

Unsupervised Unlearning of Concept Drift with Autoencoders

André Artelt^{a,b,c}, Kleanthis Malialis^b, 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, mpolyar}@ucy.ac.cy

ORCID: {0000-0003-3432-7434, 0000-0002-6476-9025, 0000-0001-6495-9171}

Abstract—Concept drift refers to a change in the data distribution affecting the data stream of future samples. Consequently, learning models operating on the data stream might become obsolete, and need costly and difficult adjustments such as retraining or adaptation. Existing methods usually implement a local concept drift adaptation scheme, where either incremental learning of the models is used, or the models are completely retrained when a drift detection mechanism triggers an alarm. This paper proposes an alternative approach in which an unsupervised and model-agnostic concept drift adaptation method at the global level is introduced, based on autoencoders. Specifically, the proposed method aims to “unlearn” the concept drift without having to retrain or adapt any of the learning models operating on the data. An extensive experimental evaluation is conducted in two application domains. We consider a realistic water distribution network with more than 30 models in-place, from which we create 200 simulated data sets / scenarios. We further consider an image-related task to demonstrate the effectiveness of our method.

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

I. INTRODUCTION

Concept drift is one of the major challenges encountered by learning algorithms in data stream mining [1]. It refers to the problem of dealing with a non-stationary 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 consequences in practical applications.

Consider, for example, a domain area in which multiple learning models have been trained, each specialised for a particular downstream task; i.e., a set of (different) models operate on the same data (stream). For instance, in water distribution networks [2] each model corresponds to a different downstream task, such as predicting water pressure at a node from neighboring nodes with pressure sensors installed, detecting water leakage, and identifying the leakage location (i.e.

isolation). 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) and unsupervised learning / anomaly detection (leakage detection).

In the presence of concept drift, most - if not all - downstream models will be incapable of adequately performing their corresponding tasks. For instance, in water distribution networks examples of concept drift constitute changes in demands, and sensor faults. As a result, the traditional approach to be followed would be to update (either by incremental learning or complete re-training) each downstream model to maintain an optimal performance. Numerous methods [1], [3] have been proposed for concept drift adaptation, which we review later in Section III. Indeed, for some types of drift, e.g. water demand changes, the traditional approach should rightfully be applied. However, for some other drift types, e.g. sensor faults, existing methods become impractical, costly, and fail to scale well when the number of models is large.

The contributions made in this work are the following:

- We propose an unsupervised learning method for global concept drift adaptation based on autoencoders. Our method aims to revert the data distribution to the state it was before the concept drift had occurred, without the need to modify or even consider any of the existing downstream models – i.e. our proposed method is completely model-agnostic. Its key advantage is that no ground truth information (e.g., labels in a classification task) is required from human experts.
- We conduct empirical evaluations, involving a diverse set of different domains and scenarios, and demonstrate the effectiveness of our proposed method. Specifically, we evaluate our method in 200 simulated data sets / scenarios from a realistic water distribution network (each

corresponding to different drift characteristics) with more than 30 downstream models. Furthermore, evaluation is performed in an image-related task as well.

The remainder of this paper is organized as follows: First (Section II), we formally introduce the problem of local and global concept adaptation which we are dealing with in this work. Next (Section III), we present related and existing work to place our work in the literature. In Section IV we introduce and describe our proposed method for implementing a global concept drift adaptation using autoencoders, which we empirically evaluate in several case studies (Section V). Section VI discusses important remarks and characteristics of the proposed method, before Section VII concludes this work including some pointers to future directions.

II. CONCEPT DRIFT ADAPTATION PROBLEM

In this work we deal with the problem of adapting a set of models to concept drift – because in a supervised scenario concept drift becomes visible by means of a drop in accuracy or increasing loss [3], we formalize the concept drift adaptation problem as follows:

Definition 1 (Local Concept Drift Adaptation Problem): We are given a set models $m_i : \mathcal{X} \rightarrow \mathcal{Y}$ which were trained on data from the distribution $p_{\mathcal{X},\mathcal{Y}}$. Next, we assume that some type of concept drift happens, and the data distribution changes to $p'_{\mathcal{X},\mathcal{Y}}$. We assume that this concept drift affects the performance of some (possibly all) models $m_i(\cdot)$:

$$\mathbb{E}_{x,y \sim p'_{\mathcal{X},\mathcal{Y}}} [\ell(m_i, x, y)] > \mathbb{E}_{x,y \sim p_{\mathcal{X},\mathcal{Y}}} [\ell(m_i, x, y)] \quad (1)$$

where $x, y \sim p_{\mathcal{X},\mathcal{Y}}$ denotes samples from the joint distribution of \mathcal{X}, \mathcal{Y} , and $\ell(\cdot) \mapsto \mathbb{R}_+$ denotes a suitable function for measuring the performance of the models (e.g., mean-squared error).

The *Local Concept Drift Adaptation Problem* is to adapt the models $m_i(\cdot)$, yielding models $m'_i(\cdot)$, to the changed distribution such that the performances increases:

$$\mathbb{E}_{x,y \sim p'_{\mathcal{X},\mathcal{Y}}} [\ell(m_i, x, y)] > \mathbb{E}_{x,y \sim p'_{\mathcal{X},\mathcal{Y}}} [\ell(m'_i, x, y)] \quad (2)$$

In this work, we propose an alternative to adapting the models $m_i(\cdot)$ directly – i.e. adapting “locally” to the concept drift –, as it is done in Definition 1. Instead of a local adaptation, we aim for a global adaptation of the data distribution that is agnostic of any used models – i.e. we want to transform the observed data $\vec{x} \in \mathcal{X}$ such that the influence of the concept drift is removed. We formalize this as follows:

Definition 2 (Global Concept Drift Adaptation Problem):

The *Global Concept Drift Adaptation Problem* is to adapt to the changed distribution by transforming the samples $\vec{x} \in \mathcal{X}$ using a mapping $f : \mathcal{X} \rightarrow \mathcal{X}$, such that the performances of $m_i(\cdot)$ increases:

$$\mathbb{E}_{x,y \sim p'_{\mathcal{X},\mathcal{Y}}} [\ell(m_i, x, y)] > \mathbb{E}_{x,y \sim p'_{\mathcal{X},\mathcal{Y}}} [\ell(m_i, f(x), y)] \quad (3)$$

Remark 1: Note that the key difference between Definition 1 and Definition 2 is the way the concept drift adaptation is

done. In the former each model $m_i(\cdot)$ is adapted separately, while in the latter a global adaptation of the observed data is computed. Local and global adaption complement each other; local adaption can be appropriate for some drift types (e.g., water network expansion, water demand changes) while global adaption is appropriate for other types (e.g., sensor faults). Definition 1 proposes to adapt/change each model while Definition 2 does not change any model at all.

Our Contribution: This work proposes a method for a global drift adaptation (Definition 2) which is completely model-agnostic and unsupervised – i.e. we do not make any assumptions on the models $m_i(\cdot)$ and do not assume, in contrast to many other methods, that we observe labeled data.

III. RELATED WORK

Existing methods on addressing concept drift (Definition 1 and Definition 2) can be (loosely) categorised into the following three categories [1], [3]: concept drift adaptation, transfer learning, and representation learning.

1) **Concept Drift Adaptation:** Methods in this category are further grouped as passive, active, and hybrid methods.

Passive methods. These methods implicitly adapt to concept drift by using incremental learning, that is, they continually update a learning model [4] – these methods constitute a solution to the concept drift adaptation problem (Definition 1) since they adapt each model $m_i(\cdot)$. Passive methods make use of memory components (also referred to as memory-based methods), such as a moving window that 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].

Besides the assumption of observing labeled data, another key challenge is class imbalance [7] and to address it, incremental learning has been used in combination with different mechanisms, e.g., memory-based models [8] like having one memory component per class [9], (adaptive) rebalancing [10], [11], and bagging [12], [13].

Furthermore, most existing methods assume supervision. Incremental learning has been used in conjunction with other learning paradigms, such as active learning [14]–[16] and unsupervised learning [17].

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 [17], [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., DDM [20] 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, active methods discard the existing model and create a new one that performs a complete retraining as

soon as a concept drift alarm is triggered – therefore, similar to passive methods, these can be interpreted as a solution to the concept drift adaptation problem (Definition 1). Moreover, **ensembling** has also been used in active methods [21].

Hybrid methods. Hybrid methods combine the advantages of both approaches, for instance, HAREBA [22] combines a threshold-based drift detection mechanism with adaptive rebalancing to cope with class imbalance, [15] uses explicit drift detection with incremental active learning, while [17] proposes strAEm++DD, an autoencoder-based incremental learning method with drift detection.

2) **Transfer Learning:** Transfer learning [23]–[25] can also be used for adapting to a changed distribution – i.e. concept drift adaptation. Unfortunately, many transfer learning methods require labeled data [25]. A few unsupervised methods, however, exist: For instance [26]–[28].

3) **Representation Learning:** Representation learning [29] aims to learn a data representation that can be used by different learning models operating on the same data domain – e.g. learning models operating on text data might work on the same text embedding (i.e. text representation). However, such learned representations are also affected by concept drift and therefore have to be adapted and/or be as robust and invariant as possible. Only very little work on adapting representations exist: For instance, in [30] a method for learning invariant representations for continual learning (a very special case where new tasks must be learned over time) is proposed.

IV. UNSUPERVISED UNLEARNING OF CONCEPT DRIFT WITH AUTOENCODERS

Our proposed method is unsupervised, makes no assumption on the models $m_i(\cdot)$, and consists of two major parts:

- 1) **Distribution Learning:** We build an autoencoder that attempts to learn the data distribution so that we can characterize the influence of concept drift on observed samples. 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 assumes that an effective and efficient concept drift detection mechanism is in place and that the influence of concept drift can be characterized by the autoencoder.
- 2) **Unsupervised Transfer Learning:** When concept drift is detected, we build a function that tries to “undo” the concept drift – i.e. implementing $f(\cdot)$ from Definition 2. That is, we try to transform samples observed from the drifted data distribution, appear like they had been observed from the original (pre-drift) data distribution, by using the autoencoder built in the beginning.

Our entire methodology is described in detail in Algorithm 1.

A. Autoencoder for Distribution Learning

Concept drift refers to a change in the data distribution $p_{\mathcal{X},\mathcal{Y}}$ which might affect downstream models $m(\cdot)$ that operate on the data from the domain \mathcal{X} . However, since the true distribution $p_{\mathcal{X},\mathcal{Y}}$ is usually not known, it is estimated from the data. Having an estimate of $p_{\mathcal{X},\mathcal{Y}}$ 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 something about the concept drift itself [31].

We use an autoencoder $ae : \mathcal{X} \rightarrow \mathcal{X}$ for modeling the data distribution $p_{\mathcal{X}}$. This autoencoder $ae(\cdot)$ 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:

$$ae : x \mapsto (dec \circ enc)(x) \quad (4)$$

Learning an autoencoder Eq. (4) from a given data set $\mathcal{D} \subset \mathcal{X}$ means finding an encoder $enc(\cdot)$ and a decoder $dec(\cdot)$ such that the reconstruction loss $\ell(\cdot)^1$ is minimized:

$$\inf_{enc(\cdot), dec(\cdot)} \sum_{x_i \in \mathcal{D}} \ell(\underbrace{(dec \circ enc)(x_i)}_{\hat{x}_i}, x_i) \quad (5)$$

In this work, both the encoder $enc(\cdot)$ and decoder $dec(\cdot)$ are parameterized functions and the problem of learning the autoencoder Eq. (5) can be rewritten as an optimization problem over the parameters $\theta_1 \in \mathbb{R}^m$ and $\theta_2 \in \mathbb{R}^m$:

$$\arg \min_{\theta_1, \theta_2 \in \mathbb{R}^m} \sum_{x_i \in \mathcal{D}} \ell(\underbrace{(dec_{\theta_2} \circ enc_{\theta_1})(x_i)}_{\hat{x}_i}, x_i) \quad (6)$$

B. Unsupervised Transfer Learning using Autoencoders

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. (4). While using the autoencoder to discriminate between samples from before and after the drift could be an obvious choice as in [32], as mentioned, the proposed approach is agnostic to the concept drift detection mechanism used.

In order to “unlearn” the concept drift, we implement the global adaptation $f(\cdot)$ from Definition 2 by learning 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 data set of unlabeled samples $\mathcal{D}_* \subset \mathcal{X}$ under the new distribution (i.e. some samples observed after the concept drift).

In this work we parameterize the mapping $f(\cdot)$, and optimize over the parameters θ in order to find the final mapping $f(\cdot)$:

$$\arg \min_{\theta} \frac{1}{|\mathcal{D}_*|} \sum_{x_j \in \mathcal{D}_*} \ell(\underbrace{(ae \circ f_{\theta})(x_j)}_{\hat{x}_j}, f_{\theta}(x_j)) \quad (7)$$

Note that it would also be possible to chain $f(\cdot)$ and the autoencoder together, however in this case the final function (i.e. $ae \circ f$) for adapting to the drift would be computationally more complex and might therefore be less suited for real-time scenarios and scenarios where adapting to drift must be performed on devices with limited hardware capabilities (i.e. on-device learning).

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

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 $\{m_i(\cdot)\}$

- 1: Solve Eq. (6) \triangleright Build the autoencoder on a batch \mathcal{D} of data before any concept drift happens.

Main:

- 2: **if** $d(t) == \text{True}$ **then** \triangleright Test for concept drift using some drift detection method $d(\cdot)$.
 - 3: Collect \mathcal{D}_* \triangleright Build a data set for “undoing” the drift.
 - 4: Solve Eq. (8) \triangleright Build $f(\cdot)$ for “undoing” the drift.
 - 5: **else**
 - 6: $y_i = m_i(f(\vec{x}_t)) \quad \forall i \quad \triangleright$ Apply current sample \vec{x}_t to $f(\cdot)$ and then to the downstream tasks $\{m_i(\cdot)\}$.
-

In order to avoid less useful solutions – e.g. the mapping $f(\cdot)$ might yield the same output or might exploit some special particularities of the autoencoder, we propose to add the l_1 regularization term $C \cdot \|f_\theta(x_j) - x_j\|_1$ to the optimization problem Eq. (7):

$$\arg \min_{\theta} \frac{1}{|\mathcal{D}_*|} \sum_{x_j \in \mathcal{D}_*} \ell(\underbrace{(\text{ae} \circ f_\theta)(x_j)}_{\hat{x}_j}, f_\theta(x_j)) + C \cdot \|f_\theta(x_j) - x_j\|_1 \quad (8)$$

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

As pointed out in the beginning of Section IV, we can use the unsupervised transfer learning using autoencoder from Section IV-B as a method for globally adapting (i.e. unlearning) to concept drift as stated in Definition 2 – i.e. removing the influence of concept drift on the data space such that all models working on the data space are still applicable and perform reasonable well – see Algorithm 1.

V. EXPERIMENTS

We empirically evaluate the performance of our proposed methodology for global concept drift adaptation on a set of diverse data sets. For technical implementation details of the following experiments, we refer to the Python implementation which is publicly available on GitHub².

A. Data

We consider two data sets for our experiments:

1) *Digits*: We use the first five digits (i.e. 0-4) from [33] as a data set and build a digit classifier (this is the *single* downstream model $m(\cdot)$) by using logistic regression. We introduce concept drift in the test data by setting all pixels in the upper half of the image to zero – see Fig. 1 for an illustration. We use a 10-fold cross-validation to get statistically meaningful results. For a comparison, we also completely retrain the

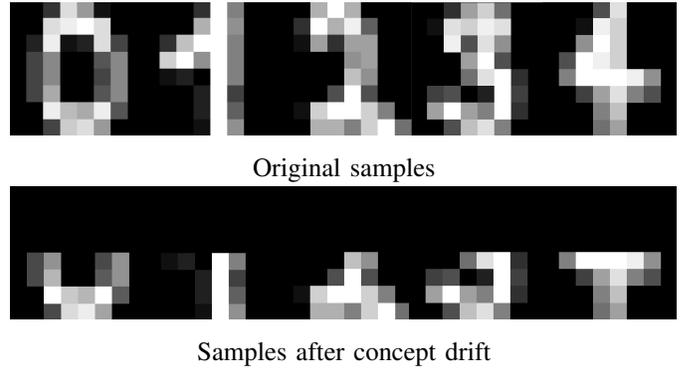


Fig. 1: Illustration of concept drift on the digits data set.

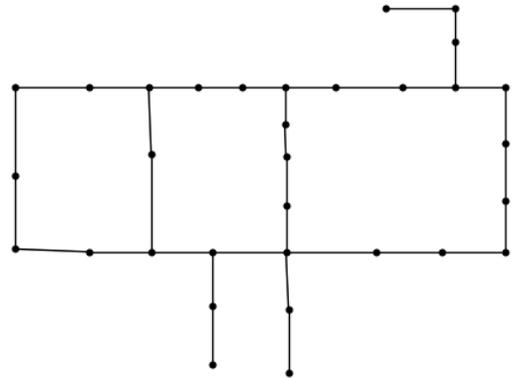


Fig. 2: Hanoi water distribution network [34].

classifier using a labeled training set – by this estimate how well a supervised drift adaptation could be, assuming labeled would be available.

2) *Hanoi*: In this experiment, we use the Hanoi water distribution network [34], shown in Figure 2. We place a pressure sensor at every node in the network (there are 32 nodes in total) and build a virtual sensor for nearly every sensor – i.e. we have 31 downstream regression tasks $m(\cdot)$ that try to predict the pressure at a particular (different) location.

In order to get statistically meaningful results, we generated a total number of 200 scenarios. 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 [35]:

- 1) A constant offset of the sensor’s measurement compared to the true measured quantity.
- 2) Gaussian noise added to the sensor measurements.
- 3) Sensor power failures, which result in the measurement

²<https://github.com/HammerLabML/UnsupervisedUnlearningConceptDriftAutoencoders>

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 $ae(\cdot)$ 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.

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 global concept drift adaptation 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 $ae(\cdot)$ as an implementation of $f(\cdot)$ – i.e. we evaluate if and how well the reconstruction of the autoencoder is already able to adapt 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 global concept drift adaptation 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 4 – the detailed numbers are given as in Table II in the appendix. 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 4, we also provide the results of a single downstream task in Figure 3 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 of the baseline where we use the autoencoder reconstruction instead of $f(\cdot)$ for undoing the drift. We observe that supervised retraining (assuming labels are available) would lead to a significant performance boost but still a bit worth than the performance before the drift which indicates that the new concept is more difficult to learn. In the case of the Hanoi data set, we also observe a significant improvement in the variance – i.e. solutions are more stable. These findings demonstrate a

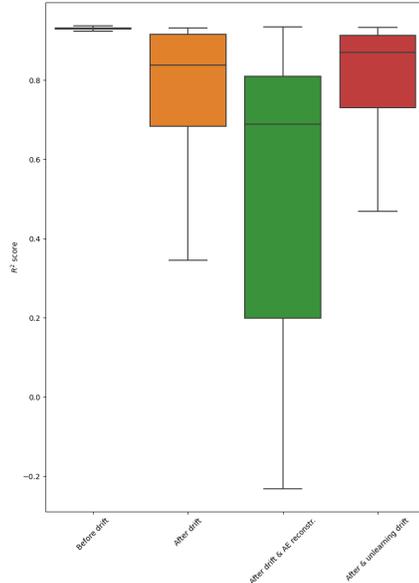


Fig. 3: Results on the Hanoi data set for downstream task 4 – we plot the R^2 scores over all scenarios.

strong performance of our proposed method for global concept drift adaptation.

VI. DISCUSSION

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 adapt to 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 adapting to the concept drift, and therefore only a small set of samples is needed which leads to a fast training of $f(\cdot)$ as well.

Limitations. 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 a larger reconstruction loss of the autoencoder, our proposed method is not able to learn a function for adapting the drift. We, therefore, suggest applying our proposed methodology only in cases where the concept drift manifests itself in a significantly large reconstruction loss of the autoencoder.

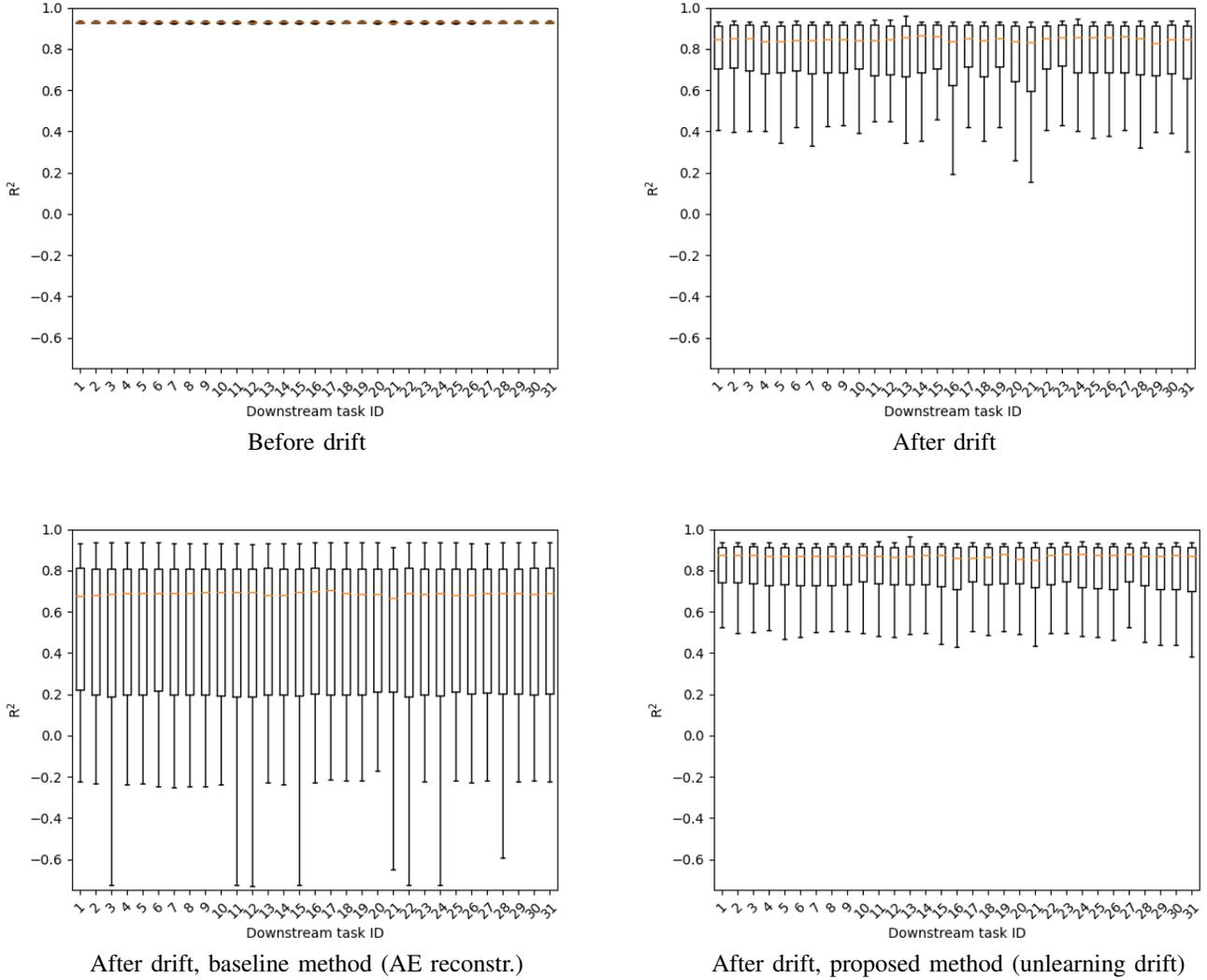
VII. CONCLUSION AND FUTURE WORK

In this work, we proposed an unsupervised methodology for global concept drift adaptation using autoencoders – i.e.

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

Accuracy↑	Before drift	After drift	After drift & AE reconstruction	After & drift adaptation	Supervised retraining
	0.98 ± 0.02	0.68 ± 0.05	0.74 ± 0.07	0.77 ± 0.08	0.93 ± 0.02

Fig. 4: Results on the Hanoi data set – we plot the R^2 scores for each of 31 downstream tasks over all scenarios.



“undoing” concept drift in an unsupervised manner, which has the advantage that no downstream model $m_i(\cdot)$ 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, it might be of interest to study if and how the performance of the downstream models 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 aspect of ML-based systems that are deployed in the real world. In this context, it would be of interest to make the adaptation $f(\cdot)$ transparent, e.g. by explaining its outputs, which may provide some insights into the nature of the observed concept drift.
- In this work, we always considered the scenario of a single concept drift. A straightforward extension to a scenario, where concept drifts occur frequently, is to completely retrain/rebuild $f(\cdot)$ from scratch after every

concept drift. In such scenarios, it would be of interest to explore incremental learning or adaptations of $f(\cdot)$ in order to speed up the process of concept drift adaptation.

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 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 non-stationary 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] 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.
- [8] 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
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [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, D. Papatheodoulou, S. Filippou, C. G. Panayiotou, and M. M. Polycarpou, “Data augmentation on-the-fly and active learning in data stream classification,” 2022.
- [17] J. Li, K. Malialis, and M. M. Polycarpou, “Autoencoder-based anomaly detection in streaming data with incremental learning and concept drift adaptation,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2023.
- [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] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.
- [21] 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.
- [22] 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.
- [23] 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>
- [24] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [25] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.
- [26] 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>
- [27] Z. Yao, Y. Wang, M. Long, and J. Wang, “Unsupervised transfer learning for spatiotemporal predictive networks,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 10778–10788.
- [28] R. K. Sanodiya and L. Yao, “Unsupervised transfer learning via relative distance comparisons,” *IEEE Access*, vol. 8, pp. 110290–110305, 2020.
- [29] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [30] G. Sokar, D. C. Mocanu, and M. Pechenizkiy, “Learning invariant representation for continual learning,” *arXiv preprint arXiv:2101.06162*, 2021.
- [31] F. Hinder, A. Artelt, V. Vaquet, and B. Hammer, “Contrasting explanation of concept drift,” in *ESANN*, 2022.
- [32] 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
- [33] 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.
- [34] 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.
- [35] 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 shows the results on the Hanoi data set, where we report the median R^2 score over all scenarios, all numbers are rounded to two decimal points.

TABLE II: Detailed results on the Hanoi data set

Downstream task ID	$R^2 \uparrow$ before drift	$R^2 \uparrow$ after drift	$R^2 \uparrow$ after drift & AE reconstruction	$R^2 \uparrow$ after & drift adaptation
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