Unsupervised Unlearning of Concept Drift with Autoencoders

André Artelt[a,b,c], Kleanthis Malialis[b, c], Christos G. Panayiotou[b, c] Marios M. Polycarpou[b, c] Barbara Hammer[a],

[a] *Faculty of Technology*
Bielefeld University, Bielefeld, Germany
Email: {aartelt, bhammer}@techfak.uni-bielefeld.de
ORCID: {0000-0002-2426-3126, 0000-0002-0935-5591}

[b] *KIOS Research and Innovation Center of Excellence*
[c] *Department of Electrical and Computer Engineering*
University of Cyprus, Nicosia, Cyprus
Email: {malialis.kleanthis, christosp, mpolycar}@ucy.ac.cy
ORCID: {0000-0003-3432-7434, 0000-0002-6476-9025, 0000-0001-6495-9171}

***Abstract*—The phenomena of concept drift refers to a change of the data distribution affecting the data stream of future samples – such non-stationary environments are often encountered in the real world. Consequently, learning models operating on the data stream might become obsolete, and need costly and difficult adjustments such as retraining or adaptation. Existing methods to address concept drift are, typically, categorised as active or passive. The former continually adapt a model using incremental learning, while the latter perform a complete model retraining when a drift detection mechanism triggers an alarm. We depart from the traditional avenues and propose for the first time an alternative approach which "unlearns" the effects of the concept drift. Specifically, we propose an autoencoder-based method for "unlearning" the concept drift in an unsupervised manner, without having to retrain or adapt any of the learning models operating on the data.**

***Index Terms*—concept drift unlearning, autoencoders, data streams, nonstationary environments.**

I. Introduction

Concept drift is undoubtedly one of the major challenges encountered by learning algorithms in data stream mining [1]. It refers to the problem of dealing with a nonstationary data distribution that evolves over time as data continually arrive in a streaming manner. If not properly addressed, learning models may become obsolete with severe practical consequences.

Consider, for example, a domain area in which multiple learning models have been trained, each specialised for a particular downstream task, as shown in Figure 1a. For instance, in water distribution networks [2] each model (M) corresponds to a different downstream task (T), such as, predicting water pressure at a node from neighbouring nodes with pressure sensors installed, detecting water leakage, and identifying the leakage location (i.e., isolation). In the presence of concept drift, most - if not all - downstream task models will be incapable of adequately performing their corresponding tasks. Furthermore, we emphasise that different types of models might have been trained, i.e., regression (pressure prediction) and classification (leakage detection and isolation) models, as well as different learning paradigms might have been used, i.e., supervised learning (pressure prediction, leakage isolation) and unsupervised learning / anomaly detection (leakage detection).

In the presence of drift, to maintain optimality in all tasks, each model should be updated or retrained (M'); this is shown with arrows in Figure 1a. Numerous methods have been proposed in the literature to adapt to drift [1], [3], which we review in a later section. Importantly, this traditional approach may be effective in small-scale scenarios, but it becomes impractical, costly, and fails to scale well in large scenarios.

The contributions made in this work are the following:

- To the best of our knowledge, we propose for the first time an alternative approach to addressing drift. We propose the *concept drift unlearning (CDU)* method which is responsible for learning (completely unsupervised) to revert the data distribution to the state it was before the concept drift had occurred, without the need to modify any of the existing models for each downstream task; this is depicted in Figure 1b.
- To realize the potential of the proposed approach and increase its practical impact, we propose to revert the effects of drift in an unsupervised learning manner using autoencoders, a special type of a neural network which learns to reconstruct its input. In other words, no ground truth information (e.g., labels in a classification task) is required from human experts.
- We conduct a rich experimental study, involving a diverse set of different domains and scenarios, and demonstrate the effectiveness of the proposed approach. Specifically, we evaluate our approach in 200 simulated data sets / scenarios from a realistic water distribution network (each corresponding to different drift characteristics) with more than 30 downstream tasks. Furthermore, evaluation

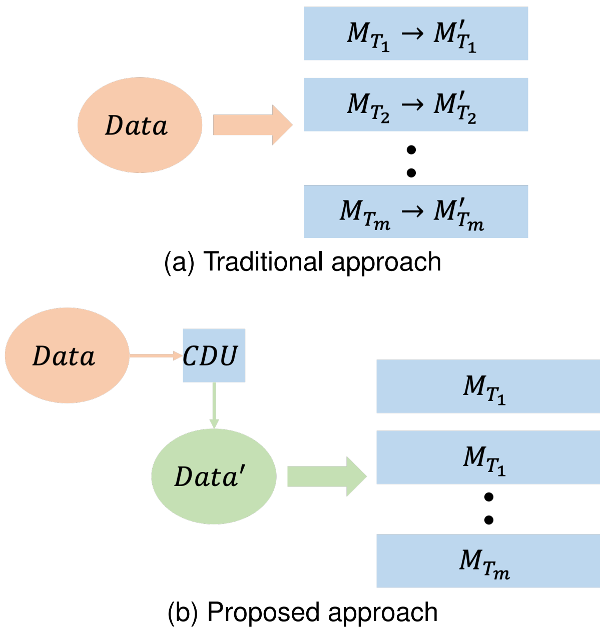


Fig. 1: Approaches to address concept drift.

is performed in an image-related task.

The rest of the paper is organized as follows. Section II presents the related work. Section III discusses our proposed method. Section IV describes our experimental study. Section V discusses important remarks and characteristics of the proposed method. Section VI concludes this work with some pointers to future directions.

II. Related Work

Existing methods on concept drift adaptation are, typically, categorised as active or passive [1], [3].

*1) **Passive** methods:* These methods implicitly adapt to concept drift by using incremental learning, that is, they continually update a learning model [4]. Passive methods make use of memory components (also referred to as **memory-based** methods), such as, a moving window which maintains a list of the most recent examples. Memory-based methods have the disadvantage of having to pre-determine the "right" memory size, therefore, mechanisms to resolve this have been introduced, such as, having an adaptive window [5]. Other passive methods use **ensembling** where a set of classifiers dynamically grows or shrinks to maintain good performance, such as, the Learn++.NSE method [6], and DDD [7] which maintains ensembles with different diversity levels, as a different level of ensemble diversity may be more appropriate before and after a drift for better generalisation.

A key challenge is class imbalance [8] and to address it, incremental learning has been used in combination with different mechanisms, e.g., memory-based models [9] like having one memory component per class [10], (adaptive) rebalancing [11], [12], and bagging [13], [14].

Furthermore, incremental learning has been used in conjunction with other learning paradigms, such as, active learning [15], [16] and unsupervised learning [17].

*2) **Active** methods:* These methods use an explicit mechanism for concept detection and are, typically, referred to as **change detection-based** methods. Some methods use statistical tests, such as independence tests [18], or JIT classifiers [19]) which propose two CUSUM-inspired tests capable of detecting abrupt and smooth changes. Some others use a threshold-based mechanism, e.g., the popular DDM [7] which uses one threshold value to raise a warning flag and another to trigger a drift alarm.

In contrast to passive methods which incrementally update a learning model, passive methods discard the existing model and create a new one which performs a complete retraining as soon as a concept drift alarm is triggered. Moreover, **ensembling** has also been used in active methods [20].

Hybrid methods combine the advantages of both approaches, for instance, HAREBA [21] combines a threshold-based drift detection mechanism with adaptive rebalancing to cope with class imbalance, while [15] uses explicit drift detection with incremental active learning. **Transfer learning** [22] can also be used for adapting to a changed distribution. For instance in [23], a method for dealing with sensor drift in hyperspectral cameras is proposed.

III. Proposed Method

Our method, which consists of two major parts, works completely unsupervised and therefore no labels are required and no assumptions on the downstream tasks are made:

1) **Distribution Learning:** We build an autoencoder that attempts to learn the data distribution so that we can later on distinguish between samples that were observed before/after a concept drift. We note that the proposed method is agnostic to the drift detection mechanism used. Our focus is not on the problem of concept drift detection, but rather assume that an effective and efficient concept drift detection mechanism is in place.
2) **Concept Drift Unlearning:** When concept drift is detected, we build a function that tries to "undo" the concept drift – i.e. we try to make samples, from after the concept drift, look like samples from before the concept drift by using the autoencoder built in the beginning.

The entire procedure is described in full detail in Algorithm 1.

A. Autoencoder for Distribution Learning

Concept drift refers to a change in the data distribution $p_{\mathcal{X}}$ which might affect downstream tasks $m(\cdot)$ that operate on the data from the domain $\mathcal{X}$. However, since the true distribution $p_{\mathcal{X}}$ is usually not known, it is estimated from the data. Having an estimate of $p_{\mathcal{X}}$ enables one not only to detect concept drift [3] but also to distinguish between samples before/after the concept drift – this, for instance, can be used to learn smth. about the concept drift itself [24].

We use an autoencoder $\text{ae} : \mathcal{X} \rightarrow \mathcal{X}$ for modeling the data distribution $p_{\mathcal{X}}$. This autoencoder $\text{ae}(\cdot)$ consists of an encoder enc : $\mathcal{X} \rightarrow \mathcal{X}'$ and a decoder dec : $\mathcal{X}' \rightarrow \mathcal{X}$, whereby $\mathcal{X}'$ denotes the space of the encoding:

$$\text{ae} : x \mapsto (\text{dec} \circ \text{enc})(x) \tag{1}$$

Learning an autoencoder Eq. (1) from a given data set $\mathcal{D} \subset \mathcal{X}$ means to find an encoder $\text{enc}(\cdot)$ and a decoder $\text{dec}(\cdot)$ such that the reconstruction loss $\ell(\cdot)$[1] is minimized:

$$\inf_{\text{enc}(\cdot),\, \text{dec}(\cdot)} \sum_{x_i \,\in\, \mathcal{D}} \ell\big(\underbrace{(\text{dec} \circ \text{enc})(x_i)}_{\hat{x}_i}, x_i\big) \tag{2}$$

In this work, both the encoder $\text{enc}(\cdot)$ and decoder $\text{dec}(\cdot)$ are parameterized functions and the problem of learning the autoencoder Eq. (2) can be rewritten as an optimization problem over the parameters θ_1 and θ_2:

$$\underset{\theta_1,\, \theta_2}{\arg\min}\, \frac{1}{|\mathcal{D}|} \sum_{x_i \,\in\, \mathcal{D}} \ell\big(\underbrace{(\text{dec}_{\theta_2} \circ \text{enc}_{\theta_1})(x_i)}_{\hat{x}_i}, x_i\big) \tag{3}$$

B. Concept Drift Unlearning – CDU

We assume that the data distribution $p_{\mathcal{X}}$ changes at some point in time, which then leads to a larger reconstruction error of the autoencoder Eq. (1). While using the autoencoder to discriminate between samples from before and after the drift could be an obvious choice as in [25], as mentioned, the proposed approach is agnostic to the concept drift detection mechanism $d(\cdot)$ used.

In order to "unlearn" the concept drift, we want to learn a mapping $f : \mathcal{X} \rightarrow \mathcal{X}$ such that the reconstruction loss becomes smaller again – we use the reconstruction error as a proxy for how well a given sample fits to the original data distribution. We infer this mapping $f(\cdot)$ from a given a data set of unlabeled samples $\mathcal{D}_* \subset \mathcal{X}$ under the new distribution (i.e. some samples observed after the concept drift). Finding such a function $f(\cdot)$ is modeled as the following optimization problem:

$$\inf_{f \,\in\, \mathcal{F}} \sum_{x_j \,\in\, \mathcal{D}_*} \ell\big(\underbrace{(\text{ae} \circ f)(x_j)}_{\hat{x}_j}, f(x_j)\big) \tag{4}$$

Note that it would also be possible to chain $f(\cdot)$ and the autoencoder together, however we propose not do this because otherwise the final function for undoing the drift would be computationally more complex and might therefore be less suited for real-time scenarios and scenarios where the "undoing" of the drift must be performed on devices with limited hardware capabilities.

In this work we parameterize the mapping $f(\cdot)$ – i.e. $f_\theta : \mathcal{X} \rightarrow \mathcal{X}$ –, and optimize over the parameters θ in order to find the final mapping $f(\cdot)$:

$$\underset{\theta}{\arg\min}\, \frac{1}{|\mathcal{D}_*|} \sum_{x_j \,\in\, \mathcal{D}_*} \ell\big(\underbrace{(\text{ae} \circ f_\theta)(x_j)}_{\hat{x}_j}, f_\theta(x_j)\big) \tag{5}$$

In order to avoid less useful solutions – e.g. the mapping $f(\cdot)$ might always yield the same output or might exploit some special particularities of the autoencoder –, we propose to add the regularization $C \cdot \|f_\theta(x_j) - x_j\|_1$ to the optimization problem Eq. (5):

$$\underset{\theta}{\arg\min}\, \frac{1}{|\mathcal{D}_*|} \sum_{x_j \,\in\, \mathcal{D}_*} \ell\big(\underbrace{(\text{ae} \circ f_\theta)(x_j)}_{\hat{x}_j}, f_\theta(x_j)\big) + C \cdot \|f_\theta(x_j) - x_j\|_1 \tag{6}$$

where $C \geq 0$ denotes regularization strengths which allow to balance between the different terms in the objective. The reasoning behind this regularisation is to minimize the number of features that are changed by $f(\cdot)$.

Algorithm 1 Unsupervised Unlearning of Concept Drift

Arguments: Data stream of unlabeled samples $(\vec{x}_t, \vec{x}_{t+1}, \dots)$; A set of already trained downstream tasks $\{m_i(\cdot)\}$

Solve Eq. (3) ▷ Build the autoencoder on a batch $\mathcal{D}$ of data before any concept drift happens.

Main:

if d(t)==True **then** ▷ Test for concept drift using some drift detection method $d(\cdot)$.

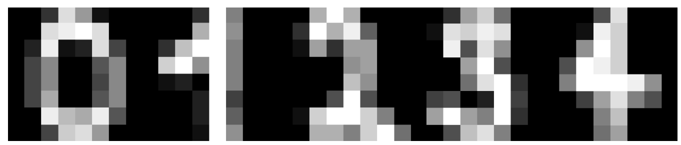
  Collect $\mathcal{D}_*$ ▷ Build a data set for "undoing" the drift.

  Solve Eq. (6) ▷ Build $f(\cdot)$ for "undoing" the drift.

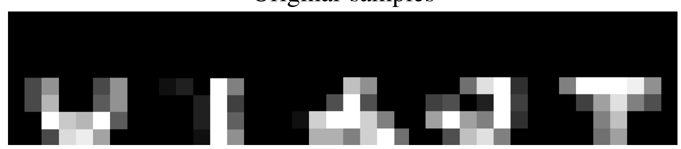
else

  $y_i = m_i(f(\vec{x}_t)) \quad \forall\, i$ ▷ Apply current sample $\vec{x}_t$ to $f(\cdot)$ and then to the downstream tasks $\{m_i(\cdot)\}$.

Fig. 2: Illustration of concept drift on the digits data set.



Original samples



Samples after concept drift

IV. Experiments

We empirically evaluate the performance of our proposed methodology on a set of diverse data sets – for technical implementation details of the experiments, we refer to the Python implementation which is publicly available on GitHub[2].

A. Data

1) Digits: We use the first five digits (i.e. 0-4) from [26] as a data set and build a digit classifier (this is the downstream task $m(\cdot)$) by using logistic regression. We introduce concept drift in the test data by setting all pixels in the upper half of

[1]E.g. mean-squared error or some other function for penalizing differences between two samples.

[2]https://github.com/HammerLabML/UnsupervisedUnlearningConceptDriftAutoencoders

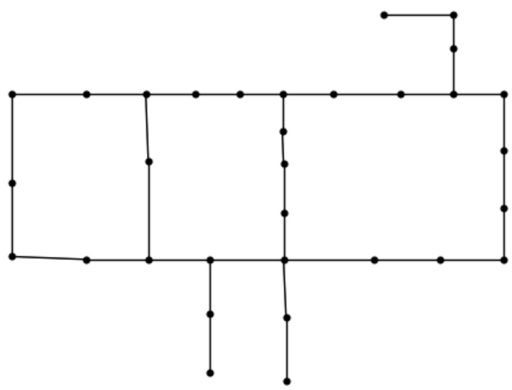


Fig. 3: Hanoi water distribution network [27].

the image to zero – see Fig. 2 for an illustration. We use a 10-fold cross validation to get statistically meaningful results.

2) Hanoi: In order to get statistically meaningful results, we generated a total number of 200 scenarios using the Hanoi water distribution network [27], which is shown in Figure 3. We place a pressure sensor at every node in the network (there are 32 nodes in the network) and build a virtual sensor for nearly every sensor[3] – i.e. we have 31 downstream regression tasks $m(\cdot)$. For each scenario we introduce concept drift by selecting a random pressure sensor to become faulty – i.e. placing a random sensor fault (with random parameters) somewhere in the network. To simulate real-world characteristics, we consider the following types of sensor faults [28]:

1) A constant offset of the sensors measurement compared to the true measured quantity.
2) Gaussian noise added to the sensor measurements.
3) Sensor power failures, which results to the measurement being equal to zero.
4) An offset of the sensor measurement, linearly proportional to the true measured quantity.

B. General Setup

The general setup of the experiments is the same for all data sets:

1) Train the downstream task models $m(\cdot)$ as well as the autoencoder $\mathrm{ae}(\cdot)$ on a time window without any concept drift.
2) Evaluate the downstream task models $m(\cdot)$ on the remaining samples before the concept drift occurs – and also on samples after the concept drift occurred for evaluating the influence of the concept drift on the downstream tasks.

[3]We exclude the first node because its pressure is always zero.

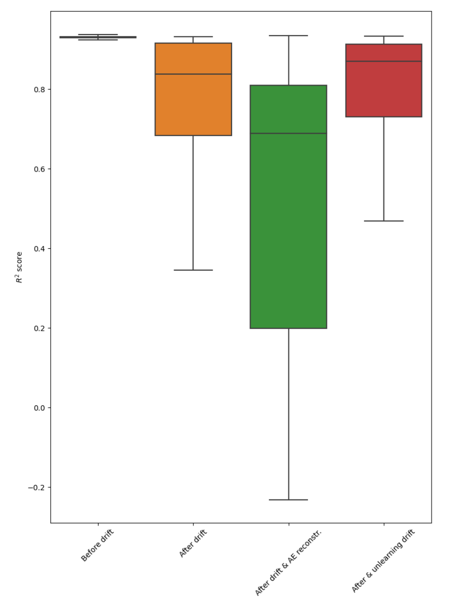


Fig. 4: Results on the Hanoi data set for downstream task 4 – we plot the R^2 scores over all scenarios.

Note that, for the purpose of evaluating our methodology, we do not build any concept drift detector but rather use the available ground truth.

3) Build the unsupervised concept drift unlearning function on a small time window after the concept drift occurred.
4) Evaluate the downstream task models $m(\cdot)$ after applying the undoing function $f(\cdot)$ on the samples after the concept drift.
5) As a baseline, we use the autoencoder $\mathrm{ae}(\cdot)$ as an implementation of $f(\cdot)$ – i.e. we evaluate if and how well the reconstruction of the autoencoder is already able to undo the concept drift.

In both cases, we implement the autoencoder as a multi-layer perceptron (MLP, i.e., a standard fully-connected feed-forward neural network) and the "undoing" function $f(\cdot)$ is implemented as an affine mapping.

C. Results

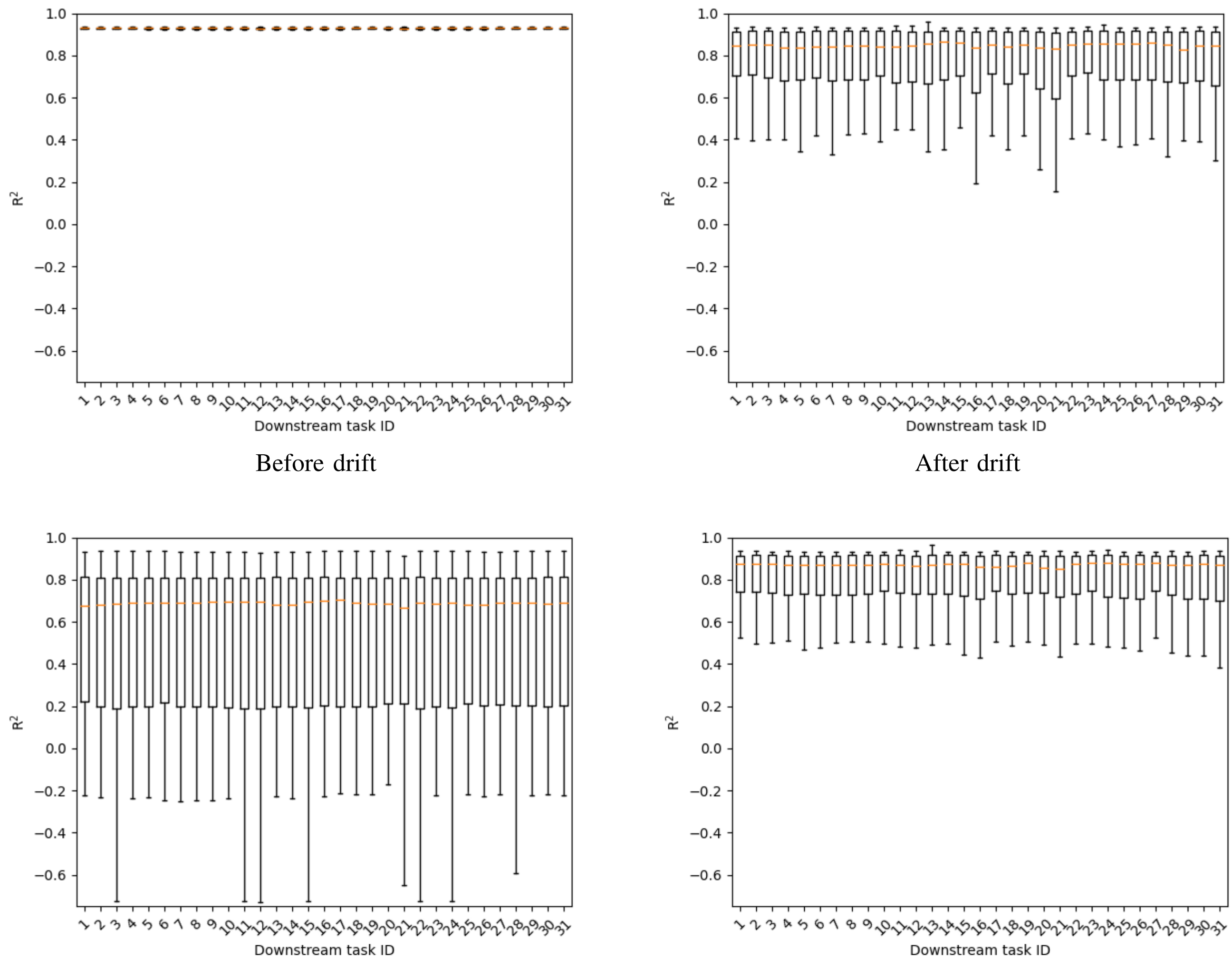
The results of the experiments on the digits data set are shown in Table I. For the Hanoi data set, we show the results for each downstream task and over all scenarios in Figure 5 – the detailed numbers are given as in Table II in the appendix A. Note that, in order to get rid of outliers distorting the box-plots, we filtered out all scenarios where the reconstruction of the autoencoder could not be improved, which leaves us with 138 scenarios left. Furthermore, since there is a lot of information in Figure 5, we also provide the results of a single downstream task in Figure 4 for illustrative purposes.

In both cases (data sets), we observe a significant improvement of the downstream task models $m(\cdot)$ after applying $f(\cdot)$ to the drifted data. We also observe a significant improvement

TABLE I: Results on the digits data set – we report the mean an variance over all ten folds, all numbers are rounded to two decimal points.

Accuracy↑	Before drift	After drift	After drift & AE reconstruction	After & unlearning drift
	0.98 ± 0.02	0.68 ± 0.05	0.74 ± 0.07	0.77 ± 0.08

Fig. 5: Results on the Hanoi data set – we plot the R^2 scores for each of 31 downstream task over all scenarios.



Before drift

After drift

After drift & AE reconstr.

After & unlearning drift

of the baseline where we use the autoencoder reconstruction instead of $f(\cdot)$ for undoing the drift. In case of the Hanoi data set, we also observe a significant improvement of the variance – i.e. solutions are more stable. These findings demonstrate a strong performance of our proposed method for undoing concept drift.

V. Discussion

Advantages. The key advantage of our proposed method is that it works completely unsupervised – i.e. no labeled samples are required, which in the real world are often not available or are very costly to obtain – and it is completely agnostic of the number and types of downstream tasks, which becomes handy in case of many "complex" downstream tasks that require large amount of (labeled) data for training.

Computational aspects. The computational complexity of our proposed method mainly depends on the implementation of $f(\cdot)$ because the autoencoder, which might be a rather complex neural network, is only trained once in the beginning and is not changed anymore afterwards. The complexity of $f(\cdot)$ (e.g. size of the neural networks that implements $f(\cdot)$) then determines how many samples are needed for building $f(\cdot)$ and consequently how much time this process of building $f(\cdot)$ takes – here time not only refers to the training time of $f(\cdot)$ but also to the time until we can "undo" the concept drift and can continue using the downstream task models $m(\cdot)$. However, in this work we observed that often a relatively simple architecture of $f(\cdot)$ is already sufficient for "undoing" the concept drift, and therefore only a small set of samples is need which leads to a fast training of $f(\cdot)$ as well.

Limitation. A potential limitation of our proposed approach is the choice of the autoencoder itself for distribution learning. If the concept drift does not lead to larger reconstruction loss of the autoencoder, our proposed method is not able to learn a function for undoing the drift. We therefore suggest to apply our proposed methodology only in cases where the concept drift manifests itself in a significantly large reconstruction loss of the autoencoder.

VI. Conclusion and Future Work

In this work we proposed a methodology for unsupervised concept drift unlearning – i.e. "undoing" concept drift in an unsupervised manner, which has the advantage that no downstream task model must be retrained or adapted for which labeled samples would be required. We empirically evaluated our proposed method on several scenarios from different domains and observed a strong performance of our proposed method.

Based on this initial work, a couple of potential directions for future research emerge:

- While the strength of our proposed method is that it does not require any labeled samples – i.e. it works completely unsupervised – it might be of interest to study if and how the performance of the downstream tasks could be improved in case a small set of labeled samples is available which can be used when building $f(\cdot)$.
- Transparency is an important aspects of ML-based systems that are deployed in the real world. In this context it would be of interest to make $f(\cdot)$ transparent, e.g. by explaining it's outputs – by this one might also be able to gain some insights into the nature of the observed concept drift.
- In this work, we always considered the scenario of a single concept drift. A straight forward extension to a scenario, where concept drifts occur frequently, is to completely retrain/rebuild $f(\cdot)$ from scratch after every concept drift. In this , it would be of interest to explore incremental learning or adaptations of $f(\cdot)$ in order to speed up the process of "undoing" the new concept drift.

Acknowledgment

A.A. and B.H. acknowledge funding from the VW-Foundation for the project *IMPACT* funded in the frame of the funding line *AI and its Implications for Future Society*, and funding from from the European Research Council (ERC) under the ERC Synergy Grant Water-Futures (Grant agreement No. 951424).

K. M., C. G. P, and M. M. P. acknowledge funding from the European Research Council (ERC) under grant agreement No 951424 (Water-Futures), the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE), and the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

References

[1] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[2] E. Kyriakides and M. Polycarpou, Eds., *Intelligent monitoring, control, and security of critical infrastructure systems*. Springer, 2014, vol. 565.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[4] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, 2018.

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

[6] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.

[8] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[9] J. Jakob, A. Artelt, M. Hasenjäger, and B. Hammer, "Sam-knn regressor for online learning in water distribution networks," in *Artificial Neural Networks and Machine Learning - ICANN 2022 - 31st International Conference on Artificial Neural Networks, Bristol, UK, September 6-9, 2022, Proceedings, Part III*, ser. Lecture Notes in Computer Science, E. Pimenidis, P. P. Angelov, C. Jayne, A. Papaleonidas, and M. Aydin, Eds., vol. 13531. Springer, 2022, pp. 752–762. [Online]. Available: https://doi.org/10.1007/978-3-031-15934-3_62

[10] K. Malialis, C. Panayiotou, and M. M. Polycarpou, "Queue-based resampling for online class imbalance learning," in *International Conference on Artificial Neural Networks (ICANN)*. Springer, 2018, pp. 498–507.

[11] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Online learning with adaptive rebalancing in nonstationary environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4445–4459, 2021.

[12] V. Vaquet and B. Hammer, "Balanced sam-knn: Online learning with heterogeneous drift and imbalanced data," in *Artificial Neural Networks and Machine Learning – ICANN 2020*, I. Farkaš, P. Masulli, and S. Wermter, Eds. Cham: Springer International Publishing, 2020, pp. 850–862.

[13] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2014.

[14] A. Cano and B. Krawczyk, "Rose: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams," *Machine Learning*, pp. 1–39, 2022.

[15] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 27–39, 2013.

[16] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Nonstationary data stream classification with online active learning and siamese neural networks," *Neurocomputing*, vol. 512, pp. 235–252, 2022.

[17] Y. Dong and N. Japkowicz, "Threaded ensembles of autoencoders for stream learning," *Computational Intelligence*, vol. 34, no. 1, pp. 261–281, 2018.

[18] F. Hinder, A. Artelt, and B. Hammer, "Towards non-parametric drift detection via dynamic adapting window independence drift detection (DAWIDD)," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4249–4259. [Online]. Available: https://proceedings.mlr.press/v119/hinder20a.html

[19] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers—part i: Detecting nonstationary changes," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145–1153, 2008.

[20] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[21] K. Malialis, M. Roveri, C. Alippi, C. G. Panayiotou, and M. M. Polycarpou, "A hybrid active-passive approach to imbalanced nonstationary data stream classification," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2022.

[22] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2021. [Online]. Available: https://doi.org/10.1109/JPROC.2020.3004555

[23] V. Vaquet, P. Menz, U. Seiffert, and B. Hammer, "Investigating intensity and transversal drift in hyperspectral imaging data," *Neurocomputing*, vol. 505, pp. 68–79, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.07.011

[24] F. Hinder, A. Artelt, V. Vaquet, and B. Hammer, "Contrasting explanation of concept drift," in *ESANN*, 2022.

[25] M. Jaworski, L. Rutkowski, and P. Angelov, "Concept drift detection using autoencoders in data streams processing," in *Artificial Intelligence and Soft Computing - 19th International Conference, ICAISC 2020, Zakopane, Poland, October 12-14, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, Eds., vol. 12415. Springer, 2020, pp. 124–133. [Online]. Available: https://doi.org/10.1007/978-3-030-61401-0_12

[26] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits data set," https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits, 1998.

[27] S. G. Vrachimis, M. S. Kyriakou *et al.*, "Leakdb: A benchmark dataset for leakage diagnosis in water distribution networks," in *WDSA/CCWI Joint Conference Proceedings*, vol. 1, 2018.

[28] V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Sensor Fault Diagnosis," *Foundations and Trends® in Systems and Control*, vol. 3, no. 1-2, pp. 1–248, 2016. [Online]. Available: http://www.nowpublishers.com/article/Details/SYS-007

Appendix

TABLE II: Detailed results on the Hanoi data set – we report the median R^2 score over all scenarios, all numbers are rounded to two decimal points.

Downstream task ID	$R^2 \uparrow$ before drift	$R^2 \uparrow$ after drift	$R^2 \uparrow$ after drift & AE reconstruction	$R^2 \uparrow$ after & unlearning drift
1	0.93	0.85	0.68	0.87
2	0.93	0.85	0.68	0.88
3	0.93	0.85	0.69	0.88
4	0.93	0.84	0.69	0.87
5	0.93	0.84	0.69	0.87
6	0.93	0.84	0.69	0.87
7	0.93	0.84	0.69	0.87
8	0.93	0.84	0.69	0.87
9	0.93	0.84	0.70	0.87
10	0.93	0.84	0.69	0.88
11	0.93	0.84	0.69	0.87
12	0.93	0.85	0.70	0.87
13	0.93	0.86	0.68	0.87
14	0.93	0.86	0.68	0.88
15	0.93	0.86	0.69	0.88
16	0.93	0.84	0.70	0.86
17	0.93	0.85	0.70	0.86
18	0.93	0.84	0.69	0.86
19	0.93	0.85	0.69	0.88
20	0.93	0.84	0.69	0.85
21	0.93	0.83	0.67	0.85
22	0.93	0.85	0.69	0.87
23	0.93	0.86	0.69	0.88
24	0.93	0.86	0.69	0.88
25	0.93	0.86	0.68	0.88
26	0.93	0.86	0.68	0.87
27	0.93	0.86	0.69	0.88
28	0.93	0.85	0.69	0.87
29	0.93	0.83	0.69	0.87
30	0.93	0.85	0.69	0.87
31	0.93	0.85	0.69	0.87