Autonomous Cross Domain Adaptation Under Extreme Label Scarcity

Weiwei Weng, Mahardhika Pratama*, Senior Member, IEEE, Choiru Za’in, Marcus de Carvalho, Rakaraddi Appan, Andri Ashfahani*, and Edward Yapp Kien Yee*

Abstract—A cross domain multistream classification is a challenging problem calling for fast domain adaptations to handle different but related streams in never-ending and rapidly changing environments. Notwithstanding that existing multistream classifiers assume no labeled samples in the target stream, they still incur expensive labeling costs since they require fully labeled samples of the source stream. This article aims to attack the problem of extreme label shortage in the cross domain multistream classification problems where only very few labeled samples of the source stream are provided before process runs. Our solution, namely, Learning Streaming Process from Partial Ground Truth (LEOPARD), is built upon a flexible deep clustering network where its hidden nodes, layers, and clusters are added and removed dynamically with respect to varying data distributions. A deep clustering strategy is underpinned by a simultaneous feature learning and clustering technique leading to clustering-friendly latent spaces. A domain adaptation strategy relies on the adversarial domain adaptation technique where a feature extractor is trained to fool a domain classifier by classifying source and target streams. Our numerical study demonstrates the efficacy of LEOPARD where it delivers improved performances compared to prominent algorithms in 15 of 24 cases. Source codes of LEOPARD are shared in https://github.com/wengweng001/LEOPARD.git to enable further study.

Index Terms—Concept drifts, data streams, incremental learning, multistream classification, transfer learning.

Manuscript received 27 October 2021; revised 14 May 2022; accepted 12 June 2022. Date of publication 23 June 2022; date of current version 6 October 2023. This work was supported by the National Research Foundation, Singapore, through Industry Alignment Fund pre-Positioning Programme (IAF-PP) in the Advanced Manufacturing and Engineering (AME) Domain, under Contract A19C1A0018. (Corresponding author: Mahardhika Pratama.)

Weiwei Weng, Marcus de Carvalho, Rakaraddi Appan, and Andri Ashfahani are with the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore 639798 (e-mail: wengweng001@e.ntu.edu.sg; marcus.decarvalho@ntu.edu.sg; appan001@e.ntu.edu.sg; andriash001@e.ntu.edu.sg).

Mahardhika Pratama was with the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore 639798. He is now with the Academic Unit of STEM, University of South Australia, Adelaide, SA 5095, Australia (e-mail: dhika.pratama@unisa.edu.au).

Choiru Za’in is with the School of Information Technology (IT), Monash University, Clayton, VIC 3800, Australia (e-mail: choiru.zaain@monash.edu).

Edward Yap Kien Yee is with the Singapore Institute of Manufacturing Technology, Agency for Science, Technology and Research (A*Star), Singapore 138634 (e-mail: edward_yapp@simtech.a-star.edu.sg).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3183356. Digital Object Identifier 10.1109/TNNLS.2022.3183356

I. INTRODUCTION

A. Background

MULTISTREAM classification problems [1] portray a classification problem across many streaming processes running simultaneously but independently. Each streaming process features different but related characteristics to be handled by a single model having a stream-invariant trait. That is, each stream suffers from the domain shift problem in which they follow different distributions. Multistream classification problem also considers the issue of labeling cost where the ground-truth access is only provided in the source stream while leaving the target stream with the absence of any labeled samples. Unlike the traditional domain adaptation problems, the multistream problem deals with continuous information flows, which must be handled in a fast and sample-wise fashion. Another typical problem is the asynchronous drift problem, which distinguishes itself from the conventional single-stream problem. The asynchronous drift problem refers to independent drifts between source and target streams taking place at different time points. The multistream classification problem distinguishes itself from the online transfer learning problem [2] in which both source and target domains are streaming in nature, whereas the online transfer learning problem assumes a static source domain although it considers a streaming problem of the target domain. The underlying goal of the multistream classification problem is to build a predictive model $f(.)$, which simultaneously performs the unsupervised domain adaptation and addresses the issue of data streams. Notwithstanding the recent progress in the multistream classification area, most works are designed from a single domain perspective in which both source and target streams are drawn from the same feature space. In addition, existing solutions incur expensive labeling costs because they require full supervision of the source stream.

B. Practical Scenario

This article puts into perspective a cross domain multistream classification problem under extreme label scarcity where, unlike conventional multistream classification problems, the source stream and the target stream are generated from different feature spaces but share the same target attributes. The extreme label scarcity issue presents in the fact that no label is provided for the target stream, while only few labeled samples are made available in the source stream during the warm-up
The Learning Streaming Process from Partial Ground Truth (LEOPARD) approach is proposed in this article and resolves the cross domain multistream classification problems under extreme label scarcity. Instead of building a machine-specific model for monitoring purposes, a single machine-invariant model is constructed, thereby saving significant developmental costs, because data collection, annotation, and preprocessing do not have to be repeated for each machine. Nevertheless, this task is challenging because data samples captured by sensors are streaming in nature. Different machines are installed with different sensors or of different types, thereby producing different feature spaces, while having different sampling rates leading to different batch sizes. Process deviations due to tool wear or any other external influencing factors occur independently in each machine at different time points leading to drifting data distributions in each machine with different rates, magnitudes, and types. The issue of labeling cost occurs because visual inspections leading to interruption of machine operations are necessitated to annotate data samples. It hinders the labeling process during the process runs. The labeling process is possible to be done only for prerecorded samples to avoid frequent stoppages of machine operations.

We visualize the significance of label’s scarcity in the context of domain adaptation in Fig. 1, where DANN [3] is evaluated under different label proportions of source streams. Our numerical results are produced in theoffice31 problem (Webcam → DSLR) using five label ratios: 10%, 20%, 30%, 40%, and 50%. It is observed that DANN’s performances are significantly compromised with reductions of label proportions, less than 10% accuracy on source and target streams under 5% and 10% label proportions. That is, its accuracy on source and target streams consistently slips.

C. Our Contribution

The Learning Streaming Process from Partial Ground Truth (LEOPARD) approach is proposed in this article and resolves the cross domain multistream classification problems under extreme label scarcity. LEOPARD is developed under the framework of a flexible deep clustering network (DCN) where it features an elastic and progressive network structure to handle changing data distributions. That is, hidden nodes, hidden layers, and hidden clusters are self-evolved with respect to the asynchronous drift problem in both source and target streams. The learning process of LEOPARD aims to achieve two objectives under shared network parameters: clustering-friendly latent space and cross domain alignment in which it minimizes three loss functions. The reconstruction loss functions as the nonlinear dimension reducer where it projects input samples into a low dimension and establishes a common latent space between the source stream and the target stream. It is achieved by the stacked autoencoder (SAE) performing the nonlinear mapping. The second component is the clustering loss creating a clustering-friendly latent space preventing the trivial solution. The cross domain adaptation loss is meant to induce the domain alignment and utilizes the adversarial domain adaptation approach [3]. This strategy relies on a domain classifier to classify the origin of data samples and a feature extractor. The feature extractor and the domain classifier compete with each other, thus resulting in domain-invariant representations. LEOPARD does not call for any labeled samples for its updates, and few prerecorded labeled samples of the source stream are only used to establish the class-to-cluster relationship.

This article presents four major contributions: 1) it proposes a new problem, namely, cross domain multistream classification problems under extreme label scarcity; 2) an algorithm, namely, LEOPARD, is developed to address the issue of label’s scarcity in the cross domain multistream classification problem; 3) a joint optimization problem is formulated to attain the clustering-friendly latent space and the domain alignment such that the target stream can be predicted accurately with very few labels of the source stream and no labels of the target stream; and 4) the source code of LEOPARD along with all datasets is made public in https://github.com/wengweng001/LEOPARD.git to enable further study. Our numerical study has substantiated the efficacy of LEOPARD in handling the issue of extreme label scarcity in the cross domain multistream classification problems. It delivers highly competitive performances compared to prominent algorithms.

II. RELATED WORKS

A. Multistream Classification

The area of multistream classification has attracted growing research interest as observed by the number of works published in the literature. Pioneering work is proposed in [1] using the kernel mean matching (KMM) method as a domain adaptation technique combined with a drift detection method to detect the concept drift in each domain. Considering high computational complexity and memory demand of [1], FUSION is proposed in [4] where the KLIEP method is implemented for domain adaptation, while a density ratio method is designed for detecting the asynchronous drifts. MSCRDR is put forward in [5] and uses the Pearson divergence method for domain adaptation. Recently, a deep learning algorithm, namely, ATL, is proposed to solve the multistream classification problem using the encoder and decoder structures under shared parameters coupled with the KL divergence method for domain adaptation [6]. ATL characterizes...
an inherent drift handling aptitude with a self-evolving network structure. MELANIE is proposed in [7] to handle the multisource multistream classification problem. This work is extended in [8]. Another solution of multisource multistream classification is offered in [9] where the CMD-based regularization is integrated. The problem of multisource unsupervised domain adaptation under both homogeneous and heterogeneous settings is discussed in [10]. The area of multistream classifications deserves an in-depth study due to at least two reasons.

1) These approaches are designed for a single domain problem where both source and target streams share the same feature space (domain). To the best of our knowledge, there exists only one work in the literature handling the cross domain multistream classification problem [11] using the empirical maximum mean discrepancy for domain adaptation. However, this approach is based on a nondeep learning solution relying on a simple linear projection for feature transformation, prone to trivial solutions.

2) Although these approaches rely on the unsupervised domain adaptation approaches where no label is offered for the target stream, full annotations are required for the source stream. On the other side, the multistream classification problem distinguishes itself from the online transfer learning problem [2] assuming a fixed and static source domain. Hence, the problem of asynchronous drift problem is absent in the online transfer learning problem.

B. Semi-Supervised Transfer Learning

The issue of labeling cost has attracted research interest in the transfer learning community. In [12], the notion of complementary labels incurring less expensive labeling costs than the true class label is implemented. Dual deep neural networks are designed in which one focuses on complementary labels, while another handles the domain adaptation. Liu et al. [13] concern on reductions of the labeling cost in the heterogeneous domain adaptation problem usually calling for some labeled samples of the target domain. Another effort is devoted to reduce the labeling cost in [14] where it concerns on an open set domain adaptation where the target domain contains unknown classes. The use of noisy labels for unsupervised domain adaptation has been investigated in [15]. Our work differs from these works in two aspects.

1) LEOPARD handles the situation of cross domain multistream classification under extreme label scarcity. That is, labeled samples are only revealed for the source stream during the warm-up period, where no labeled samples for both streams are given for model updates during the process runs.

2) The learning approach is designed for the stream learning scenario.

III. PROBLEM FORMULATION

Suppose that $D_S$ and $D_T$ stand for the source and target domains, respectively. The goal of domain adaptation is to solve a classification problem of the target domain $D_T$ without any labels by transferring the knowledge base from the source domain $D_S$ where there exist some labeled samples. Referring to [16], the generalization error of the target domain is upper bounded by how good a model $f(\cdot)$ learns the source domain and the discrepancies between the two domains

$$\epsilon_T(f) \leq \epsilon_S(f) + d_1(D_S, D_T)$$

$$+ \min \{E_{D_S}(f_S, f_T), E_{D_T}(f_S, f_T)\}$$

(1)

where the first term is the source error, the second term is the divergence between the two domains, and the last term is the difference in labeling function between the two domains, which should be small [16]. Direct minimization of the divergence is a challenging task due to a lack of correspondence between data samples of the two domains. For streaming data, it leaves a major challenge because the divergence measure works with a finite number of samples.

A cross domain multistream classification problem under extreme label scarcity is defined as a classification problem of two independent streaming data $B_S^1, B_S^2, \ldots, B_S^{k_S}$ and $B_T^1, B_T^2, \ldots, B_T^{k_T}$ termed a source stream and a target stream, respectively, where $K_S$ and $K_T$ are, respectively, the number of source stream and target stream unknowns in practice. $B_S^i$ and $B_T^i$ are drawn from the source domain $D_S$ and the target domain $D_T$, respectively. Extreme label scarcity is perceived in the limited access of ground truth where only prerecorded samples of the source stream $B_S^i = \{x_s^i, y_s^i\}_{i=1}^{N_S}$ are labeled, while no label is provided during the process runs $B_T^i = \{x_t^i\}_{i=1}^{N_T}$. $N_m$ and $N_S$ denote the number of prerecorded data samples of the source stream and the size of the source stream, respectively. On the other hand, the target stream suffers from the absence of true class labels $B_T^i = \{x_t^i\}_{i=1}^{N_T}$, where $N_T$ is the size of the target stream. Note that we consider a case where both source and target domains are streaming in nature. $x_S^i \in X_S$, $x_T^i \in X_T$, and $X_S \neq X_T$ are input vectors of the source stream and the target stream, while $y_i = [l_1, l_2, \ldots, l_m]$ is a target vector formed as one-hot vector $y_i^j \in \mathcal{Y}$. $(x_t^i, y_t^i) \in X_T \times \mathcal{Y}$, and $(x_T^i, y_T^i) \in X_T \times \mathcal{Y}$. That is, the two domains feature different feature spaces but share the same labeling function and target variables, and cross domain. The two streaming data are sampled with different speeds resulting in $N_S \neq N_T$, different batch sizes while following different distributions $P(x_S) \neq P(x_T)$, covariate shift. The source stream and the target stream are nonstationary in nature where their concepts are drifting $P(x, y)_S \neq P(x, y)_T$, $P(x, y)_T \neq P(x, y)_T^{t+1}$, and $t \neq t'$, i.e., concept drifts of the two streams might develop at different time periods $t \neq t'$, asynchronous drift.

IV. LEARNING PROCEDURE OF LEOPARD

A. Network Structure of LEOPARD

LEOPARD is structured as a DCN developed with a feature extraction layer extracting natural features $Z$ from raw input features $x$ by means of a mapping function $F_W(\cdot)$, where $W_f$ stands for the parameters of the feature extractor. The extracted
features $Z$ are passed to a fully connected layer formed as an SAE with a tied-weight constraint. That is, the decoder parameters are the inverse mapping of the encoder parameters. The natural features $Z \in \mathbb{R}^{d}$ are projected to a low dimensional latent space $h' \in \mathbb{R}^{R_l}$, where $u'$ and $R_l$ are, respectively, the number of natural features and hidden nodes at the $l$th layer $R_l \ll u'$. The decoding and encoding mechanisms are expressed as

$$h' = r(W_{enc}^l h^{l-1} + b^l); \quad h^0 = Z$$  \hspace{1cm} (2)$$

$$h^{l-1} = r(W_{dec}^l h^{l} + c^l) \quad \forall l = 1, \ldots, L$$  \hspace{1cm} (3)$$

where $W_{enc}^l \in \mathbb{R}^{R_l \times u'}$ and $b^l \in \mathbb{R}^{R_l}$ stand for the connective weights and biases of the $l$th layer of the encoder, respectively, while $W_{dec}^l \in \mathbb{R}^{u' \times R_l}$ and $c^l \in \mathbb{R}^{u'}$ denote the connective weights and biases of the $l$th layer of the decoder, respectively. The tied-weight constraint $W_{dec}^l = (W_{enc}^l)^T$ functions as a regularization mechanism preventing the issue of overfitting.

The clustering mechanism is carried out in each deep embedding space, each latent space. That is, it takes place in every hidden layer of SAE $h_l$ creating different representations of data samples. The inference mechanism is performed by first calculating the similarity degree of a data sample and a hidden cluster [17]:

$$\phi^l_j = \frac{1 + |h^l - C^l_j|_2 / \lambda}{\sum_{j=1}^{N_c} (1 + |h^l - C^l_j|_2 / \lambda)^{\alpha+1} - 1}$$  \hspace{1cm} (4)$$

where $C_j^l$ and $h^l$ are the centroid of the $j$th cluster of the $l$th layer and the latent representation of a data sample $x$ of the $l$th layer, while $\text{Clus}^l$ is the number of cluster created in the $l$th latent space, i.e., the $l$th layer of SAE. $\lambda = 1$ is chosen here. The student t-distribution is adopted to model the similarity degree, and $\phi^l_j$ is also regarded as the cluster posterior probability $P(C^l_j | X)$ [18], where $P(C^l_j | X)$ is the case of perfect match between $h^l$ and $C^l_j$. The similarity degree $\phi^l_j$ is aggregated across $N_m$ prerecorded samples having true class labels $B^{o}_m = \{x^S_n, x^T_n\}_{n=1}^{N_m}$. This operation produces the cluster's allegiance [19] measuring cluster's tendencies to a particular class. Suppose that $N_o$ stands for the number of prerecorded samples having the $o$th class as their labels, and the cluster allegiance $\text{Ale}^l_{j, o}$ is calculated as

$$\text{Ale}^l_{j, o} = \frac{\sum_{m=1}^{N_o} \phi^l_{j, o}}{\sum_{o=1}^{N_o} \sum_{m=1}^{N_o} \phi^l_{j, o}}$$  \hspace{1cm} (5)$$

where $\phi^l_{j, o}$ measures the similarity degree of the cluster $C^l_j$ and the $n$th prerecorded sample $h^l_n$ falling into the $o$th class. Equation (5) pinpoints the neighborhood dependency of the $j$th cluster to the $o$th class and implies that an unclean cluster, occupied by data samples of mixed classes, possesses low cluster allegiance. The winner-takes-all principle $\text{win} = \arg \max_{j=1, \ldots, \text{Clus}^l} \phi^l_j$ is adopted here, where a data sample is associated with the nearest cluster. The local score of the $l$th layer is defined as the allegiance of the winning cluster $\text{Score}^l = \text{Ale}^l_{\text{win}}$. The predicted class label $\hat{y}$ is determined as a class label maximizing its global score. The global score is calculated as the summation of a local score across $L$ layers

$$\hat{y} = \arg \max_{o=1}^{M} \sum_{l=1}^{L} \text{Score}^l$$  \hspace{1cm} (6)$$

where the majority voting approach is implemented here. It is evident that LEOPARD merely benefits from the labeled pre-recorded samples of the source stream $B^S_0$ to associate a cluster to a specific class. No label at all from both streams is solicited in the streaming phase where it confirms its applicability in the extreme label scarcity environments. Fig. 2 visualizes the network structure of LEOPARD. It is perceived that the clustering process occurs in every hidden layer of LEOPARD, thus producing its local outputs. The final predicted class label is aggregated across all layers making use of a summation operation. $R_l$, $L$, and Clus are self-evolved with respect to varying distributions.

B. Parameter Learning of LEOPARD

1) Adversarial Domain Adaptation: The idea of domain adaptation is to minimize the divergence between the target domain and the source domain. The concept of adversarial domain adaptation is founded on the idea of $H$ divergence [3] where it relies on a hypothesis class $H$, a set of binary classifiers.

Definition 1 [20]: Given the two domains $D_S$ and $D_T$ and the hypothesis class $H$, the $H$ divergence between $D_S$ and $D_T$ is defined as follows:

$$d_H(D_S, D_T) = 2 \sup_{\eta \in H} \left[ \Pr[\eta(x) = 1]_{x \sim D_S} - \Pr[\eta(x) = 1]_{x \sim D_T} \right].$$  \hspace{1cm} (7)$$

The $H$ divergence in (7) relies on the hypothesis class $H$ to distinguish data samples generated from $D_S$ or data samples generated from $D_T$. In [20], the empirical $H$ divergence can be used in the case of a symmetric hypothesis class $H$

$$d_H(D_S, D_T)$$

$$= 2 \left( 1 - \min_{\eta \in H} \left[ \frac{1}{n} \sum_{x \sim D_S} I[\eta(x) = 0] + \frac{1}{n'} \sum_{x \sim D_T} I[\eta(x) = 1] \right] \right).$$  \hspace{1cm} (8)$$
where $I[a]$ denotes an indicator function returning 1 if $a$ is true or 0 otherwise. This implies that the $H$ divergence can be minimized by finding a representation where the source and target samples are indistinguishable [3].

The concept of adversarial domain adaptation can be implemented by deploying a domain classifier $\zeta_{\text{dc}}(F_{W_f}(\cdot))$ working along with the feature extractor $F_{\mu}(\cdot)$ and a classifier $\zeta_{\text{cd}}(F_{W_f}(\cdot))$. The domain classifier predicts the origin of data samples whether they are generated by the source domain $D_S$ or the target domain $D_T$, while the classifier generates the final output of a network. The domain reversal layer is implemented in updating the feature extractor such that indistinguishable features of source and target domains are induced. That is, the overall loss function is written as follows:

$$L = \frac{1}{N_S} \sum_{n=1}^{N_S} L_{\text{cd}}(\zeta_{\text{cd}}(F_{W_f}(x_{ni}), d_{ni}))$$

$$- \lambda \frac{1}{N_S} \sum_{n=1}^{N_S} L_{\text{cd}}(\zeta_{\text{dc}}(F_{W_f}(x_{ni}), d_{ni}))$$

$$+ \frac{1}{N_T} \sum_{n=1}^{N_T} L_{\text{cd}}(1 - \zeta_{\text{dc}}(F_{W_f}(x_{ni}), d_{ni}))$$

(9)

where $L_{\text{cd}}(\cdot)$ is implemented as the cross entropy loss function and $d_{ni}$ is the domain identity, i.e., 1 for the source domain and 0 for the target domain. From (9), the gradient reversal layer inserts a negative constant confusing the domain classifier, i.e., generating indistinguishable samples. The parameter learning process is formulated as follows:

$$W_f = W_f - \mu (\frac{\partial L_{\text{cd}}}{\partial W_f} - \alpha_1 \frac{\partial L_{\text{cd}}}{\partial W_f})$$

(10)

where the feature extractor is trained to produce similar features of the two domains seen in the negative sign of the gradient, thereby leading to the domain invariant network.

2) Loss Function: The parameter learning strategy of LEOPARD is constructed using a joint loss function comprising two modules: clustering loss and cross domain adaptation loss. The underlying goal is to produce domain-invariant parameters and clustering-friendly latent spaces such that the online cross domain adaptation can be solved under extreme label scarcity. The overall cost function is formalized as follows:

$$L_{\text{all}} = L_{\text{cluster}} - \alpha_1 L_{\text{cd}}$$

(13)

where $L_{\text{cluster}}$ and $L_{\text{cd}}$, respectively, denote the clustering loss and the cross domain adaptation loss, while $\alpha_1$ is a tradeoff constant controlling the influence of the cross domain adaptation loss. It is an unconstrained optimization problem that can be optimized using the stochastic gradient descent approach with no epoch or epoch per batch to assure scalability in streaming environments. That is, a number of iterations are done per batch. A data batch is discarded once iterations across a number of epochs are completed to allow bounded complexity. The negative sign in (13) follows the gradient reversal strategy generating similar features across two domains. In other words, the gradient of clustering loss, and the gradient of cross domain loss and the domain classifier loss are subtracted [3].
3) Clustering-Friendly Latent Space: The clustering loss aims to achieve the clustering-friendly latent space via simultaneous feature learning and clustering. The clustering loss is formulated as the reconstruction loss and the KL divergence loss minimizing probabilistic distance of the latent space and the auxiliary target distribution [17]

\[
L_{\text{cluster}} = L_1(x_{S,T}, \hat{x}_{S,T}) + \sum_{l=1}^L \left( L_2(h_{S,T}^l, \hat{h}_{S,T}^l) + \alpha_2 \text{KL}(\phi|\Phi) \right) \tag{14}
\]

where \( \Phi^l \) is the auxiliary target distribution of the \( l \)th latent space and \( \alpha_2 \) is a regularization constant controlling the strength of the KL divergence loss. \( \phi^l \) is the similarity degree of the current sample to existing clusters. \( L_1(.) \) is the reconstruction loss formed as the mean square error (mse) loss function. It also performs nonlinear dimension reduction preventing the trivial solutions often happening in the case of linear mapping. It guarantees a data sample to be mapped back to its original representation. The key difference between the two loss functions lies in the adaptation mechanism in which \( L_1 \) is solved in the end-to-end fashion, while \( L_2 \) is carried out in the layerwise fashion.

The last term also known as the KL divergence loss minimizes the discrepancy of the distribution of a current data batch calculated via (4) and the auxiliary target distribution KL(\( \phi|\Phi \)) \( = \sum_i \sum_j \phi_{i,j}^l \log \left( \frac{\phi^l_{i,j}}{\Phi_{i,j}^l} \right) \). The auxiliary target distribution should satisfy three requirements [17]: 1) improve prediction; 2) emphasize samples of high confidence; and 3) normalize the contribution of each cluster to avoid the creations of large clusters. We adopt the same auxiliary distribution as in [17], where \( \Phi_{i,j}^l \) is raised to the second power and normalized by frequency per cluster

\[
\Phi_{i,j}^l = \left( \frac{\phi_{i,j}^l}{\sum_{n=1}^N \phi_{i,j}^l} \right)^2 / \zeta_j \tag{15}
\]

where \( \zeta_j = \sum_{i=1}^M \phi_{i,j}^l \) is the frequency of a cluster. This strategy is understood as the soft-cluster assignment [17] where all clusters are updated and differ from the hard-cluster assignment only tuning the winning cluster. The clustering mechanism is hard to conduct in the high-dimensional space [21], thus calling for feature learning steps to be committed simultaneously. The clustering process takes place in every latent space \( h(.)^l \) set as the common feature spaces between the source and target domains. That is, (14) is executed using samples of both source and target streams. This process also functions as an implicit domain adaptation strategy since the minimization of reconstruction loss across two streams with shared parameters ends up with an overlapping region of both domains [6]. The optimization procedure takes place simultaneously where the network parameters and the cluster parameters are adjusted concurrently with the SGD method.

4) Domain-Invariant Network: LEOPARD consists of three submodules: feature extractor \( F(.) \), classifier \( \zeta(.) \), and domain classifier \( \zeta(.) \) to achieve a domain invariant property, as depicted in Fig. 3. The feature extractor is parameterized by \( Wf \), and the classifier formed as the deep clustering module is parameterized by \( WC \in \{W_{enc}, W_{dec}, C\} \), while the domain classifier formed as a single hidden layer network is parameterized by \( Wdc \). The feature extractor and the domain classifier play a minimax game via the gradient reversal layer where the feature extractor is trained to fool the domain classifier via the production of similar features of source and target streams, while the domain classifier is trained to identify the origin of data samples. The cross domain adaptation loss is, thus, formulated as the domain classifier loss as follows:

\[
L_{cd} = \frac{1}{N_S} \sum_{n=1}^{N_S} L_\zeta (\zeta_{WS}(F_{Wf}(x_n)), d_n) + \frac{1}{N_T} \sum_{n'=1}^{N_T} L_\zeta (1 - \zeta_{WS}(F_{Wf}(x_{n'}))), d_{n'}) \tag{16}
\]

where \( d_n \) stands for the origin of data samples, i.e., 1 for the source stream and 0 for the target stream. The domain classifier is tasked to solve a binary classification problem, where \( L_\zeta(.) \) is set as the cross entropy loss function. This leads to similar parameter learning processes for feature extractor, domain classifier, and classifier, respectively, as in (10) - (12), except the presence of the clustering loss instead of the cross entropy loss, as defined in (14): \( Wf = Wf - \mu(\partial L_{\text{cluster}} / \partial Wf) + \alpha_1(\partial L_{cd} / \partial Wf) \); \( WC = WC - \mu(\partial L_{\text{cluster}} / \partial WC) \); and \( Wdc = Wdc - \mu\alpha_1(\partial L_{cd} / \partial Wdc) \), where \( \mu \) denotes the learning rate. Note that the gradient reversal layer has no parameters and simply alters the sign of the gradients allowing maximization process to be carried out via the stochastic gradient descent approach. This only applies to the feature extractor, as illustrated in Fig. 3.

C. Structural Learning of LEOPARD

1) Evolution of Cluster: The classifier of LEOPARD implements the self-organizing mechanism of network clusters where the clusters are flexibly grown in every hidden layer \( h^l(.) \) if changing data distributions are identified. Furthermore, it is performed for both source data samples \( h^l_{S} \) and target data samples \( h^l_{T} \). That is, the clustering mechanism does not generate stream-specific clusters. Suppose that \( D(X, Y) \) stands for the \( L_2 \) distance between two variables \( X \) and \( Y \), an \( i \)th cluster of the \( l \)th layer is parameterized by its center \( C^l_i \), and the growing condition is formulated as follows:

\[
\min_{i = 1, \ldots, \text{Ch}} D(h^l_i, C^l_i) > \mu^l_{D,i} + k_1 \sigma^l_{D,i} \tag{17}
\]

where \( \mu^l_{D,i} \) and \( \sigma^l_{D,i} \) denote the mean and standard deviation of the distance \( D(h^l_i, C^l_i) \) of the \( i \)th cluster of the \( l \)th layer while \( k_1 = 2 \exp -||h^l_i - C^l_i||_2 + 2 \) leading to a dynamic confidence degree. The dynamic confidence degree enables the cluster growing phase to be carried out in the case of a far proximity between a data sample and the winning cluster. Equation (17) examines the coverage span of existing clusters where a new cluster is inserted if a data sample is remote from the influence zone of existing clusters or a concept drift develops. A new
cluster is crafted by assigning the current sample of interest as the cluster’s center \( C_{\text{Clus}_{i+1}}^i = h_i \) and setting the cluster’s cardinality to be \( N_{\text{Clus}_{i+1}} = 1 \).

2) Evolution of Network Structure: The classifier of LEOPARD is equipped with the hidden node growing and pruning strategies adapting to the concept drifts of data streams. That is, this mechanism takes place for both the source stream and the target stream. The self-organizing mechanism is controlled by the network significance (NS) method [22] adopting the bias-variance decomposition concept of every layer. That is, a high bias situation leads to an introduction of a new node, while a high variance condition triggers the node pruning mechanism. Note that the network bias and variance, here, are evaluated with respect to the local error of a layer. All of which are carried out in an unsupervised fashion with respect to the reconstruction error. The NS method is formalized as follows:

\[
\text{NS} = (E[\hat{h}_i] - h_i)^2 + (E[\hat{h}_i^2] - E[\hat{h}_i]^2).
\]  

Equation (18) can be solved by finding the expected output \( E[\hat{h}_i] \) under a certain probability density function \( p(x) \) assumed to follow the normal distribution \( N(\mu, \sigma^2) \) with mean \( \mu \) and variance \( \sigma^2 \). The bottleneck of this approach is found in the case of drift \( p(x)_t \neq p(x)_{t-1} \) where it does not keep pace with rapidly changing distributions. To correct this shortcoming, the autonomous Gaussian mixture model (AGMM) can be used to estimate a complex probability density function \( p(x) \), as done in [6]. It is computationally expensive and often unstable in the high input dimension case due to the use of product norms. Furthermore, we deal with a multilayer network here doubling the complexity of AGMM.

The hidden unit growing and pruning steps are signaled by the statistical process control (SPC) approach [23] commonly used for anomaly detection tasks. The SPC method is applied here to detect the high bias or high variance condition and written as follows:

\[
\begin{align*}
\mu_{\text{bias}}^{n,l} + \sigma_{\text{bias}}^{n,l} & \geq \mu_{\text{min}}^{n,l} + k_1 \sigma_{\text{bias}}^{n,l} \\
\mu_{\text{var}}^{n,l} + \sigma_{\text{var}}^{n,l} & \geq \mu_{\text{min}}^{n,l} + k_2 \sigma_{\text{var}}^{n,l}.
\end{align*}
\]  

The SPC method is generalized here using \( k_2 = 1.3 \exp(-\text{Bias}^2) + 0.7 \) and \( k_3 = 1.3 \exp(-\text{Var}^2) + 0.7 \). This modification leads to dynamic confidence levels enabling flexible growing and pruning phases. That is, the node growing process is likely performed in the case of a high bias while being strict in the case of a low bias. The same case also applies to the node pruning mechanism. \( \mu_{\text{bias}}^{n,l} \) and \( \sigma_{\text{bias}}^{n,l} \) are reset if the growing condition (19) is satisfied. On the other hand, if the pruning condition is met, \( \mu_{\text{var}}^{n,l} \) and \( \sigma_{\text{var}}^{n,l} \) are reset. The initialization of a new node is carried out using the Xavier initialization strategy. The least contributing node having the least statistical contribution is subject to the pruning step if (20) is observed. Since LEOPARD is constructed under a different-depth structure where every layer performs its own clustering mechanism and produces its local output, the growing and pruning steps are independently undertaken per layer. Furthermore, this mechanism occurs for both source and target streams to anticipate the asynchronous drift problem where the network structure is shared across two domains.

The classifier of LEOPARD is fit with the hidden layer growing mechanism where it expands the network depth based on the drift detection mechanism [24]. The drift detection mechanism is designed from the concept of Hoeffding’s bound and analyzes the dynamics of latent features \( Z \) to identify the change of marginal distribution. Note that no labeled samples are offered for model updates, and the drift detection approach is executed for both source and target streams. The addition of a network layer is desired in practice because it is capable of substantially increasing network capacity significantly, thus enhancing the model’s generalization. The drift detection procedure starts by finding the cutting point, a point where the population mean increases. A cutting point is declared by the following condition:

\[
\hat{P} + \epsilon_p \geq \hat{Q} + \epsilon_q
\]  

where \( P \in \mathbb{R}^{2N} \) is a data matrix containing two consecutive data batches \( \{B_{k-1}, B_k\} \), i.e., previous and current data batches, while \( Q \in \mathbb{R}^{2N} \) is a data matrix with cut as the hypothetical cutting point of interest, cut < 2N. Two data batches are applied here to increase the sensitivity of cutting point identification because latent features are relatively stable compared to the original input space. The hypothetical cutting point is arranged as cut = [25%, 50%, 75%] × 2N instead of every point to avoid false alarm. \( \hat{P} \) and \( \hat{Q} \) denote the statistics of data matrices \( P \) and \( Q \). \( \epsilon_p, \epsilon_q \) stand for the error bound derived from the concept of Hoeffding’s bound as follows:

\[
\epsilon_{P,Q} = \sqrt{\frac{1}{2 \times \text{size}} \ln \frac{1}{\alpha_s}}
\]  

where \( \alpha_s \) is the significance level being inversely proportional to the confidence level 1 − \( \alpha_s \). While size refers to the size of the data matrix of interest \( P \) and \( Q \).

Once eliciting the cutting point of interest cut, a data matrix \( R \in \mathbb{R}^{2N-\text{cut}} \) is constructed. A drift is signaled if \( |R - \bar{Q}| \geq \epsilon_d \). Besides the drift condition, a warning condition is set and pinpoints a case where a drift needs to be confirmed by the next data batch. That is, \( \epsilon_w \leq |R - \bar{Q}| \leq \epsilon_d \), where \( \alpha_w < \alpha_d \).

The error bounds \( \epsilon_{D,W} \) are defined as follows:

\[
\epsilon_{D,W} = (b - a) \times \sqrt{\frac{\text{size} - \text{cut}}{2 \times \text{cut} \times \text{size}}} \ln \frac{1}{\alpha_{D,W}}
\]  

where \( [a, b] \) denotes the range of the data matrix \( P \). A new layer is created if a concept drift is found. That is, the number of nodes is set as half of the network width of the previous layer \( l - 1 \). This step enables the nonlinear feature reduction and avoids an overcomplete network. The domain classifier and the feature extractor have a fixed structure because the structural learning of the classifier suffices to address the asynchronous drift problem.

D. Algorithm

The learning policy of LEOPARD is visualized in Fig. 3 and Algorithm 1 where LEOPARD is driven by the feature extractor, the classifier, and the domain classifier. The forward pass procedure is done by feeding raw input attributes \( x_{S,T} \).
Algorithm 1 LEOPARD

Input: Source streaming data \( \{B_{S}^{1}, B_{S}^{2}, \ldots, B_{S}^{K_{S}}\} \), target streaming data \( \{B_{T}^{1}, B_{T}^{2}, \ldots, B_{T}^{K_{T}}\} \), initialization epochs \( E_{init} \), batch number of source and target streaming data \( b_{k} \), epoch number \( E \).

Output: Network parameters of feature extractor \( W_{f} \), classifier \( W_{c} \) and domain classifier \( W_{dc} \).

1) Initialize clustering using scarcity labelled data \( B_{0}^{S} \).

2) Initializing clusters using scarcity labelled data \( B_{0}^{S} \).

3) Average accuracy \( Acc \).

4) for \( i = 1 : E_{init} \) do

5) Network layer evolution of classifier (SAE) \( \zeta_{W_{c}} \) by Eq. (23);

6) Hidden unit of classifier (SAE) \( \zeta_{W_{c}} \) growing and pruning by Eq. (19) and (20);

7) \( L_{cluster} = L_{c}(x_{S,T}, h_{S,T}) + \sum_{l=1}^{L_{c}} (L_{c}(h_{S,T}, h_{S,T}) + \alpha_{2} K L(\phi^{l} | \Phi^{l})) \);

8) Update feature extractor parameter \( W_{f} \) and classifier parameter \( W_{c} \) in respect to \( L_{cluster} \);

9) \( L_{cd} = \frac{1}{N_{S}} \sum_{n=1}^{N_{S}} L_{c}(\zeta_{W_{c}}(f_{W_{c}}(x_{n})), d_{n}) + \frac{1}{N_{T}} \sum_{n=1}^{N_{T}} L_{c}(1 - \zeta_{W_{c}}(f_{W_{c}}(x_{T})), d_{n}) \);

10) Update feature extractor parameter \( W_{f} \) and domain classifier parameter \( W_{dc} \) in respect to \( L_{cd} \);

11) end

12) return \( W_{f}, W_{c}, W_{dc} \) and average accuracy \( Acc \);

The structural learning process occurs in both the initialization phase and the main training phase in which it includes the cluster growing process, the hidden node growing and pruning processes, and the hidden layer growing process. As with the warm-up phase, the initialization phase using \( E_{init} \) prerecorded samples over \( E \) epochs is implemented if a new layer is created. It is obvious that LEOPARD does not exploit any labeled samples for model updates except for labeled samples to be used to calculate the cluster allegiance (5). The structural learning mechanism addresses the issue of asynchronous drifts across both streams.

V. NUMERICAL STUDY

This section presents numerical validation of LEOPARD putting forward nine datasets leading to 24 independent numerical results. An ablation study is added in this section to further numerically validate the contribution of each learning component. Source codes of LEOPARD can be found in https://github.com/wengweng001/LEOPARD.git. Our analysis of label proportions and visualizations of LEOPARD’s learning performances is offered in the Supplementary Material.

A. Dataset

1) MNIST (MN) ↔ USPS (US): This problem presents a digit recognition problem having ten classes. The data samples are formed by gray-scale images of hand-written digits resized to \( 28 \times 28 \) for US → MN and \( 28 \times 28 \) for MN → US cases.

2) Amazon@X(AM): This is a multidomain sentiment analysis problem encompassing product reviews obtained from Amazon.com. \( X \) stands for the product type [25]. Five product types, namely, beauty, books, industrial, luxury, and magazine, are selected here where the cross domain multistream classification problem is formulated with two products with similar contexts but different topics. The averaged summed outputs from Google’s word2vec model pretrained on 100 billion words [26] are used to perform feature extraction.

3) Office31: This problem presents three domains: amazon (A), DSLR (D), and Webcam (W). It comprises 31 categories of office objects. We present the case of D ↔ W, where D comprises 498 images and W consists of 795 images. The characteristics of nine datasets are summed up in Table I.

B. Simulation Protocol

The numerical study is carried out using the prequential test-then-train protocol as per [23]. A model is tested first before updating it with the same data stream. The numerical evaluation is independently undertaken per batch where the numerical result is averaged across all batches. Our simulation is repeated five times to guarantee the consistency of numerical results where the final numerical results are averaged over five independent runs. The asynchronous drift problem is induced by applying the scaling hyperplane strategy [27], [28] where a data stream is scaled to \( x_{i} = ((d_{z} \times x_{i})/|x||) \). \( d_{z} \) is a randomly generated concept drift vector where \( z \) is the number of concept drifts in the stream: \( z = 1 \) for every source stream and \( z = 1 \) for every target data stream. A fixed random seed is
with a specific domain adaptation loss function, while a clustering loss and a reconstruction loss. ADCN is not equipped with a combination of a clustering loss and the adversarial domain adaptation approach as per LEOPARD without any clustering mechanism.

All of them work under the extreme label scarcity condition as with LEOPARD where access to true class labels is only provided for the pretrained samples of the source stream, while no label is offered during the process runs for both the source stream and the target stream. Comparison with ADCN is done by executing their published codes to assure fair comparisons. We utilize our own implementations of DCN, AE+KMeans, DEC, and DANN.

D. Hyperparameters

The learning rate and momentum of LEOPARD are allocated as 0.01 and 0.95, while the regularization constant of the cross domain loss \( \alpha_1 \) is set as 0.1. LEOPARD also depends on labeled pretrained samples \( B_0 \) of the source stream set as 10% of source samples proportionally taken from each class \( N_m = 10\% N_S \). That is, each class contributes the same number of samples. The numbers of initial epochs are set as \( E = 100 \) (amazon@X), \( E = 50 \) (MN ↔ US), and \( E = 500 \) (D ↔ W), respectively. The initialization phase is carried out using labeled pretrained samples of the source stream. The parameters of the drift detectors \( \alpha_s, \alpha_D, \) and \( \alpha_W \) are selected, respectively, as 0.001, 0.001, and 0.005. For amazon@X problems, LEOPARD runs in the one-pass learning procedure, whereas, for MN ↔ US experiments, the training process of the clustering loss adopts the epoch per batch strategy with 10 (MN → US, W → D) epochs and 5 epochs (US → MN), respectively. The epoch per batch strategy satisfies the online learning requirement because a data batch is discarded after training over predetermined epochs. The same setting is also applied to the baseline algorithms assuring fair comparisons.

For the MN ↔ US problem, the feature extractor is formed as convolutional neural networks. The encoder part is constructed as two convolutional layers using 16 and 4 filters, respectively, while having the max-pooling layer in between. The decoder part is built upon two transposed convolutional layers with 4 and 16 filters, respectively. For amazon@X sentiment analysis problems, the multilayer perceptron feature extractor is put forward with two hidden layers where the number of nodes is fixed as 300 and 100. For the Office31 problem, ResNet34 is applied as feature extractors. The initial nodes of the fully connected layer are simply assigned as 96 for the MNIST ↔ USPS problem, 30 for amazon@X sentiment analysis problems, and 500 for D ↔ W. The ReLU activation function is applied for the intermediate layers, while the decoder output utilizes the sigmoid activation function producing normalized reconstructed output. The network structures of baseline algorithms are set similar to ensure a fair comparison. Further details of our numerical studies are explained in the LEOPARD’s codes shared in https://github.com/wengweng001/LEOPARD.git

These parameters are fixed throughout all study cases to guarantee non ad hoc performance of LEOPARD. The hyperparameters of the baselines are selected as per the guidelines of their publications and hand-tuned if their performances are surprisingly compromised. The hyperparameters of all consolidated algorithms are listed in the Supplementary Material.

E. Numerical Results

From Table II, it is seen that LEOPARD outperforms other algorithms in 15 of 24 cases with noticeable margins. This aspect portrays the efficacy of the adversarial domain adaptation approach and the soft-cluster assignment mechanism where these two modules are absent in the baseline algorithms, i.e., ADCN, DCN, and AE+KMeans adopt the hard-cluster assignment strategy and suffer from the absence of the adversarial domain adaptation approach. This finding also confirms the advantage of the adversarial domain adaptation approach.
TABLE II
AVERAGE ACCURACY (%) OF THE TARGET STREAM ACROSS FIVE RUNS; *INDICATES STATISTICALLY SIGNIFICANT RESULTS, AND BOLD DENOTES THE BEST NUMERICAL RESULTS.

| Experiments | LEOPARD | ADCN | AE-kmeans | DCN | DEC | DANN |
|-------------|---------|------|-----------|-----|-----|------|
| AM1 → AM2  | 20.6520 ± 2.3598 | 19.8160 ± 4.7555 | 27.8012 ± 1.6392 | 27.7792 ± 1.8319 | 18.3774 ± 1.3356 | 15.2291 ± 22.7966 |
| AM1 → AM3  | *21.5300 ± 1.0819 | 57.0520 ± 10.7930 | 25.8773 ± 1.6412 | 26.1010 ± 1.9047 | 17.2324 ± 1.5599 | 34.8559 ± 29.1757 |
| AM1 → AM4  | *57.7840 ± 0.0476 | 43.9800 ± 3.3004 | 27.9307 ± 1.2602 | 28.1326 ± 1.1633 | 16.6092 ± 0.6464 | 44.4809 ± 20.6428 |
| AM1 → AM5  | *63.5100 ± 1.2016 | 60.7980 ± 3.0386 | 31.3240 ± 0.8684 | 31.9996 ± 0.8381 | 13.9947 ± 1.0473 | 41.5806 ± 13.6181 |
| AM2 → AM1  | *71.5400 ± 8.0301 | 25.2880 ± 6.6832 | 36.7685 ± 1.5799 | 36.9540 ± 2.0903 | 8.8334 ± 0.9026 | 49.4693 ± 39.4318 |
| AM2 → AM3  | 45.4920 ± 13.2055 | 19.9380 ± 15.1533 | 31.0612 ± 1.8636 | 30.9986 ± 1.8202 | 14.2652 ± 1.0375 | 43.3046 ± 25.3212 |
| AM2 → AM4  | *48.5600 ± 3.5658 | 14.1460 ± 1.7088 | 27.1297 ± 1.0116 | 27.2251 ± 0.9788 | 15.6150 ± 1.4221 | 42.8489 ± 14.2926 |
| AM2 → AM5  | 50.3680 ± 15.8943 | 31.9610 ± 5.8286 | 28.7212 ± 1.6700 | 28.4330 ± 1.8060 | 18.3333 ± 2.6212 | 60.4227 ± 8.1435 |
| AM3 → AM1  | 37.3520 ± 4.7246 | 53.0180 ± 17.7237 | 25.7504 ± 1.2193 | 25.4951 ± 1.4807 | 7.7442 ± 1.5116 | 17.1871 ± 19.2744 |
| AM3 → AM2  | 31.3120 ± 8.9392 | 8.9900 ± 2.0600 | 22.1118 ± 1.1185 | 25.3291 ± 5.8252 | 17.2165 ± 1.6547 | 40.7979 ± 13.1832 |
| AM3 → AM4  | 18.7240 ± 1.0962 | 28.4340 ± 4.5292 | 23.2826 ± 1.4628 | 23.1707 ± 1.6232 | 15.2796 ± 2.2464 | 22.2822 ± 17.1165 |
| AM4 → AM1  | *59.5520 ± 2.8339 | 37.1540 ± 8.1052 | 22.1986 ± 0.7979 | 22.3941 ± 0.6026 | 16.0360 ± 0.2062 | 20.1265 ± 21.2838 |
| AM4 → AM3  | 45.5500 ± 6.3118 | 69.4040 ± 6.0339 | 23.2919 ± 3.1591 | 23.4662 ± 3.0392 | 8.0683 ± 0.7488 | 55.4146 ± 40.1126 |
| AM4 → AM2  | 23.2340 ± 5.5658 | 62.6160 ± 3.7864 | 22.4032 ± 1.3615 | 22.6676 ± 1.4520 | 17.2892 ± 1.9414 | 54.9784 ± 22.9740 |
| AM5 → AM1  | *64.3840 ± 0.8955 | 56.0280 ± 2.6346 | 21.3376 ± 1.1374 | 21.3101 ± 0.9658 | 15.5601 ± 1.4566 | 20.0455 ± 13.7290 |
| AM5 → AM3  | *87.6760 ± 0.3484 | 56.3760 ± 19.8715 | 20.1306 ± 1.2201 | 20.4453 ± 0.8831 | 8.8645 ± 1.4946 | 61.0945 ± 17.2396 |
| AM5 → AM2  | 12.5000 ± 1.1611 | 10.8880 ± 3.7396 | 19.3033 ± 1.0140 | 19.6829 ± 0.8429 | 15.7549 ± 2.1423 | 36.7490 ± 25.9903 |
| AM5 → AM4  | *56.7900 ± 6.2585 | 27.9480 ± 3.3931 | 21.4889 ± 2.2889 | 21.1898 ± 2.2209 | 16.9898 ± 3.9668 | 51.0357 ± 32.7811 |
| AM5 → AM1  | *50.7580 ± 2.7002 | 32.5880 ± 4.3853 | 19.8205 ± 0.7625 | 19.5470 ± 0.6045 | 16.0969 ± 1.2545 | 33.3787 ± 21.3489 |

The standard for statistical significance is based on the T-Test score between LEOPARD and other baselines, 4 T scores for each experiment. We determined the result as statistically significant if t score is greater than 2.015 for at least 3 out of 4.

F. Ablation Study

This section studies the effect of LEOPARD’s learning modules by analyzing their performance when deactivating a particular learning module. LEOPARD is configured into four models.

1) The structural learning method is switched OFF leaving LEOPARD to have a static network structure. In addition, the parameter learning step utilizes both the cross domain adaptation loss and the KL divergence loss. LEOPARD is driven by the reconstruction loss only.

2) The structural learning strategy of LEOPARD is deactivated, while the parameter learning step utilizes both the cross domain adaptation loss and the KL divergence loss.

over the feature reconstruction strategy with shared parameters across the two streams implemented in all baselines. The soft-cluster assignment approach where the cluster and network parameters are simultaneously optimized via the SGD method performs better than the hard assignment strategy. It is demonstrated by the fact that LEOPARD beats ADCN in significant numbers of cases. There is no significant performance difference between AE-kMEANS and DEC. On the other hand, the importance of the structural learning component in handling data streams is clearly portrayed here where LEOPARD and ADCN are superior compared to other algorithms having static structures. Such a mechanism allows timely reactions to the asynchronous drift problem across the source stream and the target stream. The performance of DANN implemented under conventional neural network structures is far inferior to LEOPARD under extreme label scarcity conditions. This fact confirms the advantage of the clustering approach compared to the conventional neural network structure to reduce label’s dependencies. The statistical test is undertaken using the t-test (P < 0.05) confirming the advantage of LEOPARD where it beats other algorithms with statistically significant gaps in 13 of 24 cases.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
3) The structural learning mechanism of LEOPARD is activated but with the absence of the KL divergence loss and the cross domain adaptation loss.

4) BERT is applied as a feature extractor in lieu of a word2vec model. Our ablation study is carried out using four study cases: AM1 → AM3, AM1 → AM4, AM5 → AM1, and AM5 → AM4.

From Table III, LEOPARD suffers from major performance degradation for all configurations 1) - 4) confirming the efficacy of its current version. For AM5 → AM1, in configuration 1), the structural learning strategy, the KL divergence loss, and the cross domain adaptation loss are absent producing poor results. The performance improves slightly with the activation of the KL divergence loss and the cross domain adaptation loss as per configuration 2). Although the KL divergence loss and the cross domain adaptation loss are deactivated, the structural learning mechanism improves the accuracy as per configuration 3) but is not yet on par with the LEOPARD. An interesting observation presents in the case of AM5 → AM4 where configuration 3) produces poor results. The performance betters in configurations 1) and 2) without the structural learning mechanism. Nevertheless, configurations 1) - 3) are not comparable to the LEOPARD where all modules are engaged. We note that the size of the source stream is much less than the size of the target stream in the AM5 → AM4 case. This leads to very few labeled samples provided in the warm-up phase. For AM1 → AM4, the absence of structural evolution, configuration 2), drops the LEOPARD's accuracy slightly. More severe degradation than configuration 2) occurs in configurations 1) and 3) where the KL divergence loss and the cross domain adaptation loss are deactivated. The same pattern is demonstrated in the case of AM1 → AM3. This finding clearly confirms the advantage of the KL divergence loss and the cross domain adaptation loss for LEOPARD. In addition, the structural evolution boosts the performance of LEOPARD, especially in the presence of drifts.

The use of BERT as a feature extractor, as shown in Configuration 4), worsens the predictive performances of LEOPARD significantly. This finding confirms the compatibility of the word2vec model over the BERT model as a feature extractor of LEOPARD most likely due to the absence of a self-attention mechanism or recurrent connection in LEOPARD.

G. t-SNE Plots

Fig. 4 illustrates the t-SNE plots of LEOPARD [18] for the USPS → MNIST case on the target stream before and after the training process. It is observed that there does not exist any cluster structures initially in this problem, but such cluster structures are clearly present after the training process showing the effectiveness of (14). These facts confirm that LEOPARD does not call for the existence of any cluster structures beforehand, and the clustering loss $L_{\text{cluster}}$ is capable of establishing the clustering-friendly latent space.

H. Future Directions

This article has developed an algorithmic solution to multistream classification problems under extreme label shortages, LEOPARD. That is, given two different but related streaming processes, LEOPARD properly functions with few prerecorded samples of the source domain and the absence of any labels when the streaming processes run. This benefit goes one step ahead of existing multistream classifiers or unsupervised domain adaptation methods calling for fully labeled source streams. Nonetheless, the problem of multistream classification remains at the infant stages leaving several open issues for future works.

The problem of open set domain adaptation presents a case where the source and target domains do not share the same target classes [31]. Such a setting is also seen as a way to reduce the labeling cost and is beneficial in the realm of multistream classifications. Busto and Gall [31] propose a feature transformation strategy associating target classes of the target domain to those of the source domain. Liu et al. [32] put forward the concept of openness and unknown classes in the open set domain adaptation problem. The theoretical bound of the open set domain adaptation is derived in [14]. The application of the theoretical bound for deep learning is demonstrated in [33]. These approaches are limited to the off-line case calling for extensions for the multistream classification setting. Gradual domain adaptation [34] is highly relevant to the multistream classification context because the multistream classification problem still considers different but related streams and constant discrepancies. Also, the asynchronous drifts usually appear suddenly. Equation (1) assumes a small and fixed combined risk and is unrealistic because the combined risk may increase during the training process [35]. This issue is still ignored in the multistream classification problems. Few-shot hypothesis adaptation [36] is another interesting direction for the multistream classification topic and extends the few-shot domain adaptation problem without any source domain data.

VI. Conclusion

LEOPARD is proposed in this article to cope with the cross domain multistream classification problems under the lack of labeled samples. The advantage of LEOPARD has been numerically validated using 24 study cases combined from nine datasets. It is demonstrated that LEOPARD outperforms its counterparts with noticeable margins in 15 of 24 cases. The ablation study further confirms the efficacy of LEOPARD's learning modules. One limitation of LEOPARD lies in the adversarial domain adaptation strategy only performing domain's alignment. This approach is poor when there exist
big conditional distribution discrepancies. This issue is rather tricky here because LEOPARD does not benefit from any labels for its model updates. Our initial insight shows the feasibility of the pseudolabeling strategy to attack this problem by improving class inferences. Noisy labels remain an open issue and are explored in the future.

REFERENCES

[1] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, “An adaptive framework for multistream classification,” in Proc. 25th ACM Int. Conf. Inf. Knowl. Manage., New York, NY, USA, Oct. 2016, pp. 1181–1190.

[2] P. Zhao, S. C. H. Ho, J. Wang, and B. Li, “Online transfer learning,” Artif. Intell., vol. 216, pp. 76–102, Nov. 2014.

[3] Y. Ganin et al., “Domain-adversarial training of neural networks,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 2030–2096, 2016.

[4] A. Haque, Z. Wang, S. Chandra, B. Dong, L. Khan, and K. W. Hamlen, “FUSION: An online method for multistream classification,” in Proc. ACM Conf. Inf. Knowl. Manage., New York, NY, USA, 2017, pp. 919–928.

[5] B. Dong, S. Chandra, Y. Gao, and L. Khan, “Multistream classification with relative density ratio estimation,” in Proc. AAAI, 2019, pp. 3478–3485.

[6] M. Pratama, M. de Carvalho, X. Renchunzi, E. Lughofer, and H. Du, “ATL: Autonomous knowledge transfer from many streaming processes,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 3861–3870.

[7] H. Du, L. L. Minku, and H. Zhou, “Multi-source transfer learning for non-stationary environments,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2019, pp. 1–8.

[8] H. Du, L. L. Minku, and H. Zhou, “MARLINE: Multi-source mapping transfer learning for non-stationary environments,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2020, pp. 122–131.

[9] X. Renchunzi and M. Pratama, “Automatic online multi-source domain adaptation,” Inf. Sci., vol. 582, pp. 480–494, Jan. 2022.

[10] F. Liu, G. Zhang, and J. Lu, “Multisource heterogenous unsupervised domain adaptation via fuzzy relation neural networks,” IEEE Trans. Fuzzy Syst., vol. 29, no. 11, pp. 3308–3322, Nov. 2021.

[11] H. Tao, Z. Wang, Y. Li, M. Zamani, and L. Khan, “COMC: A framework for online cross-domain multistream classification,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2019, pp. 1–8.

[12] Y. Zhang, F. Liu, Z. Fang, B. Yuan, G. Zhang, and J. Lu, “ClariNet: A one-step approach towards budget-friendly unsupervised domain adaptation,” in Proc. 29th Int. Joint Conf. Artif. Intell. (IJCAI), C. Bessiere, Ed. Yokohama, Japan, Jul. 2020, pp. 2526–2532.

[13] F. Liu, G. Zhang, and J. Lu, “Heterogeneous domain adaptation: An unsupervised approach,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 12, pp. 5588–5602, Dec. 2020.

[14] Z. Fang, J. Lu, F. Liu, J. Xuan, and G. Zhang, “Open set domain adaptation: Theoretical bound and algorithm,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 10, pp. 4309–4322, Oct. 2021.

[15] F. Liu, J. Lu, B. Han, G. Niu, G. Zhang, and M. Sugiyama, “Butterfly: One-step approach towards wildly unsupervised domain adaptation,” 2019, arXiv:1905.07720.

[16] 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.

[17] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in Proc. 33rd Int. Conf. Mach. Learn., vol. 48, 2016, pp. 478–487.

[18] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, Nov. 2008.

[19] J. Smith, C. Taylor, S. Baer, and C. Dovrolis, “Unsupervised progressive learning and the STAM architecture,” in Proc. 30th Int. Joint Conf. Artif. Intell. (IJCAI), Z.-H. Zhou, Ed., Aug., 2021, pp. 2979–2987, doi: 10.24963/ijcai.2021/410.

[20] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, “Analysis of representations for domain adaptation,” in Proc. 19th Int. Conf. Neural Inf. Process. Syst. Cambridge, MA, USA: MIT Press, 2006, pp. 137–144.

[21] Z. Wang, Z. Kong, S. Changra, H. Tao, and L. Khan, “Robust high dimensional stream classification with novel class detection,” in Proc. IEEE 35th Int. Conf. Data Eng. (ICDE), Apr. 2019, pp. 1418–1429.

[22] M. Pratama, C. Za’in, A. Ashfahani, Y. S. Ong, and W. Ding, “Automatic construction of multi-layer perceptron network from streaming examples,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., Nov. 2019, pp. 1171–1180.

[23] J. Gama, Knowledge Discovery From Data Streams, 1st ed. Boca Raton, FL: CRC Press, 2010.

[24] M. Pratama, W. Pedrycz, and G. I. Webb, “An incremental construction of deep neuro fuzzy system for continual learning of nonstationary data streams,” IEEE Trans. Fuzzy Syst., vol. 28, no. 7, pp. 1315–1328, Jul. 2020.

[25] J. Ni, J. Li, and J. McAuley, “Justifying recommendations using distantly-labeled reviews and fine-grained aspects,” in Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Conf. Joint Natural Lang. Process. (EMNLP-IJCNLP), Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 188–197.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013, arXiv:1301.3781.

[27] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” ACM Comput. Surv., vol. 46, no. 4, pp. 1–44, Apr. 2014.

[28] M. de Carvalho, M. Pratama, J. Zhang, and E. K. Y. Yapp, “ACDC: Online unsupervised cross-domain adaptation,” 2021, arXiv:2110.01326.

[29] A. Ashfahani and M. Pratama, “Unsupervised continual learning in streaming environments,” IEEE Trans. Neural Netw. Learn. Syst., early access, Apr. 13, 2022, doi: 10.1109/TNNLS.2022.3163362.

[30] B. Yang, X. Fu, N. D. Sidoropoulos, and M. Hong, “Towards K-means-friendly spaces: Simultaneous deep learning and clustering,” in Proc. 34th Int. Conf. Mach. Learn. (Proceedings of Machine Learning Research), vol. 70, D. Precup and Y. W. Teh, Eds. Sydney, NSW, Australia: PMLR, Aug. 2017, pp. 3861–3870.

[31] P. P. Busto and J. Gall, “Open set domain adaptation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 754–763.

[32] H. Liu, Z. Cao, M. Long, J. Wang, and Q. Yang, “Separate to adapt: Open set domain adaptation via progressive separation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2922–2931.

[33] L. Zhong, Z. Fang, F. Liu, B. Yuan, G. Zhang, and J. Lu, “bridging the theoretical bound and deep algorithms for open set domain adaptation,” IEEE Trans. Neural Netw. Learn. Syst., pp. 1–15, 2021.

[34] A. Kumar, T. Ma, and P. Liang, “Understanding self-training for gradual domain adaptation,” 2020, arXiv:2002.11361.

[35] L. Zhong, Z. Fang, F. Liu, J. Lu, B. Yuan, and G. Zhang, “How does the combined risk affect the performance of unsupervised domain adaptation approaches?” 2020, arXiv:2101.01104.

[36] H. Chi et al., “TOHAN: A one-step approach towards few-shot hypothesis adaptation,” 2021, arXiv:2106.06526.