algorithm 1: Training Models.

Input: $\{\mathcal{D}_t\}_{t=0}^{T}$ (datasets), N (num. of stages per step).
Output: $g_T = h_{\phi_T} \circ f_{\theta_T}$.

Train f_{θ_0} and h_{ϕ_0} with L_{cc}^0, and compute Π_{new}^0.
Initialise $\Pi_{old}^{t=1,0}$ as Π_{new}^0.
for each incremental step $t \leftarrow 1$ to T **do**
  for each optimisation stage $n \leftarrow 0$ to $N-1$ **do**
    LP: Train $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ by solving (5).
    IP: Estimate $p(\mathcal{F}_{old}^n)$ and $\Pi_{old}^{t,n}$.
    Train f_{θ_t} and h_{ϕ_t} by solving (6).
  end for
  LP: Train $\Gamma_{\gamma_t^N}$ and $\Psi_{\psi_t^N}$ by solving (5).
  IP: Estimate $\Pi_{old}^{t,N}$ and Π_{new}^t.
  Initialise $\Pi_{old}^{t+1,0} := \Pi_{new}^t \cup \Pi_{old}^{t,N}$.
end for

$$L_{rd}^t \triangleq L_{rd}(\mathcal{F}_{old}^{n,t}) = \sum_{F \in \mathcal{F}_{old}^{n,t}} y_F \log h_{\phi_t}(F) \tag{3}$$

where y_F is the one-hot label vector of the class $c_F \in \mathcal{C}_{old}$ and $\mathcal{F}_{old}^{n,t}$ is the set of RERs sampled from the estimated distribution using the updated prototypes of $\mathcal{C}_{old}$. The loss L_{rd}^t approximates $\mathcal{L}_{\mu_{t'}}(\mathcal{D}_{t'}; \theta_t, \phi_t) \leq \epsilon_{t'}$ of g_t on the previous datasets $\{\mathcal{D}_{t'}\}_{t'=0}^{t-1}$ using their inferred representations.

We enhance the training objective by a distillation loss $L_{fkd}^t \triangleq L_{fkd}(\mathcal{D}_t)$ [7] to reduce the entity of representation drift across incremental tasks. L_{fkd}^t is defined by the ℓ_2 distance between representations extracted from $\mathcal{D}_t$ using f_{θ_t} and $f_{\theta_{t-1}}$, the latter inherited from the previous step and kept fixed. Thereby, L_{fkd}^t approximates the difference between $\mathcal{L}_{cc}^t$ and the loss $\mathcal{L}_{pc}^t$ of the previous model $f_{\theta_{t-1}}$ on $\mathcal{D}_t$. Although $\mathcal{L}_{pc}^t$ is not explicitly defined in (1), it provides information regarding shareability of representations among consecutive steps $t-1$ and t. Therefore, models optimising L_{fkd}^t can make use of the feature shareability for learning drifts. Then, the overall classification objective L_{cc}^t computed at each step t is $L_{cc}^t = L_{ce}^t + \lambda_{rd} L_{rd}^t + \lambda_{fkd} L_{fkd}^t$, where L_{rd}^t and L_{fkd}^t are used only for $t > 0$ with loss balancing parameters $\lambda_{rd} > 0$ and $\lambda_{fkd} > 0$.

5.2. Training Representation Drift Models

The loss functions $L_f^t \triangleq L_f(\mathcal{F}_t^0, \mathcal{F}_t^n; \gamma_t^n)$ and $L_s^t \triangleq L_s(\mathcal{F}_t^0, \Pi_{old}^{t,0}; \psi_t^n)$ denote the objectives used to individually train feature and semantic drift models $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$, respectively. The exact form of the aforementioned objectives depends on the employed network architecture identifying $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$. A more detailed description is provided in the supplementary material.

Model Fusion (MF): To estimate $p(F \in \mathcal{F}_{old}^n)$ using **GM** and **VM**, we fuse the output of $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ by jointly training them. For this purpose, we optimise model parameters by minimising a measure of discrepancy between the estimated distributions $p(F \in \mathcal{F}_{old}^n; \gamma_t^n)$ and $p(F \in \mathcal{F}_{old}^n; \psi_t^n)$ employing the training objective

$$\begin{aligned} L_{fus}^t = &\|\Pi_{old,s}^{t,n} - \Pi_{old,f}^{t,n}\|_2^2 \\ &+ \lambda_{corr} \|\rho(\Pi_{old,s}^{t,n}) - \rho(\Pi_{old,f}^{t,n})\|_2^2 \end{aligned} \tag{4}$$

where the subscript s and f denotes the updated prototypes of old classes estimated by the semantic and feature drift models, respectively, $\|\cdot\|_2^2$ is the squared ℓ_2 norm, $\lambda_{corr} > 0$ is the regularisation parameter and $\rho(\Pi)$ is the normalised correlation matrix of Π [9]. Finally, the renovated distributions $p(F \in \mathcal{F}_{old}^n; \gamma_t^n)$ and $p(F \in \mathcal{F}_{old}^n; \psi_t^n)$ are linearly combined with equal weights to obtain $p(F \in \mathcal{F}_{old}^n)$.

Then, the overall objective used to learn representation drift is defined by $L_{drift}^t = L_s^t + L_f^t + \lambda_{fus} L_{fus}^t$ where $\lambda_{fus} > 0$ is the loss balancing parameter. We note that L_{drift}^t measures the loss of current models on inferred representations of old classes. Thereby, we aim at reducing forgetting (ϵ_t), *i.e.*, the discrepancy between RERs as estimated by drift models and their evanescent (unavailable) counterparts by training models optimising L_{drift}^t.

5.3. Optimisation of Model Parameters

In the previous subsections, we designed the loss functions to capture losses induced by representation drift in CIL while training classification models. Consequently, we consider training models by minimizing $L_{cc}^t + L_{drift}^t, \forall t$. At the incremental step $t = 0$, we train f_{θ_0} and h_{ϕ_0} with $L_{ce}(\mathcal{D}_0)$. At each step $t > 0$, we train classification and drift models in an alternate fashion as follows:

- $\mathcal{F}_t^n$ is extracted from $\mathcal{D}_t$ using f_{θ_t}, and $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ are trained until convergence[5] by solving

$$\arg\min_{\gamma_t^n, \psi_t^n} L_{drift}^t(\Pi_{old}^{t,0}, \Pi_{old,\{f,s\}}^{t,n}, \mathcal{F}_t^0, \mathcal{F}_t^n; \gamma_t^n, \psi_t^n). \tag{5}$$

- First, $\mathcal{F}_{old}^n$ is estimated by drift models $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ (employed individually or fused). Then, $\Pi_{old}^{t,n}$ is computed by class-wise averaging features sampled from $p(F \in \mathcal{F}_{old}^n)$. Finally, f_{θ_t} and h_{ϕ_t} are trained on $\tilde{\mathcal{D}}_t = \mathcal{D}_t \cup \mathcal{F}_{old}^n$ by

$$\arg\min_{\theta_t, \phi_t} L_{cc}^t(\tilde{\mathcal{D}}_t; \theta_t, \phi_t). \tag{6}$$

At the end of step $t \geq 0$, we compute $\Pi_{new}^t = \{\pi_c, c \in \mathcal{C}_t\}$ by class-wise average of feature representations $f_{\theta_t}(\mathcal{D}_t)$ of input samples and initialise $\Pi_{old}^{t+1,0} = \Pi_{old}^{t,n} \cup \Pi_{new}^t$, where $\Pi_{old}^{t,n} = \emptyset$ for $t = 0$. A detailed description of the training procedure is given in Algorithm 1.

6. Experimental Results

Datasets. We evaluate our approach on multiple standard CIL benchmarks, that is, CIFAR100 [17], TinyImageNet [24] and CUB200-2011 [33] datasets. We devise 3

[5] The convergence criterion for a model is early stopping the optimisation of model parameters if the training loss does not change for τ steps.

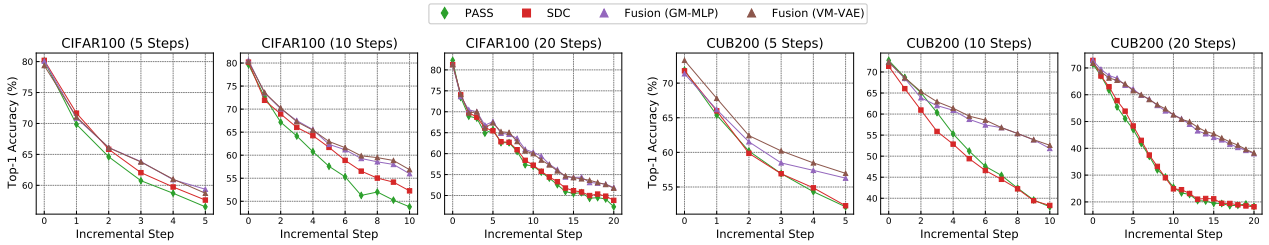

Figure 4. Per step average top-1 accuracy (%) on the CIFAR100 and CUB200-2011 datasets.

class-incremental setups; first, the framework is trained on half of the available semantic classes (except for one setup on CIFAR100, where only 40 classes are selected as the first task); then, the remaining class set is evenly divided into respectively 5, 10 or 20 incremental steps. Class order is selected randomly and then fixed at every class split.

Implementation details. ResNet-18 [10] is used as a backbone. The model is trained for 100 epochs (*i.e.*, $N = 100$ and each stage corresponds to one epoch over $\mathcal{D}_t$) at each incremental step with Adam optimiser. Learning rate is initialised to $1\mathrm{e}{-3}$ for CIFAR100 and TinyImageNet, and to $1\mathrm{e}{-4}$ for CUB200-2011. It is decreased by a factor of 0.1 after 45 and 90 epochs [43]. Images are cropped to 32×32, 64×64 and 256×256 for CIFAR100, TinyImageNet and CUB200-2011 respectively, and randomly flipped. We apply input and label augmentation [43]. We set batch size to 64, and $\lambda_{fkd} = 10$ and $\lambda_{rd} = 10$ in all experiments.

We employed lightweight DNNs to identify networks of Γ_γ and Ψ_ψ to model representation drifts. In particular, we investigate the use of GMs with an MLP, and we use a conditional VAE [30, 42] to implement VMs, where hyperparameters are experimentally tuned. We implement the fusion loss (4) using two methods for an ablation: we define ρ by (i) a normalised feature kernel matrix [9], and (ii) an identity map, *i.e.*, $A = \rho(A)$. The results obtained by (ii) are marked by † in the tables. An extensive description of the implementation and training details of Γ_γ and Ψ_ψ is provided in the supplemental material.

Comparisons. We compare our approach with several CIL methods storing exemplars of old classes (EEIL [3], iCarl [25], UCIR [12], DER [39]) and other SotA exemplar-free methods (EWC [16], LwF [19], LwM [7], PASS [43], SDC [40]). As for exemplar-based methods, we store 20 samples with *herd selection* [12, 25]. We evaluate SDC [40] by employing the prototype drift compensation proposed in [40] to update prototypes of past classes, and model old-class feature distribution by Gaussians as discussed in Sec. 4. In the following sections, we will show how our methods outperform SotA exemplar-free frameworks, while surpassing some approaches that make use of exemplars.

Evaluation metrics. We use a per-step incremental accuracy metric [43] defined as the average top-1 classification accuracy over all classes observed up to the current incremental step k: $\bar{a}^k = 1/|\mathcal{C}_{0:k}| \sum_{c \in \mathcal{C}_{0:k}} a_c^k$, $\mathcal{C}_{0:k} = \cup_{t=0}^{k} \mathcal{C}_t$, where a_c^k denotes the accuracy for class c attained at step k. Accuracy results in Table 2 and 3 are computed at the end of the last incremental step. Additional implementation details and results are provided in the supplemental material.

Table 2. Per-class average top-1 accuracy (%).

Method	CIFAR100 5 Steps	CIFAR100 10 Steps	CIFAR100 20 Steps	TinyImageNet 5 Steps	TinyImageNet 10 Steps	TinyImageNet 20 Steps
Fine-tuning	9.09	4.49	2.76	8.12	4.34	2.33
Joint	72.24	72.24	72.24	58.19	58.19	58.19
EWC [16]	26.26	19.92	3.82	14.63	6.73	3.62
LwF [19]	39.51	18.00	12.58	40.62	24.43	22.62
LwM [7]	40.49	38.39	33.65	28.39	27.18	23.55
EEIL [3]	45.26	41.36	34.84	32.03	28.93	27.25
iCarl [25]	54.06	51.11	41.20	41.81	41.39	38.68
UCIR [12]	51.13	46.00	38.31	35.73	32.95	29.23
DER [39]*	66.33	65.76	-	-	-	-
PASS [43]	56.53	47.54	47.30	47.00	41.50	29.04
SDC [40]	57.62	52.26	48.84	47.89	45.41	41.46
Feat. Drift (GM-MLP)	57.91	54.45	50.63	47.48	45.19	40.56
Sem. Drift (GM-MLP)	58.33	54.15	50.85	47.92	46.21	42.43
Fusion† (GM-MLP)	58.89	55.95	51.61	47.95	46.36	42.43
Fusion (GM-MLP)	**59.37**	55.99	**51.91**	48.56	46.50	42.81
Feat. Drift (VM-VAE)	56.99	53.69	51.09	47.88	44.67	41.05
Sem. Drift (VM-VAE)	58.17	55.38	51.65	48.60	46.24	43.44
Fusion† (VM-VAE)	58.76	55.50	51.72	**48.74**	46.46	42.72
Fusion (VM-VAE)	58.72	**56.86**	51.75	48.57	**46.92**	**44.61**

† Without using the correlation objective [9] in fusion loss.
* Numerical values were directly taken from [39].

6.1. Comparison with the State-of-the-Art

CIFAR100. Results given in Table 2 show that our models (with the best achieved accuracy) outperform the closest SotA (SDC) by 1.75%, 4.6% and 3.07% for 5, 10 and 20 steps. In Fig. 4, for the sake of clearance of presentation of results, we only provide per-step incremental accuracy of Fusion (GM-MLP and VM-VAE). The results show the improved accuracy achieved by our models with respect to the competitors throughout the incremental steps.

TinyImageNet. Table 2 shows that our framework outperforms exemplar-based competitors and the SotA methods not using exemplars [43, 40]. In particular, our drift models yield superior performance w.r.t. SDC [40]. This is especially true when semantic and feature representation drifts are jointly taken into account, showing that they both individually model crucial and complementary information by model fusion, which is not fully captured by SDC [40].

CUB200-2011. Table 3 shows that non-exemplar methods provide quite low results, especially when the number of

Table 3. Per-class average top-1 accuracy (%).

Method	CUB200 5 Steps	CUB200 10 Steps	CUB200 20 Steps
Fine-tuning	10.93	7.10	4.63
Joint	74.67	74.67	74.67
EWC [16]	10.63	6.43	4.65
LwF [19]	26.40	13.65	7.89
PASS [43]	52.14	37.97	18.29
SDC [40]	52.30	38.30	18.17
Feat. Drift (GM-MLP)	55.87	50.67	31.36
Sem. Drift (GM-MLP)	56.51	47.89	32.50
Fusion† (GM-MLP)	56.20	52.07	36.67
Fusion (GM-MLP)	56.28	51.82	37.99
Feat. Drift (VM-VAE)	57.39	51.29	32.72
Sem. Drift (VM-VAE)	**57.34**	51.88	33.34
Fusion† (VM-VAE)	56.59	52.00	36.80
Fusion (VM-VAE)	56.97	**52.58**	**38.26**

† Without using the correlation objective [9] in fusion loss.

incremental steps is increased. Adopting the method proposed in [40] to compensate for modeling shift of prototypes using a softmax classifier seems to have no beneficial effect, showing that it fails to adequately model semantic drift in a fine-grained classification setup with high semantic similarity among classes. On the other end, our framework demonstrates to successfully capture model representation drift; by injecting up-to-date knowledge of old classes, in fact, we manage to more effectively mitigate catastrophic forgetting. Per-step accuracy values displayed in Fig. 4 corroborate accuracy results given in Table 3.

6.2. Ablation Study

Modeling drifts with GMs and VMs: Our framework enables implementation of drift models using different GMs and VMs. We studied the accuracy of a GM (MLP) and VM (VAE) for modeling different drifts and their fusion in Sec. 6.1. The results suggest that the accuracy of GMs and VMs depends on statistical sufficiency of data which affects *capacity* of f_θ and learned representations as follows:

- On the Cifar100 dataset, the GM (MLP) outperforms the VM (VAE) for smaller (*e.g.*, 5) steps, where more classes are observed at each step, compared to the larger (*e.g.* 20) steps. We conjecture that this result can be attributed to training models using statistically insufficient data representing all classes at each step.
- On TinyImageNet, containing larger images than Cifar100, the VM (VAE) performs on par with and slightly outperforms the GM (MLP) for smaller steps.
- On CUB200 containing the largest images, the VM (VAE) outperforms the GM (MLP) for all steps.

Analysing semantic drift: We study how the proposed framework captures and preserves semantic relationships between representation of old and new classes by modeling semantic drift. We express inter-class relationships by the Euclidean distance between prototypes (those estimated for old classes and computed over available data for new classes) and track the evolution of such measures over an incremental step. In Fig. 5, we report distance values computed at the beginning and end of an incremental step, together with their difference (distances are normalised along new-class axis). We notice how prototypes estimated by leveraging the modelled semantic drift tend to more effectively preserve their relationships, whereas keeping prototypes fixed (as in PASS) causes impairment of inter-class relationships as new representations are learned.

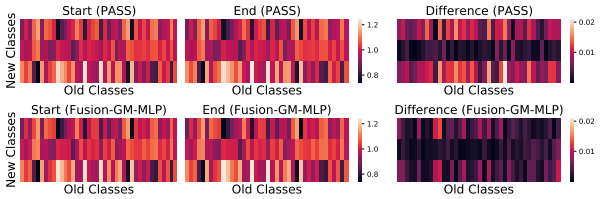

Figure 5. Normalised distance bw. estimated prototypes of classes seen at the first and second step, captured at the beginning (left) and end (mid) of the second step (CIFAR100, 20 steps). We report the absolute value of the difference of the two measures (right).

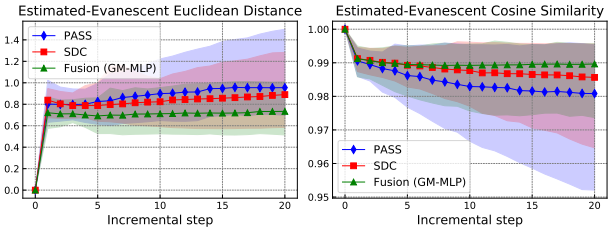

Figure 6. Average distance between estimated and evanescent prototypes of old classes (CIFAR100, 20 incremental steps).

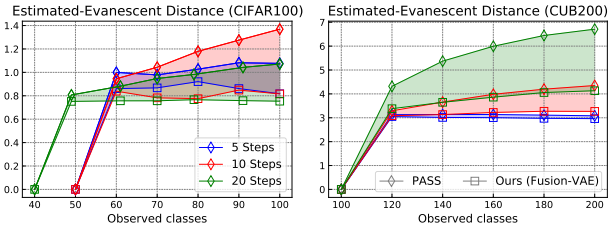

Figure 7. Average Euclidean distance between the estimated and evanescent prototypes of old classes.

Analysing feature drift of evanescent representations: We compute the Euclidean and cosine distances between estimated prototypes of old classes (*i.e.*, computed over training data and fixed [43], or updated by [40] or by drift models) and their reference (*i.e.*, evanescent) representations (computed over the test set) at each step (Fig. 6). The results show that our proposed methods can track more efficiently the trajectory of evanescent prototypes (in terms of geometric distances) compared to the SotA PASS and SDC methods, by modeling the evolution of the representations (*i.e.*, feature drift). In Fig. 7, we compare normalised distance between estimated and evanescent prototypes for different number of total steps, on CIFAR100 and CUB200. We observe that our methods always outperform PASS. This is particularly noticeable for the 20-step setup on CUB200,

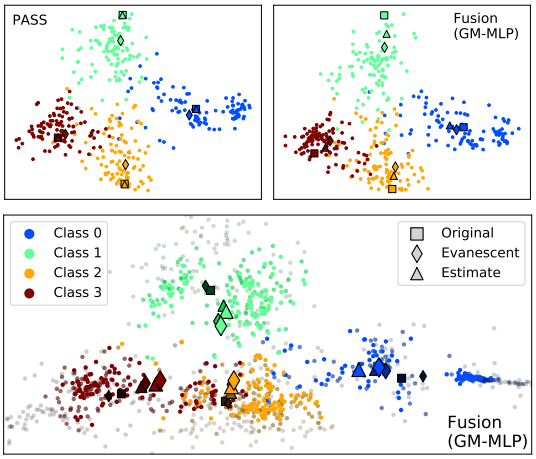

Figure 8. Feature representations of the first four learned classes (CIFAR100, 20 steps) extracted from samples of test set (dots), along with their prototypes computed over available training data (squares), over test data (diamonds) and estimated prototypes (diamonds). In the lower plot, decrease in transparency and increase in brightness indicate that representations are extracted at progressively increasing incremental steps (*i.e.*, at steps 0, 10 and 20).

where our approach jointly shows the largest improvement over SotA accuracy by $\sim 20\%$. We visualise 2D embeddings of feature vectors in Fig. 8 using Isomap [31]. We observe that our proposed methods estimate prototypes closer to their evanescent versions compared to PASS, following the trajectory of evanescent representations of old classes, without having access to training data of such classes.

Analysing how learned evanescent representations affect classification accuracy: We investigate the relationship between incremental accuracy and normalised distance between estimated and evanescent prototypes (Fig. 9). We notice that accuracy and prototype distance are negatively correlated, with similar trends for different methods. Thereby, by more accurately tracking and modeling evanescent old class prototypes, our methods yield superior performance compared to the SotA PASS and SDC methods.

Statistical analyses of representations: We first compute $p_F(c) = \exp(-||F-\pi_c)||_2/\zeta)/\sum_j \exp(-||F-\pi_j)||_2/\zeta)$, where $F \in \mathcal{F}_{old}$ is the representation of a test sample of $\mathcal{C}_{old}$ as extracted by the current feature extractor, $\{\pi_j\}_j$ are estimated prototypes of $\mathcal{C}_{old}$ and ζ is set to 0.1. We analyse the change of entropy (H) and cross-entropy (CE) of p_F across incremental steps in Fig. 10 and Fig. 11. We observe that our method provides higher H and smaller CE compared to PASS and SDC. This result suggests that information capacity of representations learned by our methods increases along with classification accuracy more compared to the SotA as models are incrementally trained.

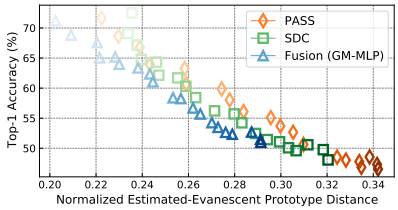

Figure 9. Relationship between top-1 accuracy (%) and normalised Euclidean distance between estimated and evanescent old-class prototypes. Each point depicts a single training phase, and the decrease in transparency indicates progressively increasing incremental steps. For each step, accuracy values have been averaged over all classes observed so far and distances are averaged over all past classes (CIFAR100, 20 steps).

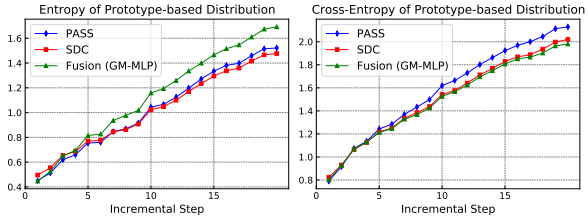

Figure 10. Avg. entropy H and CE of p_F (CIFAR100, 20 steps).

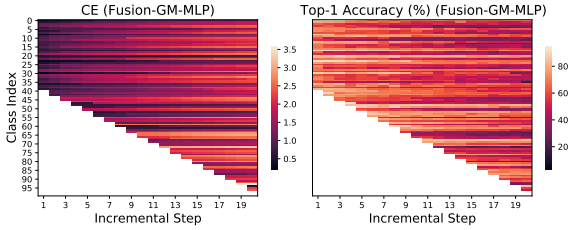

Figure 11. Class-wise average CE of p_F and top-1 accuracy computed at multiple incremental steps (CIFAR100, 20 steps).

7. Conclusion

We identify the evanescence of representations of old data as a cause of catastrophic forgetting in CIL. To employ evanescent representations in CIL and improve accuracy of classification models, we propose a new framework. First, we aim to model feature and semantic drift of representations. Then, by leveraging drift models, we are able to infer up-to-date representations of former tasks without storing any exemplar, and exploit them to preserve the past knowledge. We have evaluated our proposed framework on multiple CIL benchmarks. In the analyses, our proposed method achieved exemplar-free SotA accuracy. We further provided a detailed ablation study of geometric and statistical properties of the learned representations and drift models.

We believe that our proposed framework and approach for modeling drifts will lead to new research directions for CIL such as multi-level optimisation of hierarchical models with compositional loss functions of evolving representations, and analyses of their theoretical properties.

References

[1] Hongjoon Ahn, Jihwan Kwak, Su Fang Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. SS-IL: Separated softmax for incremental learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[3] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2, 6

[4] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co^{2}L: Contrastive continual learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[5] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[6] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 1

[7] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 5, 6

[8] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[9] Bobby He and Mete Ozay. Feature kernel distillation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 5, 6, 7

[10] 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 (CVPR)*, 2016. 6

[11] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. 2

[12] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6

[13] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[14] Metod Jazbec, Matt Ashman, Vincent Fortuin, Michael Pearce, Stephan Mandt, and Gunnar Rätsch. Scalable gaussian process variational autoencoders. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIStat)*, 2021. 3

[15] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 2

[16] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. 2, 6, 7

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[18] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[19] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2, 6, 7

[20] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[21] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the ELBO! A linear VAE perspective on posterior collapse. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3

[22] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual

learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. 1

[23] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 1

[24] Hadi Pouransari and Saman Ghili. Tiny imagenet visual recognition challenge. In *CS231N course, Stanford Univ., Stanford, CA, USA*, 2014. 5

[25] 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 (CVPR)*, 2017. 2, 6

[26] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[27] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[28] 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 International Conference on Computer Vision (ICCV)*, 2021. 2

[29] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[30] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 6

[31] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 8

[32] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[33] Catherine Wah, Peter Welinder Steve Branso and, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 5

[34] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[35] Guile Wu, Shaogang Gong, and Pan Lid. Striking a balance between stability and plasticity for class-incremental learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[36] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[37] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019. 2

[38] Yaniv Yacoby, Weiwei Pan, and Finale Doshi-Velez. Failure modes of variational autoencoders and their effects on downstream tasks. In *ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning*, 2021. 3

[39] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 6

[40] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6, 7

[41] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 2

[42] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *CoRR*, abs/1706.02262, 2017. 6

[43] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 6, 7