Reconstruction-based unsupervised drift detection over multivariate streaming data

Daniil Kaminskyi
Department of Computer Science
TU Dortmund University
Dortmund Germany
daniil.kaminskyi@tu-dortmund.de

Bin Li
Department of Computer Science
TU Dortmund University
Dortmund Germany
bin.li@tu-dortmund.de

Emmanuel Müller
Department of Computer Science
TU Dortmund University
Dortmund Germany
emmanuel.mueller@tu-dortmund.de

***Abstract*—In real-world applications, data streams are generated all the time. Real-time data processing of complex multivariate data becomes essential for many downstream analysis tasks. However, real-world data is not bound to be of the same distribution - the environment where it is recorded could be rapidly evolving, making it a challenge to apply a stationary model throughout the whole flow of data. Moreover, labels are often expensive to acquire or delayed in such scenarios. This paper considers the severe problem in an unsupervised setting, where we detect the distributional drifts in the input data stream without considering the data labels or specific classifiers.**

We propose AECDD (Autoencoder-based Concept Drift Detector), a reconstruction-based unsupervised drift detection model using an Autoencoder to track changes in data distribution. More specifically, instead of detecting drifts by tracking the classification error change as in many existing approaches, we track the reconstruction error of the Autoencoder in an unsupervised manner. Our empirical evaluation shows that AECDD captures the drifts well in multivariate data streams. Finally, we also demonstrate the drift in the reconstruction error space by an intuitive visualization.

***Index Terms*—Concept drift, Autoencoder, Streaming data, Unsupervised learning**

I. Introduction

In the era of big data, a vast amount of sensor data is collected all the time. Analyzing streaming data in real-time is essential in various application scenarios. Distributional changes in real-time data streams, that are sometimes evolving, sometimes sudden, are called concept drifts. Concept drift detection has become a prominent topic in the current research. The reliability of models used for processing big data streams depends on how well they adapt to changes in data. Distribution drifts can have a heavy impact on the performance of algorithms as the model trained on previous data does not fit the ever-changing stream. Moreover, the arriving velocity of the data stream often leaves domain expert few time to label the data in real-time.

The concept drift detection task is solved in both supervised and unsupervised learning manners. The availability of ground truth at any time narrows the detection down to observing the changes in the underlying probability distribution of $P(X, y)$. The supervised drift detection approach requires labels for the detection models, which is a severe constraint for a significant amount of real-world applications. Detecting changes in $P(X)$ itself using an unsupervised approach is desired. To this end, a multitude of methods to detect concept drift have been proposed and have been tested on effectiveness [1]. Generally, such models require parts of data with varying sizes kept in memory to perform dissimilarity measurements. This, however, brings up another problem when dealing with multivariate streaming data.

In a real-world scenario, accumulating a sufficient amount of reference data and storing it may lead to vast memory consumption. The scalability of the drift detection algorithm to a high number of dimensions is desirable to keep both memory consumption conservative and the prediction accuracy reliable. Furthermore, rare existing approaches can detect complex pattern drifts in multi-dimensional streams, which is important for some downstream tasks, e.g. anomaly detection. These complex patterns may for example include a frequency change in data, leaving the amplitude of values observed the same.

Recently, Autoencoders have been employed in unsupervised representation learning [2], and further contribute to the downstream tasks [3], [4]. Their deep model structure enables the modeling of complex data patterns. In time series data, Autoencoders constructed using Recurrent Neural Networks (RNNs) can be used to capture the temporal dependencies between data points [5].

Our goal is to develop an unsupervised approach to detect concept drift in high-dimensional streaming data. To achieve our goal, we map complex high-dimensional streaming data to and from a lower dimensional space, utilizing the Autoencoder's ability to fit to a single historical pattern and failing on newer data should its pattern change. Contributions of this paper can be summarized as follows:

- We propose a reconstruction-based unsupervised concept drift detection approach;
- Empirical study on synthetic data showing effectiveness and efficiency of our approach.

II. Related work

The drift detection problem is well studied in the big data field. The Drift Detection Method DDM [6] is one of the most well-known concept drift detection methods. It tracks the error rate of a base learner over the flow of data. The learner's error rate will decrease as the number of analyzed

2375-9259/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDMW58026.2022.00109

samples increases, as long as the data distribution is stationary. If the error rate increases, reaching the confidence interval of a significant change, a concept drift is detected. Numerous improvements of the algorithm [7]–[9] have been proposed, with the idea of detecting change using base learner's errors, e.g. classifier's misclassification rate, staying the same.

Algorithms for unsupervised drift detection use a distance function or metric to quantify the dissimilarity between the distribution of historical data and the new data. By addressing the root sources of the concept drift, these algorithms provide information about the drift time and location.

The first approach [10] in this category of algorithms utilized total variation for the distribution discrepancy analysis. The choice and application of the distance function formed a general pattern for dissimilarity measurements used in the unsupervised drift detection methods after it.

Incremental Kolmogorov-Smirnov (IKS) [11] is a distribution-based method that checks for drift by comparing instance—possibly distance between the empirical distribution function of the newest data and the cumulative distribution function of the reference data using a two-sample Kolmogorov–Smirnov test.

Neighbor-based Density Variation Identification (NN-DVI) [12] uses dissimilarities between clusters of data to check for a concept change. The approach is addressing regional density changes, which are often let out by the distribution-based drift detectors.

A discriminative classifier [13] approach trains the model to classify the reference data as belonging to a stationary concept. Should a significant portion of the newest data be marked as not part of the historical concept by the classifier, the drift is alarmed.

Several existing works conduct dimensionality reduction before concept drift detection. A PCA-based approach was proposed in [14] to alarm for changes in multivariate data. Ceci et al. [15] use both PCA and Autoencoder to detect changes from latent space. A downside, however, is that the model itself is relying on a user-defined threshold as its hyperparameter.

Jaworski et al. [16] use an Autoencoder to detect changes in phishing data. Using an Autoencoder as the base model for detecting concept drifts in streaming data is also introduced in [17]. The model shows that the usage of the Autoencoder is possible for both streaming and semi-stationary data by training the Autoencoder on a sufficient amount of data from the input, however, a user-predefined threshold is needed.

A significant shortcoming of a standard Autoencoder is that it is not aiming to capture the temporal dependencies of the data. Since concept drifts in streaming data may have an underlying time-frame principle, a different type of Autoencoder can be used to reconstruct the data. The Long Short-Term Memory Encoder-Decoder (LSTM-ED) [17] model reconstructs sequences of normal time-series. Originally proposed as an anomaly detection method, it can be applied to a concept drift task. An LSTM-based approach is introduced by [18] to adapt to concept drift. The model operates in a semi-supervised setting, as a concept drift is only detected if the attached classifier misclassifies for an instance of data, where the ground truth is required. An improvement is shown in [19], in which the model adapts to the concept drift by using the LSTM's latent space for regression, classification, and forecasting tasks. The aforementioned methods, however, do not provide explicit drift detection. The proposed model, on the other hand, implements a reconstruction-based unsupervised drift detection mechanism. The reconstruction error of streaming data is used to detect concept drifts in real-time.

III. Proposed Model

In this section, we introduce AECDD (Autoencoder-based Concept Drift Detector), an unsupervised drift detection approach for multivariate streaming data. The section is split into an overview of the general detection architecture, and a detailed description of the training and prediction phases of the algorithm.

A. Architecture overview

AECDD performs online unsupervised drift detection on streaming data. The general workflow, shown in Figure 1, is split into two parts: the training phase, where the LSTM Autoencoder model learns its parameters to represent historical data accurately, and the detection phase, during which the data stream is consecutively consumed in batches and the drift detection takes place.

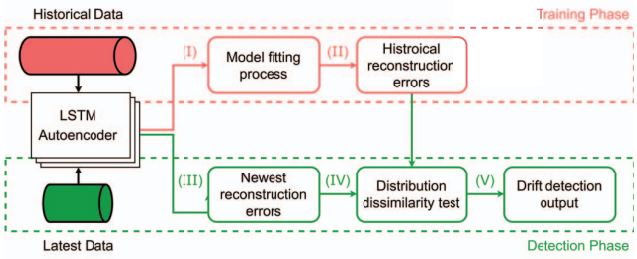

Fig. 1. Overview of AECDD workflow

In the training phase, the Autoencoder is trained to reconstruct the input data itself (I). In step (II), the model produces reconstruction errors of a subset of historical data, used as a reference sample to be compared against in the detection phase.

The detection phase is done by consuming new data in fixed-size batches. Autoencoder produces representations with the parameters learned in the training phase of the latest data inside a window and the reconstruction error is calculated (III). If a dissimilarity measurement test (IV) confirms that the underlying distribution in the newest reconstruction errors is greater than the historical, the model alarms for a concept drift (V).

B. Training phase

Given is a D-dimensional data stream $\{X_1^D, X_2^D, ..., X_N^D\}$, where $X_i^D = \{x_1, x_2, ...x_D\}$ for timestamp i.

An Autoencoder is a symmetric neural network consisting of a encoder f_{enc} and a decoder f_{dec}. We construct the Autoencoder with LSTMs to capture the temporal information in streaming data. For a sequence consisting of S elements $X \in \mathbb{R}^{S \times D}$, the Autoencoder reconstructs it as follows:

$$\begin{aligned} f_{enc} &: \mathbb{R}^{S \times D} \rightarrow \mathbb{R}^{H} \\ f_{dec} &: \mathbb{R}^{H} \rightarrow \mathbb{R}^{S \times D} \end{aligned} \tag{1}$$

We assume that the training data is sampled from a stationary distribution, such that the Autoencoder can learn unified knowledge of one single data pattern. The Autoencoder is trained to minimize the reconstruction error $e = |X - (f_{dec} \circ f_{enc})X|$ during the training phase.

The LSTM Autoencoder takes historical data of size T consisting of elements $\{X_1^D, ...X_T^D\}$ for training. It is further split into overlapping *sequences* $\{\{X_1^D, ...X_S^D\}\}, \{X_2^D, ...X_{S+1}^D\}..., \{X_{T-S}^D, ...X_T^D\}\}$. A demonstration of the sub-batching process is depicted in Figure 2 on an example with $T = 4$, $S = 2$. By splitting the data in an overlapping manner we get 3 sequences for training.

Minimizing the reconstruction error for the training set is done by learning the parameters for the hidden states in the encoding and decoding part of the Autoencoder. As the next step, we get the reference reconstruction error $e_{hist} = \{e_1^{S \times D}, e_2^{S \times D}, ..., e_{(T-S+1)}^{S \times D}\}$.

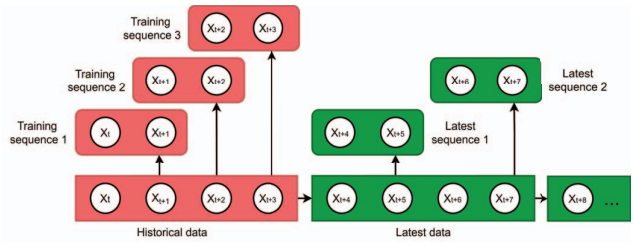

Fig. 2. Data stream processing. Red part represents the training phase, while the green one stands for online prediction.

C. Detection phase

Once the training phase is complete, we start processing the data from the real-time data stream by sliding a window of size W. It is split into sequences $\{\{X_i^D, ..., X_{i+S}^D\}, ..., \{X_{i+W-S}^D, ..., X_{i+W}^D\}\}$ in a non-overlapping manner. In the case shown in Figure 2, the sliding window size is set to $W = 4$. Contrary to the training phase, the data split here results in 2 sequences for reconstruction.

A reconstruction error $e_{new} = \{e_i^{S \times D}, ..., e_{i+W/S}^{S \times D}\}$ is calculated using the model's learned parameters. An increase in reconstruction error indicates that the input data pattern changes so that the Autoencoder fails to reconstruct it with low error. The severity of change in reconstruction errors is estimated by a statistical test.

We perform a one-tailed two-sample Kolmogorov–Smirnov Test (KS-Test) [20], [21] as a non-parametric and distribution-free statistical test. It is conducted on each of the reconstruction error dimensions to check whether historical and newer reconstruction errors are significantly different. Assuming F_{hist}, F_{new} are two empirical estimated cumulative distribution functions from e_{hist} and e_{new} respectively, the null hypothesis $H_0 : F_{new} \leq F_{hist}$ is rejected if:

$$\sup_e [F_{new}(e) - F_{hist}(e)] > c(\alpha)\sqrt{\frac{m+n}{m \cdot n}} \tag{2}$$

where n, m are the number of data points in e_{hist} and e_{new} with $n = T$ and $m = W$; α is the significance level, $c(\alpha) = \sqrt{-ln(\alpha)}$. We alarm for concept drift if the null hypothesis is rejected in at least one of the dimensions.

IV. Experiments

A. Datasets

We evaluate our approach with two synthetic datasets containing different types of concept drifts.

- Changing Sine Sudden (CS_{sud}) - D continuous sine waves with phase lengths $\{\lambda_1, ...\lambda_D\}$ are generated over a duration C. Sudden concept drift is introduced where the phase lengths switch to $\{\lambda'_1, ..., \lambda'_D\}$. Parameter C represents the length of a single concept.
- Changing Sine Incremental (CS_{incr}) - instead of switching immediately, a transition period of length tr is used, during which phase lengths gradually shift from $\{\lambda_1, ...\lambda_D\}$ to $\{\lambda'_1, ..., \lambda'_D\}$, introducing an incremental drift.

The datasets are designed to generate drifts in the data distribution using frequency changes instead of amplitude changes. It is done to emphasize the effectiveness of LSTM Autoencoder in capturing temporal dependencies of the data stream.

A noise factor of $\epsilon = 0.1$ is applied to the sine waves, which is a random value in the range $[0, 0.1]$. ϵ is added to each of the sine waves' observations. Extending both datasets, a single irrelevant dimension consisting of random values in the range $[-1, 1]$ is added to sophisticate the task of the fitting process.

B. Experiment setting

We examine different variations of the parameter setting. The test parameters for phase length were chosen as $\{\lambda_1, ..., \lambda_D\} = \{100, 150, 200, 250, 300\}$ and $\{\lambda'_1, ..., \lambda'_D\} = \{150, 100, 100, 250, 300\}$ to represent concept drift in three of the five dimensions (six including the irrelevant). The corresponding change in wave frequencies ranges between 33% and 100%, including an increase as well as a decrease in wave frequency. Single concept length C is set to 5000 and transition length tr to 500.

For the KS Test we compare the experiments under three different significance levels $\alpha = \{0.001, 0.01, 0.05\}$. This setting is responsible for the sensitivity of the detection model and we examine its influence in the next section.

Furthermore, we try different settings of hidden size and sequence length for the Autoencoder with $H = \{10 - 100\}$ and sequence length $S = \{10, 20, 50, 100, 150, 200\}$.

For every experiment, we first generate the data stream and then take T observations as our historical data. The Autoencoder is trained to minimize the reconstruction error on 75% of the historical data and the reconstruction error for the remaining 25% of the dataset is used as a reference reconstruction error. Once the training is complete, the data is taken in batches of size W, and the detection task is done on the observations inside the window. We examine the historical data size used during training phase as $T = \{500, 1000, 1500, 2000\}$ and the window length $W = \{50, 100, 150, 200, 250\}$.

All experiments are conducted on an NVIDIA Quadro RTX 6000 24GB GPU. For the model training, we set the learning rate as $1e - 3$, batch size as 20, and the number of epochs being 50. The experimental results are averaged over ten runs.

C. Competitors

To compare the effectiveness of the proposed model with other unsupervised detection techniques, we chose three algorithms as our competitors.

D3 [13] is a drift detection algorithm based on a discriminative classifier. It operates as follows: having a reference and the newest data sample a simple slack variable is introduced, namely the reference data is assigned label 0, and the newest data 1. The classifier is trained on the reference sample, learning the historical concept. A drift is alarmed if the AUC score for the predicted labels is higher than threshold τ, meaning the classifier can successfully differentiate between the new and the data distributions. For our experiments, we set the reference data to be the first 1000 elements of the respective dataset, and slide a window of size 250 through the data to test for the drift. The τ is set to the proposed default value of 0.75.

HDDDM [22], Hellinger Distance Drift Detection Method, functions by constructing histograms from the reference distribution and current data distribution and calculating the Hellinger distance between them to detect concept drift. We fix the reference data to the first 1000 elements of the dataset and use a sliding window of size 100 to check for drift. The p-value of the statistical test is set to 0.01.

CDBD [23], the Confidence Distribution Batch Detection method, operates similarly to HDDDM, however, the Kullback-Leibler divergence is used as a dissimilarity measurement. First 1000 elements of the dataset are set as a reference, and we slide a window of size 100 to detect drift. The p-value of the statistical test is set to 0.01.

To keep the comparison fair from the proposed model's perspective, the AECDD settings were chosen similarly to the other competitors. We fix the training size $T = 1000$ and window length to $W = 100$. Other settings are chosen as follows: hidden size $H = 10$, sequence length $S = 20$, and significance level $\alpha = 0.001$. These optimal settings were obtained from the proposed model's tests and discussed in Section IV-E.

D. Evaluation metrics

We use the holdout evaluation approach for our experiment [24]. The drift locations are predefined by the parameters assigned to the synthetic data generator. The data is processed in batches and the detection can occur only at intervals of length W. As such, we propose to evaluate the performance by marking all of the observations inside a window, considered to be a part of the initial concept, as negative. Should the model detect a drift after processing a window, the elements of this window are marked positive. Using this methodology, a 2×2 confusion matrix is built by comparing the prediction vector to the ground truth about the drifted and stationary values provided by the stream generator.

We use F1 score and accuracy as our metrics, derived from the confusion matrix. In case of a sub-optimal fit, the model is prone to marking the elements as belonging to another concept. To give an insight into this type of scenario, the precision value is taken into the final metrics.

E. Performance analysis

The best performance was achieved with a combination of parameters $H = 10$, $\alpha = 0.001$, $T = 2000$, $S = 20$. However, no single best option for different drift types and window lengths W was discovered, as shown in Table I.

To further interpret the effect of parameters, we investigate all metrics for different settings of T, α, and H. Analyzing the performance of the proposed model on different training sizes in Table II, we come to a conclusion, that the effectiveness of the model increases with the training size T. For the generated sine data with a maximal wavelength of 300, a training size of 500 is not sufficient, nor is 1000. The more recurring patterns the model learns during its training phase, e.g. sine phases, the better it becomes at dissecting contextual information from the stream, wave frequency in this example.

The significance level α setting for the KS Test is responsible for the sensitivity of the model. Based on the results shown in Table III, we can derive that the increase of sensitivity leads to a higher false positive rate, decreasing the performance overall. Thus, the lowest value of $\alpha = 0.001$ brings the best results.

As a next step, we fix our values $T = 2000$, $\alpha = 0.001$ for an analysis on the choice of sequence length S and hidden size H.

To better understand how the model differentiates between concepts, we investigate the reconstruction errors for one of the relevant dimensions in Figure 3. The real concept drift is located at the mark 5000 for both datasets. The reconstruction errors rise significantly after this point, which leads to the newest data being marked as drifted. On the other hand, we can observe false positive predictions for the concept drift before it occurred. This could be explained by the fact that we test each of the dimensions individually, alarming for drift if any of them shows a difference from the historical data. Notably, the reconstruction errors for the CS_{incr} grow gradually in contrast to a sudden rise for CS_{sud}, meaning the reconstruction errors depict the change in the data correctly. On the other hand,

TABLE I
BEST PERFORMANCE

Dataset	Window length (W)	Accuracy	F1	Precision
CS_{sud}	50	0.987 (+/-0.010)	0.99 (+/-0.008)	1.000 (+/-0.000)
	100	0.980 (+/-0.008)	0.984 (+/-0.006)	1.000 (+/-0.000)
	150	0.979 (+/-0.021)	0.984 (+/-0.016)	1.000 (+/-0.000)
	200	0.982 (+/-0.020)	0.986 (+/-0.015)	1.000 (+/-0.000)
	250	0.991 (+/-0.014)	0.993 (+/-0.011)	1.000 (+/-0.000)
CS_{incr}	50	0.976 (+/-0.010)	0.981 (+/-0.008)	0.984 (+/-0.007)
	100	0.978 (+/-0.015)	0.983 (+/-0.011)	0.984 (+/-0.005)
	150	0.980 (+/-0.017)	0.985 (+/-0.013)	0.992 (+/-0.013)
	200	0.976 (+/-0.024)	0.982 (+/-0.018)	1.000 (+/-0.000)
	250	0.976 (+/-0.026)	0.983 (+/-0.019)	1.000 (+/-0.000)

TABLE II
PERFORMANCE UNDER DIFFERENT TRAINING SIZE

Dataset	Training size (T)	Accuracy	F1	Precision
CS_{sud}	500	0.535 (+/-0.016)	0.693 (+/-0.008)	1.000 (+/-0.000)
	1000	0.586 (+/-0.041)	0.729 (+/-0.020)	1.000 (+/-0.001)
	1500	0.670 (+/-0.101)	0.784 (+/-0.057)	1.000 (+/-0.001)
	2000	0.761 (+/-0.142)	0.847 (+/-0.085)	1.000 (+/-0.001)
CS_{incr}	500	0.555 (+/-0.011)	0.711 (+/-0.005)	1.000 (+/-0.000)
	1000	0.600 (+/-0.038)	0.743 (+/-0.019)	1.000 (+/-0.002)
	1500	0.667 (+/-0.090)	0.789 (+/-0.050)	0.999 (+/-0.005)
	2000	0.733 (+/-0.123)	0.834 (+/-0.072)	0.999 (+/-0.004)

TABLE III
PERFORMANCE UNDER DIFFERENT SIGNIFICANCE LEVEL

Dataset	Significance level (α)	Accuracy	F1	Precision	FPR
CS_{sud}	0.001	0.671 (+/-0.150)	0.783 (+/-0.095)	1.000 (+/-0.001)	0.589 (+/-0.282)
	0.01	0.640 (+/-0.122)	0.764 (+/-0.076)	1.000 (+/-0.000)	0.642 (+/-0.239)
	0.05	0.604 (+/-0.081)	0.744 (+/-0.051)	1.000 (+/-0.000)	0.702 (+/-0.175)
CS_{incr}	0.001	0.659 (+/-0.126)	0.781 (+/-0.078)	0.999 (+/-0.005)	0.581 (+/-0.231)
	0.01	0.639 (+/-0.104)	0.770 (+/-0.064)	1.000 (+/-0.003)	0.614 (+/-0.196)
	0.05	0.617 (+/-0.070)	0.757 (+/-0.045)	1.000 (+/-0.001)	0.651 (+/-0.148)

TABLE IV
MODEL PERFORMANCE COMPARISON

Dataset	Detector	Accuracy	F1	Precision
CS_{sud}	CDBD	0.519 (+/-0.047)	0.513 (+/-0.032)	0.524 (+/-0.050)
	D3	0.500 (+/-0.004)	0.043 (+/-0.164)	0.501 (+/-0.007)
	HDDDM	0.668 (+/-0.166)	0.526 (+/-0.316)	0.681 (+/-0.338)
	AECDD	**0.704 (+/-0.024)**	**0.790 (+/-0.014)**	**1.000 (+/-0.000)**
CS_{incr}	CDBD	0.513 (+/-0.042)	0.518 (+/-0.029)	0.541 (+/-0.046)
	D3	0.476 (+/-0.000)	0.000 (+/-0.000)	-
	HDDDM	0.641 (+/-0.202)	0.480 (+/-0.355)	0.733 (+/-0.277)
	AECDD	**0.708 (+/-0.017)**	**0.798 (+/-0.009)**	**0.996 (+/-0.007)**

when looking at a dimension not affected by the concept drift in Figure 4, we can not observe a significant change in its reconstruction errors at the drift location. Still, a concept drift is detected from a statistical test conducted on other dimensions.

As demonstrated in Figures 5 and 6, an increase of both hidden size and sequence length leads to worse results for most of the parameter settings. A possible reason could lie in that the model with a smaller latent space captures contextual information better than one with over-complicated hidden state transitions. As for the sequence lengths, as its size increases, it captures a larger part of one particular sine wave. Having the Autoencoder focused on reconstructing a smaller portion of the waves may be a better solution than trying to recreate the phase as a whole.

Comparing the model to other unsupervised detection meth-

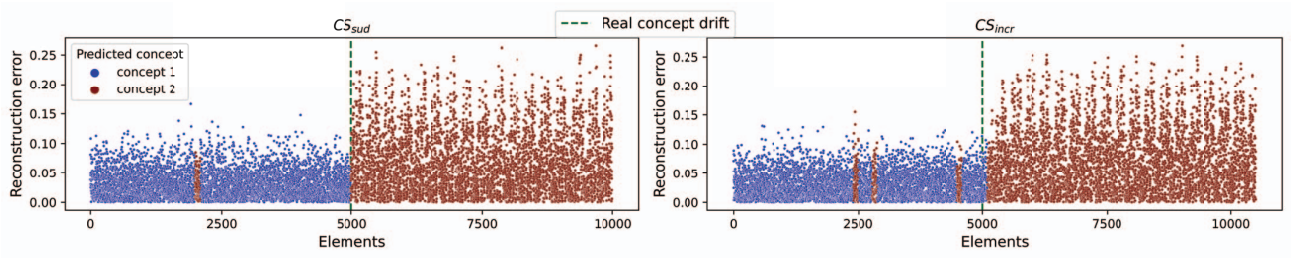

Fig. 3. Reconstruction errors in relevant dimension before and after concept drifts: the two concepts in each dataset is split by the green dashed line. The colors represent the model's prediction of concepts.

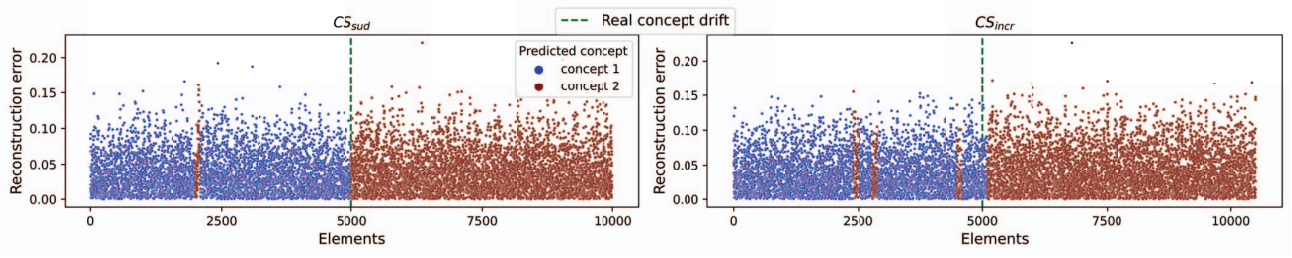

Fig. 4. Reconstruction errors in an irrelevant dimension before and after concept drifts: the two concepts in each dataset is split by the green dashed line. The colors represent the model's prediction of concepts.

ods in Table IV, we can see that the proposed model achieves better results even when operating with a non-optimal training size. The discriminative classifier approach in D3 fails to raise an alarm when the sine wave frequency changes since the observed values' range stays the same. The Kullback-Leibler divergence does not prove to be a good metric for the data distribution change either. AECDD achieves almost perfect precision score when compared to other algorithms, since the underlying Autoencoder model captures the change of data even when not fitted optimally to the reference data. Overall, using reconstruction errors instead of the raw observed values decreases the variance of data used for the dissimilarity tests, bringing a benefit to the detection accuracy.

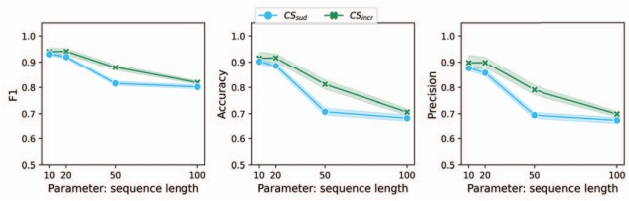

Fig. 5. Parameter sensitivity analysis: Sequence length.

In a real-world scenario, of course, the data is not bound to having only two concepts throughout the whole data stream. The next step is after a concept drift has been detected in the model adaptation. Several drift adaptation strategies have been proposed in the literature, the obvious being discarding the old model and training a new one on the latest data [25], [26]. Xu et al. [27] proposed to adapt to the concept drift by increasing the number of nodes in the hidden layers of their model to

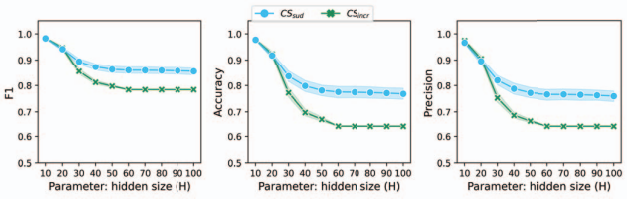

Fig. 6. Parameter sensitivity analysis: Hidden size.

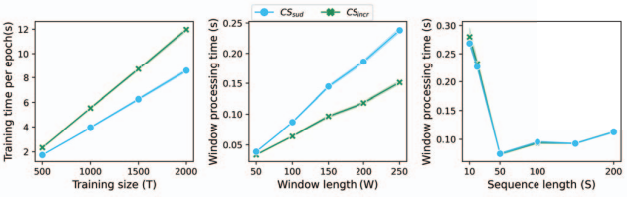

Fig. 7. Running time analysis.

increase the generalization capability. The model adaptation strategy, however, is out of our work's discussion scope as we only propose a way to detect the drift.

Finally, the processing time is an important factor in real-time applications. We track the computation time in our experiments. Figure 7 demonstrates the relation of time spent in the training phase of the algorithm, with training size T being the most important factor. Increasing training size T to boost the model's accuracy is seen as a trade-off between computation time and model performance.

The window processing time, however, takes up only a tiny

fraction of the model's training time for one epoch. Since we split the window into sequences in a non-overlapping manner, the number of sequences to be passed through LSTM is significantly lower than for one iteration of training.

An interesting finding that we discovered, is that the increase of sequence length first drastically decreases window processing time, only to increase it as the sequences become larger. A possible reason would be that the time taken by reconstructing a single sequence increases at a lower speed than the number of sequences that are contained in the window, which all need to be passed through the Autoencoder.

V. Conclusion and Future Works

This paper introduces an Autoencoder-based unsupervised drift detection algorithm for multivariate streaming data. The AECDD has been tested on synthetic data, tailored to specifically target the ability of the model to capture contextual information. The proposed model has shown good results in this setting. The application of the proposed algorithm to real-world data should include a viable strategy for the model's adaptation process once the drift has been detected since the real-world scenarios are not limited to having only two concepts throughout the whole data stream.

In our work, we assume that the historical data originates from a stationary context where no concept drift occurs. On the other hand, an insufficient amount of data taken for training leads to a bad performance of the model. This possible trade-off between the danger of potentially encountering a concept drift inside the historical data and achieving good results on new data is a subject of future research. Specifically, the behavior of the algorithm on real-world data and tuning the hyperparameters is a field of interest for future work.

Acknowledgment

This work was supported by the Research Center Trustworthy Data Science and Security, an institution of the University Alliance Ruhr.

References

[1] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[2] G. Zhou, K. Sohn, and H. Lee, "Online incremental feature learning with denoising autoencoders," in *Artificial intelligence and statistics*. PMLR, 2012, pp. 1453–1461.

[3] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pretrained deep recurrent neural network for time series classification," *arXiv preprint arXiv:1706.08838*, 2017.

[4] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, 2014, pp. 4–11.

[5] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," 07 2016.

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

[7] I. Frías-Blanco, J. d. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2015.

[8] A. Pesaranghader and H. Viktor, "Fast hoeffding drift detection method for evolving data streams," vol. 9852, 09 2016, pp. 96–111.

[9] A. Pesaranghader, H. Viktor, and E. Paquet, "Mcdiarmid drift detection methods for evolving data streams," 2018.

[10] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," 04 2004, pp. 180–191.

[11] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1545–1554. [Online]. Available: https://doi.org/10.1145/2939672.2939836

[12] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognition*, vol. 76, 11 2017.

[13] O. Gözüaçık, A. Büyükçakır, H. Bonab, and F. Can, "Unsupervised concept drift detection with a discriminative classifier," ser. CIKM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2365–2368. [Online]. Available: https://doi.org/10.1145/3357384.3358144

[14] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 935–944.

[15] M. Ceci, R. Corizzo, N. Japkowicz, P. Mignone, and G. Pio, "Echad: Embedding-based change detection from multivariate time series in smart grids," *IEEE Access*, vol. PP, pp. 1–1, 08 2020.

[16] M. Jaworski, L. Rutkowski, and P. Angelov, "Concept drift detection using autoencoders in data streams processing," in *Artificial Intelligence and Soft Computing*, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, Eds. Cham: Springer International Publishing, 2020, pp. 124–133.

[17] A. G. Menon and G. Gressel, "Concept drift detection in phishing using autoencoders," in *Machine Learning and Metaheuristics Algorithms, and Applications*, S. M. Thampi, S. Piramuthu, K.-C. Li, S. Berretti, M. Wozniak, and D. Singh, Eds. Singapore: Springer Singapore, 2021, pp. 208–220.

[18] Á. C. Lemos Neto, R. A. Coelho, and C. L. d. Castro, "An incremental learning approach using long short-term memory neural networks," *Journal of Control, Automation and Electrical Systems*, Apr 2022. [Online]. Available: https://doi.org/10.1007/s40313-021-00882-y

[19] S. Suryawanshi, A. Goswami, P. Patil, and V. Mishra, "Adaptive windowing based recurrent neural network for drift adaption in non-stationary environment," *Journal of Ambient Intelligence and Humanized Computing*, Jun 2022. [Online]. Available: https://doi.org/10.1007/s12652-022-04116-0

[20] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multidimensional data streams," *Interfaces*, 01 2006.

[21] C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, "Statistical features-based real-time detection of drifted twitter spam," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 914–925, 2017.

[22] G. Ditzler and R. Polikar, "Hellinger distance based drift detection for nonstationary environments," in *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, 2011, pp. 41–48.

[23] P. Lindstrom, B. Mac Namee, and S. Delany, "Drift detection using uncertainty distribution divergence," vol. 4, 12 2011, pp. 604–608.

[24] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[25] S. H. Bach and M. A. Maloof, "Paired learners for concept drift," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 23–32.

[26] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," vol. 7, 04 2007.

[27] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, 02 2017.