Anomaly Detection for Reoccurring Concept Drift in Smart Environments

Vincenzo Agate
Department of Engineering
University of Palermo
Palermo, Italy
vincenzo.agate@unipa.it

Salvatore Drago
Department of Engineering
University of Palermo
Palermo, Italy
salvatore.drago06@unipa.it

Pierluca Ferraro
Department of Engineering
University of Palermo
Palermo, Italy
pierluca.ferraro@unipa.it

Giuseppe Lo Re
Department of Engineering
University of Palermo
Palermo, Italy
giuseppe.lore@unipa.it

***Abstract*—Many crowdsensing applications today rely on learning algorithms applied to data streams to accurately classify information and events of interest in smart environments. Unfortunately, the statistical properties of the input data may change in unexpected ways. As a result, the definition of anomalous and normal data can vary over time and machine learning models may need to be re-trained incrementally. This problem is known as concept drift, and it has often been ignored by anomaly detection systems, resulting in significant performance degradation. In addition, the statistical distribution of past data often tends to repeat itself, and thus old learning models could be reused, avoiding costly retraining phases on new data, which would waste computational and energy resources. In this paper, we propose a hybrid anomaly detection system for streaming data in smart environments that accounts for concept drift and minimize the number of machine learning models that need to be retrained when shifts in incoming data distribution are detected. The system is multi-tier and relies on two different concept drift detection modules and an ensemble of anomaly detection models. An extensive experimental evaluation has been carried out, using two real datasets and a synthetic one; results show the high performance achieved by the system using common metrics such as F1-score and accuracy.**

***Index Terms*—concept drift, smart city, online anomaly detection, unsupervised learning**

I. Introduction

Recently, with the widespread diffusion of IoT technologies related to smart city and other smart environment scenarios, the amount of information collected through mobile devices that possess sensors and an Internet connection is growing exponentially. Data coming from smartphone sensors can also be combined with high-level information provided directly by humans to detect and proactively monitor complex real-world phenomena [1], as dictated by the Mobile Crowdsensing (MCS) paradigm [2]. At the same time, many private and public companies urgently need efficient machine learning techniques to analyze huge amounts of streaming data, enabling them to make real-time predictions and support their decisions based on such data [3].

Two key aspects must be taken into account to address this need, namely the enormous amount of data to analyze and its varied nature. First, considering that user devices are continuously generating new data that need to be collected and analyzed in real time, existing data fusion and anomaly detection methodologies should be readjusted to process incoming data in streaming [4].

Moreover, much of the data collected and analyzed are related to natural phenomena and human behavior, e.g., anomaly detection in surveillance video streams [5], environmental pollution monitoring in smart cities [6], road surface condition monitoring [7], and health monitoring [8]. In addition to being inevitably affected by noise and, potentially, tampering by selfish users [9], [10], these data are also subject to the problem of concept drift [11]. This means that the statistical properties of the input data or the target variable that the model is trying to predict can change over time in a sudden and unexpected way.

For example, consider an anomaly detection system that analyzes images extracted from the video streams of airport smart surveillance cameras. The goal of the system is to classify travelers based on certain characteristics such as clothing and face cover, and it is trained to label people wearing a mask as suspicious.

The emergency situation caused by the COVID-19 pandemic has led to a sudden increase in people wearing masks. How should the system behave in this case, if wearing a face mask is no longer abnormal behavior? We are in the presence of concept drift, and the system should be retrained accordingly.

Furthermore, what happens if the health emergency ends and masks are no longer commonly used? This is called reoccuring concept drift, and when it happens the system should return to its original operation.

Many of the traditional data fusion and anomaly detection approaches in crowdsensing systems completely ignore the concept drift problem [12], but this raises serious accuracy and reliability concerns in any real-world scenarios where concept drifts actually occur, and which would require timely and efficient real-time retraining of the machine learning models adopted.

In fact, when concept drift occurs, a model trained on past data may not be fit to properly analyze new incoming values, resulting in inaccurate predictions and poor decision results. Indeed, if the data analysis model does not explicitly handle concept drift, it will have to be manually retrained with the new data. This continuous model retraining, in turn, results in wasted time, energy, as well as network and computational resources. Moreover, until the model is manually retrained, the system will exhibit a drastic drop in performance, which could have catastrophic consequences in critical scenarios such as detecting anomalies in power management [13], online banking transactions, or intrusion detection [14] in critical network infrastructure.

In this paper, we propose an unsupervised anomaly detection system for reoccurring concept drift in data streams for smart environments. The main goal of our system is to minimize the number of machine learning models that need to be retrained, while maintaining very high accuracy in anomaly detection, even in the case of sudden and reoccurring concept drifts. The system architecture is multi-tier, and exploits two concept drift detection modules and an ensemble of specially trained models for anomaly detection. Our approach takes advantage of the fact that, even in the presence of concept drift, old concepts often recur cyclically. Thus, we maintain a history of previously trained models that could be reused in the future, in the spirit of sustainability and efficient use of available resources [15], which is especially valuable in mobile scenarios where such resources are limited and their usage must be reduced as much as possible [16].

The only case in which a new model will be trained is when none of the old models fits the new data, that is, when there is a drift toward a concept that was never observed before. However, even in the rare cases where it is necessary to train a new model, our system will do it efficiently by adopting a technique that is suitable for streaming data, using only the last window of data received.

To validate our approach, the proposed system is extensively evaluated with a series of experiments performed on three different datasets. The results show that the system is able to detect outliers with high accuracy while minimizing the number of models that have to be trained.

The main contributions of our work can be summarized as follows.

- We propose a new hybrid anomaly detection system that directly takes into account reoccurring concept drifts.
- We adapt the well-known LOF anomaly detection technique to be more suitable for analyzing data streams in real time.
- We reduce as much as possible the number of anomaly detection models that need to be trained while still maintaining high performance.

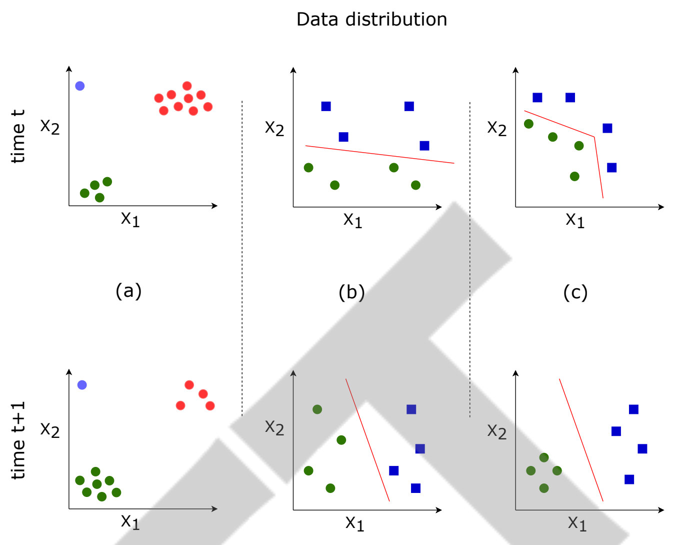


Figure 1. Sources of concept drift.

- We extensively validate our system with two different real-world datasets and a synthetic one.

The remainder of the paper is structured as follows. Section II is dedicated to related work. The proposed system is discussed in detail in Section III. The experimental results are shown in Section IV, while Section V presents our conclusions.

II. Related Work

Many works in recent years have focused on studying anomaly detection techniques, with the goal of identifying outliers in a dataset [17]. An outlier is an observation that is so different from others that it may appear to have been generated by some other mechanism than normal data. This definition is based on statistical considerations and assumes that normal data follow a common generation mechanism. Observations that deviate from this mechanism are considered outliers.

The detection of these unusual patterns is extremely important in various scenarios, such as intrusion detection systems, financial fraud detection, medical diagnosis, and e-voting systems [18].

The most widely used anomaly detection techniques in the literature can be divided into three main categories: statistical approaches, distance-based methods, and isolation-based methods.

Statistical approaches [19] assume that the data were generated according to a specific distribution. Outliers will then be those points that have a low probability of being generated according to that distribution. In distance-based methods [20], outliers are modeled as points that are isolated from normal data, without making any assumptions about their distribution. Such approaches are mainly categorized into those based on nearest neighbors and those based on clustering and properties such as cluster density and size. Isolation-based methods [21], on the other hand, try to isolate outlier observations from the dataset, since anomalous data are considered very different from normal data and are assumed to represent a small portion of the entire dataset.

Early outlier detection algorithms are known as static, in that they determine whether outliers are present once all records are available in the dataset and thus can only work *a posteriori*. In contrast, incremental outlier detection techniques [22] can work in real-time with streaming data, identifying outliers as soon as a new record is received by the system.

Traditional anomaly detection techniques have to be enriched and revised in order to detect outliers incrementally. To this end, we have modified a well-known static anomaly detection technique, known as Local Outlier Factor (LOF) [20], to adapt it to streaming data analysis, as will be explained below.

In dynamic environments such as mobile crowdsensing, the distribution of input or output data can change over time unexpectedly. This phenomenon is known as concept drift [11], [23], and can have several sources, as shown in Fig. 1. The first type of drift [24], known as virtual drift or feature space drift, occurs when only the distribution of the input data changes and there are no shifts in the predictions, as shown in Fig. 1-a.

In the second type of drift, called actual drift or decision boundary drift, the distribution of input data remains unchanged, while the predictions change, causing the decision boundary to be altered (Fig. 1-b). However, in real world settings, these two types of drift are more likely to happen simultaneously: Fig. 1-c shows the third type of drift, which occurs when both the distribution of input data and predictions change together. Another classification that is often used in the literature for concept drift concerns how the distribution of data changes over time. Generally, four types of possible drifts are distinguished, as shown in Fig. 2:

- the drift could be *sudden*, which can happen when, for example, one sensor is replaced with another that is calibrated differently, or when there is a sudden malfunction or an unexpected event, such as the outbreak of a pandemic;
- the drift could be *incremental*, with a series of intermediate concepts following one another, reflecting the case of user preferences changing over time or continuous sensor wear and tear, resulting in loss of accuracy;
- the drift could be *gradual*, if the new concept initially alternates with the previous one, before stabilizing and finally prevailing;
- the drift could be *reoccurring*, if old concepts recur cyclically, for example based on seasonal changes or periodic events.

It is generally not critical to identify the exact type of drift, since in practice concept drifts of different types can be combined in variable and unpredictable ways. However, it is essential to provide strategies to handle all types appropriately so that the learning model can be readjusted quickly and efficiently.

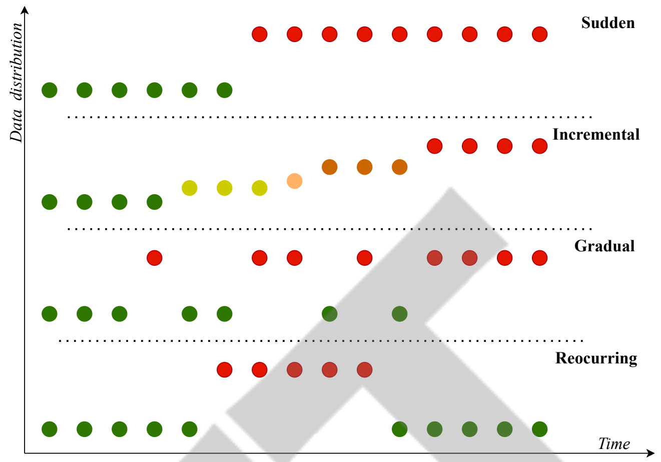


Figure 2. Types of Concept Drift.

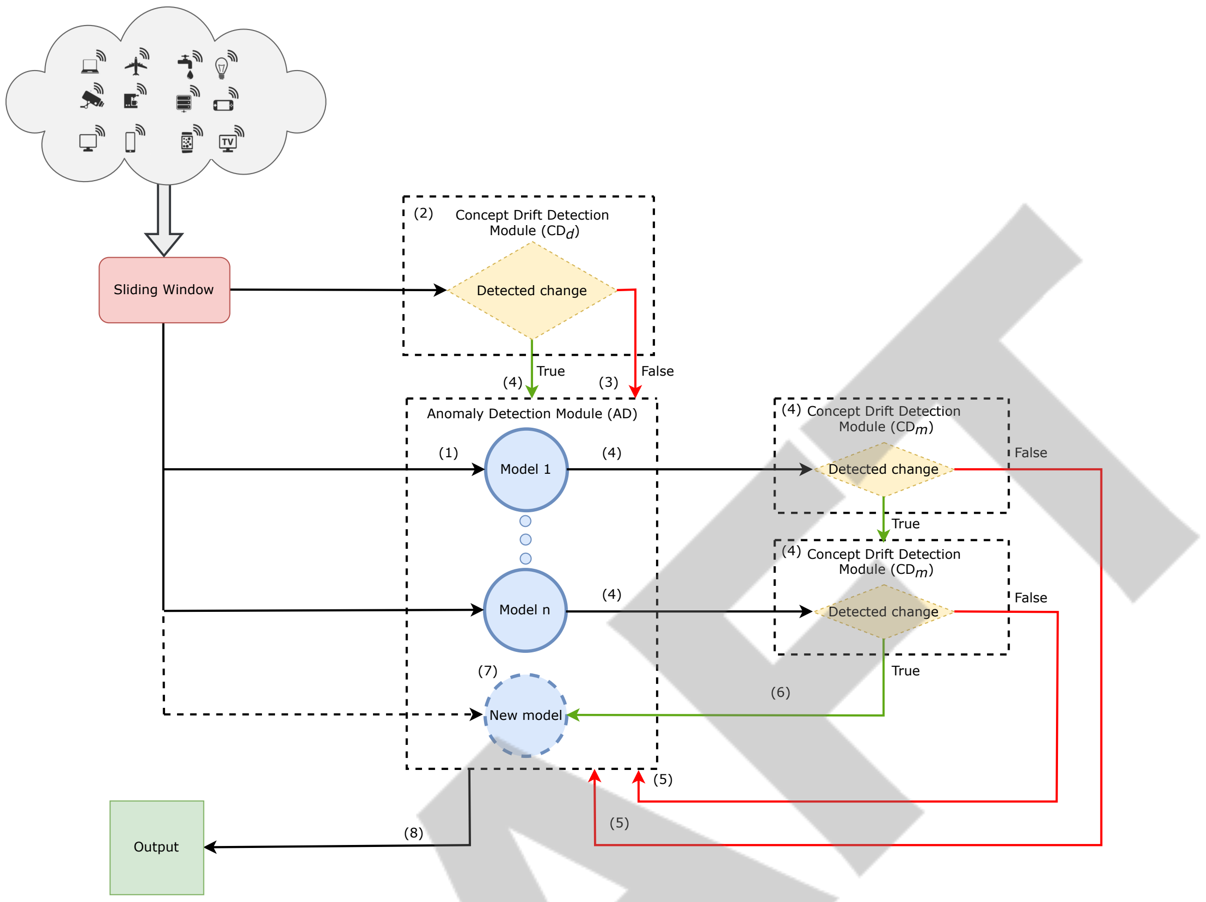
Works in the literature dealing explicitly with both anomaly detection and concept drift are not very common, although they are increasing in number in recent years. Many of these works exploit consolidated concept drift detection techniques, such as ADWIN [25] and KSWIN [26].

For example, IForestASD [27] analyzes new incoming data in batches and uses a parameter that corresponds to the anomaly rate that should not be exceeded in batches. However, in unsupervised mode without additional information about the data, choosing an appropriate value for this parameter is very difficult. If the value chosen is too low, drifts will be continuously detected and the model will be updated too often. Conversely, if the parameter is too high, the model will never be updated, resulting in performance degradation in case of concept drift.

The authors of [28] overcome this issue by proposing several approaches and comparing them through experiments. Their final idea is to use the KSWIN [26] drift detection method on incoming data features to reveal a concept drift and update the model accordingly before it evaluates them.

The main limitation of these approaches is ignoring the fact that concept drifts, especially in real cases, can be reoccurring. If two concepts alternate in the streaming data, the machine learning model will be updated with each variation. However, the replaced model may be useful later, when the new incoming data will again describe the concept on which the former model was trained.

Unlike these works, our system accounts for reoccurring concept drift, minimizing the number of re-training phases even when concept drift is detected. This results in significant savings in terms of energy and computational effort. Another limitation of most anomaly detection systems in the literature is that they are not particularly suitable for online usage. In contrast, we also adapt state-of-the-art anomaly detection techniques to optimize the analysis of streaming data in real time.



Sliding Window
(2) Concept Drift Detection Module (CD$_d$)
Detected change
True
False
(4)
(3)
Anomaly Detection Module (AD)
(1)
Model 1
(4)
Model n
(4)
(7)
New model
(4) Concept Drift Detection Module (CD$_m$)
Detected change
False
True
(4) Concept Drift Detection Module (CD$_m$)
Detected change
False
True
(6)
(5)
(5)
(8)
Output

Figure 3. Architecture of the proposed system.

III. Proposed Architecture

In this section, we present the multi-tier architecture of the proposed anomaly detection system, as shown in Fig. 3. The system is composed of two concept drift detection modules and an ensemble of anomaly detection (AD) models. The first of these models is the one currently used, while the others represent the history of old AD models that could be reused in case of reoccurring concept drift. To do this, the system reconfigures itself by autonomously choosing the models to employ [29]. When the system receives new input data, the first drift detection module checks if a drift is present. If no drift is detected, the data is simply evaluated with the current model. Otherwise, the new data are evaluated by all the old AD models saved in the history, and their outputs are examined, in turn, by a second drift detection module. The system checks whether one of the old models is again valid and consistent with the new incoming data; if so, that model will simply be reused, avoiding unnecessary retraining and thus reducing the waste of resources and energy.

The system uses a window W of fixed size w that contains the new incoming batch of data and a list ml of AD models, of fixed size N. Each AD model returns not only the prediction but also the confidence of that prediction [30].

Let us define m and cl as the current model and the confidence list for each model, respectively. In addition, we denote the concept drift detection module used on the input data as CD_d and the one that is used on the confidence of model predictions as CD_m.

The workflow of the system is shown in Fig. 3 and is detailed step-by-step as follows.

1) If W is the first data window, create the first model M_0, train it with W, add this model to ml, set $m = M_0$ and go directly to step 8.
2) Evaluate whether CD_d detects a drift between W and the previous data window.
3) If no drift is detected in step 2, go directly to step 8.
4) Otherwise, if a drift is detected in step 2, estimate the confidence for each model M_i included in ml, and evaluate whether CD_m detects a concept drift for M_i between the new confidence and the original confidence saved in cl.
5) If CD_m finds a model M_i without concept drift, set $m = M_i$ and go to step 8.
6) If a concept drift is detected for all models, create a new model M_{new} and train it with the current W that caused

the drift.

7) If there are already N models, then replace the oldest model M_0 with the newly created model M_{new}, otherwise simply add M_{new} to ml; in both cases, set $m = M_{new}$.

8) Predict the classes of all the data in W using the current model, m. If m is a newly created model, also add the confidence of its predictions to cl; in both cases, return to step 2.

The following sections will discuss in greater detail the computational complexity of the proposed system, as well as the functioning of the concept drift detection and anomaly detection modules, respectively.

A. Computational complexity

Let us assume that the proposed system is working on the n-th batch of incoming data, and that the model list ml has reached its maximum size N. To calculate the computational complexity we must distinguish three possible cases:

1) old concept: the new data batch does not introduce concept drift (workflow steps 2, 3, and 8);

2) reocurring concept: the new data batch causes a concept drift and the system can reuse one of the existing models (workflow steps 2, 4, 5, and 8);

3) new concept: the new data batch causes a concept drift and it is necessary to train a new model (workflow steps 2, 4, 6, 7, and 8).

The computational complexity (C) to evaluate a new batch of data depends on the window size, w. In the worst case, we get, respectively:

1) $C_{CD_d}(w) + C_{AD}(w)$
2) $C_{CD_d}(w) + N \cdot (C_{AD}(w) + C_{CD_m}(w))$
3) $C_{CD_d}(w) + N \cdot (C_{AD}(w) + C_{CD_m}(w)) + C_T(w) + C_{AD}(w)$

where C_{CD_d}, C_{CD_m}, C_{AD} and C_T are, respectively, the computational costs of CD_d and CD_m, the evaluation of a single data point with the anomaly detection model, and the training of a new module.

B. Concept Drift Detection Modules

The KSWIN [26] method is used to implement both concept drift detection modules (CD_d and CD_m). KSWIN is based on the well-known Kolmogorov-Smirnov (KS) statistical test [31], which is a non-parametric test that does not need any assumption on the underlying data distribution. However, this test can only handle one-dimensional data, as it compares the absolute distance between two one-dimensional data distributions.

Since we are dealing with many features, it is necessary to adapt the KSWIN method to our scenario. The solution proposed in [28] is to apply the KSWIN method to each feature separately, one by one. However, this approach has a limitation, since a relatively small variation could occur on multiple features at the same time, causing a concept drift that would not be detected by the system that only considers one feature at a time.

To address this problem, we decided to create a new synthetic feature that accounts for the variation of all other features at the same time. The KSWIN method is then used to analyze the variation of the distributions of all features, including the synthetic one. If a drift is detected for at least one of them, the system will consider using a new model or reusing one of those previously trained.

In this work, we chose the sum of the squares of the base features, appropriately normalized, as the synthetic feature. This function proved to be a good choice in practice, as will be shown by the experimental results.

The operations performed by the KSWIN method consist of adding new data to the window and then applying the KS statistical test at the end. Introducing a new single data point in the KSWIN window has an $O(1)$ cost, so adding the whole data window costs $O(w)$. The KS test [31] has a computational cost of $O(n)$, where n is the number of data points. In our case, $n = 2w$, since we compare two windows of size w. The computational complexity of CD_d and CD_m is thus $O(w)$.

C. Anomaly Detection Module

The anomaly detection module is composed of Local Outlier Factor (LOF) models. LOF is a density-based anomaly detection algorithm that is particularly suitable for use in datasets with non-uniform distribution [21]. The method identifies outliers based on the local deviation of specific points by comparing the density of each data point with that of its neighbors. When a point has a very low local density, it is considered an outlier.

The original LOF algorithm is a static method intended to be used when all input data are already available. To employ it with streaming data, three possible approaches are usually adopted: supervised, periodic, or iterated. Supervised LOF involves training the system only once at the beginning, with the first window of data. Periodic LOF requires to re-train the system periodically on the entire data set available up to that point. Finally, iterated LOF re-trains the system for each individual incoming record, and has the highest computational cost.

Periodic and iterated LOF approaches have a high computational cost since their models are trained on all the data received up to that point. The supervised method is not affected by this problem, however, it does not account for changes in the statistical distribution of the input data and is therefore not suitable for handling data with concept drifts. Periodic and iterated LOF are not ideal for handling sudden drifts either, because they do not provide a mechanism for assigning more weight to recent data.

As described above, our system overcomes these problems by adopting a hybrid approach. First, it handles concept drifts dynamically, upstream of the LOF algorithm itself, with dedicated detection modules. Then, it maintains a history to reduce the number of new models to train as much as possible, and the method only fits recent data to best identify sudden drifts while reducing the computational complexity, which depends only on the data window size. As demonstrated in [20], the computational complexity of training a new LOF model is $O(n \log n)$, where n is the number of data points. In our case, $n = w$, so the total computational complexity of training a LOF model is $O(w \log w)$.

The number of models stored in the history is of paramount importance: by using many models, it is likely that one of them will fit the input data. This results in a significant decrease in the number of models that have to be trained, especially in the presence of reoccurring concept drift in real-world situations. However, storing too many models in the history is space-consuming, so it is necessary to find an appropriate trade-off.

IV. Experimental evaluation

In this section, we will analyze the behavior of our system to confirm the validity of the proposed approach using three datasets. The first two are well-known real-world datasets commonly used to test anomaly detection systems, which were not explicitly designed to highlight concept drift, namely SMTP and SHUTTLE. The third one is a synthetic dataset, named SRdrift, which is characterized by sudden and reoccurring concept drifts.

The SMTP dataset [32] is a modified version of the original KDDCUP99 dataset [33], and contains 95156 records of network traffic data with 4 numerical features, as well as a label indicating whether each record is an outlier or an inlier. Only 0.03% of the data are outliers; thus, the two possible classes are extremely unbalanced. The dataset does not explicitly contain sudden concept drifts but, as will be shown, has some properties of gradual concept drift.

The SHUTTLE dataset [32] is a modified version of the original "Statlog (Shuttle)" dataset [33], and contains 49097 records with 9 numerical features and their respective classes. A total of 3511 outliers (7%) are present. Unlike SMTP, the SHUTTLE classes are less unbalanced and the dataset does not include any kind of concept drift.

SRdrift was created by merging two synthetic datasets that, individually, do not exhibit concept drift. The two sub-datasets present records that are grouped into a single cluster except for 10% of outliers; normal data are generated by a multivariate Gaussian distribution, while outliers are generated by a uniform distribution. SRdrift contains 2049 records with 3 numerical features, and it interleaves blocks of data from the two sub-datasets, resulting in concept drifts that are both sudden and reoccurring.

All experiments have been carried out by running the LOF algorithm with 25 neighbors, and always keeping the same maximum number of models, $N = 5$. For each dataset, the experiments have been repeated by varying the window size w from 25 to 150. The same experiments have been repeated by disabling the concept drift detection modules (CD_D and CD_M) as a baseline. Finally, all experiments were repeated 1000 times, and we present the average of the results obtained.

Table I
System evaluation with the SHUTTLE dataset when varying window size.

Drift Detection	Window size	Reused Models	Macro F1-score	Weighted F1-score	Acc
Yes	25	99%	0.50	0.90	0.93
	50	99%	0.85	0.95	0.95
	100	99%	0.89	0.97	0.97
	150	98%	0.90	0.97	0.97
No	150	-	0.92	0.98	0.98

For each experiment, we report the average accuracy, which can be defined as a function of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}.$$

We also report the F1-score, which is defined as the harmonic mean of precision and recall, as follows:

$$precision = \frac{TP}{TP + FP},$$

$$recall = \frac{TP}{TP + FN},$$

$$F1\text{-}score = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

In particular, we use the F1-score in two of its versions, namely macro and weighted. We chose to report these variants because of the huge imbalance between the two classes in the datasets.

Since the F1-score ignores true negatives, in fact, it can be misleading for unbalanced classes [34]. The macro F1-score is calculated by taking the arithmetic mean of all the F1-scores, for each class, assigning equal importance to every class. The weighted F1-score, on the other hand, also considers the support of each class, assigning a greater importance to those that occur more often.

Tables I, II and III summarize the results obtained by the proposed system with the three datasets as the window size changes, both with and without drift detection. In addition to the metrics described above, we also report the percentage of times an already trained model was reused, instead of training a new one unnecessarily. As for the system without concept drift detection, which is considered as a baseline, we are not interested in the percentage of reused models, so we directly report its best case with the window sizes considered ($w = 150$). Comparing the results shown in the three tables brings up some interesting considerations.

Table I shows the results obtained with the SHUTTLE dataset, which is completely without concept drift. As can be expected, in this case the performance of the system with

Table II
SYSTEM EVALUATION WITH THE SMTP DATASET WHEN VARYING WINDOW SIZE.

Drift Detection	Window size	Reused Models	Macro F1-score	Weighted F1-score	Acc
Yes	25	98%	0.45	0.90	0.83
	50	91%	0.48	0.95	0.90
	100	87%	0.48	0.96	0.91
	150	93%	0.48	0.95	0.90
No	150	-	0.40	0.80	0.66

Table III
SYSTEM EVALUATION WITH THE SRDRIFT DATASET WHEN VARYING WINDOW SIZE.

Drift Detection	Window size	Reused Models	Macro F1-score	Weighted F1-score	Acc
Yes	25	90%	0.74	0.92	0.93
	50	88%	0.89	0.96	0.96
	100	83%	0.86	0.95	0.95
	150	72%	0.88	0.96	0.96
No	150	-	0.57	0.73	0.66

concept drift detection is very similar to that of the baseline without drift detection.

This is positive, as it means that the system performs very well even in the absence of concept drift. In particular, Table I shows that the average accuracy and average weighted F1-score for the systems with drift detection are 0.955 and 0.9475, respectively.

We note that the system with an excessively small window size ($w = 25$) is the one with the worst results, especially for the macro F1-score. The percentage of reused models is obviously very high (99%) since no drifts are present in the dataset.

Surprisingly, Table II shows that our system obtains considerably better results than the baseline with the SMTP dataset. Notably, we observe that the average accuracy goes from 88.5% for systems with drift detection to 66% for the baseline. This suggests that the dataset is actually affected by a slight incremental concept drift, which is difficult to detect by direct inspection. Such insight is also confirmed by the percentage of reused models, averaging 92%, which is lower if compared to the SHUTTLE dataset.

Table III shows the results obtained with the SRdrift dataset, which contains sudden and reoccurring concept drifts. We note how the percentage of reused models (83% on average) is lower than in previous cases, which is to be expected given that SRdrift explicitly includes drifts.

Nevertheless, the percentage is still very high, proving that the history of past models is extremely effective in minimizing the number of re-training needed. Table III also shows that the system with drift detection again achieves significantly better performance than the baseline. In fact, the average accuracy of

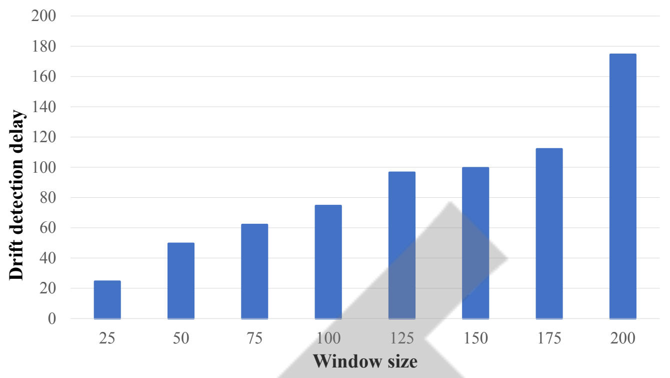


Figure 4. Drift detection delay of the proposed system in the SRDrift dataset, when varying window size.

the proposed system is 0.95 compared to 0.66 for the baseline. The macro and weighted F1-score values are also much higher (0.84 and 0.95 vs. 0.57 and 0.73, respectively).

Finally, Figure 4 shows the drift detection delay as the window size changes. Unsurprisingly, when the window size increases, the time required to report drifts grows accordingly. On the other hand, excessively small windows are not very representative of the incoming data and cause overall system performance to decline, as can be seen from the three tables. Consequently, it is necessary to achieve a trade-off between concept drift detection speed and accuracy. Intermediate window sizes such as 50 and 100 are those that appear most promising in this respect.

V. CONCLUSIONS

In this paper, we studied the phenomenon of concept drift in the context of streaming data for smart systems, highlighting the importance of explicitly handling reoccurring concepts. To this end, we proposed a new system that combines anomaly detection and concept drift detection techniques, using the static LOF algorithm in a hybrid way to handle streaming data. We also proposed a novel method to explicitly address reoccurring concept drift, by maintaining a history of past models and minimizing the number of re-training phases required, even in the case of sudden drifts.

The proposed system is characterized by a multi-tier architecture, and is composed of two concept drifts detection modules and an ensemble of LOF models for anomaly detection. To validate our approach, we have extensively evaluated our system with multiple experiments on three datasets.

The results show that the system is effective in detecting outliers whether the input data have drift or not, achieving high accuracy and F1-score values.

In addition, the percentage of reused models is extremely high, proving that the adopted approach is effective in handling reoccurring drifts while minimizing the number of new models that need to be trained. As future work, we would like to dynamically use data windows of variable size, depending on the frequency and severity of the detected drift in a certain time frame.

Another improvement to the proposed system would be to also consider a dynamic number of models in the history, to better adapt to periods when the ratio of model reuse is particularly high or low, or to devise a more sophisticated model replacement policy, for example using reputation mechanism [35], [36].

This would allow the system to reduce the window size as much as possible and increase the number of models in periods where many different concepts occur, and conversely, increase the window size and reduce the number of models in more stable periods, leading to an overall increase in performance.

ACKNOWLEDGMENTS

This work is partially funded by the European Union - FESR o FSE, PON Ricerca e Innovazione 2014-2020 - DM 1062/2021.

REFERENCES

[1] A. De Paola, P. Ferraro, S. Gaglio, G. Lo Re, M. Morana, M. Ortolani, and D. Peri, "A context-aware system for ambient assisted living," in *International Conference on Ubiquitous Computing and Ambient Intelligence*. Springer, 2017, pp. 426–438.

[2] Y. Liu, L. Kong, and G. Chen, "Data-oriented mobile crowdsensing: A comprehensive survey," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2849–2885, 2019.

[3] P. Ferraro and G. Lo Re, "Designing ontology-driven recommender systems for tourism," in *Advances onto the Internet of Things*. Springer, 2014, pp. 339–352.

[4] R. Al-amri, R. K. Murugesan, M. Man, A. F. Abdulateef, M. A. Al-Sharafi, and A. A. Alkahtani, "A review of machine learning and deep learning techniques for anomaly detection in IoT data," *Applied Sciences*, vol. 11, no. 12, p. 5320, 2021.

[5] K. K. Santhosh, D. P. Dogra, and P. P. Roy, "Anomaly detection in road traffic using visual surveillance: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–26, 2020.

[6] S. Ali, T. Glass, B. Parr, J. Potgieter, and F. Alam, "Low cost sensor with IoT LoRaWAN connectivity and machine learning-based calibration for air pollution monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[7] V. Agate, F. Concone, and P. Ferraro, "A resilient smart architecture for road surface condition monitoring," in *The Proceedings of the International Conference on Smart City Applications*. Springer, 2021, pp. 199–209.

[8] D. Santani, T.-M.-T. Do, F. Labhart, S. Landolt, E. Kuntsche, and D. Gatica-Perez, "DrinkSense: Characterizing youth drinking behavior using smartphones," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2279–2292, 2018.

[9] F. Restuccia, P. Ferraro, S. Silvestri, S. K. Das, and G. Lo Re, "IncentMe: Effective mechanism design to stimulate crowdsensing participants with uncertain mobility," *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1571–1584, 2018.

[10] V. Agate, A. De Paola, G. Lo Re, and M. Morana, "Vulnerability Evaluation of Distributed Reputation Management Systems," in *InfQ 2016 - New Frontiers in Quantitative Methods in Informatics*. ICST, Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 1–8.

[11] 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.

[12] A. De Paola, P. Ferraro, S. Gaglio, G. Lo Re, and S. K. Das, "An adaptive bayesian system for context-aware data fusion in smart environments," *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1502–1515, 2016.

[13] A. Timilsina, A. R. Khamesi, V. Agate, and S. Silvestri, "A reinforcement learning approach for user preference-aware energy sharing systems," *IEEE Transactions on Green Communications and Networking*, 2021.

[14] V. Agate, F. M. D'Anna, A. De Paola, P. Ferraro, G. Lo Re, and M. Morana, "A behavior-based intrusion detection system using ensemble learning techniques." in *ITASEC*, 2022.

[15] V. Agate, A. R. Khamesi, S. Silvestri, and S. Gaglio, "Enabling peer-to-peer user-preference-aware energy sharing through reinforcement learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.

[16] A. De Paola, P. Ferraro, G. Lo Re, M. Morana, and M. Ortolani, "A fog-based hybrid intelligent system for energy saving in smart buildings," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 7, pp. 2793–2807, 2020.

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[18] V. Agate, A. De Paola, P. Ferraro, G. Lo Re, and M. Morana, "Secure-ballot: A secure open source e-voting system," *Journal of Network and Computer Applications*, vol. 191, 2021.

[19] P. D. Domański, "Study on statistical outlier detection and labelling," *International Journal of Automation and Computing*, vol. 17, no. 6, pp. 788–811, 2020.

[20] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[21] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proceedings of the conference on research in adaptive and convergent systems*, 2019, pp. 161–168.

[22] J. Gao, W. Ji, L. Zhang, A. Li, Y. Wang, and Z. Zhang, "Cube-based incremental outlier detection for streaming computing," *Information Sciences*, vol. 517, pp. 361–376, 2020.

[23] 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, pp. 1–37, 2014.

[24] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[25] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[26] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, pp. 340–351, 2020.

[27] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013.

[28] M. U. Togbe, Y. Chabchoub, A. Boly, M. Barry, R. Chiky, and M. Bahri, "Anomalies detection using isolation in concept-drifting data streams," *Computers*, vol. 10, no. 1, p. 13, 2021.

[29] V. Agate, P. Ferraro, and S. Gaglio, "A cognitive architecture for ambient intelligence systems," in *AIC*, 2018, pp. 52–58.

[30] L. Perini, V. Vercruyssen, and J. Davis, "Quantifying the confidence of anomaly detectors in their example-wise predictions," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 227–243.

[31] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[32] S. Rayana, "ODDS library," 2016. [Online]. Available: http://odds.cs.stonybrook.edu

[33] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: https://archive.ics.uci.edu/ml

[34] D. M. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[35] V. Agate, A. De Paola, G. Lo Re, and M. Morana, "A platform for the evaluation of distributed reputation algorithms," in *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, 2018, pp. 1–8.

[36] ——, "A simulation software for the evaluation of vulnerabilities in reputation management systems," *ACM Transactions on Computer Systems (TOCS)*, vol. 37, no. 1-4, pp. 1–30, 2021.