Nonparametric and Online Change Detection in Multivariate Datastreams using QuantTree

Luca Frittoli, Diego Carrera, Giacomo Boracchi

Abstract—We address the problem of online change detection in multivariate datastreams, and we introduce QuantTree Exponentially Weighted Moving Average (QT-EWMA), a nonparametric change-detection algorithm that can control the expected time before a false alarm, yielding a desired Average Run Length (ARL_0). Controlling false alarms is crucial in many applications and is rarely guaranteed by online change-detection algorithms that can monitor multivariate datastreams without knowing the data distribution. Like many change-detection algorithms, QT-EWMA builds a model of the data distribution, in our case a QuantTree histogram, from a stationary training set. To monitor datastreams even when the training set is extremely small, we propose QT-EWMA-update, which incrementally updates the QuantTree histogram during monitoring, always keeping the ARL_0 under control. Our experiments, performed on synthetic and real-world datastreams, demonstrate that QT-EWMA and QT-EWMA-update control the ARL_0 and the false alarm rate better than state-of-the-art methods operating in similar conditions, achieving lower or comparable detection delays.

Index Terms—online change detection, nonparametric monitoring, multivariate datastreams, histograms, false alarms.

1 Introduction

Change detection in datastreams [1] is a challenging problem with relevant applications in many domains including quality control [2], security [3], cryptographic attacks [4], finance [5], and in several engineering problems, where control charts have been employed for decades [6]. A relevant example is industrial process monitoring, where production machinery is equipped with multiple sensors that measure vibration frequency, flow, temperature, pressure, etc. These observations form datastreams that must be monitored since any distribution change might indicate failures or ongoing deterioration of specific components such as bearings and gears, thus change detection can be key for predictive maintenance [7]. Change detection is also studied in the machine learning literature since classification and recommendation systems often operate on streaming data. Here, changes are called *concept drifts* and classifiers must be adapted to an evolving data-generating process [8].

Many of these applications require *multivariate* datastreams to be processed *online*, i.e., while acquiring new observations. This condition represents a crucial challenge when designing and implementing change-detection algorithms. On the one hand, a device implementing a change-detection algorithm has limited memory and can perform a limited number of operations at each time t, while datastreams are virtually unlimited. On the other hand, to increase the detection power, online change-detection algorithms would need in principle to analyze, at each time t, all the data observed until t, and this typically implies an increase of computational and memory requirements [5]. Another fundamental challenge is to monitor multivariate datastreams in a *nonparametric* manner, which enables change-detection algorithms to operate when the initial distribution ϕ_0 is unknown. Unfortunately, most nonparametric online change-detection algorithms can only monitor *univariate* datastreams [5]. A few nonparametric detectors for multivariate datastreams have been proposed in the literature [9], [10], but most of these address change-detection in a *one-shot* scheme by performing independent statistical tests over fixed-sized batches of data. Thus, these algorithms do not leverage the whole datastream and usually perform worse than their online counterparts [5].

We present *QuantTree Exponentially Weighted Moving Average (QT-EWMA)*, a nonparametric online change-detection algorithm that can effectively monitor multivariate datastreams while controlling the frequency of false alarms, namely detections that do not correspond to any distribution change. As in statistical hypothesis testing, having a certain number of false alarms is unavoidable in change detection. In particular, any change-detection algorithm is characterized by a trade-off between the frequency of false alarms and the detection power. In many applications, including our previous example of industrial process monitoring [7], promptly detecting changes is crucial (e.g., to avoid a failure). Still, any false alarm might trigger a costly intervention (e.g., the replacement of functioning machinery). Therefore, one typically sets a false alarm probability compatible with the resources allocated for these interventions and implements a change-detection algorithm that minimizes the detection delay subject to this bound on false alarms [5]. Moreover, controlling false alarms enables a fair comparison between the detection power of different solutions. Unfortunately, most online change-detection algorithms for multivariate datastreams from the literature, especially the nonparametric ones, cannot control false alarms effectively.

Typically, a change-detection algorithm has three main ingredients: *i)* a *model* $\widehat{\phi}_0$ of the initial distribution ϕ_0 to be fitted on a training set, *ii)* a *statistic* T, based on $\widehat{\phi}_0$, that yields a known response when data are drawn from ϕ_0, and *iii)* a *decision rule* to analyze the values of T and report changes. Typically, the decision rule consists in comparing T with a threshold h defined to yield the desired false alarm probability. One-shot detectors, which analyze a fixed amount of data, have thresholds that do not depend on t

and are simply defined as quantiles of T, which can either be computed analytically [11] or by Monte Carlo simulations [9]. In contrast, the test statistic in online algorithms depends on t, as T_t takes into account all the data points acquired until t. In this case, computing the thresholds is more complicated since one typically wants to control the *Average Run Length* (ARL_0), i.e. the expected time before raising a false alarm [6].

The proposed QT-EWMA is a novel online nonparametric change-detection algorithm for multivariate datastreams. QT-EWMA combines a *QuantTree (QT)* histogram [9], used as a model $\widehat{\phi}_0$, and a novel online statistic T_t based on *Exponentially Weighted Moving Average (EWMA)* [12]. In particular, by QT-EWMA we monitor the proportion of incoming samples falling in each bin of the histogram, and use this to build an efficient and practical online change-detection algorithm. The theoretical properties of QuantTree [9] guarantee that QT-EWMA is completely nonparametric since the distribution of our statistic does not depend on ϕ_0, hence its thresholds $\{h_t\}_t$ controlling the ARL_0 can be set *a priori*. Moreover, these thresholds guarantee by design a constant false alarm probability over time and, consequently, a fixed false alarm rate at any time instant during monitoring. Thus, QT-EWMA controls both ARL_0 and false alarm rate.

We also introduce *QT-EWMA-update*, a new change-detection algorithm based on QT-EWMA that enables online monitoring even when the training set is extremely small, e.g. in concept-drift adaptation [8] when the change-detection algorithm has to be re-configured after a detection. In QT-EWMA-update we use new samples to update the bin probabilities of our initial QuantTree histogram, as long as no change is detected. This update improves the model $\widehat{\phi}_0$, thus increasing the detection power. Our updating procedure is compatible with the computational requirements of online monitoring schemes, and the distribution of the QT-EWMA-update statistic is also independent from ϕ_0, enabling the computation of thresholds controlling the ARL_0 through the same procedure as in QT-EWMA. Hence, QT-EWMA-update overcomes a major limitation characterizing several online and nonparametric change-detection algorithms, i.e., requiring a large training set to fit $\widehat{\phi}_0$ before monitoring. This is particularly useful when acquiring stationary data from ϕ_0 is difficult or costly. Our main contributions are:

- We present QT-EWMA, an online nonparametric change-detection algorithm for multivariate datastreams based on a novel EWMA statistic (Section 4.1).
- We prove that the bin probabilities of QuantTree histograms follow a Dirichlet distribution, and this allows us to compute the thresholds $\{h_t\}_t$ of QT-EWMA by an efficient Monte Carlo scheme. These thresholds enable controlling the ARL_0 and false alarm rates for any ϕ_0 (Section 4.2).
- We propose QT-EWMA-update, which enables monitoring when the training set is extremely small by updating the QuantTree histogram online (Section 5).
- We propose two simple yet theoretically sound procedures to extend a generic one-shot detector to monitor datastreams controlling the ARL_0 (Section 6), which we employ as baselines in our experiments.

Our experiments, performed on both synthetic and real-world datastreams, show that QT-EWMA controls the ARL_0 better than the baselines and *Scan-B* [13], a competing algorithm based on a *Maximum Mean Discrepancy (MMD)* statistic [10], regardless of the training set size. Our results also show that QT-EWMA operates at the expected false alarm rate, which Scan-B does not guarantee. Moreover, QT-EWMA achieves similar or lower detection delays than Scan-B, especially on real-world datastreams. Most importantly, our QT-EWMA-update achieves significantly better detection performance compared to all the nonparametric alternatives when the training sets are extremely small. Our code and the thresholds of QT-EWMA and QT-EWMA-update are available at: https://boracchi.faculty.polimi.it/Projects

This paper extends our previous work [14], where we introduced QT-EWMA. The major original contribution of this paper is QT-EWMA-update, which enables monitoring when an extremely small training set is provided. Moreover, we extend the results presented in [9] by proving that the bin probability vector of a QuantTree histogram is a realization of a Dirichlet random vector with known parameters, and this allows us to derive a very efficient Monte Carlo scheme to compute thresholds, reducing the runtime of the simulations by 25% compared to [14].

The rest of the paper is organized as follows: in Section 2 we survey the change-detection literature, focusing on methods operating on multivariate datastreams, and in Section 3 we provide a formal definition of the online change-detection problem. In Sections 4 and 5 we introduce the QT-EWMA and QT-EWMA-update algorithms, respectively, and our procedure to compute thresholds controlling the ARL_0. In Section 6 we illustrate how to extend one-shot change detectors to monitor datastream controlling the ARL_0, and discuss the theoretical guarantees and limitations of these approaches. In Section 7 we show that the computational complexity and memory requirements of QT-EWMA and QT-EWMA-update favorably compare to those of the alternative solutions and finally in Section 8 we demonstrate the effectiveness of our solutions by testing it on both synthetic and real-world datastreams.

2 Related Work

Most change-detection algorithms in the literature employ models and statistics designed to analyze univariate datastreams [2], [5], [12]. The vast majority of these methods lack straightforward extensions to multivariate data, especially those leveraging nonparametric statistics based on ranks [5]. Change detection in multivariate datastreams has often been addressed in *multi-stream* (*multi-channel*) settings, i.e., by separately analyzing each component of the datastream [15], [16], [17]. However, the hypotheses underpinning multi-stream monitoring are fundamentally different from those of change-detection in multivariate datastreams. In fact, [15], [16], [17] assume that the components of the input vector are generated by a 1-dimensional random variables, and ignore correlations among them. Moreover, in multi-stream settings, changes typically affect the distribution of a subset of these random variables [15], while, in multivariate settings, more general distribution changes are considered [18],

TABLE 1
Properties of the most relevant change-detection algorithms designed for multivariate datastream monitoring.

	algorithm	nonpar.	online	control ARL_0	update
Gauss	Hotelling CPM [19]		✓	✓	✓
	SS-CPD [20]		✓	✓	
	SPLL [21]	semipar.	with mod.	with mod.	
PCA	PCA-SPLL [22]	✓			
	PCA-CD [23]	✓	✓		
	Martingale [24], [25]	✓	✓		
MMD	Scan-B [13]	✓	✓	✓	✓
	NEWMA [26]	✓	✓	iff ϕ_0 known	
	NTK-MMD [27]	✓	✓		
Hist.	BG-CuSum [28]	✓	✓	iff ϕ_0 known	
	QuantTree [9]	✓	with mod.	with mod.	
	QT-EWMA	✓	✓	✓	
	QT-EWMA-update	✓	✓	✓	✓

including subtle changes in the correlation between components that are hard to detect by multi-stream analysis.

Many change-detection algorithms specifically designed for multivariate datastreams are parametric: two remarkable examples are the *Change Point Model (CPM)* [19] based on the Hotelling test statistic [29], and *Sequential Subspace Change Point Detection (SS-CPD)* [20]. Both methods perform online monitoring while controlling the ARL_0, but rely on the hypothesis that ϕ_0 is Gaussian. A popular approach to handle multivariate datastreams is to reduce the data dimension by computing the likelihood of the observations with respect to a Gaussian [3] or Gaussian mixture model $\widehat{\phi}_0$ [21], [30], which is fitted on a training set and therefore is quite flexible in modeling ϕ_0. As in [21], we call these methods *semiparametric*. The main limitation of parametric and semiparametric methods is the implicit assumption that ϕ_0 belongs to a known family of probability distributions, which typically does not hold in real-world datastreams. Some nonparametric approaches reduce the data dimension by *Principal Component Analysis (PCA)* [22], [23], or by a *strangeness measure* [24], [25] to monitor a univariate datastream, e.g. by Martingale-based permutation tests [24]. However, none of these methods based on dimensionality reduction can be set to maintain the target ARL_0.

The Maximum Mean Discrepancy (MMD) is a nonparametric statistic that was originally introduced for hypothesis testing [10], and has recently been employed for online change detection [13], [26], [27] following a sliding-window approach. Usually, these methods do not fit a model $\widehat{\phi}_0$, but compare the new observations directly to the training set, which has to be stored during monitoring [13]. Among these methods, Scan-B [13] is the only one where the ARL_0 can be set before deployment for any unknown distribution ϕ_0. However, Scan-B has a unique threshold, i.e. $h_t \equiv h$, defined by the asymptotic behavior of the ARL_0 when $h \to \infty$ [13], which does not guarantee an accurate control of the ARL_0 and the false alarm rate, as we show in our experiments. *NEWMA* [26] detects changes by analyzing the relation between two EWMA statistics based on MMD having different forgetting factors. Unfortunately, thresholds controlling the ARL_0 can be set only when the analytical expression of ϕ_0 is known [26], which limits the applicability of NEWMA. *Neural Tangent Kernel MMD (NTK-MMD)* [27] approximates the MMD statistic by training a neural network on samples from ϕ_0, to reduce the computational and memory overhead in online testing. In this case, the thresholds are computed by training multiple networks with different training/validation splits, and then bootstrapping over validation data, a procedure that does not control the ARL_0. A major limitation of algorithms based on MMD is that they require a large amount of reference data [13], [27]: our experiments show that Scan-B [13] yields poor performance when the training set is small, even though the algorithm includes the incoming samples into the reference data, thus updating over time. In contrast, QT-EWMA does not require such large training sets, and QT-EWMA-update yields even lower detection delays by incrementally updating $\widehat{\phi}_0$.

Histograms are very flexible nonparametric models to describe ϕ_0 [31]. A remarkable example is QuantTree [9], which adaptively defines a histogram over a training set drawn from ϕ_0. QuantTree histograms have been employed in a one-shot change-detection test [9], which cannot be directly used in online settings, leveraging a nonparametric statistic to assess whether a single batch of test data follows ϕ_0 or not. Another change-detection algorithm based on histograms is the *Binned Generalized Cumulative Sum (BG-CuSum)* [28], which can operate online controlling the ARL_0. However, this algorithm has been tested only on univariate datastreams, and it is infeasible to extend to multivariate data because the number of bins scales exponentially with the data dimension. Moreover, it requires to know the cumulative function of ϕ_0, or an accurate approximation, to enable controlling the ARL_0. Thus, when ϕ_0 is unknown, BG-CuSum requires a huge training set, especially when the data dimension is high [28]. The proposed QT-EWMA and QT-EWMA-update overcome all these limitations, enabling to control the ARL_0 regardless of ϕ_0 and the data dimension. Moreover, QT-EWMA-update is specifically designed to operate when the training set is small.

Table 1 summarizes the main properties of the most relevant change-detection algorithms designed to monitor multivariate datastreams. In particular, we consider the following properties: being nonparametric, being executed online, controlling the ARL_0, and being able to incrementally update the model using the incoming data. To the best of our knowledge, Scan-B [13] is the only nonparametric and online change-detection algorithm from the literature in which the target ARL_0 can be set independently of ϕ_0. As we show in Section 6, one-shot change-detection methods such as QuantTree [9] and *Semi-Parametric Log-Likelihood (SPLL)* [21] can be modified to operate online while controlling the ARL_0. Other methods that control the ARL_0 are either parametric [19], [20], or require the analytical expression of ϕ_0 [26], [28].

QT-EWMA-update shares some similarity with *incremental learning* methods [32], where streaming data samples are used to improve previously learned models [33]. In incremental learning and *learning in non-stationary environments* literature, the model to be improved is typically a classifier [34], [35], hence a certain amount of supervised data is required. In contrast, we consider an unsupervised setting in which we do not know whether the incoming samples follow ϕ_0 or not, and we incrementally update a model

$\widehat{\phi}_0$ using streaming data, similarly to other online change-detection algorithms [5], [19].

We remark that all the models and statistics described here might not be suitable to detect distribution changes in high-dimensional data such as signals or images. This is primarily due to the fact that models such as Gaussian mixtures [21] and histograms [9] cannot describe complicated structures, and in many statistics the computational overhead increases with the data dimension [13], [28]. Moreover, the higher the data dimension, the harder it is to detect distribution changes, an effect known as *detectability loss* [30]. For this reason, high-dimensional data samples typically undergo a feature-extraction procedure to reduce their dimension before being analyzed by any change-detection algorithm. This is a standard procedure that has been followed to prepare the Credit Card Fraud Detection dataset [36] and the INSECTS dataset [37].

3 Problem Formulation

We address the online change-detection problem in a virtually unlimited multivariate datastream $x_1, x_2 \ldots \in \mathbb{R}^d$. We assume that, as long as there are no changes, all the data samples are i.i.d. realizations of a random variable having unknown distribution ϕ_0. In the case of time series, this hypothesis is typically met after some pre-processing [5]. We define the *change point* τ as the unknown time instant when a change $\phi_0 \to \phi_1$ takes place:

$$x_t \sim \begin{cases} \phi_0 \text{ if } t < \tau \\ \phi_1 \text{ if } t \geq \tau \end{cases}. \tag{1}$$

We assume that both ϕ_0 and $\phi_1 \neq \phi_0$ are unknown, and that a training set TR containing N realizations of ϕ_0 is provided to fit $\widehat{\phi}_0$, which in our case is a QuantTree histogram [9]. After fitting $\widehat{\phi}_0$, an online change-detection algorithm assesses, for each new incoming sample x_t, whether the sequence $\{x_1, \ldots x_t\}$ contains a change point. Typically, a statistic T_t based on $\widehat{\phi}_0$ is computed at each incoming x_t, then a decision rule is applied. Usually, the rule consists in controlling whether $T_t > h_t$ for a certain threshold h_t, and the detection time t^* is defined as the first time instant when there is enough statistical evidence to claim that the datastream $\{x_1, \ldots x_{t^*}\}$ contains a change point, namely:

$$t^* = \min\{t : T_t > h_t\}. \tag{2}$$

As in any statistical test, the sequence of thresholds $\{h_t\}_t$ employed in change detection should be defined to control the probability of having a false alarm, namely a detection on data drawn from ϕ_0. In online settings, we measure the amount of false alarms by the Average Run Length [6], defined as $\text{ARL}_0 = \mathbb{E}_{\phi_0}[t^*]$, where the expectation is taken assuming that the whole datastream is drawn from ϕ_0. Thus, the ARL_0 is the average time before a false alarm. Ideally, the *target* ARL_0 of an online change-detection method should be set *a priori*, similarly to Type I error probability in hypothesis testing. The goal is to detect a distribution change as soon as possible, i.e., to minimize the *detection delay* $t^* - \tau$, while *controlling the* ARL_0, i.e. having an *empirical* ARL_0 that approaches the target ARL_0 set before monitoring. We remark that controlling the ARL_0 also provides an upper bound on the expected detection delay.

Algorithm 1 QT-EWMA

Input: datastream $x_1, x_2, \ldots$, target probabilities $\{\pi_j\}_{j=1}^K$, thresholds $\{h_t\}_t$, TR
Output: detection flag `ChangeDetected`, detection time t^*

1: `ChangeDetected` $\leftarrow$ False, $t^* \leftarrow \infty$
2: estimate QT histogram $\{(S_j, \pi_j)\}_{j=1}^K$ from TR and define $\{\tilde{\pi}_j\}$ as in (5)
3: $Z_{j,0} \leftarrow \tilde{\pi}_j \ \forall j = 1, \ldots, K$
4: **for** $t = 1, \ldots$ **do**
5: $\quad y_{j,t} \leftarrow \mathbb{1}(x_t \in S_j)$
6: $\quad Z_{j,t} \leftarrow (1-\lambda) Z_{j,t-1} + \lambda y_{j,t}, \quad j = 1 \ldots, K$
7: $\quad T_t \leftarrow \sum_{j=1}^K (Z_{j,t} - \tilde{\pi}_j)^2 / \tilde{\pi}_j$
8: $\quad$ **if** $T_t > h_t$ **then**
9: $\quad\quad$ `ChangeDetected` $\leftarrow$ True, $t^* \leftarrow t$
10: $\quad\quad$ **break**;
11: $\quad$ **end if**
12: **end for**
13: **return** `ChangeDetected`, t^*

When TR is small, the model $\widehat{\phi}_0$ is typically inaccurate and the change-detection algorithm yields high detection delays. In this case, the algorithm should be able to incrementally update $\widehat{\phi}_0$ using new samples, as in [5], [19], thus improving the detection performance.

4 QuantTree Exponentially Weighted Moving Average

Here we introduce the QT-EWMA algorithm (Section 4.1) and illustrate the procedure we follow to compute its thresholds controlling the ARL_0 (Section 4.2).

4.1 The QT-EWMA Algorithm

We propose QT-EWMA (Algorithm 1) to extend to online monitoring the QuantTree algorithm [9], which was originally designed for one-shot change detection. QuantTree models ϕ_0 by a histogram made of K bins $\{S_j\}_{j=1}^K$ constructed by splitting $\mathbb{R}^d$ along random directions. The splits are defined so that each bin S_j contains $\pi_j N$ samples from the training set TR, where $\{\pi_j\}_{j=1}^K$ is a given set of target probabilities. QuantTree histograms can model both univariate and multivariate distributions and, most importantly, enable nonparametric monitoring. In fact, the distribution of any statistic defined by the number of test samples falling in each bin S_j of a QuantTree histogram does not depend on ϕ_0 nor on the data dimension d, as demonstrated in [9]. Further details on QuantTree – including how to define the bins when TR cannot be exactly split to match the target probabilities – can be found in [9].

Here we define a novel online statistic T_t to monitor the proportion of samples falling in each bin of a QuantTree histogram constructed over TR (line 2). In particular, when a new sample x_t is acquired, we define K binary statistics from the indicator functions of each bin S_j, namely

$$y_{j,t} = \mathbb{1}(x_t \in S_j), \quad j = 1, \ldots, K, \tag{3}$$

to track in which bin x_t falls. Denoting the true bin probabilities $p_j = \mathbb{P}_{\phi_0}(S_j)$, namely, the probability of a point sampled from ϕ_0 to belong to S_j, we have that

$$\mathbb{E}_{\phi_0}[y_{j,t}] = p_j, \ j = 1, \ldots, K, \tag{4}$$

where the expected value $\mathbb{E}_{\phi_0}$ is computed under the assumption that $x_t \sim \phi_0$. Since ϕ_0 is unknown, so are the bin probabilities $(p_1, \dots, p_K)$, which are a realization of a random vector [9] and can be approximated by $\tilde{\pi}_j \approx p_j$, where $\tilde{\pi}_1, \dots, \tilde{\pi}_K$ are defined as:

$$\tilde{\pi}_j := \frac{\pi_j N}{N+1}, j < K \text{ and } \tilde{\pi}_K := \frac{\pi_K N + 1}{N+1}. \quad (5)$$

After evaluating the statistics $y_{j,t}$ for the incoming sample x_t (line 5), we compute the EWMA statistic [12] $Z_{j,t}$ (line 6), to monitor the proportion of data in S_j, for $j \in \{1, \dots, K\}$:

$$Z_{j,t} = (1-\lambda) Z_{j,t-1} + \lambda y_{j,t} \quad \text{where} \quad Z_{j,0} = \tilde{\pi}_j \,. \quad (6)$$

Finally, we define the QT-EWMA change-detection statistic:

$$T_t = \sum_{j=1}^{K} \frac{(Z_{j,t} - \tilde{\pi}_j)^2}{\tilde{\pi}_j}, \quad (7)$$

which is similar to the Pearson statistic [38]. In fact, T_t measures the overall difference between the proportion of samples $x_1, \dots, x_t$ falling in each bin S_j, represented by $Z_{j,t}$, and $\tilde{\pi}_j$, which represent their estimated expected values under ϕ_0. This difference naturally increases when $t > \tau$ as a consequence of a change $\phi_0 \to \phi_1$ since this modifies the probability of some bin S_j. The QT-EWMA statistic is computed at each incoming sample (line 7) and then compared against the corresponding threshold h_t to detect changes (line 8).

The distribution of any statistic defined over a QuantTree histogram does not depend on ϕ_0 nor on d, thus QT-EWMA is a nonparametric change-detection algorithm. This claim is substantiated by the theoretical results in [9], which we extend here by fully characterizing the probability distribution of $(p_1, \dots, p_K)$:

Proposition 1. *Let $\{S_j\}_{j=1}^K$ be a partitioning built by the QuantTree algorithm with target probabilities $\{\pi_j\}_{j=1}^K$ on a training set $TR \sim \phi_0$ of size N. Then, the bin probability vector $(p_1, \dots, p_K)$ is drawn from the Dirichlet distribution:*

$$(p_1, \dots, p_K) \sim \mathcal{D}(\pi_1 N, \pi_2 N, \dots, \pi_K N + 1). \quad (8)$$

Proof. We leverage the result in [39] linking the Dirichlet distribution to the *stick-breaking process*. In particular, the stick-breaking process generates a sequence of K random variables $q_1, \dots, q_K$ as

$$q_j = \prod_{k=1}^{j-1} (1 - \tilde{q}_k) \cdot \tilde{q}_j, \; j < K, \quad q_K = 1 - \sum_{j=1}^{K-1} q_j, \quad (9)$$

where $\tilde{q}_j$ for $j = 1, \dots, K-1$ are defined as

$$\tilde{q}_j \sim \text{Beta}\left(\gamma_j, \sum_{k=j+1}^{K} \gamma_j\right), \quad (10)$$

and $\gamma_1, \dots, \gamma_K$ are the parameters that define the stick-breaking process. In [39] it has been shown that

$$(q_1, \dots, q_K) \sim \mathcal{D}(\gamma_1, \dots, \gamma_K). \quad (11)$$

To prove the proposition it is enough to show that there exists a specific configuration of γ_j such that the bin probabilities p_j of a QuantTree histogram can be expressed as q_j in (9). To this purpose, we recall the result in [9] where it has been shown that p_j can be written as

$$p_j = \prod_{k=1}^{j-1} (1 - \tilde{p}_k) \cdot \tilde{p}_j, \; j < K, \quad p_K = 1 - \sum_{j=1}^{K-1} p_j, \quad (12)$$

where $\tilde{p}_j$ are independent and follow Beta distributions:

$$\tilde{p}_j \sim \text{Beta}\left(\pi_j N, \left(1 - \sum_{k=1}^{j} \pi_k\right) N + 1\right) \; j = 1, \dots, K-1. \quad (13)$$

Now, we only need to find a suitable choice of $\gamma_1, \dots, \gamma_K$ to express the $\tilde{p}_j$ as the $\tilde{q}_j$ in (10). If we define $\gamma_j = \pi_j N$ for $j < K$ as in (13) and $\gamma_K = \pi_K N + 1$, we obtain that:

$$\sum_{k=j+1}^{K} \gamma_k = \sum_{k=j+1}^{K} \pi_k N + 1 = \left(1 - \sum_{k=1}^{j} \pi_k\right) N + 1, \quad (14)$$

where the last equality follows from $\sum_{j=1}^K \pi_j = 1$. Equation (14) ensures the correspondence between $\tilde{p}_j$ in (13) and $\tilde{q}_j$ in (10), which implies the thesis. □

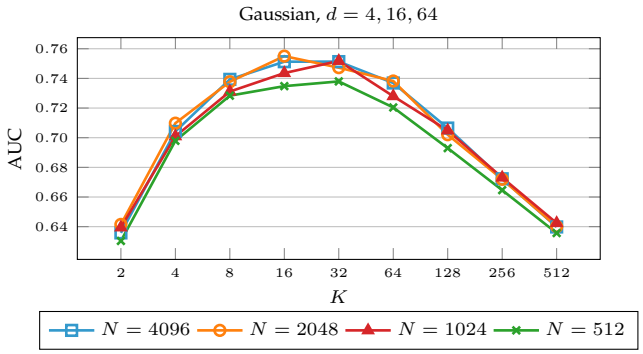

Fig. 1. Detection power of QT-EWMA on Gaussian datastreams with different training set size N when varying the number of bins K. The results, which are averaged over $d = 4, 16, 64$, show that histograms with a small K cannot describe ϕ_0 accurately and yield low detection performance. Also setting a very large K harms detection performance since, at a fixed N, increasing K yields inaccurate estimates $\{\tilde{\pi}_j\}_j$.

Proposition 1 means that, whenever we construct a QuantTree over a training set, we are partitioning $\mathbb{R}^d$ into K bins with probabilities $(p_1, \dots, p_K)$ drawn from the Dirichlet distribution in (8). Since the expected value of the j-th component of a random vector drawn from the Dirichlet distribution $\mathcal{D}(\gamma_1, \dots, \gamma_K)$ is $\gamma_j / \sum_{k=1}^K \gamma_k$, by simple algebraic manipulation we have that $\mathbb{E}_{\phi_0}[p_j] = \tilde{\pi}_j$, where $\tilde{\pi}_j$ are defined as in (5). Therefore, the values $\tilde{\pi}_1, \dots, \tilde{\pi}_K$ can be used as estimates of the bin probabilities $p_1, \dots, p_K$. Moreover, from the property of the Dirichlet distribution we have that $\text{var}[p_j] \to 0$ when $N \to \infty$, thus $\tilde{\pi}_j$ is a good estimate of p_j. We remark that the statistics employed in QuantTree [9] estimate the bin probability p_j by its target value π_j since it is assumed that a large training set is provided, and $\tilde{\pi}_j \to \pi_j$ as $N \to \infty$ by definition (5). Here we also consider cases where N is small, thus in the QT-EWMA statistics (6) and (7) we employ $\tilde{\pi}_j$, which is a more accurate estimate of p_j.

Impact of the choice of K. The number of bins K of the QuantTree histogram is a fundamental parameter that influences the change-detection performance of QT-EWMA. To analyze the impact of the choice of K, we test QT-EWMA

with $K = 2, 4, 8, \ldots, 512$. In particular, we compute the QT-EWMA statistic over 5000 stationary Gaussian datastreams and 5000 datastreams containing a change point at $\tau = 500$, and we measure the detection power by the *Area Under the ROC Curve (AUC)* given by the statistic values at $t = 1000$, namely T_{1000}. The distribution changes $\phi_0 \to \phi_1$ consist in random roto-translations of ϕ_0 generated by CCM [18] (see Section 8.1). Since the detection performance depends also on the training set size N, we employ different values of $N = 512, 1024, 2048, 4096$.

In Fig. 1 we report the average results obtained on datastreams with $d = 4, 16, 64$. Setting $K = 2, 4$ yields low AUC because histograms having such few bins cannot describe ϕ_0 well. Increasing the number of bins ($K = 128, 256, 512$) while keeping N fixed increases the variance of the bin probabilities p_j due to the properties of the Dirichlet distribution (8), harming the detection performance. Intermediate values – especially $K = 16, 32$ for the considered values of N – yield the best results, thus we in our experiments we select $K = 32$ as in [9]. As expected, the detection performance increases with N since $\text{var}[p_j] \to 0$ when $N \to \infty$. However, the improvement is substantial only when increasing N from 512 to 1024, while using a larger N yields a marginal improvement, see Fig. 1.

4.2 Computing Thresholds to Control the ARL_0

In online monitoring, the thresholds $\{h_t\}_t$ should guarantee the target $\text{ARL}_0 = \mathbb{E}_{\phi_0}[t^*]$, where t^* is the detection time, as defined in (1). Thanks to the properties of QuantTree [9], the distribution of any statistic based on QuantTree, including T_t, does not depend on ϕ_0 nor on d. Therefore, the QT-EWMA thresholds $\{h_t\}_t$ defined to yield the target ARL_0 will only depend on the EWMA parameter λ, the target bin probabilities $\{\pi_j\}_{j=1}^K$, and the training set size N. Following [40], we define $\{h_t\}_t$ to guarantee a fixed false alarm probability α at each time instant t. This implies that the detection time t^* under ϕ_0 is a Geometric random variable with parameter α [40], hence its expected value is

$$\text{ARL}_0 = \mathbb{E}_{\phi_0}[t^*] = \frac{1}{\alpha}. \tag{15}$$

To this purpose, as noted in [40], the thresholds $\{h_t\}_t$ must satisfy the following condition:

$$\mathbb{P}_{\phi_0}(T_t > h_t \mid T_k \leq h_k \; \forall k < t) = \alpha \quad \forall t \geq 1. \tag{16}$$

Since it is infeasible to exactly compute the conditional probabilities in (16), we resort to Monte Carlo simulations as in [5]. Leveraging Proposition 1, we simulate the construction of a QuantTree histogram on a training set $TR \sim \phi_0$ of size N by drawing its bin probabilities $(p_1, \ldots, p_K)$ from the Dirichlet distribution (8). Then, for each probability vector, we simulate the binary statistics $(y_{1,t}, \ldots, y_{K,t})$ in (3) of a stationary datastream of length $L = 5000$ by drawing them from the following multinomial distribution $\mathcal{M}$:

$$(y_{1,t}, \ldots, y_{K,t}) \sim \mathcal{M}(p_1, \ldots, p_K). \tag{17}$$

Then, we use these values $\{(y_{1,t}, \ldots, y_{K,t})\}_{t=1}^{5000}$ to compute the QT-EWMA statistics $\{T_t\}_{t=1}^{5000}$ by (6)–(7). To compute the thresholds $\{h_t\}_t$ yielding the desired ARL_0, we repeat the procedure above 1,000,000 times, and define h_1 as the empirical $(1-\alpha)$-quantile of all the values of T_1, where $\alpha = 1/\text{ARL}_0$ as in (15). Similarly, we define h_t with $t > 1$ as the $(1-\alpha)$-quantiles of the values T_t, using only those sequences $\{(y_{1,k}, \ldots, y_{K,k})\}_{k=1}^t$ whose statistics T_k have never exceeded any of the previous thresholds h_k for $k = 1, \ldots, t-1$. Computing the thresholds $\{h_t\}_t$ in this way guarantees that, for each time t, the empirical quantiles of T_t are conditioned to $T_k \leq h_k \; \forall k < t$, which in turn implies (16), hence the target ARL_0 is preserved [40].

We compute the thresholds h_t for $t = 1, \ldots, 5000$ and then fit a polynomial in powers of $1/t$ to these values that returns h_t for a given t, as suggested in [5]. This allows to both estimate h_t for $t > 5000$ and to improve the estimates $\{h_t\}_{t=1}^{5000}$ by leveraging correlation among thresholds. In our code we provide the polynomial expressions of the thresholds maintaining $\text{ARL}_0 = 500, 1000, 2000, 5000, 10000, 20000$, which can be very useful to control false alarms in high-throughput applications.

This procedure based on Proposition 1 is substantially more efficient than that presented in [14], where we computed the QT-EWMA statistics $\{T_t\}_{t=1}^{5000}$ from synthetic univariate Gaussian datastreams, i.e. $\phi_0 = \mathcal{N}(0, 1)$ after constructing a QuantTree histogram on a synthetic training set $TR \sim \phi_0$. Directly generating the bin probabilities $(p_1, \ldots, p_K)$ from the Dirichlet distribution (8) and the sequences $\{(y_{1,t}, \ldots, y_{K,t})\}_{t=1}^{5000}$ from the multinomial distribution (17) replaces the construction of a QuantTree histogram on each training set and the computation of the binary statistics $(y_{1,t}, \ldots, y_{K,t})$ (3) for each synthetic datastream, reducing by 25% the average runtime of the Monte Carlo simulations compared to [14].

Control over False Alarm Rates. An important consequence of setting a constant false alarm probability in (16) is that our thresholds can also control the false alarm rate at any time instant t. In fact, being t^* a Geometric random variable [40] with parameter α, the probability of having a false alarm before t corresponds to the following geometric sum:

$$\mathbb{P}_{\phi_0}(t^* \leq t) = \sum_{k=1}^{t} \alpha(1-\alpha)^{k-1} = 1 - (1-\alpha)^t. \tag{18}$$

This property enables us to assess the control of false alarms on datastreams containing a change point at τ by computing the proportion of datastreams in which $t^* \leq \tau$. This can then be compared to the target false positive rate in (18), which depends on the target ARL_0 (see Section 8.3).

5 Updating the QuantTree Histogram

Here we present QT-EWMA-update (Section 5.1), evaluate the impact of the updating speed on the detection performance (Section 5.2), and discuss stopping the update to avoid including post-change samples (Section 5.3).

5.1 The QT-EWMA-update Algorithm

In QT-EWMA we model the distribution ϕ_0 by means of a QuantTree histogram [9] constructed on a training set TR of size N. Then, during monitoring, we compute the statistic T_t (7) to compare the proportion of samples falling in each bin S_j with the estimated bin probabilities $\tilde{\pi}_j$. Being ϕ_0 unknown, we approximate the true bin probabilities

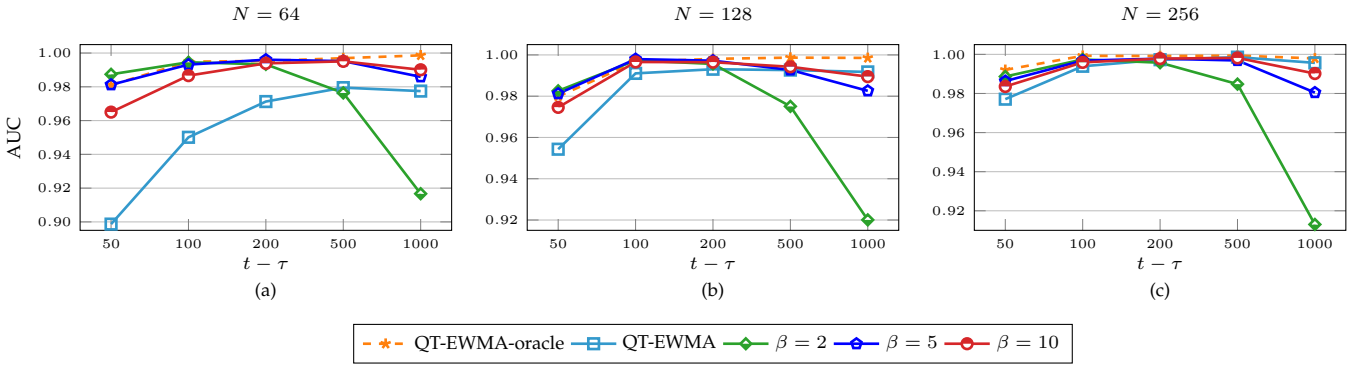

Fig. 2. Detection power of QT-EWMA-update ($\beta = 2, 5, 10$) compared to QT-EWMA ($\beta = \infty$) and the oracle QT-EWMA over univariate datastreams containing a change point at $\tau = 1000$, setting $N = 64, 128, 256$. In particular, we compute the AUC of the statistic T_t at different times $t > \tau$. QT-EWMA-update outperforms QT-EWMA right after the change, especially when $N = 64$. However, the performance of QT-EWMA-update decreases over time since it updates the bin probabilities using $x_t \sim \phi_1$ when $t > \tau$, and this is very apparent when the updating speed is high ($\beta = 2$).

$p_j = \mathbb{P}_{\phi_0}(S_j)$ by $\tilde{\pi}_j$, which is reasonable since $\mathbb{E}_{\phi_0}[p_j] = \tilde{\pi}_j$ and $\text{var}[p_j] \to 0$ as $N \to \infty$ thanks to Proposition 1. However, when N is small, the variance of p_j is high, thus $\{\tilde{\pi}_j\}_{j=1}^K$ yield inaccurate estimates of the true bin probabilities $\{p_j\}_{j=1}^K$, which harms the detection performance.

To overcome such limitation in online settings, we propose to update the model $\widehat{\phi}_0$ every time a new observation x_t arrives, which increases the detection power as in [5], [19]. In particular, we present *QT-EWMA-update*, where we replace each $\tilde{\pi}_j$ in (7) with an estimate $\hat{p}_{j,t}$ of the bin probability p_j that is incrementally updated when a new observation x_t becomes available, as long as no changes are detected. We define $\hat{p}_{j,t}$ as:

$$\hat{p}_{j,0} = \tilde{\pi}_j, \quad \hat{p}_{j,t} = (1 - \omega_t)\hat{p}_{j,t-1} + \omega_t y_{j,t} \quad t > 0, \qquad (19)$$

where $y_{j,t} = \mathbb{1}(x_t \in S_j)$, $\omega_t = 1/\beta(N + t)$ is a parameter representing the *updating speed* as it regulates the weight of the latest sample in the average, and $\beta \geq 1$ is a tuning parameter. We remark that all the quantities involved in our QT-EWMA-update statistic, including $\hat{p}_{j,t}$, are computed from a QuantTree histogram, thus the distribution of the statistic T_t does not depend on ϕ_0 nor on d [9]. Therefore, we can compute the thresholds of QT-EWMA-update for a given β by the same Monte Carlo procedure presented in Section 4.2, guaranteeing the control of the ARL_0.

5.2 The Role of the Updating Speed

The parameter β allows tuning the updating speed ω_t of QT-EWMA-update, which has a crucial impact on the detection performance. Setting $\beta = 1$ guarantees that $\hat{p}_{j,t} \to p_j$ when $t \to \infty$, as long as $x_t \sim \phi_0$, since (19) becomes the cumulative average of $y_{j,t}$, whose expected value is $\mathbb{E}_{\phi_0}[y_{j,t}] = p_j$ by definition. However, samples $x_t \sim \phi_1$ acquired after the change τ introduce a severe bias that harms the detection performance. Therefore, we propose to set $\beta > 1$, as this reduces the contribution of the most recent samples when updating $\hat{p}_{j,t}$. Setting $\beta > 1$ slightly biases the estimate $\hat{p}_{j,t}$ in stationary conditions, but turns out to be very beneficial in terms of detection delay, as shown in our experiments. We remark that QT-EWMA corresponds to the case $\beta = \infty$ since $\hat{p}_{j,t} \equiv \tilde{\pi}_j$.

To illustrate the trade-off regulated by β, we perform a simple experiment on univariate datastreams, setting $\phi_0 = \mathcal{U}(0, 1)$ and $\phi_1 = \mathcal{N}(0.5, 0.5)$. We generate 1000 univariate training sets from ϕ_0, 500 stationary univariate datastreams of length $L = 2000$ from ϕ_0, and 500 datastreams with initial distribution ϕ_0 containing a change point at $\tau = 1000$.

We monitor each datastream by the QT-EWMA-update algorithm setting $\beta = 2, 5, 10$, and QT-EWMA. We measure the detection power of these algorithms by the AUC of the statistics T_t computed at different times t after the change such that $t - \tau = 50, 100, 200, 500, 1000$. Since in this case we set $\phi_0 = \mathcal{U}(0, 1)$, we can easily compute the bin probabilities p_j of each QuantTree histogram, so we also test the "oracle" QT-EWMA algorithm, which uses p_j instead of $\tilde{\pi}_j$ in (6) and (7), and that is never updated. The oracle is an upper bound in terms of AUC, as it uses the analytical expression of ϕ_0.

In Fig. 2 we show the results of this experiment, using training set size $N = 64, 128, 256$. We observe that the QT-EWMA algorithm steadily improves its detection power when more samples drawn from ϕ_1 are considered. We also observe that the oracle QT-EWMA yields much better results than QT-EWMA when the training set is small, while the gap reduces when N is sufficiently large, as $\tilde{\pi}_j$ becomes an accurate estimate of p_j. In all cases, QT-EWMA-update yields a higher detection power compared to QT-EWMA right after τ, but shows a substantial decrease of the AUC over time due to the fact that the $\hat{p}_{j,t}$ are updated using samples from ϕ_1. Using a larger β mitigates this effect, at the cost of slightly reducing the detection power right after τ. In our experiments (Section 8) we set $\beta = 5$ as it yields the best detection performance in Fig. 2.

5.3 Stopping the Update

All in all, QT-EWMA-update outperforms QT-EWMA when the training set is small ($N = 64, 128$), but the update yields only a marginal advantage when $N = 256$ because $\tilde{\pi}_j$ is already a good estimate of p_j. Hence, when a large training set is available, QT-EWMA is preferable since it avoids the risk of updating the model using samples from the post-change distribution. QT-EWMA-update is preferred when N is small, but the update of $\widehat{\phi}_0$ should stop as soon as a sufficient number S of samples have been acquired without detecting a change, i.e., when $N + t = S$. This allows to reduce the risk of updating using samples $x_t \sim \phi_1$ once the estimated bin probabilities $\hat{p}_{j,t}$ are sufficiently accurate.

Since stopping the update does not change the the fact that the distribution of the statistic is independent from ϕ_0 and d, which is guaranteed by the properties of QuantTree [9], we compute thresholds controlling the ARL_0 for given values of β and S using the same Monte Carlo scheme illustrated in Section 4.2. The only difference with respect to QT-EWMA-update is that we update the bin probabilities $\hat{p}_{j,t}$ by (19) only for $t < S - N$, using $\hat{p}_{j,S-N}$ when $t \geq S - N$.

6 Online One-shot Change Detection

In this section we show how to adapt one-shot algorithms, namely statistical tests to assess whether a fixed amount of samples was generated by ϕ_0, to online change detection controlling the ARL_0. We focus on algorithms that operate batch-wise (Section 6.1), and element-wise (Section 6.2).

6.1 Datastream Monitoring by Batch-wise Detectors

Several change-detection algorithms process the datastream in separate non-overlapping batches W_t of ν samples:

$$W_t = [x_{(t-1)\nu+1}, \dots, x_{t\nu}]. \tag{20}$$

In particular, these algorithms compute for each incoming batch W_t a test statistic $T^\nu(W_t)$ based on a model $\widehat{\phi}_0$ fit over TR. For example, in QuantTree [9] $\widehat{\phi}_0$ is a histogram, while SPLL [21] employs a Gaussian mixture. These algorithms detect a change as soon as $T^\nu(W_t) > h^\nu$, where the threshold h^ν does not depend on t and is defined to control the false alarm probability over each batch W_t. In what follows we show how to set the threshold h^ν in batch-wise monitoring algorithms to maintain the target ARL_0 in online change detection, leveraging the following results:

Proposition 2. *Let W_t be any batch of ν samples drawn from ϕ_0 and let the detection threshold h^ν be such that*

$$\mathbb{P}_{\phi_0}(T^\nu(W_t) > h^\nu) = \alpha. \tag{21}$$

Then, the monitoring scheme $T^\nu(W_t) > h^\nu$ yields $\text{ARL}_0 \geq \nu/\alpha$.

Proof. Reported in the supplementary material. □

Therefore, setting $\alpha = \nu/\text{ARL}_0$, any batch-wise monitoring algorithm can be transformed into a conservative online change-detection algorithm, guaranteeing that the ARL_0 is greater than or equal to the target. A slightly different result holds when the threshold h^ν is conditioned on TR, e.g. when h^ν is computed by bootstrap. The following Proposition shows that, in this case, setting $\alpha = \nu/\text{ARL}_0$ guarantees that the ARL_0 is equal to the target.

Proposition 3. *Let W_t be any batch of ν samples drawn from ϕ_0 and let the detection threshold h^ν be such that*

$$\mathbb{P}_{\phi_0}(T^\nu(W_t) > h^\nu \mid TR) = \alpha, \tag{22}$$

Then, the monitoring scheme $T^\nu(W_t) > h^\nu$ yields $\text{ARL}_0 = \nu/\alpha$.

Proof. Reported in the supplementary material. □

Leveraging these results, we adapt two well-known batch-wise change-detection methods to monitor datastreams online while controlling the ARL_0: QuantTree [9] and SPLL [21]. The properties of QuantTree [9] guarantee that it is possible to set h^ν for (21) to hold for the Pearson statistic [38], independently from ϕ_0 and TR. Hence, Proposition 2 allows to set a lower bound on the ARL_0. In contrast, the distribution of the SPLL statistic, namely the log-likelihood, depends on ϕ_0, so h^ν has to be computed by bootstrapping over a portion of TR that was not used to fit $\widehat{\phi}_0$. In this case, the hypothesis of Proposition 3 holds since the false positive probability is conditioned on the provided TR, thus the online version of SPLL yields the target ARL_0. The main drawback of this bootstrap procedure is that it requires a large TR to fit $\widehat{\phi}_0$ and to compute h^ν.

TABLE 2
Computational complexity for each update of the statistic and memory requirement of QT-EWMA and QT-EWMA-update compared to the other methods, depending on the configuration (Section 8.2).

algorithm	complexity	memory
QT-EWMA	$\mathcal{O}(K)$	K
QT-EWMA-update	$\mathcal{O}(K)$	$2K$
QuantTree [9]	$\mathcal{O}(K)$	K
SPLL [21]	$\mathcal{O}(md)$	1
SPLL-CPM	$\mathcal{O}(md + w \log w)$	w
Scan-B [13]	$\mathcal{O}(nBd)$	$(n+1)Bd$

6.2 Datastream Monitoring by Element-wise Detectors

As pointed out in Section 2, an approach to change detection in multivariate datastreams consists in reducing the data dimension, constructing a univariate datastream that can be monitored by standard change-detection algorithms. Here we reduce the dimension of each incoming sample x_t as in SPLL [21] by computing the log-likelihood $-\log(\widehat{\phi}_0(x_t))$, where $\widehat{\phi}_0$ is a Gaussian mixture model fit on the entire TR. Then, we monitor the resulting univariate sequence by a nonparametric online CPM [5] leveraging the Lepage test statistic [41]. This algorithm, which we call SPLL-CPM, maintains the desired ARL_0 thanks to the CPM, which controls the ARL_0 on any univariate datastream [5].

7 Computational Complexity

Since efficiency is key in online monitoring [5], we analyze the computational complexity and memory requirements of QT-EWMA and QT-EWMA-update. We perform the same analysis on the online versions of QuantTree [9] and SPLL [21] (Section 6.1), SPLL-CPM (Section 6.2), and Scan-B [13] (Section 2). The results are summarized in Table 2.

QT-EWMA, QT-EWMA-update and QuantTree. These algorithms are extremely efficient and require an amount of memory that is constant over time and does not depend on the data dimension d. Before monitoring, a QuantTree histogram is constructed, requiring to rank the training set K times according to a specific component, resulting in $\mathcal{O}(KN \log N)$ operations [9], where K is the number of bins and N is the training set size. During monitoring, these three algorithms find the bin of the QuantTree histogram where each incoming sample x_t falls, resulting in $\mathcal{O}(K)$ operations [9]. Then, QT-EWMA and QT-EWMA-update compute the test statistics (3), (6), (7) at a constant overhead that falls within $\mathcal{O}(K)$. QT-EWMA-update also updates the

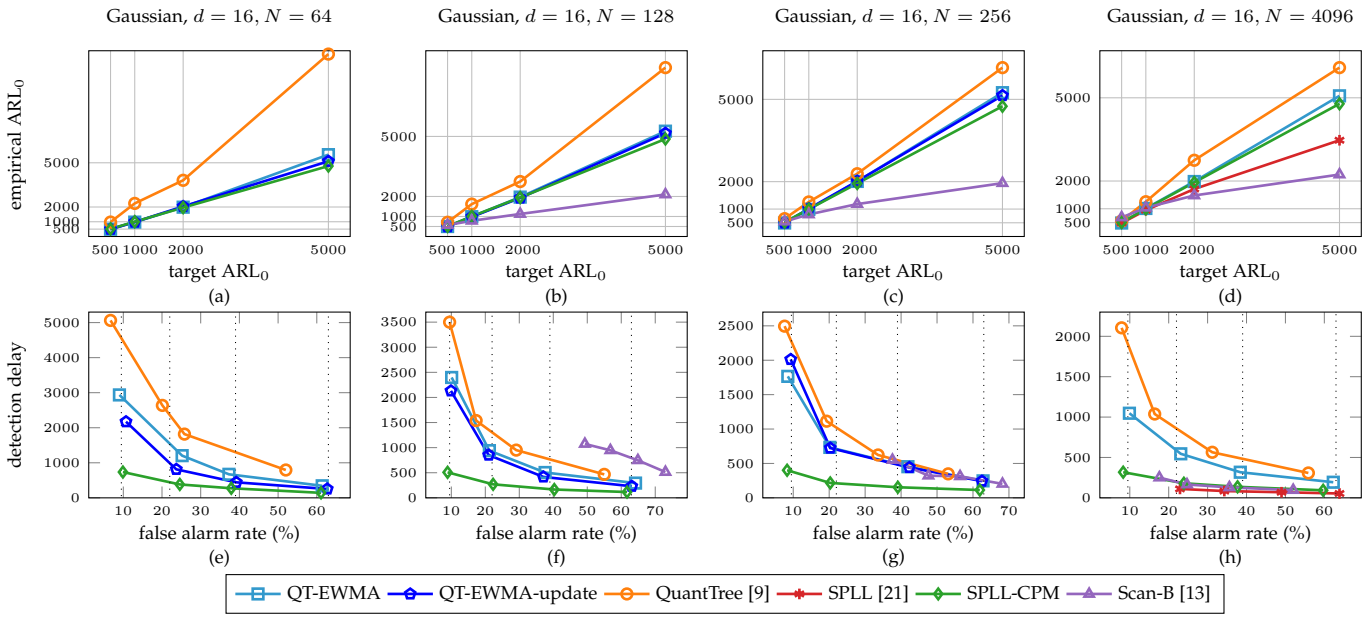

Fig. 3. Experimental results over Gaussian datastreams ($d = 16$). (a,b,c,d) show that the empirical ARL$_0$ of QT-EWMA, QT-EWMA-update and SPLL-CPM approaches the target, while the other methods do not maintain the target ARL$_0$. (e,f,g,h) show that, in terms of detection delay, the best-performing method is SPLL-CPM when using small training sets ($N = 64, 128, 256$) and SPLL when using large training sets ($N = 4096$). We observe that only QT-EWMA, QT-EWMA-update and SPLL-CPM achieve the target false alarm rates given by (18), which are represented by the vertical dotted lines.

bin probabilities of the QuantTree histogram by (19), requiring K additional operations, which also fall within $\mathcal{O}(K)$. The QuantTree algorithm instead computes the Pearson statistic at the end of each batch, and this does not increase the order of computational complexity either, resulting in $\mathcal{O}(K)$ operations as in QT-EWMA and QT-EWMA-update. In terms of memory requirement, QT-EWMA only stores the K values $Z_{j,t-1}, j = 1, \dots, K$ to compute (6) for each new sample x_t. QT-EWMA-update stores also the K estimated bin probabilities $\hat{p}_{j,t-1}, j = 1, \dots, K$, hence it requires to store $2K$ values in total. Similarly to QT-EWMA, QuantTree stores only the proportion of points in the batch belonging to each of the K bins to compute the Pearson statistic.

SPLL and SPLL-CPM. Both these algorithms are based on a Gaussian mixture model $\widehat{\phi}_0$ with m components fitted on TR. In SPLL, the likelihood of an incoming batch W_t is computed incrementally (before applying the logarithm) as the average likelihood of the samples $x_{(t-1)\nu+1}, \dots, x_{t\nu}$, requiring $\mathcal{O}(md)$ operations per sample [21]. Hence, only 1 value has to be stored in memory, namely the likelihood computed in the previous step. In contrast, the SPLL-CPM algorithm leverages the CPM framework [5] to monitor the stream of log-likelihood values $\{-\log(\widehat{\phi}_0(x_t))\}_t$. In particular, the Lepage test statistic [41] used in the CPM requires to sort the whole log-likelihood sequence obtained until time t, resulting in $\mathcal{O}(t \log t)$ operations on top of the $\mathcal{O}(md)$ operations required to compute $-\log(\widehat{\phi}_0(x_t))$. In this case, all the t values of the log-likelihood sequence have to be processed and stored at each time t, thus the computational complexity and memory requirement steadily increase over time. Since this is not desirable in online settings, the ranks of older observations can be discretized and stored in a histogram, yielding an approximation of the Lepage statistic [5] using only the most recent w samples.

Scan-B. The Scan-B algorithm [13] operates on sliding windows of size B, using n windows sampled from TR as a reference. For each incoming sample x_t, Scan-B updates n Gram matrices by computing B times the MMD statistic, resulting in $\mathcal{O}(nBd)$ operations [26]. The n reference windows and the current window have to be stored, yielding $(n+1)Bd$ values in memory [26]. Thus, the computational and memory requirements of Scan-B increase with d.

8 Experiments

In this section we show that QT-EWMA and QT-EWMA-update can control the ARL$_0$ and false alarm rates substantially better than competing methods, while achieving lower or comparable detection delays. We perform our experiments in two configurations: large ($N = 4096$) and small training sets ($N = 64, 128, 256$), to show the advantages of QT-EWMA-update when N is small.

8.1 Considered Datasets

We simulate Gaussian datastreams of dimension $d = 4, 16, 64$, choosing an initial Gaussian distribution ϕ_0 with random mean and covariance matrix, and roto-translating ϕ_0 to obtain the post-change distribution $\phi_1 = \phi_0(Q \cdot + v)$. We randomly select the roto-translation parameters Q and v using the CCM framework [18] to guarantee a symmetric Kullback-Leibler divergence $\text{sKL}(\phi_0, \phi_1) = 0.5, 1, 1.5, 2, 2.5, 3$. These settings are very useful to compare the detection performance at a different d [30]. For brevity, here we report only the results on Gaussian data with $d = 16, 64$, while $d = 4$ is in the supplementary material.

We also test on seven real-world multivariate datasets: Credit Card Fraud Detection ("credit", $d = 28$) from [36],

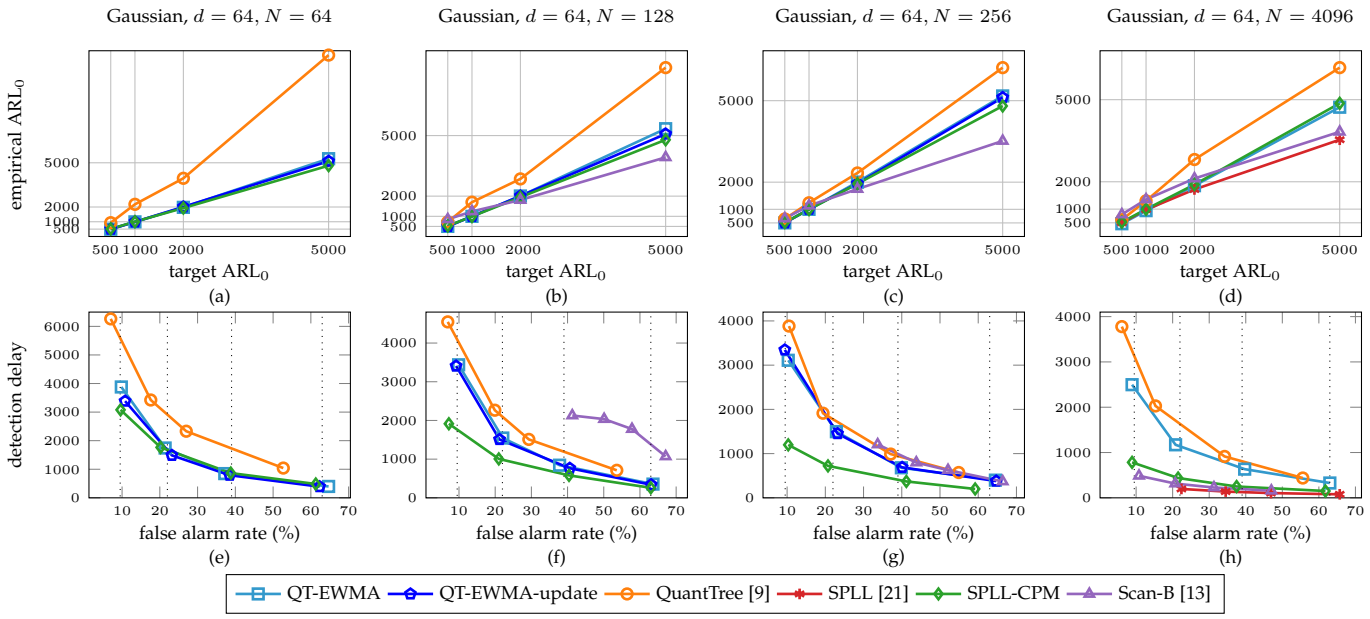

Fig. 4. Experimental results over Gaussian datastreams ($d = 64$). (a,b,c,d) show that the empirical ARL$_0$ of QT-EWMA, QT-EWMA-update and SPLL-CPM approaches the target, while the other methods do not maintain the target ARL$_0$. (e,f,g,h) show that, in terms of detection delay, the best-performing method is SPLL-CPM when using small training sets ($N = 64, 128, 256$) and SPLL when using large training sets ($N = 4096$). We observe that only QT-EWMA, QT-EWMA-update and SPLL-CPM achieve the target false alarm rates given by (18), which are represented in the plots by vertical dotted lines.

Sensorless Drive Diagnosis ("sensorless", $d = 48$), MiniBooNE particle identification ("particle", $d = 50$), Physicochemical Properties of Protein Ternary Structure ("protein", $d = 9$), El Niño Southern Oscillation ("niño", $d = 5$), and two of the Forest Covertype datasets ("spruce" and "lodgepole", $d = 10$) from the UCI Machine Learning Repository [42]. As in [9], we standardize the datasets and sum to the samples of "sensorless", "particle", "spruce" and "lodgepole" an imperceptible Gaussian noise to avoid repeated values, which harm the construction of QuantTree histograms. We prepare datastreams by randomly sampling these datasets, whose distribution can be considered stationary, and we introduce a change by applying a shift of a random vector drawn from a standard d-dimensional Gaussian distribution, scaled by the total variance of the dataset, as in [9], [22]. For brevity, we report only the average results over the "UCI+credit" datasets, while the results over individual datasets are in the supplementary material.

We also test on the INSECTS dataset [37] ($d = 33$), which contains features describing the wing-beat frequency of different species of flying insects, extracted from high-dimensional signals acquired by optical sensors. This dataset is meant as a classification benchmark for datastreams affected by concept drift. The dataset contains six concepts, each referring to data acquired under different environmental conditions affecting the insects' behavior. We assemble data from different concepts to form datastreams that include 30 types of realistic changes: we start sampling observations from one concept (ϕ_0) and switch to another (ϕ_1) introducing a change point.

To make sure that training and test data do not have samples in common, we generate Gaussian training and test data from different seeds, and sample real-world datastreams after removing TR from the datasets [36], [37], [42].

8.2 Considered Methods

To enable a fair comparison, we only consider change-detection methods where the target ARL$_0$ can be set before monitoring, regardless of ϕ_0. As shown in Section 2, the vast majority of the existing methods do not control the ARL$_0$ or do so only when ϕ_0 is known [20], [26], which is not guaranteed in general. For this reason, we compare QT-EWMA and QT-EWMA-update against QuantTree [9], SPLL [21], SPLL-CPM (described in Section 6), and Scan-B [13], which is the only method from the literature where the ARL$_0$ can be set independently on ϕ_0. Here we illustrate the configuration of the considered methods.

QT-EWMA, QT-EWMA-update and QuantTree. In our experiments we adopt the standard configuration of QuantTree [9], with $K = 32$ bins and uniform target probabilities $\pi_j = 1/K$. In fact, [31] shows that uniform histograms are very effective for change detection purposes. In QT-EWMA-update we set the parameter β, which regulates the updating speed in (19), to $\beta = 5$, which yields the best results in our preliminary experiment (see Fig. 2). In Section 5.2 we have shown that updating the QuantTree histogram is beneficial when the training set is extremely small, hence we test QT-EWMA-update with $N = 64, 128, 256$. In QuantTree we set the batch size $\nu = 32$ as in [9].

SPLL and SPLL-CPM. When monitoring datastreams sampled from Gaussian distributions and from the UCI+credit datasets, we set the number of components of the Gaussian mixture $\widehat{\phi}_0$ to $m = 1$. To maximize the performance on the INSECTS dataset, which contains data from 6 different species of insects [37], we set $m = 6$. In SPLL, we set the batch size $\nu = 32$ as for QuantTree, and employ $1/4$ of the training set to fit $\widehat{\phi}_0$ and the remaining samples to compute the threshold by bootstrap, as illustrated in Section 6.1. Since the threshold computation for SPLL requires a relatively

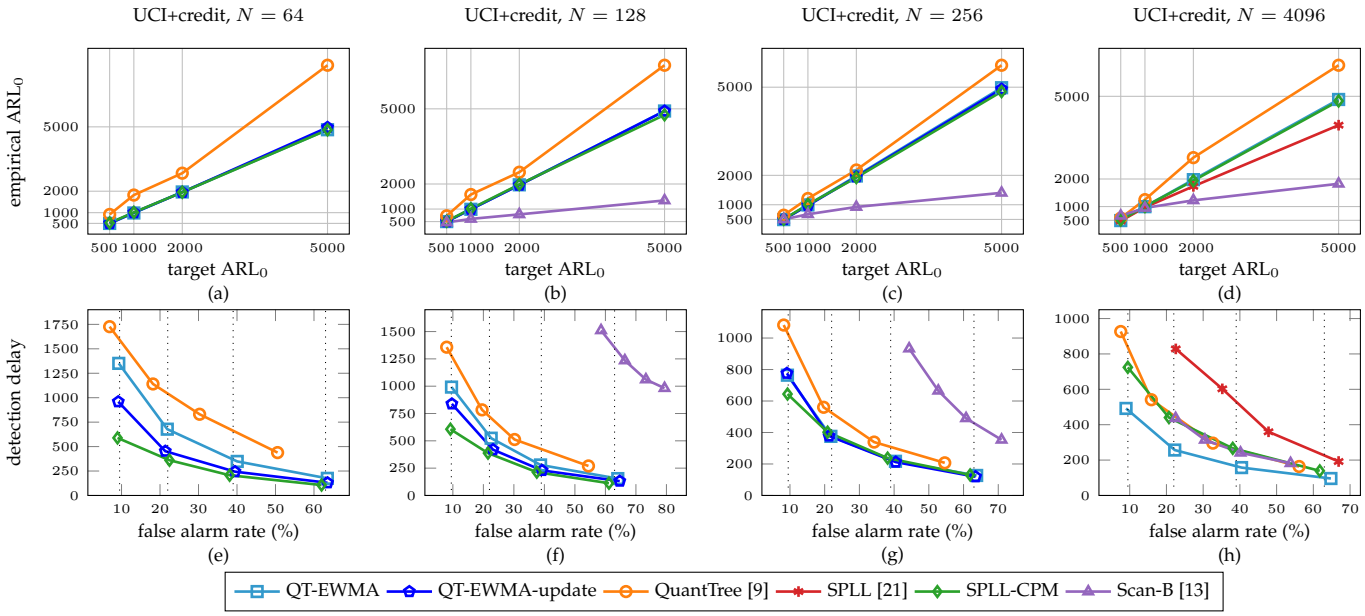

Fig. 5. Experimental results averaged over the UCI+credit datasets [36], [42]. (a,b,c,d) show that the empirical ARL_0 of QT-EWMA, QT-EWMA-update and SPLL-CPM approaches the target, while the other methods do not maintain the target ARL_0. (e,f,g,h) show that, in terms of detection delay, the best-performing methods are QT-EWMA-update and SPLL-CPM when using small training sets ($N = 64, 128, 256$) and QT-EWMA when using large training sets ($N = 4096$). We observe that only QT-EWMA, QT-EWMA-update and SPLL-CPM achieve the target false alarm rates given by (18), which are represented in the plots by vertical dotted lines.

large amount of data, we only test SPLL with large training sets ($N = 4096$). In SPLL-CPM we use the entire training set to fit $\hat{\phi}_0$, since the CPM employed to monitor the log-likelihood does not require a training set [5].

Scan-B. In all the experiments with large training sets ($N = 4096$) we test Scan-B [13] in its standard configuration, with $n = 5$ windows of size $B = 100$. This configuration cannot be employed when the training set is extremely small since Scan-B requires $N \geq nB$ [13]. For this reason, we set $B = 50$ when $N = 256$ and $B = 20$ when $N = 128$, keeping $n = 5$. All these configurations are among those suggested in [13]. Since no configurations with $B < 20$ are reported in [13], we do not test Scan-B when $N = 64$.

8.3 Figures of Merit

Empirical ARL_0. To assess whether QT-EWMA and the other considered methods maintain the target ARL_0, we compute the empirical ARL_0 as the average time before raising a false alarm. In particular, we set the target $ARL_0 = 500, 1000, 2000, 5000$, and prepare 5000 datastreams of length $L = 6 \cdot ARL_0$. According to (18), the probability of having a detection in these stationary datastreams is $\mathbb{P}_{\phi_0}(t^* \leq L) \approx 0.9975$.

Detection delay. We also evaluate the average detection delay, i.e. $ARL_1 = \mathbb{E}_{\phi_1}[t^* - \tau]$, where the expectation is taken assuming that a change point τ is present [6]. We run the considered methods configured with $ARL_0 = 500, 1000, 2000, 5000$ on 1000 datastreams of length $L = 10000$, each containing a change point at $\tau = 500$. We compute the empirical ARL_1 as the average difference $t^* - \tau$ over these datastreams, excluding those yielding false alarms.

False alarm rate. To assess whether the desired false alarm probability is achieved, we compute the percentage of datastreams in which a detection occurs at $t^* < \tau$. Setting the the target $ARL_0 = 500, 1000, 2000, 5000$ should yield a false alarm in 63%, 39%, 22% and 9.5% of the datastreams (18).

8.4 Results and Discussion

Empirical ARL_0. Fig. 3 (a,b,c,d) and Fig. 4 (a,b,c,d) plot the empirical ARL_0 achieved on Gaussian datastreams with $d = 16$ and $d = 64$, respectively, against the target ARL_0. These plots show that QT-EWMA, QT-EWMA-update can control the ARL_0 very accurately, independently from the data dimension d and the training set size N. This can be seen from the fact that the lines are close to the diagonal (note that axis scales are different). The empirical ARL_0 of QuantTree is higher than the target, and this is consistent with the statement of Proposition 2. In contrast, the empirical ARL_0 of Scan-B substantially departs from the target, in particular when the target ARL_0 is large. The limitations of Scan-B in controlling the ARL_0 are due to the fact that its threshold h is defined by an asymptotic approximation of the ARL_0 as $h \to \infty$ and $h/\sqrt{B} \to c$, where c is a constant [13]. Therefore, a larger target ARL_0 requires a larger threshold h and, in principle, a larger window size B. However, increasing B is infeasible because it would increase the computational and memory requirements (Table 2), and also the training set size since $N \geq nB$. Despite the theoretical guarantees of Proposition 3, we observe that also SPLL cannot maintain the target ARL_0 accurately, and this is due to inaccurate estimate of its thresholds, which are computed by bootstrap over a limited training set. In contrast, SPLL-CPM accurately controls the ARL_0 thanks to the properties of the CPM that monitors the log-likelihood [5]. Results obtained on the UCI+credit and INSECTS datasets (Fig. 5 (a,b,c,d) and Fig. 6 (a,b,c,d)) are consistent with those achieved on synthetic data.

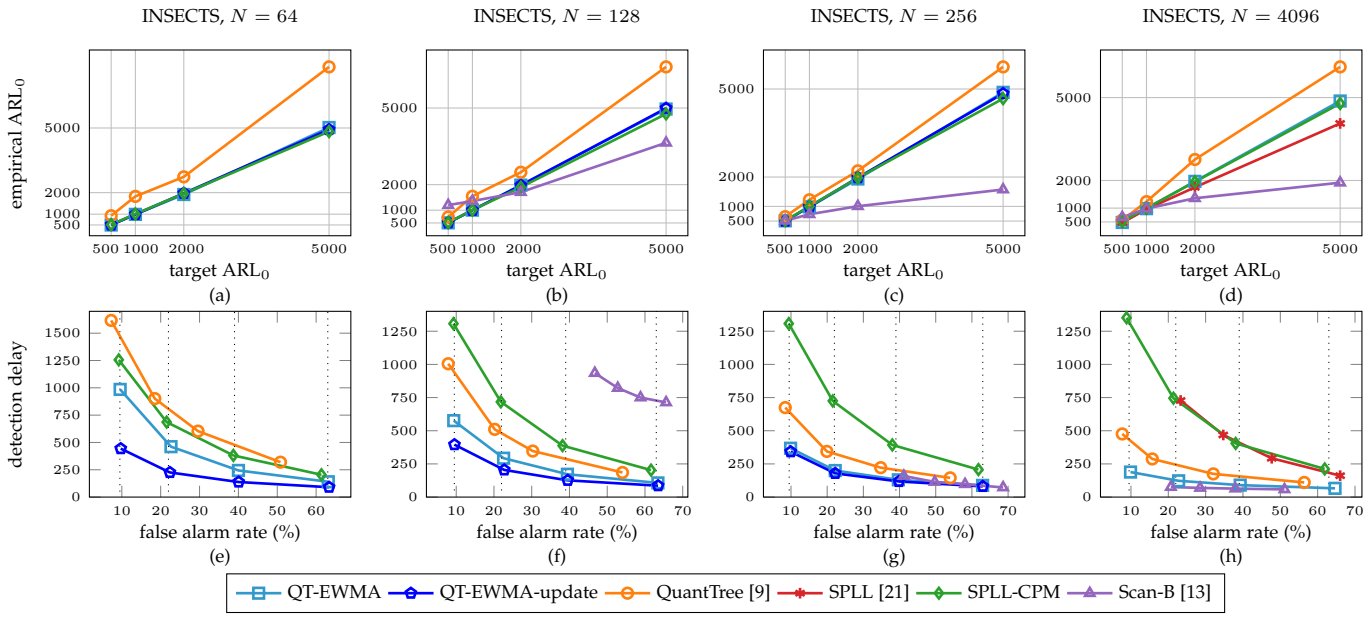

Fig. 6. Experimental results averaged over the INSECTS dataset [37]. (a,b,c,d) show that the empirical ARL$_0$ of QT-EWMA, QT-EWMA-update and SPLL-CPM approaches the target, while the other methods do not maintain the target ARL$_0$. (e,f,g,h) show that, in terms of detection delay, the best-performing method is QT-EWMA-update when using small training sets ($N = 64, 128, 256$) and Scan-B when using large training sets ($N = 4096$). We observe that only QT-EWMA, QT-EWMA-update and SPLL-CPM achieve the target false alarm rates given by (18), which are represented in the plots by vertical dotted lines.

Detection delay vs false alarm rate. We plot the average detection delay against the percentage of false alarms to assess the trade-off between these two quantities. Fig. 3 (e,f,g,h) and Fig. 4 (e,f,g,h) illustrate the performance on Gaussian datastreams with a change point at $\tau = 500$ and dimension $d = 16, 64$, respectively.

In terms of detection delay, QT-EWMA-update is the best nonparametric method when the training set is small ($N = 64, 128, 256$), being outperformed only by SPLL-CPM, which operates in ideal conditions since its parametric assumptions are met (ϕ_0 is a Gaussian). For the same reason, when the training set is large ($N = 4096$), both SPLL and SPLL-CPM outperform QT-EWMA. Also Scan-B outperforms QT-EWMA in these settings since statistics defined on histograms (such as that of QT-EWMA) are known to be less powerful than those based on MMD (such as that of Scan-B), as they perceive only changes affecting bin probabilities, and are totally blind to distribution changes inside each bin. Nevertheless, our experiments show that both QT-EWMA and QT-EWMA-update yield lower detection delays than Scan-B when the training set is small. All methods achieve higher detection delays as d increases due to detectability loss [30], which becomes apparent when we set the change magnitude to $\text{sKL}(\phi_0, \phi_1) = 0.5, 1, 1.5, 2, 2.5, 3$ in the CCM framework [18]. In the supplementary material we show that the detection delays of all the considered methods decreases when the change magnitude increases.

On the UCI+credit datasets (Fig. 5 (e,f,g,h)), QT-EWMA-update is the second-best performing method (slightly outperformed by SPLL-CPM) when the training set is small, and QT-EWMA is the best-performing method when $N = 4096$. On the INSECTS dataset (Fig. 6 (e,f,g,h)), QT-EWMA-update achieves the best detection delays when $N = 64, 128, 256$, and QT-EWMA approaches the performance of Scan-B when the training set is large ($N = 4096$). Moreover, QT-EWMA and QT-EWMA-update substantially outperform SPLL and SPLL-CPM, meaning that a QuantTree histogram can model the distribution of the INSECTS datasets much better than a Gaussian mixture. Remarkably, QT-EWMA and QT-EWMA-update consistently outperform QuantTree in all the considered scenarios, confirming that our sequential statistics are more powerful than the batch-wise Pearson statistic computed online.

As expected, the detection power of the methods based on QuantTree and Scan-B increases significantly with the training set size N. In contrast, SPLL-CPM achieves similar performance at different values of N since the Gaussian mixture model $\widehat{\phi}_0$ fitted on TR is sufficiently accurate even when N is small, and the CPM does not require a training set [5]. Most remarkably, QT-EWMA and QT-EWMA-update outperform Scan-B when N is small, meaning that in these settings our online statistics have higher detection power than monitoring sliding windows by the MMD statistic. Finally, we observe that QT-EWMA-update substantially outperforms QT-EWMA when the training set is extremely small ($N = 64, 128$), while it yields only a marginal improvement when $N = 256$, meaning that when $N \geq 256$ the values $\tilde{\pi}_j$ are sufficiently good estimates of p_j.

In all the considered scenarios, QT-EWMA, QT-EWMA-update and SPLL-CPM approach the target false alarm rates computed by (18). In contrast, QuantTree and SPLL have, respectively, lower and higher false alarm rates than the target in all the considered monitoring scenarios. This is due to the fact that the empirical ARL$_0$ of QuantTree is higher than the target as in Proposition 2, while the empirical ARL$_0$ of SPLL is lower than the target due to inaccurate threshold estimation, as observed previously. The false alarm rates of Scan-B, instead, exhibit a completely different behavior,

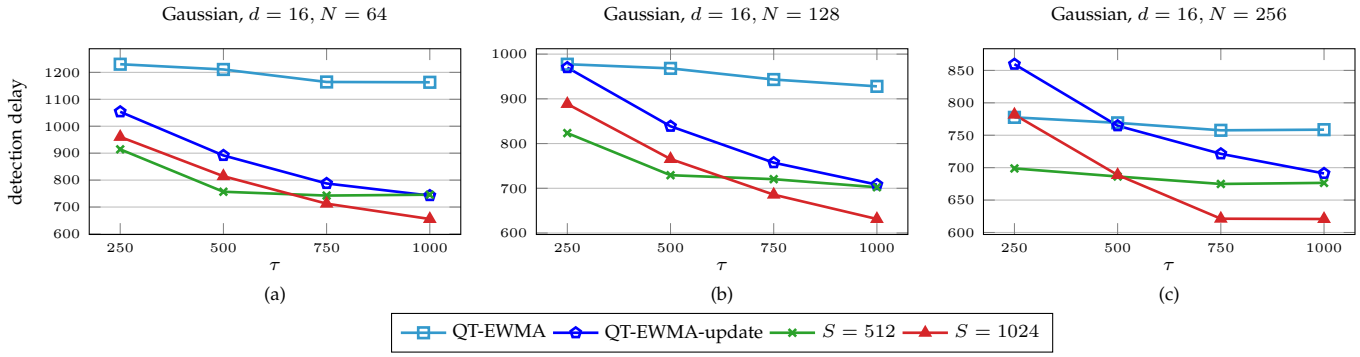

Fig. 7. Detection delays of QT-EWMA-update ($\beta = 5$) stopping the update after acquiring $S = 512, 1024$ samples, compared to QT-EWMA-update, QT-EWMA over Gaussian datastreams ($d = 16$) with change points at $\tau = 250, 500, 750, 1000$ with sKL$(\phi_0, \phi_1) = 2$. We set initial training set sizes $N = 64, 128, 256$ and target ARL$_0 = 2000$. We observe that QT-EWMA-update outperforms QT-EWMA, and that stopping the update improves the performance by reducing the risk of updating when $t > \tau$.

which also depends on ϕ_0 since its thresholds do not yield a constant false alarm probability.

Stopping the update. As discussed in Section 5.2, we can stop the update of the QuantTree histogram after acquiring a sufficient amount of data. This mitigates the problem of updating the estimated bin probabilities $\hat{p}_{j,t}$ when $t > \tau$, i.e., when $x_t \sim \phi_1$. To demonstrate this, we measure the detection delay of QT-EWMA-update where we stop the update after analyzing S samples, i.e., when $N + t = S$. In this experiment, we compare QT-EWMA-update stopping at $S = 512, 1024$ against QT-EWMA and QT-EWMA-update. Fig. 7 shows the detection delays achieved on Gaussian datastreams with $d = 16$ and length $L = 10000$ containing a change point at $\tau = 250, 500, 750, 1000$. We consider different training set sizes $N = 64, 128, 256$, and set target ARL$_0 = 2000$. As observed in the previous experiments, when $N = 64, 128$, QT-EWMA-update performs better than QT-EWMA, while the two algorithms have similar results when $N = 256$, confirming that updating the histogram is not necessary when N is sufficiently large. In all cases, the detection delay of QT-EWMA-update decreases when the change occurs later in the datastream since more samples $x_t \sim \phi_0$ are used to update $\hat{p}_{j,t}$.

The fact that the detection delays of QT-EWMA-update with stopping rule are lower than those of QT-EWMA-update confirms that reducing the amount of samples $x_t \sim \phi_1$ used to update $\hat{p}_{j,t}$ is beneficial. The detection performance of QT-EWMA-update with stopping rule improves when the change point τ occurs later in the datastream, unless the change occurs after having stopped the update, i.e. when $\tau > S - N$. In Fig. 7(a,b) we observe that setting $S = 512$ yields similar detection delays when $\tau = 500, 750, 1000$ since these changes occur after having stopped the update, thus all the $S - N$ samples used to update $\hat{p}_{j,t}$ are drawn from ϕ_0. In contrast, when $\tau = 250$ the detection delay is higher since samples from ϕ_1 might bias the estimates $\hat{p}_{j,t}$. We observe the same effect in Fig. 7(c) for $S = 1024$, where the detection delays for $\tau = 750, 1000$ are very similar and lower than those obtained when $\tau = 250, 500$.

9 Conclusions

We introduce QT-EWMA, a novel nonparametric online change-detection algorithm for multivariate datastreams. Our solution is efficient and effectively controls the ARL$_0$ and false alarm rates, which is very useful in practical applications. We also design an updating scheme for QT-EWMA and implement QT-EWMA-update. Here we update the estimated bin probabilities of the QuantTree histogram online, as soon as new data becomes available, enabling monitoring using a very small training set. Our experiments on synthetic and real-world datastreams show that alternative solutions do not provide such guarantees in nonparametric settings, and that QT-EWMA and QT-EWMA-update achieve excellent performance, especially on real-world data.

References

[1] C. C. Aggarwal, "On change diagnosis in evolving data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 587–600, 2005.

[2] D. M. Hawkins, P. Qiu, and C. W. Kang, "The changepoint model for statistical process control," *Journal of Quality Technology*, vol. 35, no. 4, pp. 355–366, 2003.

[3] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and H. Kim, "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, pp. 3372–3382, 2006.

[4] L. Frittoli, M. Bocchi, S. Mella, D. Carrera, B. Rossi, P. Fragneto, R. Susella, and G. Boracchi, "Strengthening sequential side-channel attacks through change detection," *Transactions on Cryptographic Hardware and Embedded Systems*, vol. 3, pp. 1–21, 2020.

[5] G. J. Ross, D. K. Tasoulis, and N. M. Adams, "Nonparametric monitoring of data streams for changes in location and scale," *Technometrics*, vol. 53, no. 4, pp. 379–389, 2011.

[6] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.

[7] T. D. Popescu, D. Aiordachioaie, and A. Culea-Florescu, "Basic tools for vibration analysis with applications to predictive maintenance of rotating machines: an overview," *International Journal of Advanced Manufacturing Technology*, vol. 118, pp. 2883–2899, 2022.

[8] 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.

[9] G. Boracchi, D. Carrera, C. Cervellera, and D. Macciò, "QuantTree: histograms for change detection in multivariate data streams," in *International Conference on Machine Learning*, 2018, pp. 639–648.

[10] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[11] A. Lung-Yut-Fong, C. Lévy-Leduc, and O. Cappé, "Robust change-point detection based on multivariate rank statistics," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 3608–3611.

[12] S. Roberts, "Control chart tests based on geometric moving averages," *Technometrics*, vol. 1, no. 3, pp. 239–250, 1959.
[13] S. Li, Y. Xie, H. Dai, and L. Song, "M-statistic for kernel change-point detection," *Advances in Neural Information Processing Systems*, vol. 28, pp. 3366–3374, 2015.
[14] L. Frittoli, D. Carrera, and G. Boracchi, "Change detection in multivariate datastreams controlling false alarms," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML–PKDD)*. Springer, 2021, pp. 421–436.
[15] Y. Xie and D. Siegmund, "Sequential multi-sensor change-point detection," in *2013 Information Theory and Applications Workshop*. IEEE, 2013, pp. 1–20.
[16] G. Fellouris and A. G. Tartakovsky, "Multichannel sequential detection–part I: Non-iid data," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4551–4571, 2017.
[17] Z. Sun, S. Zou, R. Zhang, and Q. Li, "Quickest change detection in anonymous heterogeneous sensor networks," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1041–1055, 2022.
[18] D. Carrera and G. Boracchi, "Generating high-dimensional datastreams for change detection," *Big Data Research*, vol. 11, pp. 11–21, 2018.
[19] K. Zamba and D. M. Hawkins, "A multivariate change-point model for statistical process control," *Technometrics*, vol. 48, no. 4, pp. 539–549, 2006.
[20] L. Xie, Y. Xie, and G. V. Moustakides, "Sequential subspace change point detection," *Sequential Analysis*, vol. 39, no. 3, pp. 307–335, 2020.
[21] L. I. Kuncheva, "Change detection in streaming multivariate data using likelihood detectors," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1175–1180, 2011.
[22] L. I. Kuncheva and W. J. Faithfull, "PCA feature extraction for change detection in multidimensional unlabeled data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 69–80, 2013.
[23] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A PCA-based change detection framework for multidimensional data streams," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 935–944.
[24] S.-S. Ho, "A Martingale framework for concept change detection in time-varying data streams," in *International Conference on Machine Learning*, 2005, pp. 321–327.
[25] N. Mozafari, S. Hashemi, and A. Hamzeh, "A precise statistical approach for concept change detection in unlabeled data streams," *Computers & Mathematics with Applications*, vol. 62, no. 4, pp. 1655–1669, 2011.
[26] N. Keriven, D. Garreau, and I. Poli, "NEWMA: a new method for scalable model-free online change-point detection," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3515–3528, 2020.
[27] X. Cheng and Y. Xie, "Neural tangent kernel maximum mean discrepancy," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6658–6670, 2021.
[28] T. S. Lau, W. P. Tay, and V. V. Veeravalli, "A binning approach to quickest change detection with unknown post-change distribution," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 609–621, 2018.
[29] H. Hotelling, "A generalized t test and measure of multivariate dispersion," in *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*. University of California, 1951.
[30] C. Alippi, G. Boracchi, D. Carrera, and M. Roveri, "Change detection in multivariate datastreams: Likelihood and detectability loss," *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, pp. 1368–1374, 2016.
[31] G. Boracchi, C. Cervellera, and D. Macciò, "Uniform histograms for change detection in multivariate data," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1732–1739.
[32] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Machine Learning*, vol. 1, no. 3, pp. 317–354, 1986.
[33] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, 2011.
[34] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2283–2301, 2012.
[35] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 620–634, 2013.
[36] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2017.
[37] V. Souza, D. M. dos Reis, A. G. Maletzke, and G. E. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1805–1858, 2020.
[38] E. L. Lehmann and J. P. Romano, *Testing statistical hypotheses*. Springer, 2006.
[39] B. A. Frigyik, A. Kapila, and M. R. Gupta, "Introduction to the Dirichlet distribution and related processes," *Technical Report UWEETR-2010-0006*, 2010.
[40] T. M. Margavio, M. D. Conerly, W. H. Woodall, and L. G. Drake, "Alarm rates for quality control charts," *Statistics & Probability Letters*, vol. 24, no. 3, pp. 219–224, 1995.
[41] Y. Lepage, "A combination of Wilcoxon's and Ansari-Bradley's statistics," *Biometrika*, vol. 58, no. 1, pp. 213–217, 1971.
[42] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

Luca Frittoli graduated in Mathematics at Università degli Studi di Milano in 2018, and is currently working towards the Ph.D. in Information Technology at Politecnico di Milano. His research interests include change detection in multivariate datastreams, concept-drift detection, and deep learning methods for anomaly detection and open-set recognition in images and point clouds.

Diego Carrera Diego Carrera graduated in Mathematics at Università degli Studi di Milano in 2013 and received the Ph.D. in Information Technology in 2018. In 2015 he has been visiting researcher at the Tampere University of Technology. Currently he is an Application Development Engineer at STMicroelectronics, where he is developing quality inspection systems to monitor the wafer production. His research interests are mainly focused on unsupervised learning algorithms, in particular change detection in high dimensional datastreams, anomaly detection in signal and images, and domain adaptation.

Giacomo Boracchi is an Associate Professor of Computer Engineering at Politecnico di Milano - DEIB, where he also received the Ph.D. in Information Technology (2008), after graduating in Mathematics (Università degli Studi di Milano, 2004). His research interests concern machine learning and image processing, and in particular change/anomaly detection, domain adaptation, image restoration and analysis. Since 2015 he is leading industrial research projects concerning outlier detection systems, X-ray systems, and automatic quality inspection systems. He has published more than 70 papers in international conferences and journals, he is currently associate editor for IEEE Transactions on Image Processing and in 2019 - 2020 he served as an associate editor for IEEE Computational Intelligence Magazine. In 2015 he received an IBM Faculty Award, in 2016 the IEEE Transactions on Neural Networks and Learning Systems Outstanding Paper Award, in 2017 the Nokia Visiting Professor Scholarship, and in 2021 the NVIDIA Applied Research Grant. He has held tutorials in major IEEE conferences: ICIP 2020, ICASSP 2018 and IJCNN 2017 and 2019.