# A Two-Stream Continual Learning System With Variational Domain-Agnostic Feature Replay

Qicheng Lao, Xiang Jiang, Mohammad Havaei, and Yoshua Bengio

***Abstract*— Learning in nonstationary environments is one of the biggest challenges in machine learning. Nonstationarity can be caused by either task drift, i.e., the drift in the conditional distribution of labels given the input data, or the domain drift, i.e., the drift in the marginal distribution of the input data. This article aims to tackle this challenge with a modularized two-stream continual learning (CL) system, where the model is required to learn new tasks from a support stream and adapted to new domains in the query stream while maintaining previously learned knowledge. To deal with both drifts within and across the two streams, we propose a variational domain-agnostic feature replay-based approach that decouples the system into three modules: an *inference module* that filters the input data from the two streams into domain-agnostic representations, a *generative module* that facilitates the high-level knowledge transfer, and a *solver module* that applies the filtered and transferable knowledge to solve the queries. We demonstrate the effectiveness of our proposed approach in addressing the two fundamental scenarios and complex scenarios in two-stream CL.**

***Index Terms*— Continual learning (CL), generative replay, learning systems, nonstationary environments, unsupervised domain adaptation, variational inference.**

## I. Introduction

ONE of the biggest challenges in machine learning is to learn in nonstationary environments, in which the underlying data distribution [i.e., the joint distribution of the input data and labels $P(X, Y)$] changes over time, also referred to as *concept drift* in previous literature [1], [2]. One source of the change can be from the drift in the conditional distribution of labels given the input data [i.e., $P(Y|X)$], often resulting from the change in the task definition, where the predictive function from the input space to label space may vary. Therefore, this drift in $P(Y|X)$ can be named *task drift*. Another source of the distribution change is the drift in the marginal distribution of the input data [i.e., $P(X)$], which we name *domain drift* here, with one additional assumption that $P(Y|X)$ remains the same, in lieu of the aforementioned task drift. The domain drift problem has been identified in many practical scenarios that tackle stream data, often with different terminologies, e.g., virtual concept drift [3], [4], feature change [5], and many others summarized in [6] and [7].

Current continual learning (CL) systems [8]–[14] proposed for nonstationary environments often assume a single nonstationary data stream for training [see Fig. 1(a)], without considering the possible domain drifts between the training data and test data. In real-world applications, however, we normally have two streams of data arriving sequentially, i.e., a training or *support* stream (with labels) and a test or *query* stream (without labels), where both task drift and domain drift can be present within and/or across the two streams [see Fig. 1(b)]. For example, in the healthcare sector, the sequentially arrived query data from different hospitals may be subject to domain drift due to the acquisition system settings, patient demographics, and so on. As such, the CL system with a single support stream, lacking the consideration of the potential domain drift across the support and query stream, would be at risk to provide predictive service to the queries since the previous knowledge accumulated in the system is not necessarily reusable on the new queries.

In this article, we propose a modularized two-stream CL system [see Fig. 1(b)] for learning in nonstationary environments, taking into account the possibility of having both task and domain drifts within and across the two streams (support and query). The two streams are motivated toward practical use cases and also designed to separate the concerns of domain drift across streams from the drifts within streams. Ideally, the system should continuously accumulate knowledge from two perspectives: first, the knowledge should be *filtered* to be domain-agnostic for addressing the domain drift between the support stream and query stream; second, the knowledge should be modularized in a *transferable* form that can be revisited at any time as needed, either to avoid catastrophic forgetting [15] by retaining competence on previously seen environments, or as part of domain-independent generic prior knowledge to help solve the current query of interest.

To achieve the above, we propose a variational domain-agnostic feature replay-based approach, which decouples the two-stream CL system into three modules: 1) the *inference module* that transforms the input data into filtered knowledge that is represented by variational domain-agnostic features; 2) the *generative module* as a

Manuscript received 3 August 2020; revised 5 January 2021; accepted 27 January 2021. Date of publication 3 March 2021; date of current version 2 September 2022. This work was supported by MEDTEQ grant 10-30 IA Multicentriq. *(Corresponding author: Qicheng Lao.)*

Qicheng Lao is with West China Biomedical Big Data Center, West China Hospital, Sichuan University, Chengdu 610041, China, also with the Montreal Institute for Learning Algorithms (MILA), Université de Montréal, Montreal, QC H2S 3H1, Canada, and also with Imagia, Montreal, QC H2S 3G9, Canada (e-mail: qicheng.lao@gmail.com).

Xiang Jiang and Mohammad Havaei are with Imagia, Montreal, QC H2S 3G9, Canada.

Yoshua Bengio is with the Montreal Institute for Learning Algorithms (MILA), Université de Montréal, Montreal, QC H2S 3H1, Canada.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3057453.

Digital Object Identifier 10.1109/TNNLS.2021.3057453

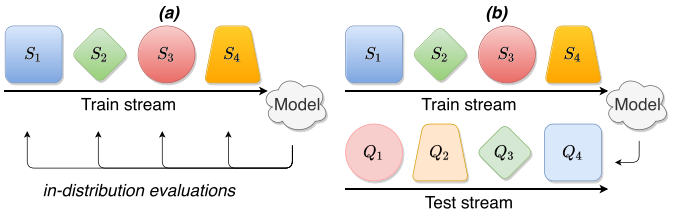



Fig. 1. (a) Comparison of current CL system and the (b) proposed two-stream CL system. $S_i$ represents a training (i.e., support) set, while $Q_j$ represents a test (i.e., query) set. We want to maintain our knowledge on $S$ and transfer to $Q$. The changes in *shape* and *color* in the streams, respectively, denote the task drift and domain drift in nonstationary environments.

means to enable knowledge transfer by replaying the learned knowledge sampled from feature distributions, similar to the high-level replay found in animal brains [16]; and 3) the *solver module* that applies the filtered and transferable knowledge to solve the queries. Intuitively, the synergy between the inference module and solver module creates a knowledge information bottleneck, where the inference module minimizes the mutual information between the input and the filtered knowledge, while the solver module maximizes the mutual information between the filtered knowledge and the labels.

It is also worth mentioning that, unlike many previous works in CL [10], [11], [17] that address catastrophic forgetting by using unfiltered input data (real or generated) to simulate a stationary environment, we emphasize the high-level knowledge transfer for selective remembering, similar to the concept learning proposed in a concurrent work [14]. This is in line with the fact that, although the human brain has a huge amount of capacity, forgetting seems an evolutionarily correct mechanism. What we need is some form of filtered and transferable knowledge at the abstract level, such as a textbook or dictionary, rather than data-level raw information.

We validate the proposed approach on two fundamental scenarios and a more generic scenario in two-stream CL (as illustrated in Fig. 2), by enforcing the nonstationarity in the task space or/and in the domain space. The experimental results demonstrate the effectiveness of the proposed approach on addressing both task drift and domain drift in all scenarios. Moreover, we show the possibility of using the generative module independently as a means of data augmentation toward better query performance and discuss the relationship between domain similarity and the performance gain obtained from reusing the generative module.

The contributions of this article are fourfold.

1) We pinpoint the limitation of current CL systems in neglecting the domain drift between the support stream and the query stream for practical usage and introduce a two-stream CL system.
2) We propose a variational domain-agnostic feature replay-based approach, which modularizes the two-stream CL system and demonstrate its effectiveness for addressing both task and domain drifts within and across the two streams in nonstationary environments.
3) We give a theoretical analysis on the error bound of the query stream in two-stream CL with the proposed approach.

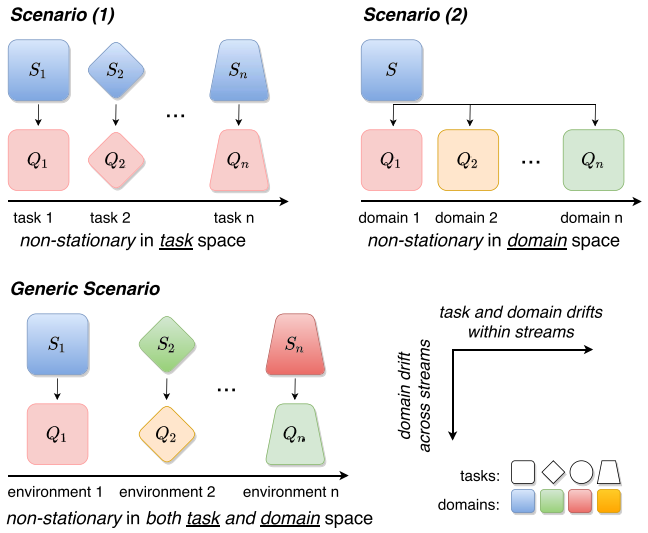



Fig. 2. Scenarios in two-stream CL. Scenario (1) and Scenario (2) compose the two fundamental elements in building up most of the complex learning scenarios in nonstationary environments. Notably, an environment in the generic scenario is defined by the combination of task and domain, where the task and domain drifts within streams can be additive. In Scenario (1), the *shape* represents different tasks changes, while, in Scenario (2), the *color* represents different domains changes.

4) We show the additional benefit of the modularized two-stream CL system in providing the generative module independently as a means of data augmentation toward better query performance.

## II. Two-Stream Continual Learning

Let $S$ denote the support stream and $Q$ the query stream, both composed of sequentially arriving data sets, i.e., $S_i = \{(\boldsymbol{x}_{S_i}^{(n)}, y_{S_i}^{(n)})\}_{n=1,\ldots,N_{S_i}}$ for $i \in [1, 2, \ldots, T_S]$, and $Q_j = \{\boldsymbol{x}_{Q_j}^{(n)}\}_{n=1,\ldots,N_{Q_j}}$ for $j \in [1, 2, \ldots, T_Q]$, and the goal of a two-stream CL system is to maintain the knowledge on $S$ and transfer to $Q$. Since the data streams can be acquired from any acquisition systems for any tasks, they may be subject to both task drift and domain drift.

The task drift can be illustrated in Scenario (1) [as shown in Fig. 2 (top left)] with nonstationarity in the task space, where for the support stream $S$, the predictive mapping from the input data $X$ to class labels $Y$ can be continuously changing, i.e., $P_{S_i}(Y|X) \neq P_{S_j}(Y|X)$ for $i \neq j$, therefore resulting in different tasks, and the same nonstationarity also applies to the query stream $Q$. The challenge here is that the model needs to not only retain knowledge learned from past data for solving previous tasks (or queries) but also address the domain shift between the query stream and the support stream since $P_Q(X) \neq P_S(X)$. The latter problem is well known as domain adaptation[1] [18], [19], and in order for the query data to be solvable, a typical assumption is that there exists an unobserved latent variable $X_0$ such that $P_Q(Y|X_0) = P_S(Y|X_0)$ holds [19].

On the other hand, we can also assume nonstationarity in the domain space, where we have continuously arriving data in the query stream, each coming from a different domain

[1]In this work, we focus on unsupervised domain adaptation since the data in the query stream are unlabeled.

(i.e., $P_{Q_i}(X) \neq P_{Q_j}(X)$ for $i \neq j$), as shown in Scenario (2) [see Fig. 2 (top right)]. This scenario also represents a group of common use cases in practice, with the assumption that the underlying predictive mechanism remains unchanged (i.e., $P_{Q_i}(Y|X) = P_{Q_j}(Y|X)$ for $i \neq j$). As a result, the model is required to be continuously updated to bridge the new domain shift for the current query while maintaining the ability to solve previously seen queries. Note that, in both Scenario (1) and Scenario (2), we assume the usual restriction in CL that the data seen in previous environments are hidden, and we only have access to data in the current environment.

The above two scenarios compose the fundamental elements in building up most of the complex learning scenarios [illustrated as the generic scenario in Fig. 2 (bottom)] in nonstationary environments. Therefore, in this article, we first consider to tackle these two fundamental scenarios (see Sections IV-C and IV-D). In addition, to show the generality of the proposed approach on complex scenarios, we also make an example scenario that is derived from combining Scenario (1) and Scenario (2), where both task drift and domain drift are present within both the support and query streams (see Section IV-E).

Moreover, in the two-stream CL system, we are interested in continuously solving the unlabeled queries, whose solvability also depends on whether the corresponding support data are available; therefore, to make sure that the queries are actually solvable, we assume that, for each $Q_i$, the corresponding $S_i$ (i.e., $S_i$ shares the same labeling function with $Q_i$) arrives earlier than $Q_i$. In practice, if $S_i$ arrives later, the adaptation of $Q_i$ can be held until $S_i$ is available. In other words, we synchronize the tasks across the two streams.

## III. Approach

Here, we present the proposed variational domain-agnostic feature replay-based approach. We propose to tackle the two-stream CL by decoupling the system into three modules (see Fig. 3).

1) *Inference module* (see Section III-A) that uses variational inference to learn a domain-agnostic feature distribution for addressing the domain drift between the support stream and query stream.
2) *Generative module* (see Section III-B) that allows replay by sampling from previously learned domain-agnostic feature distributions. The sampled features can be used either as our filtered knowledge to reinforce the solver on remembering the knowledge required to solve the previous queries in nonstationary environments or as a data augmentation tool to augment the knowledge about the current query.
3) *Solver module* (see Section III-C) that focuses only on the downstream task of interest, provided that the inferred or generated data from the above two modules are already domain-agnostic.

This decoupling aligns with the two-stream CL that aims to separate the concerns of domain drift from task drift in complex nonstationary environments and also brings up the possibility of transferring domain-agnostic knowledge among different environments, which, to the best of our knowledge, has not yet been investigated. As the first step, we will show later in our experiments that this is possible with our proposed approach. In Section III-D, we give a theoretical analysis on the error bound of the query stream in two-stream CL with the proposed approach. Algorithm 1 presents the end-to-end training of the three modules.

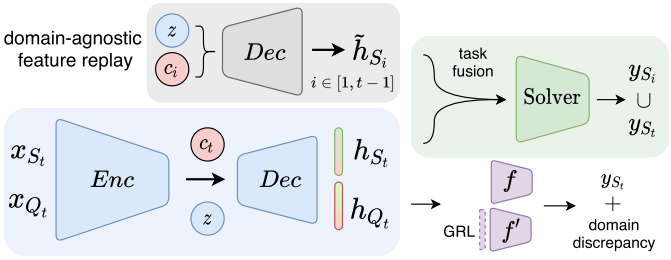



Fig. 3. Decoupling the two-stream CL system into three modules: inference module (blue), generative module (gray), and solver module (green). The inference module infers the domain-agnostic features for the current environment through variational inference; the generative module generates domain-agnostic feature samples for previously seen environments; and the solver module is able to continuously solve all seen environments.

### A. Variational Domain-Agnostic Feature Inference

In this module, we are given an environment at time step $t$ with some labeled input data $\boldsymbol{x}_{S_t}$ from the support stream $S$ and unlabeled query data $\boldsymbol{x}_{Q_t}$ from the query stream $Q$. We aim to map both support and query data into a shared stochastic feature space using a mapping function $\Psi$: $\boldsymbol{x} \to \boldsymbol{h}$ such that the following two conditions hold: 1) $\boldsymbol{h}$ maximally preserves the necessary information to predict the label $y$ and 2) the domain discrepancy between the support and query stream on $\boldsymbol{h}$ is minimum. The first condition respects the task of the current environment, while the second condition deals with the domain drift across the two streams. Note that, for simplicity of notation, we ignore the subscripts for the random variables here.

To achieve this, we use stochastic variational inference [20] through first encoding $\boldsymbol{x}$ into a latent variable $\boldsymbol{z}$, which is then decoded into $\boldsymbol{h}$. We also introduce a hierarchical conditioning factor $c$ (conditioned on the environment and its associated classes) in order to enable the conditional generation in the generative module (see Section III-B) for different environments, similar to [21]. The two conditions on $\boldsymbol{h}$ can be then formulated as

$$\max_{\Psi, f} \ \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x},c)}[\log p(y|\boldsymbol{h})p(\boldsymbol{h}|\boldsymbol{z},c)] - \mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x},c)||p(\boldsymbol{z}))$$
$$\text{s.t. } d_{\mathcal{H}\Delta\mathcal{H}}(H_{S_t}, H_{Q_t}) \leq \Lambda_t \tag{1}$$

where $H_{S_t}$ and $H_{Q_t}$ are the marginal feature distributions for the support domain $S_t$ and query domain $Q_t$, respectively, and

$$d_{\mathcal{H}\Delta\mathcal{H}}(H_{S_t}, H_{Q_t}) = 2 \sup_{f, f' \in \mathcal{F}} |\mathbb{E}_{\boldsymbol{h}\sim H_{Q_t}}[f(\boldsymbol{h}) \neq f'(\boldsymbol{h})] - \mathbb{E}_{\boldsymbol{h}\sim H_{S_t}}[f(\boldsymbol{h}) \neq f'(\boldsymbol{h})]| \tag{2}$$

is the $\mathcal{H}\Delta\mathcal{H}$ divergence that measures domain discrepancy with $f$ and $f'$ denoting two classifier hypotheses in the

**Algorithm 1** End-to-End Training of the Three Modules

**Input:** support stream $S$ and query stream $Q$;
a mapping function $\Psi$ and two hypothesis classifiers $f$, $f'$;
solver classifier $f_{solver}(\cdot;\boldsymbol{\theta})$
**while** query $Q_t$ is **not** processed **do**
  *# sample $Q_t$ and the corresponding support data $S_t$*
  $S_t \leftarrow \{(\boldsymbol{x}_{S_t}^{(n)}, y_{S_t}^{(n)})\}_{n=1,\ldots,N_{S_t}}$
  $Q_t \leftarrow \{\boldsymbol{x}_{Q_t}^{(n)}\}_{n=1,\ldots,N_{Q_t}}$
  *# inference*
  $\boldsymbol{h}_{S_t} \overset{\Psi}{\leftarrow} \boldsymbol{x}_{S_t}$, $\boldsymbol{h}_{Q_t} \overset{\Psi}{\leftarrow} \boldsymbol{x}_{Q_t}$
  optimize $\min_{\Psi,f} \max_{f'} \big(\epsilon + \beta d_{\mathcal{H}\Delta\mathcal{H}}(\boldsymbol{h}_{S_t}, \boldsymbol{h}_{Q_t})\big)$*
  infer domain-agnostic features $\boldsymbol{h}_t \leftarrow \boldsymbol{h}_{S_t} \approx \boldsymbol{h}_{Q_t}$,
  *# generative feature replay*
  **for** $i$ in $[1, t-1]$ **do**
    sample domain-agnostic features $\tilde{\boldsymbol{h}}_i$
  **end for**
  *# solver*
  optimize $\min_{\boldsymbol{\theta}} \Big(\mathbb{E}_{\boldsymbol{h}_t}[\log p(y_{S_t}|\boldsymbol{h}_t)] + \sum_{i=1}^{t-1} \mathbb{E}_{\tilde{\boldsymbol{h}}_i}[\log p(y_{S_i}|\tilde{\boldsymbol{h}}_i)]\Big)$
**end while**

*$\epsilon$ is the support domain error and $d_{\mathcal{H}\Delta\mathcal{H}}$ is the feature divergence between $S_t$ and $Q_t$.

hypothesis space $\mathcal{F}$ [22]. $\Lambda_t$ is a threshold constant for the $\mathcal{H}\Delta\mathcal{H}$ divergence between $H_{S_t}$ and $H_{Q_t}$. Note that we denote $q()$ as a variational approximation to the true distribution $p()$.

Solving (1) is equivalent to minimizing the following objective function:

$$\mathcal{L} = \epsilon + \beta d_{\mathcal{H}\Delta\mathcal{H}} \tag{3}$$

where $\epsilon$ denotes the error of satisfying the first condition in (1), and $\beta$ is a Lagrange multiplier. Note that, although derived from a new perspective, our objective function has a similar form as the upper bound of the target domain error[2] in domain adaptation theory [22]. In our case, we focus on constructing the domain-agnostic feature space that can be sampled from later on in the generative replay module (see Section III-B). Following the works in adversarial domain adaptation [23]–[28], we also minimize the domain disparity discrepancy via a minimax optimization process:

$$\begin{aligned} &\min_{\Psi,f} \ \epsilon + \beta d_{\mathcal{H}\Delta\mathcal{H}} \\ &\max_{f'} \ d_{\mathcal{H}\Delta\mathcal{H}}. \end{aligned} \tag{4}$$

### B. Generative Domain-Agnostic Feature Replay

Assuming that a variational mapping from the input space to the domain-agnostic feature space has been learned, we can then use the decoder function $g$ to conditionally generate domain-agnostic features based on the conditioning factor $c$ for each learned environment

$$\tilde{\boldsymbol{h}} = g(\boldsymbol{z}, c), \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{5}$$

[2]Equivalent to the query domain error in this article.

Since $\tilde{\boldsymbol{h}}$ is domain-agnostic, it represents the knowledge of interest filtered from a previously seen environment, and this knowledge can be transferred through the generative replay process, to further guide the training of the solver module (see Section III-C). Therefore, at each time step $t > 1$, even when the real data seen in previous environments ($i \in [1, t-1]$) are not available, we can still replay $\tilde{\boldsymbol{h}}_i$ to address the catastrophic forgetting by regularizing the solver to remember how to solve previous queries. This is similar to generative replay or pseudorehearsal [29], which has been widely used as an effective approach to addressing catastrophic forgetting in CL. However, in most works [11], [30], the original input data $\boldsymbol{x}$ are replayed. While $p(\boldsymbol{x})$ is often difficult to approximate especially when $\boldsymbol{x}$ is in high dimension, our feature replay can be viewed as a means of high-level knowledge replay. Furthermore, it is also a filtered knowledge replay with the elimination of the domain drift between the support and query stream, and the knowledge filtration is guided by the inference module described above.

In addition to addressing catastrophic forgetting, our filtered feature replay can also act as an alternative approach of transferring part of domain-independent generic prior knowledge among different environments. More specifically, the domain-agnostic features $\tilde{\boldsymbol{h}}_i$ resulting from solving a previous query $Q_i$ can be used as augmented data in addition to the inferred $\boldsymbol{h}_t$ when solving the current query $Q_t$, given the assumption that $Q_t$ shares some domain similarity with $Q_i$. We discuss more details in an empirical study (see Section IV-F).

### C. Solver Module

Our solver is a unified model that continuously integrates knowledge filtered from seen environments and is designed to solve all the queries in the query stream. Since the solver operates on the domain-agnostic feature space, i.e., there is no domain shift between the support and query data in the feature space, it is, therefore, able to solve all the unlabeled queries in the query stream, once trained on the support stream.

At time step $t$, our solver sees both the inferred features $\boldsymbol{h}_t$ from the current environment, i.e., features inferred from the input data $\boldsymbol{x}_{S_t}$ in the support stream, and the generated features $\tilde{\boldsymbol{h}}_i$ from the generative module for previously seen environments. Let $\boldsymbol{\theta}$ be the parameters of the solver, and the objective can be given as

$$\min_{\boldsymbol{\theta}} \ \Bigg\{ \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x}_{S_t}, c_t), \boldsymbol{h}_t \sim p(\boldsymbol{h}|\boldsymbol{z}, c_t)}[\log p(y_{S_t}|\boldsymbol{h}_t)] + \sum_{i=1}^{t-1} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), \tilde{\boldsymbol{h}}_i \sim p^*(\boldsymbol{h}|\boldsymbol{z}, c_i)}[\log p(y_{S_i}|\tilde{\boldsymbol{h}}_i)] \Bigg\} \tag{6}$$

where $p^*$ denotes feature distributions learned from previous environments and stored as snapshot decoders in the system.

Note that our solver is independent of the hypothesis classifier $f$ used in the inference module. Although $f$ also aims to predict the class label $y_{S_t}$ given a domain-agnostic

feature $\boldsymbol{h}_t$, it cannot replace the role of the solver in the system for solving previous queries even with the replay of $\tilde{\boldsymbol{h}}_i$. We will show later in the ablation experiment (see Section IV-C1) that the feature replay of $\tilde{\boldsymbol{h}}_i$ on $f$ can, in fact, interfere with the adversarial learning when minimizing the domain disparity discrepancy between the support and query data for the current environment, thus resulting in impaired performance. This is due to the possible task conflicts of the features from different environments, i.e., despite being domain-agnostic, the features for different tasks could conflict with each other for not being task-aware. Therefore, our solver module is indispensable as a way to fuse the domain-agnostic features into task hierarchies by the conditioning factors for smoothly integrating knowledge learned from all seen environments. We name this integration process as *task fusion*, e.g., to fuse different tasks with expanded label space.

### D. Theoretical Analysis

In this section, we analyze the theoretical guarantee for the two-stream CL system with our proposed approach based on the variational domain-agnostic feature replay.

*Theorem 1 [22]:* Given a source domain and a target domain, with the input data distributions $X_{\text{source}}$ and $X_{\text{target}}$, we have the target domain error bounded by

$$\varepsilon_{\text{target}} \leq \varepsilon_{\text{source}} + d_{\mathcal{H}\Delta\mathcal{H}}(X_{\text{source}}, X_{\text{target}}) + C^* \tag{7}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}(X_{\text{source}}, X_{\text{target}})$ is the $\mathcal{H}\Delta\mathcal{H}$ divergence that measures domain discrepancy between the source and target domains on the input data, and $C^*$ is the error of an optimal classifier for both source and target domains.

*Corollary 2:* For each time step $i$ in the support stream $S$ and query stream $Q$, let $\lambda_i = d_{\mathcal{H}\Delta\mathcal{H}}(H_{S_i}, H_{Q_i})$ be the domain discrepancy of the feature distributions $H_{S_i}$ and $H_{Q_i}$. The error of the query domain $Q_i$ is bounded by

$$\varepsilon_{Q_i} \leq \varepsilon_{S_i} + \lambda_i + C_i^* \tag{8}$$

where $C_i^* = \min_{\boldsymbol{\theta}}(\varepsilon_{S_i} + \varepsilon_{Q_i})$ is the error of an optimal solver for both the support domain $S_i$ and the query domain $Q_i$.

*Theorem 3:* Let $\lambda_i$ be the domain discrepancy of the marginal feature distributions $H_{S_i}$ and $H_{Q_i}$ measured by the $\mathcal{H}\Delta\mathcal{H}$ divergence, i.e., $\lambda_i = d_{\mathcal{H}\Delta\mathcal{H}}(H_{S_i}, H_{Q_i})$, and $P_r^{(i)}$, $P_g^{(i)}$, respectively, denote the conditional distributions of labels $y_{S_i}$ given the real features $H_{S_i}$ and generated features $\tilde{H}_{S_i}$. The total error of the query stream $\varepsilon_Q$ at time step $t$ is bounded by

$$\varepsilon_Q \leq \sum_{i=1}^{t}(\varepsilon_{S_i} + \lambda_i) + \sum_{i=1}^{t-1}\text{KL}\big(P_g^{(i)} \,\big\|\, P_r^{(i)}\big) + C^* \tag{9}$$

where $C^* = \min_{\boldsymbol{\theta}} \sum_{i=1}^{t}(\varepsilon_{S_i} + \varepsilon_{Q_i})$ is the error of an optimal solver for both support and query streams.

With the setup introduced in Corollary 2, we now prove Theorem 3.

*Proof:* At time step $t$, the error of previous query domains $Q_i$ for $i \in [1, t-1]$ is estimated by $\hat{\varepsilon}_{Q_i}$ since we use $\tilde{H}_{S_i}$ to approximate $H_{S_i}$ during the training of the solver. The total error of the query stream is

$$\begin{aligned}\varepsilon_Q &= \varepsilon_{Q_t} + \sum_{i=1}^{t-1}\hat{\varepsilon}_{Q_i} \\ &\leq \varepsilon_{S_t} + \lambda_t + \sum_{i=1}^{t-1}(\hat{\varepsilon}_{S_i} + \lambda_i) + C^*.\end{aligned} \tag{10}$$

On the other hand, for each support domain $S_i$, the KL divergence between the real and generated conditional distributions of labels $y_{S_i}$ given the features, i.e., $\text{KL}(P_g^{(i)}(y|\boldsymbol{h})||P_r^{(i)}(y|\boldsymbol{h}))$, can be interpreted as

$$\begin{aligned}&\text{KL}\big(P_g^{(i)}(y|\boldsymbol{h}) \,\big\|\, P_r^{(i)}(y|\boldsymbol{h})\big) \\ &= \sum_y P_g^{(i)}(y|\boldsymbol{h}) \log \frac{P_g^{(i)}(y|\boldsymbol{h})}{P_r^{(i)}(y|\boldsymbol{h})} \\ &= \mathbb{E}\left[\log \frac{P_g^{(i)}(y|\boldsymbol{h})}{P_r^{(i)}(y|\boldsymbol{h})}\right] \\ &= \mathbb{E}\big[\log P_g^{(i)}(y|\boldsymbol{h})\big] - \mathbb{E}\big[\log P_r^{(i)}(y|\boldsymbol{h})\big] \\ &= \hat{\varepsilon}_{S_i} - \varepsilon_{S_i}.\end{aligned} \tag{11}$$

Therefore, the error for each support domain $S_i$ at the time step $t$ is estimated by

$$\hat{\varepsilon}_{S_i} = \varepsilon_{S_i} + \text{KL}\big(P_g^{(i)} \,\big\|\, P_r^{(i)}\big). \tag{12}$$

Combining 10 and 12, the total error of the query stream $\varepsilon_Q$ at time step $t$ is

$$\begin{aligned}\varepsilon_Q &\leq \varepsilon_{S_t} + \lambda_t + \sum_{i=1}^{t-1}(\hat{\varepsilon}_{S_i} + \lambda_i) + C^* \\ &= \varepsilon_{S_t} + \lambda_t + \sum_{i=1}^{t-1}\big(\varepsilon_{S_i} + \text{KL}\big(P_g^{(i)} \,\big\|\, P_r^{(i)}\big) + \lambda_i\big) + C^* \\ &= \sum_{i=1}^{t}(\varepsilon_{S_i} + \lambda_i) + \sum_{i=1}^{t-1}\text{KL}\big(P_g^{(i)} \,\big\|\, P_r^{(i)}\big) + C^*.\end{aligned} \tag{13}$$

□

As reflected in Theorem 3, the error bound of the query stream has different components, which can also be explained by our three different modules. For example, $\lambda_i$ represents how well our inference module has learned a domain-agnostic feature space; the KL term measures the degree of the generative module approximating the real feature distributions since we assume the previous real data is without access; and finally, $\varepsilon_{S_i}$ evaluates the solver's performance on the support stream data, and $C^*$ is the capacity of the solver in finding an optimal solution for both streams.

## IV. Experiments

We validate our proposed approach for the modularized two-stream CL system on two benchmark data sets that are used to evaluate the domain adaptation performance since our primary focus in this work is to continuously adapt the support stream to the query stream in nonstationary environments. To simulate the two streams of data, we introduce nonstationarity into the data sets. For scenario (1), we split the data

sets into sequentially arriving tasks based on the class labels to simulate the task drift in nonstationary environments, and we consider one domain as the support stream data with labels and another domain as the query stream data without labels; for scenario (2), we choose one domain as the support stream and consider the remaining domains in the data set as sequentially arriving queries in the query stream so that the domain drift is present both within and across the two streams.

### A. Setup

*1) Data Sets:* We transform the office data sets into nonstationary data sets by task split. Office-31 [31] has three domains: Amazon (A), DSLR (D), and Webcam (W), which, in total, contains 31 classes and 4652 images. We split the data set into five tasks, with six classes in the first four tasks and seven classes in the last task (task split details in Appendix VIII). Office-Home [32] is a more challenging data set that contains 65 classes and 15 500 images in four distinct domains: Artistic images (Ar), Clip art (Cl), Product images (Pr), and Real-World images (Rw). Similarly, we split the data set into 13 tasks, each with five classes (task split details in Appendix IX).

*2) Model Architectures:* We adopt ResNet [33] models pretrained on ImageNet [34] as part of our encoder in the inference module, e.g., ResNet-34 for the Office-31 data set and ResNet-50 for the Office-Home data set. The extracted features are used to infer the latent code with one additional linear layer. Our decoder consists of a linear layer, a ReLU layer, and a batch norm layer. The dimension of the output domain-agnostic features is 1024 for Office-31 and 2048 for Office-Home. The solver module is a two-layer neural network with a width of 1024. We fine-tune the pretrained ResNet model used in Scenario (2) in order to learn better class representations since more classes are involved within a single task compared to Scenario (1) where the extracted features directly from the ResNet model are sufficient enough to represent the classes. We follow the same architecture choices as in [28] for the two hypothesis classifiers $f$ and $f'$, i.e., two-layer neural networks.

*3) Optimization:* We use margin disparity discrepancy (MDD) [28] in our inference module for minimizing the domain disparity discrepancy. Two optimizers are used: the Adam optimizer for the inference module with the learning rate 1e-4 and the SGD optimizer for the two hypothesis classifiers ($f$ and $f'$) and the solver module, with the Nesterov momentum of 0.9 and the weight decay of 5e-4. The initial learning rate of the SGD optimizer is set to 4e-4 for Scenario (1) and 4e-3 for Scenario (2). We use the gradient reversal strategy [35] for the minmax optimization (4), and the training scheduler for the coefficient in the gradient reversal layer is defined by

$$\text{coeff} = 2.0 \times \frac{0.3}{1.0 + \exp\left(-\frac{i}{2000}\right)} - 0.3 \tag{14}$$

where $i$ is the iteration step number. The three modules (the inference module, generative module, and solver module) are trained end-to-end with 1k iteration steps for each task in the split Office-31 and 5k iteration steps for each task in the split Office-Home. We warm-up the inference module by fine-tuning it for the first 500 iteration steps while fixing other modules.

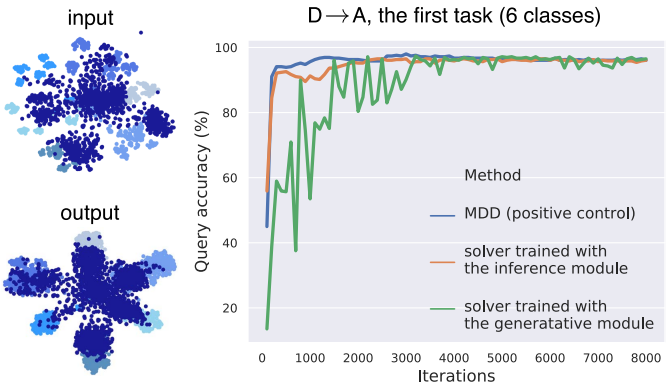



Fig. 4. Left: comparison of features representing the support data (in *light* colors; each light color represents a different class in Office-31, domain D) and the unlabeled query data (in *dark* color; domain A), before (left top) and after (left bottom) the inference module; right: the performance comparison of the solver trained with the generated features from the generative module to that trained with the real features from the inference module (domain D as the support), evaluated on the query data (domain A as the query).

### B. Evaluating the Features in the Two-Stream System

We first evaluate the features in the two-stream CL system from two perspectives: 1) whether the features from the inference module can be domain-agnostic representations of filtered knowledge and 2) whether the generated features from the generative module can be a functional replacement of the real data for high-level knowledge transfer, i.e., whether we can potentially use the proposed generative feature replay to facilitate the solver in remembering previously learned knowledge. As shown in Fig. 4 (left), the output features from the inference module are aligned between the support data and the unlabeled query data in a classwise manner [see Fig. 4 (left bottom)] compared to the shifted distributions of the raw features directly from a ResNet model pretrained on ImageNet [see Fig. 4 (left top)]. This indicates that the features from the inference module are indeed domain-agnostic between the two streams. To address the second question, we train a solver with the generated features from the generative module and evaluate them on the real features. We show in Fig. 4 (right) that, although slower in the convergence, the solver trained with the generated features can predict the class labels and the solver trained with real features from the inference module, suggesting that the feature replay approach is effective in approximating the real features for the downstream solver. It is also worth mentioning that the introduction of the inference module does not impair the domain adaptation performance, with respect to the MDD method [28] that is without the inference module, as a positive control [see Fig. 4 (right)].

### C. Scenario (1): Nonstationarity in Tasks

Having shown the merits of the features from both the inference and generative modules in Fig. 4, we now use

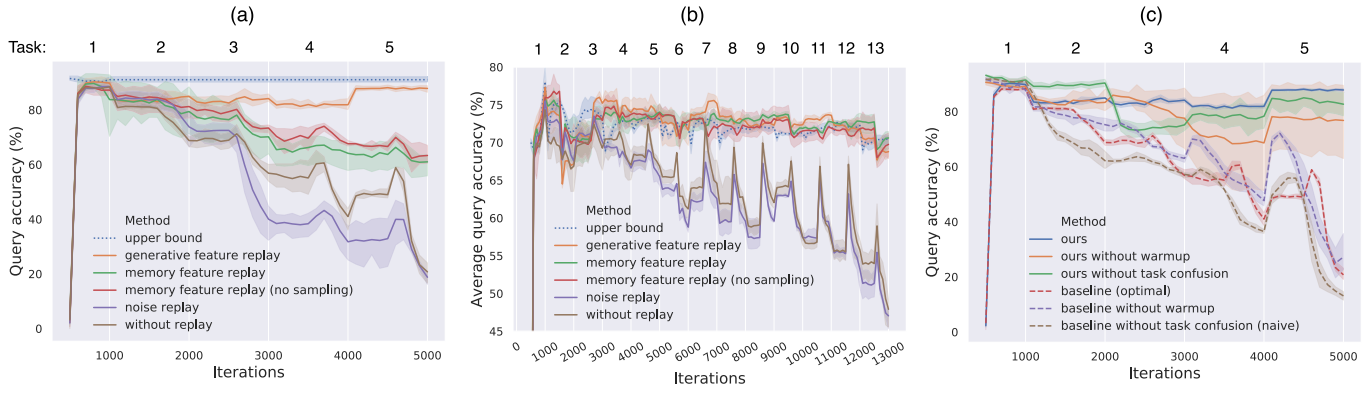



Fig. 5. (a) Query accuracy of the first task during sequential training (five tasks, D → A, split Office-31) [first task only (split Office-31)]. (b) Average query accuracy of all learned tasks during sequential training (13 tasks, Ar → Cl, split Office-Home) [average of all learned tasks (split Office-Home)]. (c) Ablation study on different components and strategies of our proposed approach [Ablation study (split Office-31)]. The results show that our proposed approach reaches the upper bound and has a better in (a) or comparable in (b) performance to memory feature replay.

the proposed variational domain-agnostic feature replay to address Scenario (1), where we assume task drift within both streams and domain drift across the two streams [see Fig. 2 (top left)]. For the solver to be able to continuously solve the nonstationary queries, we replay the sampled domain-agnostic features learned from previous tasks while learning the current task. This allows the solver to operate on both previous tasks and the current task simultaneously and thus function as a *task fuse*, i.e., to fuse tasks in the domain-agnostic feature space. To compute the upper bound, we train independent classifiers for each query assuming full access to both support and query data, and the average accuracy.

As shown in Fig. 5(a), the generative feature replay (the orange line) helps the solver remember the first task in the split Office-31 data set as the training progresses across different tasks, whereas the solver suffers from catastrophic forgetting without replay [the brown line in Fig. 5(a)]. Similarly, the average query accuracy on all learned tasks can also be significantly improved by the replay process [see the orange and brown lines in Fig. 5(b)]. Surprisingly, we also find that, in the case of the split Office-31 data set, as shown in Fig. 5(a), the generative feature replay works even better than the memory feature replay (the green and red lines), where we store the features from real data in memory without any constraint on the memory buffer size. One of the possible explanations could be that the generative feature distribution has learned the missing data points and acts as a regularizer in the feature space, which helps with overfitting especially when the training data examples are few (e.g., Office-31). This can also be evidenced by the comparable performance between the generative feature replay and memory feature replay on the split Office-Home data set [see Fig. 5(b)], where more data examples are available. As a negative control, we also experiment with noise replay where the generated features for previous tasks are replaced with random noises, and as expected, the solver suffers from catastrophic forgetting [the purple lines in Fig. 5(a) and (b)].

*1) Ablation Study:* In Table I, we analyze the different model components and notable strategies used in our approach, e.g., the replay strategies for the feature replay module,

TABLE I
COMPONENTS AND STRATEGIES OF OUR PROPOSED APPROACH

| Method | Replay | | | Warmup | Task fusion |
|---|---|---|---|---|---|
| | G | M | N | | |
| Generative feature replay (ours) | ✓ | | | ✓ | ✓ |
| Memory feature replay | | ✓ | | ✓ | ✓ |
| Noise replay | | | ✓ | ✓ | ✓ |
| Baseline (optimal) | | — | | ✓ | ✓ |
| Baseline (w/o warmup) | | — | | ✗ | ✓ |
| Baseline (w/o task fusion)* | | — | | ✗ | ✗ |

* also referred to as *naive baseline*.

the warm-up strategy for the inference module, and the solver module as a task fuse. The warm-up strategy is designed to first train the inference module independently for a few iterations to learn domain-agnostic features, before integrating it with the training of the solver module in an end-to-end fashion. Fig. 5(c) shows the results of the ablation study on both our proposed approach and the baseline that is without feature replay. It is shown in the figure that both the warm-up strategy and the task fusion component improve the performance of our proposed approach, while all baselines without feature replay suffer severely from forgetting. Based on the findings in Fig. 5(c), we then refer to the baseline with both the warm-up strategy and task fusion as *optimal baseline* and the baseline without task fusion as *naive baseline*.

We summarize the performance comparisons of the average query accuracy of all learned tasks between our approach and multiple baselines (e.g., naive baseline and optimal baseline) in Tables II (split Office-31) and III (split Office-Home). The superior performance of the proposed approach shown in the tables (i.e., outperforming the baselines and comparable to the upper bound) demonstrates its effectiveness in addressing both task drift and domain drift that are present in Scenario (1). In addition, we show that, although outperforming the naive baseline, the deep generative replay (DGR [11]), which is one of the state-of-the-art methods with raw data replay in CL, is still worse than our carefully engineered optimal baseline, further validating our proposed idea with high-level feature replay.

TABLE II
AVERAGE QUERY ACCURACY (%) OF ALL LEARNED TASKS (FIVE TASKS) ON SPLIT OFFICE-31

| Method | Setting | A → W | D → W | W → D | A → D | D → A | W → A | Average |
|---|---|---|---|---|---|---|---|---|
| joint training* | i.i.d. | 94.5±0.3 | 98.4±0.1 | 100.0±.0 | 93.5±0.2 | 74.6±0.3 | 72.2±0.1 | 88.9 |
| upper bound (task split) | non i.i.d. | 83.2±2.7 | 96.2±0.5 | 96.3±2.7 | 84.5±1.4 | 70.5±1.0 | 67.3±2.6 | 83.0 |
| naive baseline† | non i.i.d. | 69.4±4.1 | 88.6±1.0 | 88.7±3.6 | 68.6±4.4 | 48.3±1.0 | 49.6±4.0 | 68.9 |
| DGR [11] | non i.i.d. | 81.5±5.7 | 82.8±3.9 | 83.3±3.1 | 83.8±1.4 | 61.0±1.4 | 56.0±2.1 | 74.7 |
| optimal baseline† | non i.i.d. | 90.8±1.2 | 91.6±1.1 | **97.6±0.7** | **86.9±3.1** | 56.0±1.7 | 63.3±3.7 | 81.0 |
| ours | non i.i.d. | **92.0±2.3** | **95.6±0.5** | 95.7±2.7 | 85.2±0.6 | **73.3±0.4** | **67.3±3.1** | **84.9** |

* results cited from MDD [28]; † defined in Table I. The results are provided with mean±std based on three independent experiments.

TABLE III
AVERAGE QUERY ACCURACY (%) OF ALL LEARNED TASKS (13 TASKS) ON SPLIT OFFICE-HOME

| Method | Ar→Cl | Ar→Pr | Ar→Rw | Cl→Ar | Cl→Pr | Cl→Rw | Pr→Ar | Pr→Cl | Pr→Rw | Rw→Ar | Rw→Cl | Rw→Pr | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| joint training* | 54.9 | 73.7 | 77.8 | 60.0 | 71.4 | 71.8 | 61.2 | 53.6 | 78.1 | 72.5 | 60.2 | 82.3 | 68.1 |
| upper bound (task split) | 69.7 | 91.1 | 92.4 | 76.8 | 88.6 | 88.5 | 82.1 | 73.8 | 94.2 | 86.6 | 74.6 | 95.1 | 84.5 |
| naive baseline† | 7.2 | 15.5 | 23.6 | 5.5 | 27.2 | 16.7 | 30.1 | 27.6 | 33.7 | 30.2 | 6.5 | 40.1 | 22.0 |
| DGR [11] | 23.9 | 32.3 | 44.2 | 31.3 | 27.0 | 36.7 | 43.7 | 35.4 | 50.4 | 55.2 | 42.6 | 51.7 | 39.5 |
| optimal baseline† | 49.3 | 68.5 | 79.1 | 49.4 | 64.9 | 67.7 | 66.0 | 49.7 | 78.8 | 70.0 | 54.2 | 79.9 | 64.8 |
| ours | **69.0** | **87.4** | **88.3** | **71.6** | **87.2** | **87.2** | **76.2** | **70.0** | **92.2** | **84.5** | **72.7** | **93.8** | **81.7** |

* results cited from MDD [28]; † defined in Table I.

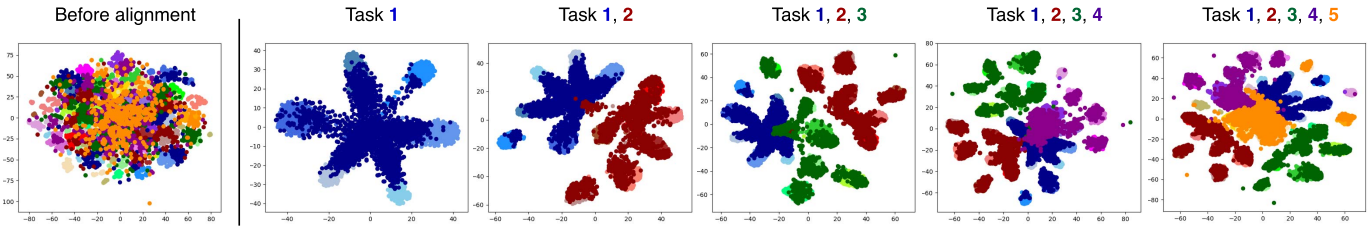



Fig. 6. t-SNE visualizations of the domain-agnostic features from both the support stream data (in *light* colors; each light color represents a different class in Office-31, domain D) and the unlabeled query stream data (in *dark* colors; each dark color represents a different task in split Office-31, domain A). The features are being continuously aligned between the support and query streams given sequentially arriving tasks in the split Office-31 data set.

TABLE IV
AVERAGE QUERY ACCURACY (%) OF ALL LEARNED DOMAINS (THREE DOMAINS) ON OFFICE-HOME

| Method | | Ascending order | | | | | Descending order | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Support stream: | Ar | Cl | Pr | Rw | Avg. | Ar | Cl | Pr | Rw | Avg. |
| | Query stream: | Rw,Pr,Cl | Rw,Pr,Ar | Rw,Ar,Cl | Pr,Ar,Cl | | Cl,Pr,Rw | Ar,Pr,Rw | Cl,Ar,Rw | Cl,Ar,Pr | |
| upper bound (domain split) | | 65.45 | 61.98 | 60.04 | 67.18 | 63.66 | 64.88 | 60.04 | 58.24 | 67.46 | 62.66 |
| without replay | | 18.41 | 18.86 | 16.90 | 20.31 | 18.62 | 25.84 | 21.31 | 25.21 | 28.42 | 25.20 |
| ours | | **62.00** | **59.17** | **57.09** | **62.63** | **60.22** | **60.00** | **57.99** | **54.95** | **65.05** | **59.50** |

*2) Visualizations:* Fig. 6 shows the t-SNE visualizations of the domain-agnostic features from both the support stream data (in *light* colors, with each color representing a different class in Office-31) and the unlabeled query stream data (in *dark* colors, with each color representing a different task in split Office-31) at each task step, where the class space is gradually expanding as the number of learned tasks increases. The classwise alignment between the support stream and

TABLE V
SETUP OF THE EXAMPLE SCENARIO DERIVED FROM COMBINING SCENARIO (1) AND SCENARIO (2)

| Dataset | | | Task & Domain | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Office-31 | Task* | | 1 | 2 | 3 | 4 | 5 | | | | | | | | |
| | Domain | Support stream: | D | A | W | D | W | | | | | | | | |
| | | Query stream: | A | W | D | W | A | | | | | | | | |
| Office-Home | Task* | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| | Domain | Support stream: | Ar | Cl | Pr | Rw | Ar | Rw | Pr | Ar | Pr | Rw | Cl | Ar | Cl |
| | | Query stream: | Cl | Pr | Rw | Ar | Rw | Pr | Cl | Pr | Rw | Cl | Ar | Cl | Rw |

* Note that the task split details are provided in Table VIII for Office-31 and Table IX for Office-Home in the appendix.

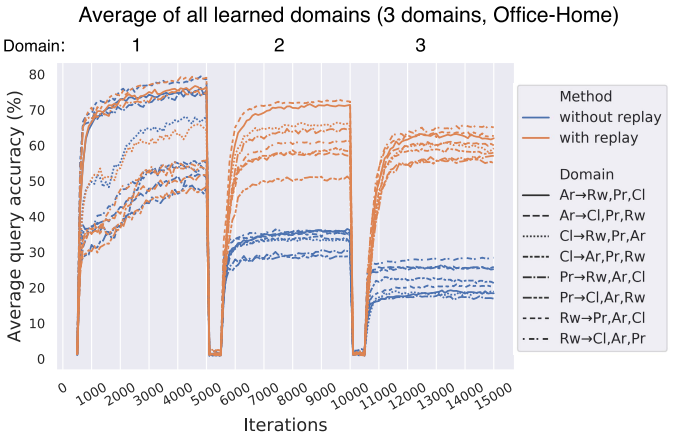



Fig. 7. Average query accuracy of all learned domains during sequential training (Office-Home). We set one domain as the support domain and the remaining three domains as the query stream domains arriving sequentially.

query stream guarantees the solver's performance on the query stream since the solver is only trained with the support stream data in our two-stream CL system.

### D. Scenario (2): Nonstationarity in Domains

In this section, we address Scenario (2) where we assume a single domain in the support stream and sequentially arriving query domains in the query stream, as shown in Fig. 2 (top right). As such, the domain drift exists both within and across the two streams. We perform experiments on Office-Home by selecting one domain as the support and the remaining three domains as the queries in the query stream ordered by the adaptation difficulty level,[3] either ascending (i.e., Rw, Pr, Ar, and Cl) or descending (i.e., Cl, Ar, Pr, and Rw). For example, if domain Ar is chosen as the support, the adaptation of the query stream can be written as Ar $\rightarrow$ Rw, Pr, and Cl in ascending order and Ar $\rightarrow$ Cl, Pr, and Rw in descending order. We evaluate the effectiveness of generative feature replay in the worst case scenario of catastrophic forgetting where the solver deteriorates into a complete forgetting. To simulate this, we train the solver from scratch for each query domain and investigate the effect of the generative domain-agnostic feature replay on the overall performance of all seen domains.

Fig. 7 shows the curves of the average query accuracy of all learned domains, where the generative feature replay (the orange lines) is shown to facilitate the solver to generalize to all previously seen domains when the data for previous query domains are without access. Conversely, the performance of the solver without replay (the blue lines in Fig. 7) deteriorates quickly as the number of query domains increases in all the listed permutation settings. This again demonstrates the ability of the proposed approach in a de novo transfer of high-level knowledge to the solver without relying on the real data. The quantitative numbers comparing the average query accuracy of our approach to that of the baseline and upper bound are given in Table IV, where we can see the dramatic improvements with the feature replay, effectively approaching the upper bound.

[3]Based on the query accuracy marginalized over the support.

TABLE VI
EVALUATION OF THE PROPOSED APPROACH ON THE EXAMPLE OF GENERIC SCENARIO ON SPLIT OFFICE-31 (FIVE TASKS AND THREE DOMAINS)

| Method | Query accuracy (%) | |
|---|---|---|
| | The first task only | Average of all learned tasks |
| upper bound | 88.8±1.9 | 82.5±2.4 |
| optimal baseline† | 15.7±10.9 | 41.8±7.1 |
| ours | **84.0±3.1** | **67.2±1.7** |

† defined in Table I.
The results are provided with mean±std based on three independent experiments.

### E. Generic Scenario: Nonstationarity in Both Tasks and Domains

Scenario (1) and Scenario (2) are the two most fundamental blocks that build up most of the complex learning scenarios in nonstationary environments. Upon the success of addressing both scenarios in Sections IV-C and IV-D, in this section, we further show an example of the generic scenario [see Fig. 2 (bottom)] that is derived from combining Scenario (1) and Scenario (2). More specifically, we integrate the domain drift within streams from Scenario (2) into Scenario (1) that has task drift within streams, and we make random combinations of the available tasks and domains to constitute the two streams. Therefore, the new scenario has both task drift and domain drift within streams and the domain drift across the two streams. The setup details of the example scenario experimented in this work are shown in Table V.

Tables VI (split Office-31) and VII (split Office-Home) show both the query accuracy of the first task and the average query accuracy of all learned tasks, evaluated at the end of

TABLE VII

EVALUATION OF THE PROPOSED APPROACH ON THE EXAMPLE OF GENERIC SCENARIO ON SPLIT OFFICE-HOME (13 TASKS AND FOUR DOMAINS)

| Method | Query accuracy (%) | |
|---|---|---|
| | The first task only | Average of all learned tasks |
| upper bound | 78.4±0.7 | 81.8±1.0 |
| optimal baseline† | 2.7±0.2 | 15.4±2.9 |
| ours | **74.5±6.6** | **61.3±2.0** |

† defined in Table I.

The results are provided with mean±std based on three independent experiments.

the query stream on the new example scenario. We compare our proposed approach to both the optimal baseline (defined in Table I) and the upper bound. It is noticed that although our proposed approach significantly outperforms the optimal baseline on both data sets, there is still a margin between the proposed approach and the upper bound, suggesting that further improvement could be explored for this challenging scenario. We leave the exploration for future work.

### *F. Generative Module for Data Augmentation*

In this section, we explore the possibility of using the generative module in our two-stream CL system independently as a means of data augmentation to transfer part of domain-independent prior knowledge for better query performance. The motivation comes from Scenario (2), where the solver module is required to eventually capture generalized features that are agnostic to all seen domains as the learning progresses in the query stream. This is analogous to learning class-specific features that lay in the intersection of different domains, as shown in Fig. 8(a). In our proposed approach, we learn domain-agnostic features $\boldsymbol{h}_i$ between the support domain $S$ and the query domain $Q_i$ at time step $i$, which can be used as augmented knowledge when solving a subsequent query domain $Q_t$ ($t > i$). However, whether replaying the generated features $\tilde{\boldsymbol{h}}_i$ can be beneficial for solving $Q_t$ depends on the relation among $S$, $Q_i$, and $Q_t$. For example, as illustrated in Fig. 8(a), the support domain and "query 1" share more domain similarity with "query 2" than "query 3"; therefore, heuristically speaking, the knowledge learned from solving "query 1" would be more transferable to "query 2."

To empirically illustrate this, we first visualize the features of the first class (class details in Table IX) from the four different domains (Ar, Cl, Pr, and Rw) in the Office-Home data set as an estimation of domain relation. The features are extracted using a ResNet-50 model pretrained on ImageNet. As seen from the t-SNE plot in Fig. 8(b), domains Ar and Rw share more overlap with domain Pr than domain Cl, suggesting that the knowledge learned from the adaptation of Ar → Rw should be more transferable to domain Pr than domain Cl. Correspondingly, we find, in our experiment, that, given domain Ar as the support domain, the generative replay of the features learned from solving Ar → Rw improves the performance of Ar → Pr by around 3% in the query accuracy, while no significant improvement is observed for Ar → Cl [see Fig. 8(c)]. However, generally speaking, the improvement

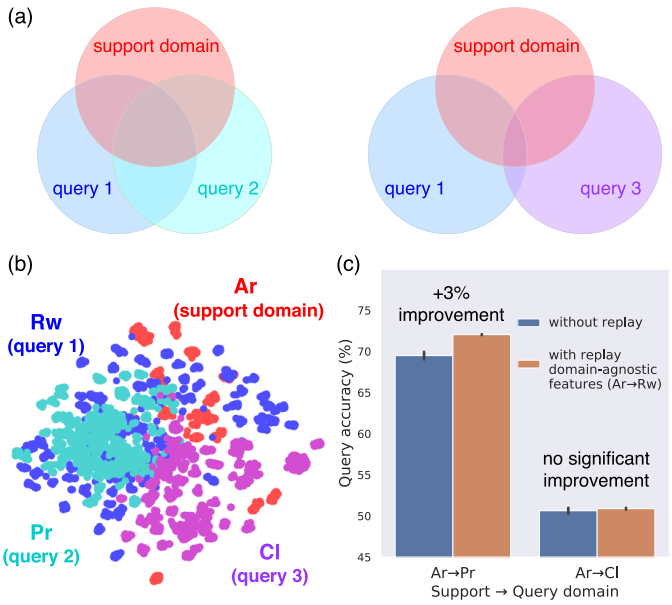



Fig. 8. (a) Examples of possible relation among domains, where the support domain and "query 1" share more similarity with "query 2" than "query 3". (b) t-SNE visualization of the features from the four different domains in the Office-Home data set (showing the first class only), illustrating the domain relation. (c) Generative replay of the domain-agnostic features learned from solving Ar → Rw improves the performance of Ar → Pr, but not Ar → Cl.

is found to be a more common phenomenon; for example, replaying the domain-agnostic features learned from Cl → Rw improves both Cl → Pr and Cl → Ar [see Fig. 9 (left)]; replaying the features from Ar → Cl also improves both Ar → Pr and Ar → Rw [see Fig. 9 (middle)]; and the same for Pr → Cl to Pr → Ar [see Fig. 9 (right)]. The observed improvements indicate that our generative feature replay, by providing more augmented feature samples that are domain-agnostic, imposes an additional regularization to constrain the solver in capturing more generalized features for solving all the domains in the query stream. However, this is based on the assumption that the solver has a fixed amount of capacity.

## V. RELATED WORK

### *A. Continuous Domain Adaptation*

Continuous domain adaptation is a core part of our proposed two-stream CL system, especially in Scenario (2) where no task drift is involved. The problem of continuous domain adaptation has been studied before but in different contexts with different emphases. For example, Mancini *et al.* [36] attempt to solve a specific scenario in continuous domain adaptation, where no target/query data are available, but with metadata provided for all domains; Gong *et al.* [37] proposed to bridge two domains by generating a continuous flow of intermediate domains between the two original domains; and Hoffman *et al.* [38] and Wulfmeier *et al.* [39] present continuous domain adaptation with the emphasis to generalize on a transitioning target/query domain. Other similar work has also been focused on the detection of the drift in the environment and then fast adapting to the new environment [40]–[42]. Closely related to our Scenario (2) is the recent work of [43],

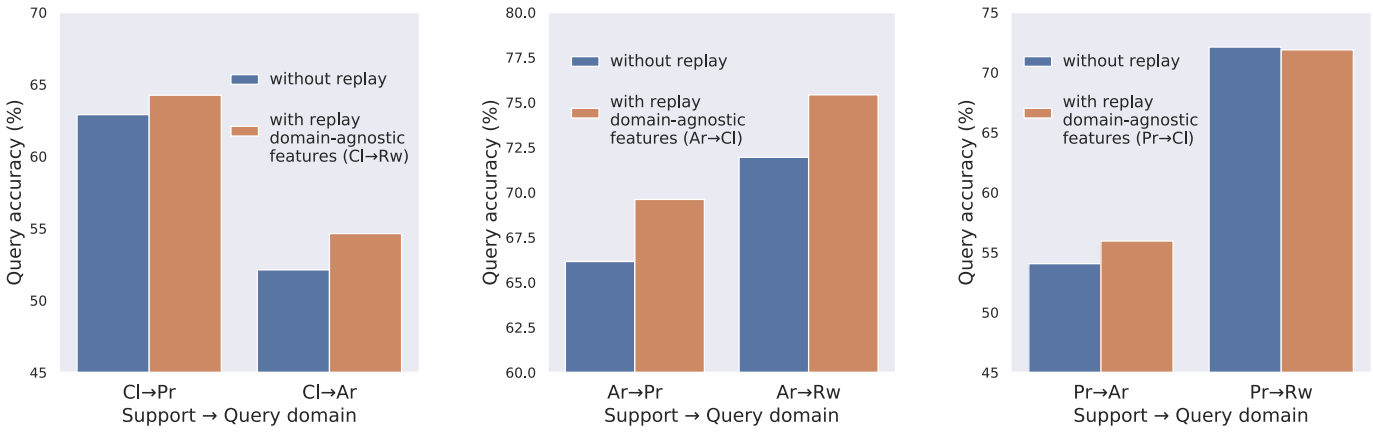



Fig. 9. More examples of generative replay of previously learned domain-agnostic features benefiting the adaptation of a subsequent query domain.

where they also aim to address catastrophic forgetting, but with an implicit assumption that the domain drift follows a specific pattern, i.e., induced by gradually changing weather or lighting condition, which is a reasonable assumption in applications, such as autonomous driving. We focus on more general use cases for solving any arriving queries in the system without imposing extra constraints on the relationship among the queries.

### B. Variational Information Bottleneck

Our work is also related to variational information bottleneck [44], in the sense that we address the domain drift across streams via a variational inference that can be viewed as maximizing the mutual information between the domain-agnostic features and labels while minimizing the mutual information between the input data and domain-agnostic features. A concurrent work [45] adopts the idea of variational information bottleneck for domain adaptation, where the one time-step domain adaptation performance is shown to be improved. Similarly, Luo *et al.* [46] showed that the integration of information bottleneck improves domain adaptive segmentation task. In our approach, we constrain the bottleneck on the decoded feature rather than directly on the latent code and require no additional regularization on the target/query data, as in [45].

Variational autoencoder [47] has also been extensively exploited in domain adaptation to learn disentangled representations for better adaptation performance, where different types of latent variables are proposed to better capture the variations in the data set (e.g., domain-relevant and class-relevant information), and the reconstruction is either on the image level [48], [49] or feature level [50]. In our inference module, the domain-agnostic features are learned through supervision from labels instead of reconstruction.

### C. Replay in Continual Learning

Replay has been widely used as an effective approach to addressing catastrophic forgetting in the CL research, such as example replay [10], deep generative replay [11], [30], and experience replay [51]. However, in these approaches, the generative process is unfiltered and operates on the raw data level directly, while our generative process is domain-agnostic and operates on the abstract feature level. A concurrent work [52] that uses latent replay is closely related to our feature replay, both emphasizing the high-level knowledge transfer; however, their latent replay stands for the replay of the activation volumes in some of the intermediate layers without stochasticity.

TABLE VIII
OFFICE-31 TASK SPLIT

| Task | Classes (number of classes) | Label range |
|---|---|---|
| 1 | back pack, bike, bike helmet, bookcase, bottle, calculator (6) | 0-5 |
| 2 | desk chair, desk lamp, desktop computer, file cabinet, headphones, keyboard (6) | 6-11 |
| 3 | laptop computer, letter tray, mobile phone, monitor, mouse, mug (6) | 12-17 |
| 4 | paper notebook, pen, phone, printer, projector, punchers (6) | 18-23 |
| 5 | ring binder, ruler, scissors, speaker, stapler, tape dispenser, trash can (7) | 24-30 |

## VI. CONCLUSION

In this article, we tackle the challenge of learning in nonstationary environments by proposing a modularized two-stream CL system, where we have two streams of data arriving sequentially (i.e., support and query steam) that can be subject to both task drift and domain drift, within and across the two streams. To address both drifts, we propose a variational domain-agnostic feature replay-based approach, which allows the model in the system to continuously accumulate the filtered and transferable knowledge for solving all queries. We present two fundamental scenarios and a more complex scenario in two-stream CL and demonstrate the effectiveness of the proposed approach on addressing both the task and domain drifts.

## APPENDIX

### A. Data Set Task Split

We provide the task split details of the two benchmark data sets, namely, Office-31 and Office-Home, which are widely used to evaluate the domain adaptation performance. We split

TABLE IX
OFFICE-HOME TASK SPLIT

| Task | Classes (number of classes) | Label range |
|---|---|---|
| 1 | Drill, Exit Sign, Bottle, Glasses, Computer (5) | 0-4 |
| 2 | File Cabinet, Shelf, Toys, Sink, Laptop (5) | 5-9 |
| 3 | Kettle, Folder, Keyboard, Flipflops, Pencil (5) | 10-14 |
| 4 | Bed, Hammer, Toothbrush, Couch, Bike (5) | 15-19 |
| 5 | Postit Notes, Mug, Webcam, Desk Lamp, Telephone (5) | 20-24 |
| 6 | Helmet, Mouse, Pen, Monitor, Mop (5) | 25-29 |
| 7 | Sneakers, Notebook, Backpack, Alarm Clock, Push Pin (5) | 30-34 |
| 8 | Paper Clip, Batteries, Radio, Fan, Ruler (5) | 35-39 |
| 9 | Pan, Screwdriver, Trash Can, Printer, Speaker (5) | 40-44 |
| 10 | Eraser, Bucket, Chair, Calendar, Calculator (5) | 45-49 |
| 11 | Flowers, Lamp Shade, Spoon, Candles, Clipboards (5) | 50-54 |
| 12 | Scissors, TV, Curtains, Fork, Soda (5) | 55-59 |
| 13 | Table, Knives, Oven, Refrigerator, Marker (5) | 60-64 |

the two data sets into multiple tasks based on the class labels to simulate the task drift in nonstationary environments.

*1) Split Office-31:* Office-31 [31] has in total 31 classes and 4652 images. We split the data set into five tasks, with six classes in the first four tasks and seven classes in the last task. Table VIII shows the classes and assigned labels in each task.

*2) Split Office-Home:* Office-Home [32] contains 65 classes and 15 500 images in four distinct domains. Similarly, we split the data set into 13 tasks, each with five classes. The classes and assigned labels for each task are shown in Table IX.

## REFERENCES


[1] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Mach. Learn.*, vol. 1, no. 3, pp. 317–354, Sep. 1986.

[2] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[3] G. Widmer and M. Kubat, "Effective learning in dynamic environments by explicit context tracking," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 1993, pp. 227–243.

[4] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Comput. Sci. Dept., Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[5] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2007, pp. 3–14.

[6] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–37, 2014.

[7] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, Apr. 2015.

[8] K. James *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[9] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[10] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2001–2010.

[11] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2990–2999.

[12] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–18.

[13] J. Schwarz *et al.*, "Progress & compress: A scalable framework for continual learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4528–4537.

[14] M. Rostami, S. Kolouri, P. K. Pilly, and J. McClelland, "Generative continual concept learning," in *Proc. AAAI*, 2020, pp. 5545–5552.

[15] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motivat.*, vol. 24, pp. 109–165, Dec. 1989.

[16] W. E. Skaggs and B. L. McNaughton, "Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience," *Science*, vol. 271, no. 5257, pp. 1870–1873, Mar. 1996.

[17] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6467–6476.

[18] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[19] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.

[20] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1303–1347, 2013.

[21] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3483–3491.

[22] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, nos. 1–2, pp. 151–175, May 2010.

[23] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1180–1189.

[24] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7167–7176.

[25] J. Hoffman *et al.*, "Cycada: Cycle-consistent adversarial domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1989–1998.

[26] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3723–3732.

[27] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1640–1650.

[28] Y. Zhang, T. Liu, M. Long, and M. Jordan, "Bridging theory and algorithm for domain adaptation," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Long Beach, CA, USA: PMLR, Jun. 2019, pp. 7404–7413.

[29] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Sci.*, vol. 7, no. 2, pp. 123–146, Jun. 1995.

[30] C. Wu *et al.*, "Memory replay GANs: Learning to generate new categories without forgetting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5962–5972.

[31] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 213–226.

[32] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5018–5027.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[34] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[35] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2030–2096, May 2015.

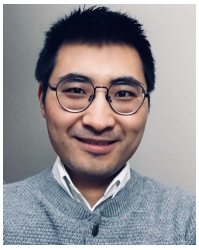
[36] M. Mancini, S. R. Bulo, B. Caputo, and E. Ricci, "AdaGraph: Unifying predictive and continuous domain adaptation through graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6568–6577.

[37] R. Gong, W. Li, Y. Chen, and L. Van Gool, "DLOW: Domain flow for adaptation and generalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2477–2486.

[38] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 867–874.

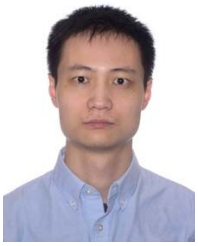
[39] M. Wulfmeier, A. Bewley, and I. Posner, "Incremental adversarial domain adaptation for continually changing environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.

[40] X. Wang, Q. Kang, M. Zhou, L. Pan, and A. Abusorrah, "Multiscale drift detection test to enable fast learning in nonstationary environments," *IEEE Trans. Cybern.*, early access, Jun. 16, 2020, doi: 10.1109/TCYB.2020.2989213.
[41] L. Zheng, G. Liu, C. Yan, C. Jiang, M. Zhou, and M. Li, "Improved TrAdaBoost and its application to transaction fraud detection," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 5, pp. 1304–1316, Oct. 2020.
[42] J. Long, H. Wang, P. Li, and H. Fan, "Applications of fractional lower order time-frequency representation to machine bearing fault diagnosis," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 734–750, 2017.
[43] A. Bobu, E. Tzeng, J. Hoffman, and T. Darrell, "Adapting to continuously shifting domains," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2018, pp. 1–4.
[44] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–19.
[45] Y. Song *et al.*, "Improving unsupervised domain adaptation with variational information bottleneck," 2019, *arXiv:1911.09310*. [Online]. Available: http://arxiv.org/abs/1911.09310
[46] Y. Luo, P. Liu, T. Guan, J. Yu, and Y. Yang, "Significance-aware information bottleneck for domain adaptive semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6778–6787.
[47] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[48] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling, "DIVA: Domain invariant variational autoencoders," 2019, *arXiv:1905.10427*. [Online]. Available: http://arxiv.org/abs/1905.10427
[49] R. Cai, Z. Li, P. Wei, J. Qiao, K. Zhang, and Z. Hao, "Learning disentangled semantic representation for domain adaptation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, p. 2060.
[50] X. Peng, Z. Huang, X. Sun, and K. Saenko, "Domain agnostic learning with disentangled representations," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5102–5112.
[51] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 348–358.
[52] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," 2019, *arXiv:1912.01100*. [Online]. Available: http://arxiv.org/abs/1912.01100



**Qicheng Lao** received the bachelor's degree in medicine from Fudan University, Shanghai, China, in 2011, the M.Sc. degree in experimental medicine from McGill University, Montreal, QC, Canada, in 2015, and the Ph.D. degree in computer science (with Thomas Fevens) from Concordia University, Montreal, in 2019.

He joined the West China Hospital, Sichuan University, Chengdu, China, as an Assistant Professor. He is currently a Post-Doctoral Fellow with the Montreal Institute for Learning Algorithms (MILA), Université de Montréal, Montreal, under the supervision of Yoshua Bengi, also in collaboration with Imagia. His research interests include multimodal representation learning, learning in nonstationary environments, transfer learning, and knowledge transfer between machine learning and medicine, particularly in neuroscience and cancer immunology.



**Xiang Jiang** received the Ph.D. degree in computer science from Dalhousie University, Halifax, NS, Canada, in 2021, under the supervision of Stan Matwin and Daniel Silver. During his Ph.D. degree, he worked as a Research Scientist Intern at Imagia, Montreal, QC, Canada. His research interests are in the areas of machine learning, deep learning, domain adaptation, and metalearning.

**Mohammad Havaei** received the Ph.D. degree from the University of Sherbrooke, Sherbrooke, QC, Canada, in 2017, under the supervision of Hugo Larochelle and Pierre-Marc Jodoin.

From 2016 to 2018, he was a member of the Montreal Institute for Learning Algorithms (MILA), Université de Montréal, Montreal, QC, Canada, as a Post-Doctoral Fellow, under the supervision of Aaron Courville. He is currently a Research Scientist with Imagia, Montreal, where he focuses on developing machine learning methods applied to healthcare.

**Yoshua Bengio** has been a Professor with the Université de Montréal, Montreal, QC, Canada, since 1993. He is also the Founder and a Scientific Director of Mila—Quebec AI Institute, Montreal, the world's largest university-based research group in deep learning. He is also the Scientific Director of IVADO, Montreal.

Dr. Bengio is a fellow of the Royal Society of London and Canada. Concerned about the social impact of AI, he actively contributed to the Montreal Declaration for the Responsible Development of Artificial Intelligence. He is recognized as one of the world's leading experts in AI and a pioneer in deep learning notably for its neural networks rebirth. In 2018, he ranked as the computer scientist with the most new citations worldwide, due to his many high-impact contributions. His contributions to research are undeniable. In 2018, he ranked as the most cited computer scientist worldwide. He received the prestigious Killam Prize, plus the ACM A.M. Turing Award, "the Nobel Prize of Computing," jointly with Geoffrey Hinton and Yann LeCun for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing. He also holds the Canada-CIFAR AI Chair and co-directs the Learning in Machines and Brains Program of the Canadian Institute for Advanced Research (CIFAR) as a Senior Fellow.