Multimodal Batch-Wise Change Detection

Diego Stucchi, Luca Magri, Diego Carrera, and Giacomo Boracchi, *Member, IEEE*

***Abstract*— We address the problem of detecting distribution changes in a novel batch-wise and multimodal setup. This setup is characterized by a stationary condition where batches are drawn from potentially different *modalities* among a set of distributions in $\mathbb{R}^d$ represented in the training set. Existing change detection (CD) algorithms assume that there is a unique—possibly multipeaked—distribution characterizing stationary conditions, and in batch-wise multimodal context exhibit either low detection power or poor control of false positives. We present MultiModal QuantTree (MMQT), a novel CD algorithm that uses a single histogram to model the batch-wise multimodal stationary conditions. During testing, MMQT automatically identifies which modality has generated the incoming batch and detects changes by means of a modality-specific statistic. We leverage the theoretical properties of QuantTree to: 1) automatically estimate the number of modalities in a training set and 2) derive a principled calibration procedure that guarantees false-positive control. Our experiments show that MMQT achieves high detection power and accurate control over false positives in synthetic and real-world multimodal CD problems. Moreover, we show the potential of MMQT in Stream Learning applications, where it proves effective at detecting concept drifts and the emergence of novel classes by solely monitoring the input distribution.**

***Index Terms*— Change detection (CD), concept drift, histograms, multimodal monitoring, Stream Learning.**

I. Introduction

DETECTING distribution changes is a fundamental problem in machine learning and stream mining, as it is key to concept drift detection and adaptation [1]. Change detection (CD) is central in many applications like industrial/environmental [2] and urban [3] monitoring, security [4] and finance [5]. In this work, we address the *batch-wise* CD problem in a novel and challenging *multimodal* setup.

In batch-wise CD problems, data come in batches, typically of fixed size, and each batch contains independent and identically distributed (i.i.d.) samples drawn from a unique stationary distribution ϕ_0. In this scenario, the goal is to monitor incoming batches to detect any distribution change $\phi_0 \rightarrow \phi_1$, which yields batches from a different distribution ϕ_1. An illustrative example of a batch-wise CD problem is predictive maintenance, where inertial sensors mounted on a piece of machinery (e.g., a robotic arm) acquire a multivariate stream of high-throughput measurements for integrity monitoring purposes. Typically, these data are processed in *batches*, which are intermittently acquired over short time intervals. In this context, detecting distribution changes in batches is very important, as changes might indicate faults in the sensors or an unexpected evolution of the machinery operating conditions. Batch-wise CD is also popular in the concept-drift detection literature, where it has been tackled by various statistical tools, from simple two-sample tests to more sophisticated density-based models [6].

All the existing CD algorithms assume that ϕ_0 corresponds to a single stationary distribution. Unfortunately, this assumption does not match situations where multiple conditions are deemed normal, and stationary batches can be drawn from different distributions $\{\phi_{0,m}\}$, which we refer to as *modalities*. For example, the robotic arm previously considered might—in normal conditions—operate different types of movement at different speeds. These settings are illustrated in Fig. 1 (left), where the blue and green samples refer to batches acquired during different operating modes, thus are drawn from different modalities. As long as batches are drawn from any modality $\{\phi_{0,m}\}$ characterizing ϕ_0, no change has to be detected. In contrast, distribution changes modifying even a single modality, for instance $\phi_{0,1} \rightarrow \phi_{1,1}$, as shown in Fig. 1 (right), should be detected, as they possibly indicate an unexpected behavior.

This example illustrates the *batch-wise multimodal* CD problem we address, where stationary batches are drawn from a set of unknown modalities $\{\phi_{0,m}\}$. Multimodal CD requires a specific algorithm to: 1) describe the stationary condition $\phi_0 = \{\phi_{0,m}\}$ and 2) associate each batch to the modality $\phi_{0,i}$ it comes from. Such multimodal CD problem has never been addressed, neither in the CD nor concept drift literature. The most general CD algorithms, which model ϕ_0 as a multipeaked distribution (say Gaussian mixtures [7], [8], [9]), assume that in stationary conditions each batch is drawn from the whole mixture. As a consequence, in the batch-wise multimodal settings, they would consider batches from a single modality $\phi_{0,m}$ unusual, resulting in a false alarm.

We fill this gap and present MultiModal QuantTree (MMQT), a novel histogram-based algorithm that solves the batch-wise multimodal CD problem. This is a very powerful algorithm, as it is: 1) *nonparametric*, meaning that it does not make any assumption on the distributions of modalities $\{\phi_{0,m}\}$; 2) *efficient*, as it employs a single histogram to compute detection statistics; and 3) *practical*, since we show it can operate at a controlled false alarm rate, a property inherited by QuantTree [10], which provides a sound theoretical background.

In order to train an MMQT, we construct a *modality-agnostic* partitioning of the input domain (see Fig. 2, left)

Manuscript received 6 February 2023; revised 28 June 2023; accepted 3 July 2023. Date of publication 15 August 2023; date of current version 6 October 2023. This work was supported by the PNRR-PE-AI FAIR Project funded by the NextGeneration EU Program. *(Corresponding author: Diego Stucchi.)*

Diego Stucchi, Luca Magri, and Giacomo Boracchi are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy (e-mail: diego.stucchi@polimi.it).

Diego Carrera is with STMicroelectronics, 20041 Agrate Brianza, Italy.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2023.3294846.

Digital Object Identifier 10.1109/TNNLS.2023.3294846

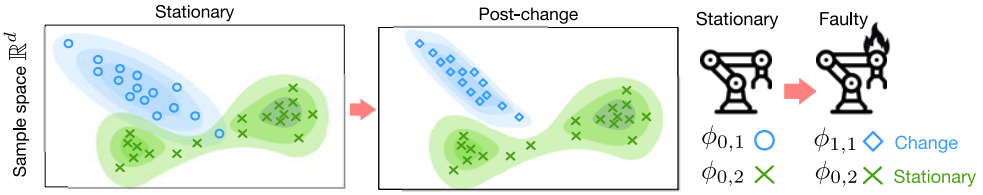

Fig. 1. In stationary conditions (left), the measurements, possibly corresponding to different operating scenarios, follow two modalities $\phi_{0,1}$ and $\phi_{0,2}$. After a fault (right), the distribution of the measurements changes ($\phi_{0,1} \rightarrow \phi_{1,1}$), potentially affecting a single modality ($\phi_{0,2}$ is unchanged).

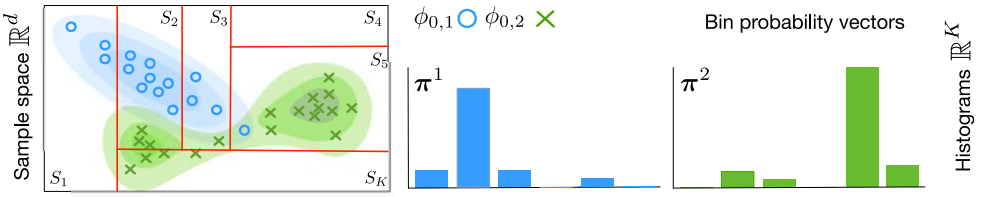

Fig. 2. Left: two stationary batches drawn from two modalities $\phi_{0,1}$ and $\phi_{0,2}$, and their contour plot. Note that here $\phi_{0,2}$ is non-Gaussian and multipeaked. A QuantTree partitioning is drawn in red lines. Right: corresponding bin-probability vectors $\boldsymbol{\pi}^1$ and $\boldsymbol{\pi}^2$. MMQT provides CD capabilities in a multimodal batch-wise setting, where any batch drawn from $\phi_{0,1}$ or $\phi_{0,2}$ is considered stationary.

and use this to transform training batches into bin-probability vectors π^i (Fig. 2, right). Then, we estimate the number of modalities characterizing ϕ_0 by clustering the bin-probability vectors. During monitoring, we detect distribution changes $\phi_0 \rightarrow \phi_1$ that affect any modality $\{\phi_{0,m}\}$ (as in Fig. 1) by computing a *modality-specific* test statistic that is compared against a predefined detection threshold. To this purpose, we define a batch-to-modality mapping that uses the centroids of the clusters of bin-probability vectors.

The main contributions of this work are the following.

1) We formulate the multimodal batch-wise CD problem, and solve this by MMQT.
2) We show that MMQT can control the false alarms by setting detection thresholds that do not depend on ϕ_0.
3) We present an automatic procedure for estimating the number of modalities in ϕ_0 by leveraging, for the first time, the control over false alarms as a criterion.
4) We propose a principled calibration procedure to adjust the estimated bin probabilities, which is particularly effective when the training set is scarce.

Our experiments demonstrate that MMQT can successfully estimate the number of modalities and effectively perform multimodal batch-wise CD while controlling the false positive rate (FPR). Interestingly, MMQT proves effective also on real-world datastreams, where ϕ_0 is an unknown collection of modalities. Moreover, we demonstrate that MMQT can successfully address key problems in concept drift detection, like detecting distribution changes in a completely unsupervised manner or the emergence of novel classes.

This article is structured as follows. In Section II, we formulate the novel batch-wise multimodal CD problem, and in Section III, we describe the related CD algorithms, which however cannot cope with our multimodal settings, and we introduce the QuantTree [10], which sets the basis for our work. In Section IV, we present MMQT, discuss its main theoretical result (Proposition 1), and analyze its computational complexity. In Section V, we assess the detection performance of MMQT in both CD and Stream Learning problems. Finally, Section VI discusses the limitations of MMQT, and Section VII wraps up our major contributions.

II. Problem Formulation

We address CD in a *batch-wise* and *multimodal* setup, where data $\mathbf{x} \in \mathbb{R}^d$ are, in stationary conditions, multivariate realizations of one of M random vectors drawn from modalities $\{\phi_{0,m},\ m = 1, \ldots, M\}$. We denote by ϕ_0 the multimodal stationary distribution such that a sample is said *stationary* when it comes from any modality $\phi_{0,m}$, as defined by the following relation:

$$\mathbf{x} \sim \phi_0 \iff \exists m \in \{1, \ldots, M\} \quad \text{s.t. } \mathbf{x} \sim \phi_{0,m}. \tag{1}$$

We further assume that data come in batches $W = \{\mathbf{x}_i, i = 1, \ldots, \nu\}$ containing ν i.i.d. samples each. In stationary conditions, each batch is drawn from a single modality, i.e.,

$$W \sim \left[\phi_{0,m}\right]^{\nu} \iff \mathbf{x} \sim \phi_{0,m} \quad \forall \mathbf{x} \in W \tag{2}$$

but different batches might come from different modalities. For the sake of simplicity, in the following, we drop the exponent ν, and simply write $W \sim \phi_{0,m}$ to denote that all the ν samples in the batch W are drawn from $\phi_{0,m}$. We also adopt the compact notation $W \sim \phi_0$ to indicate that $W \sim \phi_{0,m}$ for some m.

Our goal is to design an algorithm able to detect batches that are not drawn from any modality in ϕ_0. Thus, we address the following hypothesis testing problem:

$$H_0 : W \sim \phi_0 \quad \text{versus} \quad H_1 : W \sim \phi_1 \neq \phi_{0,m} \quad \forall m \tag{3}$$

which we solve while guaranteeing a controlled FPR $\alpha \in [0, 1]$, i.e., the proportion of type I errors [11]. By ϕ_1 in (3), we denote any post-change distribution including at least a modality other than the stationary ones $\{\phi_{0,m}\}$.

We remark that modalities $\{\phi_{0,m}\}$ do not necessarily coincide with peaks of ϕ_0, since each modality can itself be non-Gaussian and feature multiple peaks, as illustrated in Fig. 2 (see $\phi_{0,2}$). Moreover, for the sake of generality, we frame our detection problem in unsupervised settings, where the

stationary distribution ϕ_0, the modalities $\{\phi_{0,m}\}$, their number M, and the post-change distribution ϕ_1 are unknown. We only assume a training set containing N batches drawn from ϕ_0, namely $\text{TR} = \{W_i \sim \phi_0, i = 1, \ldots, N\}$, is provided.

III. Related Work and Background

CD has been abundantly addressed in many monitoring scenarios involving univariate/multivariate datastreams and streaming/batch-wise operations. Still, to the best of our knowledge, the CD literature is entirely focused on the single-modality setting, and a solution to the *batch-wise multimodal* CD problem described in Section II has never been proposed. In what follows, we summarize approaches to the multivariate CD, which is our primary interest.

Most CD algorithms for multivariate datastreams use a distance metric or a density measure to quantitatively assess the dissimilarity between the distributions of historical and new data. *Parameteric methods* assess the dissimilarity between batches by computing a two-sample statistic [12] that requires knowing the distribution ϕ_0, which is often approximated by a Gaussian mixture model (GMM) [7], [8], [9]. Changes are then detected by computing a test statistic leveraging on the fit model (e.g., the loglikelihood). Since the distribution of these test statistics is unknown, detection thresholds need to be defined by bootstrap to control false positives. Two prominent examples in this family are semi-parametric log-likelihood (SPLL) [7] and principal-component analysis (PCA)-SPLL [8]. SPLL fits a GMM to the training set and uses an upper bound of the loglikelihood as the test statistic. PCA-SPLL performs a PCA projection before applying SPLL to the low-variance components, which are supposed to be primarily affected by the change. Unfortunately, assuming that ϕ_0 belongs to a parametric family of distributions can be a severe limitation leading to a poor approximation when the hypothesis is violated. MMQT features several advantages with respect to a GMM: 1) it is not parametric; 2) it controls the FPR of each individual modality; and 3) performs a theoretically sound automatic estimation of the number of modalities. In our experiments, we consider SPLL and PCA-SPLL as representative parametric methods, with detection thresholds computed by bootstrap.

Nonparametric methods provide a more flexible alternative, as they do not restrict ϕ_0 to a parametric family of distributions. A few nonparametric alternatives have been designed for multivariate data, such as those relying on kernel test statistics [13], [14], kernel ratio test [15], or generalized log-likelihood statistic [16], like PCA-CD [17] and statistical density test (SDT) [18]. Other nonparametric approaches to multimodal CD are based on density estimation and assess the discrepancy between batch distributions [19], [20], [21], [22]. However, none of these methods tackle the multimodal CD problem described in Section II, which we solve by a histogram-based algorithm. Thus, we focus on the nonparametric methods based on histograms [6], [10], [23], [24], partitioning $\mathbb{R}^d$ and monitoring bin probabilities to detect changes. In particular, while MMQT also belongs to nonparametric algorithms, it is the first CD algorithm specifically designed for these settings.

A. Batch-Wise Monitoring by Means of Histograms

A histogram h is a collection of subset-probability pairs $h = \{(S_k, \pi_k)\}_{k=1,\ldots,K}$. The partitioning $\mathcal{S} = \{S_i\}$ is a collection of bins yielding a disjoint cover of the input domain $\mathbb{R}^d$, namely $\cup S_k = \mathbb{R}^d$, and $S_k \cap S_j = \emptyset$ when $k \neq j$. The vector $\boldsymbol{\pi} \in \mathbb{R}^K$ stacks all probabilities $\pi_k \in [0, 1]$ for samples drawn from ϕ_0 to fall inside S_k, hence $\sum_k \pi_k = 1$. In particular, we define a function Γ mapping a batch $W \subset \mathbb{R}^d$ to the corresponding *bin-probability vector* $\mathbf{p} \in \mathbb{R}^K$

$$\Gamma(W) = \mathbf{p}, \quad p_k = \frac{\#(W \cap S_k)}{\nu}, \quad k \in \{1, \ldots, K\} \tag{4}$$

being p_k the kth component of $\mathbf{p}$ and $\#(\cdot)$ the set cardinality. Histogram-based CD algorithms monitor incoming batches by computing a test statistic defined over h, i.e., a statistic that uniquely depends on the bin-probability vector $\mathbf{p}$ in (4). Examples of such statistics are the Pearson [11] or the Total Variation statistic. We use the latter, defined as

$$\mathcal{T}_{\mathcal{S},\pi}(W) = \frac{1}{2} \sum_{k=1}^{K} |p_k - \pi_k|. \tag{5}$$

Any batch deviating from ϕ_0 is detected as long as $\mathcal{T}_{\mathcal{S},\pi}(W) > \tau$, for some threshold $\tau \in \mathbb{R}$.

Histograms give rise to nonparametric CD methods, and here we discuss the most relevant examples. The kdq-tree [23] constructs a partition by iteratively halving the input space until it reaches a minimum bin density, and measures the discrepancy between historical and incoming data as the Kullback–Leibler distance. The detection thresholds for kdq-tree are computed by bootstrap over TR. Liu et al. [6] identify some limitations of histogram-based methods, such as the non-compactness of the bins and the presence of cluster gaps, and propose EIkM, a κ-means-based partitioning. Finally, we consider Density Tree [25, Ch. 5], which consists of a binary splitting partitioning where each leaf is a multivariate Gaussian, and each split is defined to maximize an information-gain metric. None of the above algorithms can natively cope with the batch-wise multimodal scenario presented in Section II, as they all compute test statistics that consider ϕ_0 as a single distribution without distinguishing between modalities. As we will show in our experiments, all these methods that set their detection thresholds by bootstrap can only control the average FPR and achieve modality-specific FPR much higher/lower than the target. Instead, MMQT solves the issue by using modality-specific detection thresholds. In particular, our method is a multimodal extension of QuantTree (described in what follows), improving on the theoretical guarantees of the original method and natively designed for the batch-wise multimodal scenario.

B. QuantTree

QuantTree [10] is an algorithm that builds histograms for CD. QuantTree takes as input TR and a set of target bin probabilities $\{\pi_k\}$, and iteratively defines a partitioning $\mathcal{S}$ whose bin probabilities under ϕ_0 are given by $\{\pi_k\}$. At each iteration, QuantTree computes a new bin by splitting the remaining portion of the input domain along a single covariate,

and choosing a sample quantile of the marginal distribution as the splitting point. This splitting scheme ensures that: 1) the bin probabilities are random variables whose distribution depends only on $\{\pi_k\}$, the batch-size ν and the size of TR, and 2) the distribution of any statistic defined over such a histogram does not depend on ϕ_0. These properties enable very practical nonparametric monitoring, as the detection thresholds to control the FPR can be set via Monte Carlo simulations on synthetically generated data and do not require bootstrapping over TR. Specifically, [26] proves that the reference bin-probability vector of a QuantTree is drawn from a Dirichlet distribution with parameters depending only on N and $\{\pi_k\}$. Moreover, the bin-probability vector $\Gamma(W)$ associated with a batch $W \sim \phi_0$ is distributed as a multinomial that only depends on ν. Therefore, we simulate the construction of a QuantTree and compute the statistic associated with a batch W by sampling random vectors from the Dirichlet and the multinomial distributions, respectively. In particular, any QuantTree constructed with the same parameters ν, N, and $\{\pi_k\}$ uses the same detection threshold, independently of ϕ_0. Despite the nice properties of QuantTree and the recent extensions via kernel functions [27], this algorithm is not able to monitor multimodal batches, as shown in our experiments.

C. Non-Stationary Datastreams

Stream Learning models are expected to detect and react to *concept drifts* [6], namely changes in the statistical properties of the monitored datastream. In Stream Learning, a classifier is trained over labeled data drawn from an initial distribution ϕ_0, potentially collected in batches, and needs to accommodate any distribution change $\phi_0 \rightarrow \phi_1$ that would disrupt its performance. In this scenario, *active approaches* [28], [29], [30] employ a CD test to detect changes and trigger an adaptation strategy to counteract the performance loss caused by a concept drift or by the emergence of novel classes [31], [32], [33]. The most popular approach to concept drift detection consists in monitoring the error rate of a classifier trained on annotated stationary samples and reporting a detection when the error rate increases [34], [35]. Such a monitoring scheme requires constant supervision to detect changes, which might not be viable in many conditions. Other methods rely on pairs of streams, which might receive different types of supervision (if any) to detect distribution changes while constantly adapting the classifiers to the evolving stream [36], [37]. A third approach to concept drift detection is to steadily monitor the distribution of unlabeled input data. This can be done in many ways, either by approximating ϕ_0 by some parametric model [7] or by employing nonparametric statistics [38], [39]. It has to be mentioned another approach, often encountered in fault detection literature, which consists in leveraging annotated examples [40] or prior knowledge to detect specific types of changes [41]. Unfortunately, these methods are very application specific and are based on assumptions that are rarely encountered in general Stream Learning settings.

Therefore, even in the non-stationary environment (NSE) literature, we are the first to address the multimodal batch-wise CD problem described in Section II. Our setting has a minor overlap with the literature on recurrent concepts, where classifiers adapt to concept drifts by learning new models, and use previously learned ones in the case of recurrent distributions [28], [42]. However, algorithms assuming the existence of recurrent concepts switch between states only when a drift is detected and struggle when ϕ_0 is the combination of a set of modalities $\{\phi_{0,m}\}$.

IV. MultiModal QuantTree

MMQT performs multimodal batch-wise CD by defining a *modality-agnostic* partitioning $\mathcal{S}$ of the input space and computing a *modality-specific* test statistic over incoming batches. The partitioning $\mathcal{S}$ is used to describe all the modalities in ϕ_0 and to associate each batch W to the modality that most likely generated it. The test statistic is efficiently computed by comparing the bin-probability vector of the tested batch against the reference vector of selected modality. Remarkably, MMQT inherits the nonparametric nature of QuantTree and can cope with modalities characterized by any multivariate distribution $\phi_{0,m}$, even those presenting multiple peaks (like $\phi_{0,2}$ in Fig. 2). Indeed, the nonparametric trait of MMQT is guaranteed by the batch-to-modality mapping (7), which allows us to compute test statistics as in QuantTree.

In the following, we delineate the training phase of MMQT (Section IV-A) and illustrate the batch-wise monitoring by modality-specific test statistics (Section IV-B). Then, we illustrate some peculiarities of MMQT that are backed by the theoretical properties of QuantTree: the calibration procedure (Section IV-C) and the estimation of the number of modalities (Section IV-D). Finally, Section IV-F reports the computational complexity of MMQT, and Section IV-G describes an alternative scheme consisting of using M-QuantTrees.

A. MMQT Construction

Histogram-based methods construct a partitioning $\mathcal{S}$ of the input space, and model ϕ_0 through a reference bin-probability vector π that describes the percentage of points drawn from ϕ_0 that fall in each bin. In principle, this scheme would apply to each modality $\phi_{0,m}$, giving rise to many histograms. However, MMQT constructs a unique partitioning $\mathcal{S}$ to define $\widehat{M}$ reference bin-probability vectors, each characterizing an individual modality as described in Algorithm 1. For simplicity, we describe all the operations assuming the number of modalities M to be known, i.e., $\widehat{M} = M$, while the automatic estimation of M is detailed in Section IV-D.

We first partition $\mathbb{R}^d$ via the QuantTree algorithm (line 1) to obtain $\mathcal{S}$, a partitioning of K disjoint bins with probability $\pi_k = 1/K$ under ϕ_0. To this purpose, all samples in TR are used, disregarding the batch they come from. Then, we process TR in a batch-wise manner, and map each W to its bin-probability vector $\mathbf{p} \in \mathbb{R}^K$ (line 2) by computing bin probabilities $\Gamma(W)$ as in (4). We obtain a set $\Gamma(\text{TR}) \subset \mathbb{R}^K$ collecting the bin-probability vectors associated with the training batches

$$\Gamma(\text{TR}) = \{\mathbf{p}_i = \Gamma(W_i), i = 1, \ldots, N\}. \tag{6}$$

Since we assume that M is provided, we do not initialize $\widehat{M}$ (line 3) and do not iterate the while loop at line 4. For each

Algorithm 1 Training MMQT and Estimating M

Require: Training set TR $= \{W_i, i = 1, \ldots, N\}$, target FPR α
Ensure: Number of clusters $\widehat{M}$, partitioning $\mathcal{S}$
1: Define $\mathcal{S}$ as a QuantTree, using TR and K bins with bin probabilities $1/K$.
2: Map each batch W in TR via Γ, yielding $\Gamma(\text{TR}) = \{\mathbf{p}_i, i = 1, \ldots, N\}$.
3: Set $\widehat{M} = 0$, $FPR = 1$
4: **while** $FPR > \alpha$ **do**
5: $\quad \widehat{M} = \widehat{M} + 1$
6: $\quad$ Run κ-means clustering on $\Gamma(\text{TR})$, using $\widehat{M}$
7: $\quad$ Calibrate bin probabilities $\{\widehat{\boldsymbol{\pi}}^m\}$ as in (12)
8: $\quad$ Compute thresholds $\{\tau_m\}$ for target α (Section IV-A)
9: $\quad$ **for all** $W \in$ TR **do**
10: $\quad\quad$ Change detection on W
11: $\quad$ **end for**
12: $\quad$ Compute the achieved FPR
13: **end while**
14: Return $\widehat{M}$, $\mathcal{S}$

modality, we compute a reference bin-probability vector and the detection thresholds as follows. First, we run (line 6) the κ-means algorithm [43] to divide the bin-probability vectors in $\Gamma(\text{TR}) \subset \mathbb{R}^K$ into $\widehat{M}$ clusters. We take the centroids of the resulting clusters as preliminary estimates of the reference bin-probability vectors $\{\boldsymbol{\pi}^m\}$ associated with the modalities. Then (line 7), we adjust each $\boldsymbol{\pi}^m$ through the calibration procedure (12) described in Section IV-C to account for the multimodal nature of the problem. Finally (line 8), for each modality $\phi_{0,m}$ we compute a detection threshold τ_m via Monte Carlo simulations, as discussed in Section III-B.

B. Batch-Wise Monitoring Using MMQT

During monitoring, MMQT computes the bin probabilities $\mathbf{p} = \Gamma(W) \in \mathbb{R}^K$ for each incoming batch W according to the partitioning $\mathcal{S}$ as in (4), and performs the *batch-to-modality association* by determining the modality ϕ_{0,m^*} which most likely has generated W as

$$m^* = \underset{m=1,\ldots,\widehat{M}}{\arg\min} \left\| \mathbf{p} - \boldsymbol{\pi}^m{}_2 \right\| \tag{7}$$

where $\mathbf{p} = \Gamma(W)$ is the bin-probability vector of the test batch and $\{\boldsymbol{\pi}^m\}$ are the calibrated reference bin-probability vectors constructed during training. Then, we compute the Total Variation (5) as a modality-specific statistic

$$\mathcal{T}_{\mathcal{S},\boldsymbol{\pi}^{m^*}}(W) = \frac{1}{2}\sum_{k=1}^{K} \left| p_k - \pi_k^{m^*} \right| \tag{8}$$

to assess the statistical distance between $\mathbf{p}$ and $\boldsymbol{\pi}^{m^*}$. We detect a change when

$$\mathcal{T}_{\mathcal{S},\boldsymbol{\pi}^{m^*}}(W) > \tau_{m^*}. \tag{9}$$

We observe that (7) might associate a batch W to a wrong bin-probability vector $\boldsymbol{\pi}^{m^*}$. This phenomenon is not compensated by MMQT and might impact the false positive control.

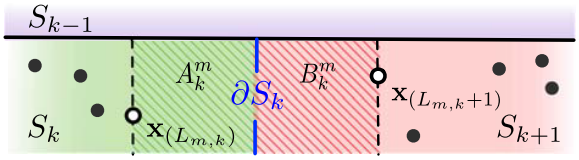

Fig. 3. Calibration of bin probabilities. We report the partitioning $\mathcal{S}$ constructed by QuantTree over the entire TR and samples from the mth modality only. The boundary ∂S_k does not meet any point from $\phi_{0,m}$, as opposed to what would happen with a QuantTree constructed over batches from $\phi_{0,m}$ (dashed lines). Calibration compensates for this difference by estimating γ_k in (14) as the ratio between the expected values of $P_{\phi_{0,m}}[A_k]$ and $P_{\phi_{0,m}}[A_k \cup B_k]$.

However, our experiments show that the FPR is well controlled both in case of synthetic and real-world datasets.

C. Calibration of Bin Probabilities

The properties of QuantTree (Section III-B) guarantee that each threshold τ_m is independent of the data dimension d and the distribution $\phi_{0,m}$, and depends only on the reference bin-probability vector $\boldsymbol{\pi}^m$, the batch size ν and the number N_m of training batches associated with that $\phi_{0,m}$. However, when we train MMQT, we do not construct the partitioning $\mathcal{S}$ specifically for $\phi_{0,m}$ but we use the entire TR, which is drawn from all the modalities in ϕ_0. Therefore, we have devised a calibration procedure to mitigate the resulting discrepancy between the bin probabilities estimated by the κ-means and the actual bin probabilities. For simplicity, in what follows, we assume to know which modality generated each batch, while in practice we estimate it by the batch-to-modality mapping (7).

During training, for each bin S_k and modality $\phi_{0,m}$, the κ-means estimates the probability $P_{\phi_{0,m}}[S_k]$ of a sample drawn from $\phi_{0,m}$ to fall in S_k. These probabilities correspond to each component of $\boldsymbol{\pi}^m$, computed as $\pi_k^m = L_{m,k}/n_m$, where $n_m = \nu N_m$ is the number of training points associated with the mth modality by (7), and $L_{m,k}$ is the number of these points falling in S_k. When we use a single partitioning $\mathcal{S}$ to monitor all the modalities $\{\phi_{0,m}\}$, each bin probability π_k^m does not accurately represent the probability for a sample drawn from $\phi_{0,m}$ to fall in S_k, since the histogram bins have not been constructed on points from the same modality. To better illustrate the issue, Fig. 3 shows an example of QuantTree partitioning, where we report training points drawn from $\phi_{0,m}$ only. To construct the kth bin S_k, all the training samples $\{\mathbf{x}_j\}$ are sorted according to the i_kth covariate, denoted as $[\cdot]_{i_k}$. Then, the algorithm defines the split by selecting a value $z_{\partial S_k}$ among the sorted covariates $z_{(1)} \leq z_{(2)} \leq, \ldots$, where each $z_{(j)}$ correspond to a sample $\mathbf{x}_{(j)} \in$ TR such that $[\mathbf{x}_{(j)}]_{i_k} = z_{(j)}$. In this example, the boundary ∂S_k between S_k and S_{k+1} does not pass through any training point drawn from $\phi_{0,m}$, which means that the split was defined by selecting a training point $\mathbf{x}$ drawn from another modality.

In the illustration in Fig. 3, we can identify the two samples drawn from $\phi_{0,m}$ that are closest to ∂S_k, namely, $\mathbf{x}_{(L_{m,k})}$ and $\mathbf{x}_{(L_{m,k}+1)}$, which are inside and outside S_k, respectively. Let A_k^m and B_k^m be the stripes between $\mathbf{x}_{(L_{m,k})}$ and ∂S_k, and between $\mathbf{x}_{(L_{m,k}+1)}$ and ∂S_k, formally defined as

$$A_k^m = \left\{ x \in S_k | [x]_{i_k} > z_{(L_{m,k})} \right\} \tag{10}$$

$$B_k^m = \left\{ x \notin \bigcup_{j=1}^{k-1} S_j | [x]_{i_k} \leq z_{(L_{m,k}+1)} \right\}. \quad (11)$$

Note that when $\mathcal{S}$ is built using QuantTree over training points associated with $\phi_{0,m}$ only, the bin boundary ∂S_k would pass through $\mathbf{x}_{(L_{m,k})}$, and A_k^m would be empty.

The proposed *calibration procedure* consists in updating the marginal bin probabilities $\boldsymbol{\pi}^m$ by taking into account the stripes A_k^m and B_k^m as follows:

$$\pi_k^m \to \mathcal{C}(\pi_k^m) = \frac{L_{m,k} + g_k^m}{n_m + 1} \prod_{j<k} \left(1 - \frac{g_j^m}{q_{j+1} n_m + 1} \right) \quad (12)$$

where $q_j = 1 - \sum_{i=1}^{j-1} L_{m,i}/(N_m \nu)$ and the values g_k^m are defined as

$$g_k^m = \frac{z_{\partial S_k} - z_{(L_{m,k})}}{z_{(L_{m,k}+1)} - z_{(L_{m,k})}}. \quad (13)$$

Intuitively, g_k^m represents the expected value of the ratio between the probabilities of A_k^m and $A_k^m \cup B_k^m$, which is the ratio between the amount of points expected to fall in A_k^m and $A_k^m \cup B_k^m$, respectively. In (12), the factor in parenthesis takes into account the probabilities of bins S_j for $j < k$ after the calibration. For the sake of simplicity, we omit $\mathcal{C}(\cdot)$ from the notation of the calibrated bin probabilities unless necessary.

The calibration procedure in (12) derives from the following proposition, which expresses the expected probability of a point to fall in a bin of MMQT. Precisely, we show that (12) is the same as (14) from Proposition 1, where γ_k^m is substituted by an estimate g_k^m.

Proposition 1: Let $L_{m,k}$ be the number of points drawn from modality $\phi_{0,m}$ falling into the bin S_k, let n_m be the overall number of samples drawn from $\phi_{0,m}$ and let $n_{m,k} = n_m - \sum_{j=1}^{k-1} L_{m,j}$ be the number of training samples drawn from $\phi_{0,m}$ not falling into bins $1, 2, \ldots, k-1$. Then, there exists $\gamma_k^m \in [0, 1]$ such that the expectation of the probability of the kth bin under $\phi_{0,m}$ can be expressed as

$$E\left[P_{\phi_{0,m}}[S_k]\right] = \frac{L_{m,k} + \gamma_k^m}{n_m + 1} \prod_{j<k} \left(1 - \frac{\gamma_j^m}{q_{j+1} n_m + 1} \right) \quad (14)$$

for $k = 1, \ldots, K-1$, where $q_j = 1 - \sum_{i=1}^{j-1} L_{m,i}/(N_m \nu)$ and the expectation is considered with respect to the training set realizations.

The proof of Proposition 1 is in the supplementary material, together with the required theoretical results. We remark that the calibration of the reference bin-probability vectors is only possible because from [10] we know the distribution of the bin probabilities in QuantTree. Other tree structures, such as kd-trees [44], do not provide the same theoretical properties.

To conclude, we observe that the impact of the calibration procedure becomes more evident when TR is small. Intuitively, as TR gets smaller, we expect the stripe A_k to become larger, thus the denominators of (12) to become smaller. This effect is shown in Section V-E, where we assess the influence of the number of training batches N_m and the distance between stationary distributions over the ability of MMQT to control the FPR with and without calibration.

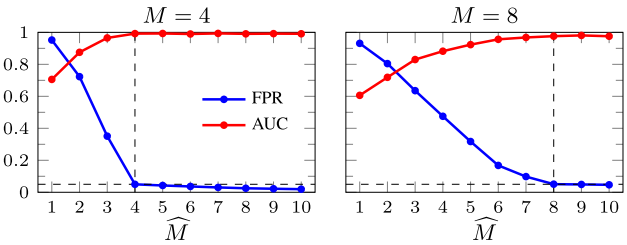

Fig. 4. Plots showing the influence of $\widehat{M}$ on FPR and AUC on synthetic dataset in $\mathbb{R}^2$. The average FPR and AUC achieved by MMQT when $M = 4, 8$ and MMQT estimates $\widehat{M} \in \{1, 2, \ldots, 10\}$ modalities. When $\widehat{M} = M$, MMQT achieves the target FPR of 5% and good AUC. The empirical FPR is extremely large when $\widehat{M} < M$ and steadily drops when $\widehat{M} > M$.

D. Estimation of the Number of Modalities

In the multimodal problem solved by MMQT, the number of modalities represented in TR is unknown. Classical model-selection procedures used in clustering validation [45], [46], [47] select the optimal number of clusters by optimizing a quality measure that usually assesses the compactness of the data in the clusters. For the first time, we use the FPR control guaranteed by MMQT to estimate the number of clusters, i.e., of modalities, characterizing ϕ_0. Our procedure to estimate M builds on the intuition, confirmed by the experiments, that when we underestimate M, the empirical FPR exceeds the target α'. In this case, in fact, batches drawn from different modalities would be associated with the same bin-probability vector, resulting in many false alarms. To prove our intuition, we run experiments on synthetic data in $\mathbb{R}^2$ where ϕ_0 is characterized by different numbers of modalities $M \in \{1, 2, \ldots, 8\}$. Here, we simulate under- and overestimations of $\widehat{M}$ by configuring MMQT with $\widehat{M} \in \{1, 2, \ldots, 10\}$. Fig. 4 illustrates the results for $M = 4$ and 8, reporting the empirical FPR achieved by MMQT plotted against the number $\widehat{M}$ of modalities used to construct $\mathcal{S}$. These plots show that when $\widehat{M} < M$, the empirical FPR is larger than the target $\alpha' = 5\%$ (dashed horizontal line), and the FPR steadily drops below the target as soon as $\widehat{M} = M$ (dashed vertical line). Moreover, the FPR decreases at a slower rate once $\widehat{M} > M$, meaning that even overestimating the number of modalities prevents us from effectively controlling the FPR. In the supplementary material, we report the results for $M \in \{1, 2, \ldots, 8\}$.

Our procedure for estimating $\widehat{M}$ consists in constructing a histogram and then computing the empirical FPR over TR for increasing values of $\widehat{M} = 1, \ldots,$ until we obtain an MMQT that yields an FPR lower than the target α'. The entire training procedure of MMQT is unsupervised and described in Algorithm 1. The number of modalities $\widehat{M}$ is estimated at lines 3–13, where multiple values of $\widehat{M}$ are tested. First (line 3), we initialize $\widehat{M} = 0$. Then, in the while loop of line 4, we test increasing values of $\widehat{M}$ until the empirical FPR achieved over TR drops below α'. At each iteration, we increase $\widehat{M}$ (line 5), compute the bin probabilities via κ-means (line 6), calibrate bin probabilities (line 7), and compute thresholds $\{\tau_m\}$ for each identified modality (line 8). Ultimately, we select the first value $\widehat{M}$ yielding an FPR lower than the target α'. In the experiments, we repeat the estimation process multiple times and set $\widehat{M}$ by majority voting. We remark that such an estimation technique would not have

been possible without the theoretical properties of QuantTree, which allows setting the detection thresholds controlling the FPR.

We want to emphasize that modalities $\{\phi_{0,m}\}$ in MMQT do not correspond to peaks of a distribution (say, the components of a GMM) but are rather any distribution of batches, not necessarily unimodal. For this reason, the problem of estimating the number of modalities in $\mathbb{R}^d$ is more challenging, as it requires distinguishing between a single modality having multiple peaks and several unimodal modalities. Note that MMQT conveniently estimates $\widehat{M}$ in the space of bin probabilities $\mathbb{R}^K$, where it can easily handle multipeaked modalities in $\mathbb{R}^d$, like $\phi_{0,2}$ in Fig. 2, which are mapped to a unimodal bin-probability distribution.

E. Parameters of MMQT

In MMQT, α is the target FPR—i.e., the expected proportion of type I errors, as in any Hypothesis Test—and is used to define detection thresholds via Monte Carlo simulations. In principle, the value of α' used to estimate the number of modalities (Section IV-D) can differ from α and, in Section V-F, we explore the impact of disentangling them. In the following, unless differently stated, we set $\alpha' = 5\%$.

Other crucial parameters for MMQT are the number ν of samples in a batch W and the number of bins K. On the one hand, the estimation of the bin-probability vector from training or testing batches improves when bins contain many samples from W, namely, when the ratio ν/K is large. Moreover, thanks to the consistency property of Hypothesis Tests, we know that the power of MMQT would increase when ν grows, as we discuss in the experiments and show in the supplementary material. On the other hand, since each bin of MMQT considers only a (randomly selected) dimension, using a large K increases the probability that every dimension is considered in the histogram construction but lowers the average number of points per bin. This drawback is especially significant for high-dimensional data, where some input components might be ignored by all the splits. We choose ν and K in each experiment to balance the number of samples per bin and the risk of ignoring some input components in the splits.

F. Computational Complexity

We analyze the computational complexity of the MMQT training following Algorithm 1. The complexity of building the QuantTree partitioning (line 1) is $O(KN\nu\log(N\nu))$, as reported in [10], while computing the probability vector of every batch in TR (line 2) costs $O(KN\nu)$. Then, inside each of the $\widehat{M}$ iterations of the while loop at line 4, we have to perform κ-means clustering, which requires $O(tK^2N)$, being t the maximum number of κ-means iterations. The cost of the bin-probability calibration is negligible, being $O(K)$. Thresholds (line 8) for each of the QuantTrees constructed during training are computed via Monte Carlo simulations as in [26], and require $O(BK + B\nu\log K)$ operations, being B the number of Monte Carlo iterations. Therefore, the overall complexity of training an MMQT becomes $O(KN\nu\log(N\nu)+\widehat{M}tK^2N+\widehat{M}^2(BK+B\nu\log K))$. During monitoring, the computational complexity is dominated by the computation of the bin-probability vector and the selection of the corresponding centroid, thus it turns out to be $O(\nu K + \widehat{M}K)$.

G. CD by M-QuantTrees

MMQT monitors incoming batches W using a single partitioning $\mathcal{S}$, which is used to compute the bin-probability vector $\mathbf{p} = \Gamma(W)$. Then, $\mathbf{p}$ is first used to identify the modality generating W (7) and then to compute the modality-specific statistic (8), which is compared against the corresponding threshold to detect changes. Such a detection scheme requires the calibration procedure of Section IV-C.

A viable alternative to MMQT is to employ M QuantTrees, one per modality, thus avoiding calibrating the bin probabilities. This corresponds to constructing modality-specific histograms to monitor batches of each modality. In the following, we refer to this solution as the M-QuantTrees. During training, the M-QuantTrees algorithm estimates $\widehat{M}$ in the same way as MMQT and uses the same batch-to-modality association in (7) to split the training set into modality-specific disjoint subsets $\{\text{TR}_m\}$. Then, each subset TR_m is used to construct a modality-specific QuantTree partitioning $\mathcal{S}^m$ and to compute a detection threshold τ_m as in [26]. During testing, each incoming batch W is associated with a modality m^* via (7) so that the detection is performed by the corresponding QuantTree $\mathcal{S}^{m^*}$.

Despite avoiding the calibration procedure, the M-QuantTrees alternative is less efficient in terms of computational complexity since, during training, it requires building $\widehat{M}$ QuantTrees more than MMQT, thus increasing the cost by $O(\widehat{M}(KN\nu\log(N\nu)))$. Moreover, during testing, each batch W needs to pass through two QuantTrees instead of just an MMQT, doubling the inference cost. To conclude, we remark that M-QuantTrees is a viable option developed within the MMQT framework, as this latter provides a procedure for estimating $\widehat{M}$ and the batch-to-modality mapping. In our experiments, we will compare MMQT and M-QuantTrees in terms of control of the FPR and detection performance.

V. Experiments

In this section, we assess the performance of MMQT on synthetic and real-world datasets (Section V-A) by the figures of merit described in Section V-B. We show that MMQT outperforms competing methods (Section V-C) and can accurately estimate the number of modalities (Section V-D), and effectively detect changes while controlling the FPR (Section V-E and V-F). We also show the benefits of MMQT in Stream Learning, where it can detect concept drifts and the emergence of novel classes in non-stationary datastreams (Section V-G). MMQT code is publicly available for download.[1]

A. Datasets

We test MMQT using both synthetic and real-world datasets.

[1]GitHub repository: github.com/diegocarrera89/quantTree

1) Synthetic Dataset: We generate datasets where the stationary $\{\phi_{0,m}\}$ and post-change $\{\phi_{1,m}\}$ modalities are Gaussians with a minimum Kullback–Leibler distance $\text{sKL}_0 \leq \text{sKL}(\phi_{i,m}, \phi_{j,n})$, $\forall i, j, m, n$. In the supplementary material, we detail the procedure to generate these modalities, which is inspired by the CCM framework [48].

2) CWRU Bearing Dataset: The Bearing dataset is provided by the Case Western Reserve University (CWRU) Bearing Data Center [49] and consists of vibration data collected by an accelerometer placed onto an electric ball-bearing motor. This annotated dataset is an example of an industrial monitoring CD application and comprises acquisitions performed at three different speeds $\{\text{Sp}_1, \text{Sp}_2, \text{Sp}_3\}$ both in normal (used for training) and faulty conditions (used only to assess the detection performance). Accelerometer measurements are preprocessed by a short-time Fourier transform over windows of size 32 extracted with eight samples overlap. We monitor the absolute values of the first $d = 8$ coefficients and split the resulting datastream in batches of $\nu = 32$ samples in $\mathbb{R}^8$.

3) INSECTS Dataset: The INSECTS dataset [50] is a recent classification benchmark for datastreams affected by concept drift. It contains feature vectors ($d = 33$) describing the wing-beat frequency of different species of flying insects. We consider only the abrupt-change dataset, which contains a datastream of six concepts with annotated change points. Each concept contains measurements acquired from six (annotated) insect species at different unknown temperatures, which are known to have different feature distributions.

B. Figures of Merit

We consider two classical figures of merit in batch-wise CD: the empirical FPR and the area under the receiving operating characteristic curve (AUC). We also assess the average error $\Delta M = \widehat{M} - M$ when estimating the number of modalities. In Section V-G, we test Stream Learning performance and we consider, on top of detection metrics, the classification accuracy and the expected detection delay (EDD). Since there is no specific temporal order among testing batches in our experiments, the EDD is computed as the expected value of a geometric distribution having as a parameter the empirical true positive rate (TPR) for a fixed detection threshold. All the results are averaged over multiple runs and reported together with the 95%-confidence interval, proving the statistical significance of our findings.

C. Methods

Most of the algorithms mentioned in Section III are limited to dealing with a single distribution ϕ_0 and, therefore, cannot cope with our batch-wise multimodal setup described in Section II. Since there are no alternatives that directly operate in our setting, we compare against methods that can be adapted to handle multivariate data from possibly multipeaked distributions. In what follows, we illustrate the considered methods and their configurations as summarized in Table I.

1) MMQT: In all our experiments, the partitioning of the input domain $\mathcal{S}$ is defined by a QuantTree to yield a uniform density coverage of the training set $\text{TR} \subset \mathbb{R}^d$ ($\pi_k = 1/K$).

TABLE I

PARAMETERS SETTING FOR THE CONSIDERED METHODS. THE NUMBER OF BINS K AND OF GAUSSIAN COMPONENTS C DEPEND ON THE DATASET. IN THE SUPPLEMENTARY MATERIAL, WE TEST OTHER VALUES OF ν AND α'

Dataset	MMQT (w/ cal) MMQT (w/o cal)	QuantTree M-QuantTrees	Density Tree	SPLL PCA-SPLL
All	$\nu = 128$			
All	$\alpha' = 5\%$	-	-	-
All	π_k = automatic	$\pi_k = 1/K\ \forall k$	-	-
Synthetic (low d)	$K = 4d$			$C = M$
Synthetic (high d)	$K = 64$			$C = M$
INSECTS	$K = 64$			$C = M$
CWRU	$K = 16$			$C = M$

In low-dimensional synthetic data ($d \in \{2, 4, 8, 16\}$), we set $K = 4d$, while in high-dimensional synthetic data ($d \in \{32, 64, 128\}$) we fix $K = 64$ since we keep a fixed batch size and we need to preserve a minimum amount of training samples per bin. For the same reason, in the CWRU ($d = 8$) and INSECTS ($d = 33$) datasets, we set $K = 16$ and $K = 64$, respectively, due to the small size of the datasets. To estimate the number of modalities $\widehat{M}$, we run the κ-means algorithm using the κ-means++ initialization [51], repeating the centroid initialization ten times. To increase the robustness, we repeat the entire estimation of the number of modalities (Algorithm 1) ten times, choosing the most often selected $\widehat{M}$ and the partitioning $\mathcal{S}$ achieving the closest FPR to α' among those yielding $\widehat{M}$. Finally, we calibrate the bin probabilities as in (12).

2) MMQT (w/o Calibration): This is exactly like MMQT, with the only difference being that we do not apply the bin-probability calibration. This baseline allows us to assess the impact of the calibration on the detection performance.

3) M-QuantTrees: As an alternative to MMQT, we test CD by using M-QuantTrees as in Section IV-G. We employ $\widehat{M}$ QuantTrees, with parameters set as in MMQT.

4) QuantTree: We test a single QuantTree [10] configured with the same parameters and statistic as MMQT. This batch-wise baseline is not multimodal, thus ignoring the fact that batches might be drawn from different modalities.

5) Semi-Parametric Log-Likelihood: SPLL [7] is perhaps the closest solution in the literature to our CD problem, as it adopts a multipeaked density model. We use SPLL in the general setting where a GMM $\widehat{\phi}_0$ is fit to TR with possibly different covariance matrices, as in [7], and we consider the ideal conditions where the number of Gaussians C is given. For the synthetic data, we fit a mixture of exactly $C = M$ Gaussian densities to TR. For the CWRU dataset, we set $C = 1$ (Single Speed scenario) or $C = 3$ (Multi Speed scenario), while for the INSECTS dataset, we set C equal to the number of insect species. During testing, SPLL associates each sample to the closest Gaussian, and computes an upper bound of the loglikelihood. The SPLL statistic is then averaged over the batch to obtain the test statistic. We estimate every time the detection threshold via bootstrapping over TR.

6) PCA-SPLL: This variant of SPLL was proposed in [8] consists of computing a PCA transformation over TR to project data over the components with the lowest variance. Then, a GMM is fit to the projected training data, and SPLL is adopted for detection. In our experiments, we keep the principal components corresponding to the lowest 20% of the variance, and set up SPLL as in the previous paragraph.

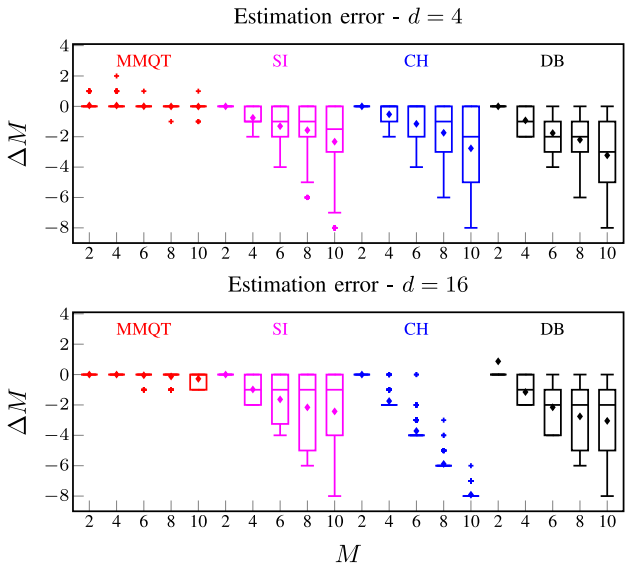

Fig. 5. Automatic estimation of the number of modalities on synthetic datasets in $\mathbb{R}^4$ (top) and $\mathbb{R}^{16}$ (bottom) with $M \in \{2, 4, 6, 8, 10\}$ modalities. Boxplots depict the distributions of $\Delta M = \widehat{M} - M$ where $\widehat{M}$ has been estimated via MMQT, SI, CH, and DB indices. MMQT yields the most accurate estimates.

7) Density Tree: This non-multimodal baseline consists of a binary splitting tree where each split is defined by maximizing an information-gain metric. Such a method is inspired by [25] and was also used as a baseline in [10]. Our implementation uses the Total Variation statistic (5), and we define a detection threshold by bootstrapping over TR.

As a closing remark, we observe that the estimation of $\widehat{M}$ and the histogram construction performed by QuantTree and MMQT are stochastic processes, characterized by a significant degree of randomness. To ensure a fair comparison, in each run, we construct the partitioning $\mathcal{S}$, we estimate $\widehat{M}$, and we use these for MMQT, MMQT w/o calibration, and M-QuantTrees. QuantTree uses the same partitioning $\mathcal{S}$.

D. Estimating the Number of Modalities

We benchmark our solution for estimating the number of modalities M against classical clustering-validation methods, namely, the Silhouette Index (SI) [45], Calinski–Harabasz (CH) Index [46], and Davies–Bouldin (DB) Index [47]. While the definitions of these indexes are different, they all aim to select the number of clusters that strike the best balance between cluster cohesion and separation. Instead, we rely on the theoretical properties of MMQT to derive the number of modalities (Section IV-D) by controlling false alarms.

We test the accuracy of estimating $\widehat{M}$ by MMQT over the synthetic datasets generated as in Section V-A, varying both the number of ground-truth modalities $M \in \{2, 4, 6, 8, 10\}$ and the data dimension $d \in \{2, 4, 8, 16\}$. Fig. 5 shows the distribution of the differences $\Delta M = \widehat{M} - M$ over 200 trials as a function of M. Due to space limitations, we report only $d \in \{4, 16\}$. Other configurations yield consistent results. All the baselines struggle in high dimensions, as they underestimate $\widehat{M}$ and suffer high variability when M and d increase. This result is not surprising, given the known limitations of these indexes in high-dimensional settings [52]. In contrast, our solution retrieves exactly $\widehat{M} = M$ in most cases, with only few overestimates having little impact on the detection performance, as we commented in Section IV-D.

TABLE II

ESTIMATION ERROR $\Delta M = \widehat{M} - M$, AVERAGED OVER 200 RUNS, ACHIEVED BY MMQT IN HIGH-DIMENSIONAL DATA. IN PARENTHESIS, THE WIDTH OF THE 95%-CONFIDENCE INTERVAL

d	M				
	2	4	6	8	10
32	0.000 (0.000)	0.000 (0.000)	-0.070 (0.035)	-0.260 (0.061)	-0.735 (0.148)
64	0.000 (0.000)	-0.055 (0.032)	-0.690 (0.073)	-0.980 (0.071)	-1.649 (0.171)
128	0.000 (0.020)	-0.615 (0.073)	-1.590 (0.092)	-2.640 (0.103)	-3.600 (0.114)

To further investigate the performance of Algorithm 1, we consider synthetic data in $\mathbb{R}^d$ for $d \in \{32, 64, 128\}$. Table II reports the estimation error ΔM averaged over 200 changes $\phi_0 \to \phi_1$ and shows that MMQT underestimates the number of modalities in this high-dimensional setting. As commented in Section IV-E, we argue that this is the consequence of the setup of MMQT with only $K = 64$ bins. In the supplementary material, we report additional results to show that different values of α' yield the same results when $M = 10$. We also assess the accuracy of $\widehat{M}$ on real-world datasets, assuming that the number of modalities corresponds to the number of insect species or motor speeds represented in the data. In the results to the experiments on real-world data, we report ΔM alongside the detection performance of the methods and shows that MMQT accurately estimates three modalities for the CWRU Multi Speed dataset and the exact number of species in the INSECTS dataset.

E. CD in Synthetic Datastreams

In this experiment, we generate the $2M$ Gaussian distributions $\{\phi_{0,m}\}_{m=1}^{M}$ and $\{\phi_{1,m}\}_{m=1}^{M}$ characterizing the stationary ϕ_0 and the post-change condition ϕ_1 for $M \in \{2, 4, 6, 8, 10\}$. We generate distributions with different dimensions $d \in \{2, 4, 8, 16, 32, 64, 128\}$ and a minimum pairwise distribution distance sKL_0 (see Section V-A). The generation procedure is illustrated in detail in Section 1 of the supplementary material. For each change $\phi_0 \to \phi_1$, we generate a training set TR of $N_m = 64$ batches of $\nu = 128$ samples per stationary modality $\phi_{0,m}$, for a total of $64 \cdot M$ batches, thus $8192 \cdot M$ training samples. To compute stable FPR and AUC values, we process 2000 batches per modality in both stationary and post-change conditions. Fig. 6 reports the FPR and AUC achieved by all the methods for $d \in \{4, 16, 32, 128\}$ averaged over 200 changes, together with a line indicating the target FPR $\alpha = 5\%$.

1) Low Dimensions ($d \in \{2, 4, 8, 16\}$): The first row of Fig. 6 reports the results for $d \in \{4, 16\}$ which are in line with $d \in \{2, 8\}$ reported in the supplementary material.

2) FPR Control: First of all, we notice that MMQT (both with and without calibration) guarantees an empirical FPR close to the target α for all the values of d and M. Nevertheless, when d increases, the FPR exceeds the target. We argue that this is a consequence of the shrinking ratio between ν and K, as commented in Section IV-E. In M-QuantTrees, histograms have uniform bins, resulting in the Total Variation statistics having discrete values as commented in [10]. Therefore, the empirical FPR achieved by M-QuantTrees does not

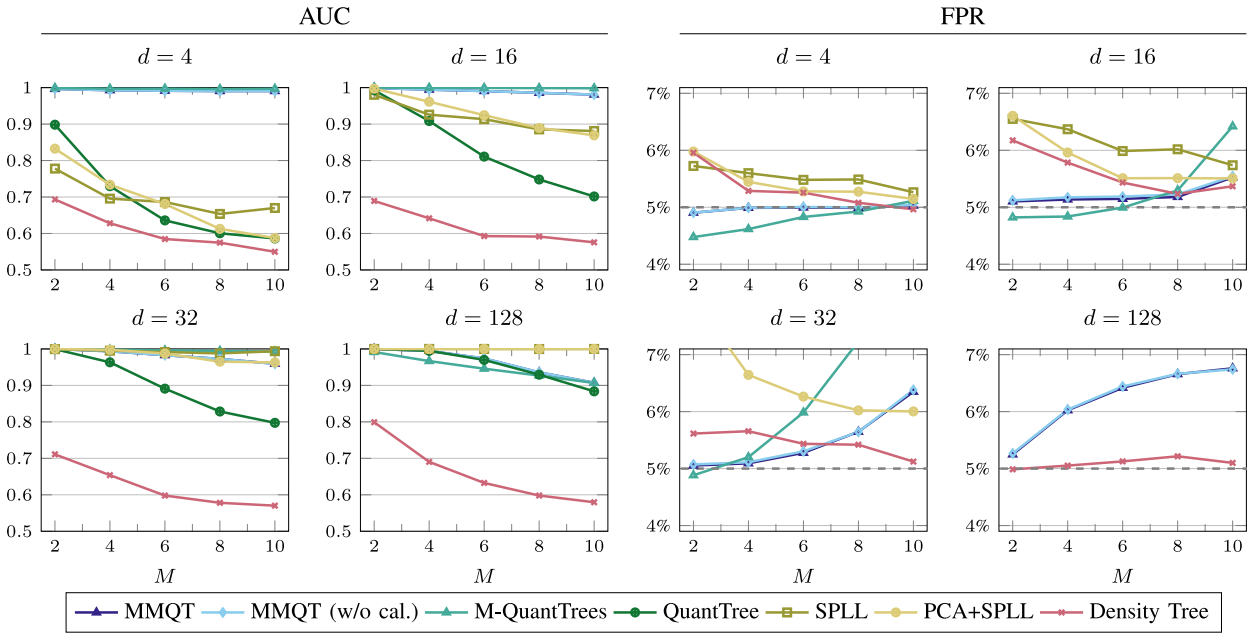

Fig. 6. AUC (left) and FPR (right) values achieved on the synthetic dataset where $M \in \{2, 4, 6, 8, 10\}$ modalities, and $d \in \{4, 16, 32, 128\}$. The dashed lines in the FPR plots indicate the target FPR of 5%. Values out of the considered range are not reported. At low dimensions (top row), MMQT achieves very high AUC scores by using the number of modalities $\widehat{M}$ estimated from TR, and at the same time controls the FPR. Instead, on high-dimensional data (bottom row) MMQT struggles to maintain its performance.

TABLE III

MODALITY-SPECIFIC AND AVERAGE FPR ACHIEVED BY THE CONSIDERED METHODS ON SYNTHETIC DATA ($d = 8$, $M = 4$). MMQT-BASED METHODS ALLOW FOR A MODALITY-SPECIFIC CONTROL OF FPR. METHODS USING BOOTSTRAP CAN ONLY CONTROL THE AVERAGE FPR

	$\phi_{0,1}$	$\phi_{0,2}$	$\phi_{0,3}$	$\phi_{0,4}$	Average
MMQT	4.80%	4.80%	4.70%	5.70%	5.00%
MMQT (w/o cal.)	5.20%	5.00%	4.50%	5.40%	5.03%
M-QuantTrees	3.20%	2.20%	3.60%	3.10%	3.02%
QuantTree	95.70%	94.80%	100.00%	88.70%	94.80%
SPLL	8.80%	0.60%	0.20%	11.90%	5.38%
PCA+SPLL	0.20%	0.00%	0.00%	18.00%	4.55%
Density Tree	0.10%	0.00%	21.00%	0.00%	5.27%

match the target. As expected, QuantTree does not control the FPR, since it considers test batches from a single modality as extremely rare with respect to the stationary distribution ϕ_0.

CD methods computing the detection threshold via bootstrap on TR achieve an overall empirical FPR close to α. However, the thresholds computed by these methods consider the distribution of the test statistic as a whole, disregarding the modality generating each batch, thus cannot control the FPR on each modality. The empirical, modality-specific, FPR typically departs substantially from α. To demonstrate our intuition, we compute the modality-specific FPR for changes $\phi_0 \to \phi_1$ where $d = 8$, $M = 4$, and we report a representative run in Table III. The results of SPLL, PCA-SPLL, and Density Tree confirm that thresholds computed by bootstrap only guarantee an average FPR control, while MMQT-based solutions control the FPR on each modality.

3) Detection Power: In terms of AUC, MMQT outperforms all the considered alternatives, except for M-QuantTrees. This is probably due to the fact that the latter constructs histograms having uniform probabilities, which typically achieve higher detection power [6], [24]. Nevertheless, using M-QuantTrees is more computationally demanding than MMQT, so the overhead might not be worth the slight gain in AUC. Remarkably, while the performance of other methods decreases as the number of modalities M increases, MMQT performance are stable, indicating that $\widehat{M}$, the bin-probability vectors $\{\boldsymbol{\pi}^m\}$, and thresholds $\{\tau^m\}$ have been correctly estimated.

4) Calibration of Bin Probabilities: Fig. 6 shows that, in this scenario, the calibration has a negligible impact on the detection performance of MMQT. We further investigate the impact of bin-probability calibration in terms of FPR control when varying the TR size and the distance between the stationary modalities. In these experiments, we compute the empirical FPR achieved by MMQT with and without calibration over a synthetic dataset in $\mathbb{R}^8$ comprising $M = 2$ stationary distributions $\phi_{0,1}$ and $\phi_{0,2}$ with Kullback–Leibler distance $\text{sKL}(\phi_{0,1}, \phi_{0,2}) \in \{1, 2, 4, 8\}$. We set $K = 32$, $\nu = 128$ and $\alpha = 5\%$, and we consider different numbers of training batches $N_m \in \{4, 8, 16, 32, 64, 128, 256\}$ per modality.

Table IV reports the average FPR achieved over 100 different ϕ_0, and shows that the calibration procedure reduces the gap between the empirical and the target FPR when TR is small. In general, both methods achieve an empirical FPR closer to α when N_m increases and we also notice that the control of the FPR improves when $\text{sKL}(\phi_{0,1}, \phi_{0,2})$ is larger. We also remark that the impact of the calibration procedure seems to be independent of the sKL distance between modalities, as it can be appreciated from the *Difference* rows of Table IV. We anticipate that these findings are in line with the results obtained with the INSECTS dataset (presented in Section V-F), where the calibration proved to be key when only $N_m = 10$ training batches per modality are given (see Table VI, bottom-left). We finally measure the norm of the

TABLE IV

FPR ACHIEVED BY MMQT (WITH AND WITHOUT CALIBRATION) WHEN VARYING THE SKL DISTANCE BETWEEN STATIONARY MODALITIES AND THE NUMBER OF TRAINING BATCHES N_m

$\mathrm{sKL}(\phi_{0,1}, \phi_{0,2})$	1	2	4	8	N_m
w/ calibration	5.88%	5.69%	5.30%	5.15%	4
w/o calibration	6.26%	6.16%	5.72%	5.45%	
Difference	0.38%	0.47%	0.41%	0.30%	
w/ calibration	5.36%	5.32%	5.33%	5.19%	8
w/o calibration	5.50%	5.45%	5.42%	5.35%	
Difference	0.14%	0.13%	0.09%	0.16%	
w/ calibration	5.20%	5.01%	5.07%	5.15%	16
w/o calibration	5.26%	5.00%	5.13%	5.21%	
Difference	0.07%	-0.01%	0.06%	0.07%	
w/ calibration	4.84%	5.06%	4.97%	5.03%	32
w/o calibration	4.85%	5.07%	5.00%	5.05%	
Difference	0.02%	0.01%	0.03%	0.02%	
w/ calibration	4.91%	4.96%	4.93%	5.08%	64
w/o calibration	4.92%	4.97%	4.93%	5.10%	
Difference	0.01%	0.01%	0.00%	0.02%	
w/ calibration	5.02%	4.91%	5.00%	5.01%	128
w/o calibration	5.01%	4.93%	5.00%	5.03%	
Difference	-0.01%	0.02%	0.00%	0.02%	
w/ calibration	4.95%	4.96%	5.03%	5.00%	256
w/o calibration	4.95%	4.97%	5.02%	5.01%	
Difference	0.00%	0.01%	-0.01%	0.01%	

difference between the reference bin-probability vectors before and after the calibration $\|\boldsymbol{\pi}^m - \mathcal{C}(\boldsymbol{\pi}^m)\|_2$. Table V confirms that the calibration changes the bin probabilities more substantially when the training set is small.

5) High Dimension ($d \in \{32, 64, 128\}$): The second row of Fig. 6 reports the results for $d \in \{32, 128\}$, while results for $d = 64$ are in line with these and reported in the supplementary material. Even though MMQT can be seamlessly applied to any dimension, our experiments show that its performance degrades when operating on high-dimensional data. As discussed in Section IV-E, employing $K = 64$ bins with high-dimensional data harms the accuracy of the histogram, as shown in the estimation error ΔM reported in Table II, which gets worse when d and M grow. By underestimating $\widehat{M}$, MMQT monitors batches drawn from different distributions using the same test statistic. As a result, the achieved empirical FPR exceeds the target α, and the AUC tends to decrease. The effect of the estimation error is more apparent on M-QuantTrees, which achieves extremely large FPR values because it monitors multiple modalities with the same QuantTree. The challenges of high dimensionality also affect the other methods. The thresholds computed by SPLL and PCA-SPLL through bootstrap lead to larger and larger FPR when the dimension and number of modalities grow. Density Tree controls the FPR at the expense of the detection power.

F. CD in Real-World Datastreams

1) CWRU Data: We consider two different CD experiments involving vibration data processed as described in Section V-A. The first one is the Single Speed scenario, where we train the methods on TR acquired when the ball bearing motor is operating at a specific speed Sp_i, $i \in \{1, 2, 3\}$. The second one is the Multi Speed, where all of the speeds $\{\mathrm{Sp}_1, \mathrm{Sp}_2, \mathrm{Sp}_3\}$ are represented in the training set. In both experiments, we introduce distribution changes $\phi_0 \to \phi_1$ using batches of faulty data. Moreover, we perform these experiments setting the target FPR to $\alpha \in \{0.5\%, 1\%, 2\%, 5\%\}$.

TABLE V

AVERAGE $L2$-NORM OF THE DIFFERENCE BETWEEN THE BIN PROBABILITIES $\|\boldsymbol{\pi}^m - \mathcal{C}(\boldsymbol{\pi}^m)\|_2$ BEFORE AND AFTER THE CALIBRATION PROCEDURE WHEN VARYING THE SKL DISTANCE BETWEEN STATIONARY MODALITIES AND THE NUMBER OF TRAINING BATCHES N_m

N_m	sKL			
	1	2	4	8
4	0,01183	0,00579	0,00382	0,00416
8	0,00535	0,00364	0,00185	0,00285
16	0,00512	0,00182	0,00147	0,00104
32	0,00336	0,00107	0,00080	0,00066
64	0,00115	0,00050	0,00054	0,00027
128	0,00078	0,00059	0,00012	0,00010
256	0,00036	0,00010	0,00010	0,00007

Table VI (top) reports the detection performance of the considered methods averaged on 100 changes. The Single Speed scenario corresponds to a traditional CD problem, where stationarity is described by a single distribution and as such, most methods control the FPR. In contrast, in the Multi Speed scenario, which represents a multimodal CD problem, only MMQT controls the FPR, while bootstrap-based methods are less accurate. This proves that MMQT is the only solution able to operate seamlessly on multimodal batches.

2) INSECTS Data: In this experiment, we monitor insects' wing-beat features to detect five distribution changes $\phi_0 \to \phi_1$ that are triggered by temperature changes from T_j to T_{j+1} as in the abrupt drift dataset [50]. Each modality $\phi_{0,m}$ corresponds to one of the six classes of insects. To this purpose, we rearrange the datastream to obtain batches containing instances from a single species. We consider both the *known-M* scenario, where M is provided and CD algorithms are trained on $N_m = 10$ batches per modality, and the *estimated-M* scenario, where MMQT estimates $\widehat{M}$ and all the CD methods are trained on $N_m = 60$ batches per modality. More training batches are required in the second case in order to estimate M. We discard all those species that do not contain at least N_m training batches, hence concepts can contain a different number M of stationary modalities $\{\phi_{0,m}\}$. We perform these experiments setting the target FPR to $\alpha \in \{0.5\%, 1\%, 2\%, 5\%\}$.

Table VI (bottom) reports the detection performance of the considered methods averaged on 100 changes $\phi_0 \to \phi_1$. In both the known- and estimated-M scenarios, MMQT outperforms all the competitors in terms of AUC and achieves an FPR close to the considered targets α. All the methods that set detection thresholds by bootstrap struggle when TR is small, while this is not a problem for MMQT, where thresholds are defined a priori. Finally, in the known-M scenario, the calibration procedure prevents MMQT from losing the FPR control when the training set is small, confirming this is key when TR is small, as discussed in Section V-E.

G. Concept-Drift Detection in Stream Learning Scenarios

In these experiments, we arrange the INSECTS dataset to test MMQT and competing methods in two Stream Learning problems, namely detecting: 1) the emergence of a novel class and 2) distribution changes due to concept drift.

TABLE VI

PERFORMANCE ON REAL-WORLD DATASETS: CWRU (TOP) AND INSECTS (BOTTOM). SINCE WE PROVIDE THE SAME ESTIMATE OF $\widehat{M}$ TO MMQT, MMQT W/O CALIBRATION AND M-QUANTTREE, ΔM IS CONSTANT. IN PARENTHESIS, THE WIDTH OF THE 95%-CONFIDENCE INTERVAL

	CWRU - Single speed						CWRU - Multi Speed					
	AUC	FPR (α=0.5%)	FPR (α=1%)	FPR (α=2%)	FPR (α=5%)	ΔM	AUC	FPR (α=0.5%)	FPR (α=1%)	FPR (α=2%)	FPR (α=5%)	ΔM
MMQT (w/ cal)	1.000 (.000)	0.49% (.04%)	0.98% (.06%)	1.78% (.09%)	4.72% (.16%)	0.27 (.05)	0.999 (0.000)	0.50% (.04%)	0.98% (.07%)	2.04% (.09%)	4.93% (.15%)	-0.04 (.05)
MMQT (w/o cal)	1.000 (.000)	0.49% (.04%)	0.98% (.06%)	1.78% (.09%)	4.78% (.16%)	0.27 (.05)	0.999 (0.000)	0.50% (.04%)	1.00% (.07%)	2.06% (.10%)	4.90% (.15%)	-0.04 (.05)
M-QuantTrees	1.000 (.000)	0.26% (.03%)	0.27% (.03%)	1.22% (.07%)	4.19% (.15%)	0.27 (.05)	1.000 (0.000)	0.98% (.12%)	1.01% (.13%)	2.39% (.14%)	6.17% (.29%)	-0.04 (.05)
QuantTree	1.000 (.000)	0.27% (.03%)	0.27% (.03%)	1.30% (.07%)	4.29% (.17%)	-	0.999 (0.001)	3.39% (.35%)	3.84% (.43%)	9.33% (.76%)	17.48% (1.01%)	-
SPLL	1.000 (.000)	1.11% (.07%)	1.70% (.11%)	3.04% (.15%)	3.89% (.09%)	-	1.000 (0.000)	1.26% (.01%)	1.59% (.06%)	4.29% (.02%)	3.90% (.02%)	-
PCA+SPLL	1.000 (.000)	1.26% (.04%)	1.85% (.01%)	2.67% (.11%)	2.78% (.04%)	-	1.000 (0.000)	0.88% (.01%)	1.83% (.01%)	3.81% (.01%)	4.78% (.02%)	-
Density Tree	0.918 (.008)	1.54% (.12%)	2.30% (.14%)	3.55% (.17%)	6.47% (.22%)	-	0.918 (0.001)	0.99% (.09%)	1.69% (.12%)	2.77% (.13%)	6.35% (.19%)	-

	INSECTS - M known					INSECTS - M estimated					
	AUC	FPR (α=0.5%)	FPR (α=1%)	FPR (α=2%)	FPR (α=5%)	AUC	FPR (α=0.5%)	FPR (α=1%)	FPR (α=2%)	FPR (α=5%)	ΔM
MMQT (w/ cal)	0.926 (.006)	0.74% (.06%)	1.48% (.08%)	2.87% (.16%)	5.79% (.14%)	0.976 (.001)	0.54% (.08%)	1.03% (.13%)	2.13% (.16%)	4.93% (.23%)	0.00 (.00)
MMQT (w/o cal)	0.926 (.006)	0.92% (.07%)	1.78% (.11%)	3.40% (.21%)	6.37% (.15%)	0.976 (.001)	0.60% (.11%)	1.12% (.16%)	2.13% (.17%)	5.02% (.23%)	0.00 (.00)
M-QuantTrees	0.901 (.006)	0.72% (.07%)	1.03% (.09%)	2.11% (.14%)	4.56% (.13%)	0.957 (.002)	0.31% (.06%)	0.76% (.13%)	1.61% (.24%)	4.63% (.24%)	0.00 (.00)
QuantTree	0.769 (.007)	84.59% (.79%)	88.21% (.66%)	91.44% (.56%)	94.76% (.42%)	0.867 (.013)	46.00% (2.94%)	51.36% (2.97%)	57.61% (3.01%)	64.11% (3.18%)	-
SPLL	0.709 (.015)	9.51% (.51%)	10.60% (.54%)	14.01% (.61%)	20.16% (.70%)	0.806 (.018)	1.84% (.20%)	2.45% (.20%)	4.04% (.28%)	7.35% (.40%)	-
PCA+SPLL	0.698 (.017)	8.23% (.50%)	9.38% (.53%)	11.80% (.58%)	17.45% (.66%)	0.713 (.021)	1.83% (.20%)	2.47% (.21%)	3.98% (.28%)	7.93% (.41%)	-
Density Tree	0.808 (.006)	1.84% (.18%)	2.34% (.21%)	3.00% (.22%)	5.47% (.31%)	0.910 (.007)	1.22% (.20%)	1.44% (.20%)	2.74% (.31%)	5.01% (.41%)	-

1) Novel Class Detection: In the INSECTS dataset, we denote the insect classes with letters A–F. We consider two stationary modalities $\phi_{0,1}$, containing insects A and B and modality $\phi_{0,2}$ containing insects D and E, always in equal proportions. Change $\phi_0 \rightarrow \phi_1$ corresponds to introducing a novel class, thus batches drawn from $\phi_{1,1}$ ($\phi_{1,2}$) contain 50% of insects from species C (F), while the remaining ones are from $\phi_{0,1}$ ($\phi_{0,2}$). We configure MMQT and M-QuantTrees with $\widehat{M} = 2$ clusters, one per stationary modality. In SPLL and PCA-SPLL, the GMMs use four Gaussian components to take into account the bimodal nature of each stationary modality. For each change $\phi_0 \rightarrow \phi_1$, the algorithms have been provided with $N_m = 32$ training batches per modality.

Table VII (left) reports the detection performance and the EDD averaged over 100 distribution changes and across all the concept drifts. MMQT is the best method in terms of AUC and FPR control. Similar AUC is attained by M-QuantTrees with a slightly worse FPR control, followed by Density Tree, which ranks third in terms of AUC but achieves an empirical FPR significantly far from the target. As expected, QuantTree and SPLL-based methods cannot manage these settings, as it is evident from the EDD and the poor control of FPR.

2) Concept-Drift Detection: This experiment is designed to show the potential of MMQT as an unsupervised drift detector in a Stream Learning problem. In particular, we train an active classifier over streaming data collected in batches, and we employ a k-nearest neighbors (k-NN) classifier paired with MMQT, which triggers corrective actions after detecting a drift. We denote as C_{MMQT} our active classifier using MMQT as a drift detector. As long as MMQT does not detect any change, C_{MMQT} classifies the incoming samples and tests the batch for CD. Then, if no detection is triggered, C_{MMQT} updates the classifier using supervised samples that amount to 10% of the batch. After the first detection, the trained k-NN shifts to a short-memory classifier trained in a sliding-window manner, exclusively using the labels from the two most recent batches. In this experiment, we compare our method with three baseline classifiers: C_{const} which is never updated, C_{UP} which is continuously updated with the supervised 10% of the incoming batch, and C_{SW} that uses the short-memory classifier since the beginning. To initially train the classifiers, we provide annotations to 25% of TR, except for C_{SW} which is always trained on the two most recent batches.

The stationary modalities $\phi_{0,1}$ and $\phi_{0,2}$ are mixtures of two species of insects at a specific temperature, while the concept drifts $\phi_0 \rightarrow \phi_1$ consist of temperature changes. Hence, the post-change distributions $\phi_{1,1}$ and $\phi_{1,2}$ are drawn from the same insects as $\phi_{0,1}$ and $\phi_{0,2}$ but at a different temperature. The training set TR comprises 16 batches of 128 samples for each modality, and we set $k = 4$ in the k-NN. To reduce false alarms triggering an unnecessary adaptation, we compute detection thresholds with a target FPR of 0.5%.

In Fig. 7, we report the accuracy of the classifiers averaged over windows of ten batches for two of the tested concept drifts. The drift's severity varies with the temperature change, and we report only the changes that most affect the classification accuracy. The supplementary material contains the results for all the settings. The thick, dark-colored lines represent the accuracy averaged over 1000 datastreams generated by sampling the INSECTS dataset, and the light-colored green lines represent the results of individual runs of C_{MMQT}, showing the effect of a false alarm on the classification performance. Moreover, we highlight with a gray background the time interval following the drift, where the averaged accuracy of the methods overlaps with the change point.

All the classifiers suffer an accuracy drop following the concept drift. However, when using active approaches, promptly detecting the drift enables quick recovery of the classification performance. In this regard, MMQT achieves an average detection delay of 1.51 batches, corresponding to 194 samples. Thus, C_{MMQT} quickly switches to the short-memory classifier after the change, limiting the impact of the drift over its classification accuracy. As expected, the (rare) false alarms raised by MMQT affect the classification performance before the change-point and consequently the average pre-change accuracy of C_{MMQT} is slightly lower than those of C_{const} and C_{UP}. Before the drift, the short-memory classifier C_{SW} cannot achieve the same accuracy as the other methods, which are trained over 25% of TR and, in the case of C_{MMQT} and C_{UP}, are updated on streaming data. After the drift, C_{const} does not recover from the accuracy loss. C_{UP} slowly gains accuracy by incorporating new annotated samples, while C_{SW} quickly settles at a post-change accuracy level. As expected, after the change, C_{MMQT} and C_{SW} achieve the same performance since they implement the same update strategy. This experiment demonstrates that MMQT can be successfully employed in

TABLE VII

CD PERFORMANCE ON NOVEL CLASS (LEFT) AND CONCEPT DRIFT (RIGHT) DETECTION. IN PARENTHESIS, THE WIDTH OF THE 95%-CONFIDENCE INTERVAL

	Novel Class Detection ($\alpha = 5\%$)			Concept Drift Detection ($\alpha = 0.5\%$)		
	AUC	FPR	EDD	AUC	FPR	EDD
MMQT	0.891 (.005)	5.20% (.06%)	209.8	0.942 (.004)	0.53% (.05%)	198.2
MMQT (w/o cal)	0.892 (.005)	5.25% (.07%)	209.0	0.942 (.004)	0.54% (.06%)	197.0
M-QuantTrees	0.898 (.005)	3.32% (.05%)	204.9	0.941 (.005)	0.40% (.06%)	207.1
QuantTree	0.679 (.013)	36.63% (1.47%)	204.9	0.831 (.009)	10.66% (1.17%)	196.4
SPLL	0.551 (.008)	13.03% (.38%)	773.3	0.759 (.021)	14.65% (1.10%)	211.1
PCA+SPLL	0.428 (.008)	12.51% (.38%)	1312.8	0.748 (.021)	11.52% (.96%)	238.0
Density Tree	0.764 (.015)	7.26% (.28%)	250.0	0.877 (.014)	4.63% (.54%)	172.1

a Stream Learning problem, once paired with a classifier that is updated upon detecting a drift.

To further investigate the benefit of MMQT in Stream Learning, we run a concept drift detection experiment in the same setting as the classification one, comparing the performance of MMQT against the CD methods used in the previous experiments. All the methods monitor the data distribution disregarding the labels, and are trained on 16 batches. Detection thresholds are set using $\alpha = 0.5\%$. Table VII (right) reports the performance in terms of AUC and FPR averaged over 100 drifts $\phi_0 \to \phi_1$ and clearly shows that methods defining thresholds by bootstrap struggle in this setting, where few training batches are provided and α is very low. Since each false positive triggers an unnecessary adaptation, these bootstrap-based methods are unfit for a Stream Learning problem where the training set is scarce. In contrast, MMQT relies on strong theoretical results to compute the detection thresholds and can control the FPR in both the classification and detection scenarios where it achieves the highest AUC. In our analysis, we focused on abrupt changes. However, MMQT might also be employed to detect gradual, incremental, or transient drifts at the expense of larger EDD. This is common to all CD methods since, for these types of drifts, the impact of change on their test statistic is lower than an abrupt change once the drift starts.

To conclude, the major drawback of MMQT in the Stream Learning scenario consists of the fact that it is a one-shot detector, processing each incoming batch independently of the previous ones. Interestingly, the research on MMQT could follow the path of QT-EWMA [26], [38], which recently combined QuantTree with a sequential monitoring scheme based on the Exponential Weighted Moving Average.

H. Sensitivity Analysis of CD Algorithms

We finally perform a sensitivity analysis on critical parameters for the CD algorithms, to provide additional insights on their configuration. First of all, the batch size ν, which we set to 128 in all the experiments, is tightly related to the change-detection power. In fact, all the Hypothesis Test underpinning the detection methods are consistent, meaning that the detection power increases (resp. worsens) when ν grows (resp. decreases). This is confirmed by the additional experiments, reported in the supplementary material, where we set $\nu \in \{64, 128, 256\}$ on synthetic datasets ($d = 32$). As for the algorithm-specific parameters, the results reported here and in the supplementary material show that the value of α' does not significantly affect the accuracy of the estimation of the number of modalities in MMQT, thus we have safely set

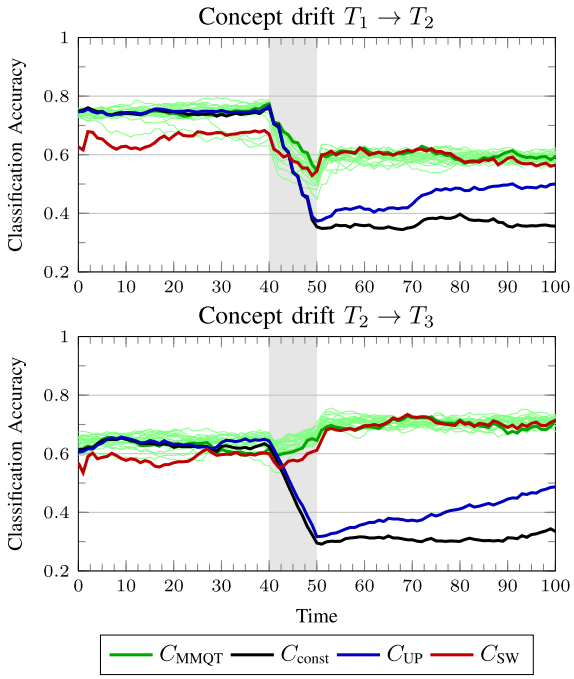

Fig. 7. Classification accuracy averaged over windows of ten consecutive batches in the Stream Learning experiment. The light-green lines represent individual runs of C_{MMQT}, the thick dark lines represent the average results. The gray area highlights the time interval immediately following the drift.

$\alpha' = 5\%$ in all our experiments. We have already discussed guidelines for regulating the number of histogram bins K, which requires taking into account ν and the data dimension d. Since the number of bins is key in MMQT, M-QuantTrees, QuantTree, and Density Tree, we have carefully defined K to yield the best performance (see Table I). To enable a fair comparison, the number of Gaussian components C in SPLL-based methods was also defined to achieve the best performance in each dataset.

VI. LIMITATIONS

The most apparent limitation of MMQT concerns high-dimensional datastreams. It is well known [9] that change detectability worsens when d increases and histograms further suffer when ν is small and d is large, since it becomes difficult to estimate bins that accurately match the target probabilities [10]. MMQT is no exception, and, as expected, its performance degrades when the data dimension scales, as shown in Fig. 6 and Table VI. However, these limitations emerge only in synthetic datasets at very high dimensions ($d = 64\,128$) and do not affect experiments on the INSECTS dataset, which represents a real-world and high-dimensional ($d = 33$) benchmark for CD. We remark that in this challenging setting, the performance drop is common to all the considered CD algorithms.

Another issue we experienced consists in estimating the number of modalities when M is large. In these cases, the set of bin-probability vectors computed from TR, namely $\Gamma(\text{TR})$ in (6), is cluttered, and the κ-means falls short. Moreover, in MMQT, $\widehat{M}$ is estimated by an exhaustive search, which might become computationally demanding. However, under the assumption that we know a maximum number of modalities $M_{\max}$, more efficient strategies can be implemented, e.g.,

performing bisection on the interval $[1, M_{\max}]$. It is worth noting that, so far, all the CD methods in the literature have been limited to the case $M = 1$, and MMQT is the first algorithm addressing multiple stationary modalities, i.e., $M > 1$. We hope our work will pave the way toward improved methods for handling a large number of modalities.

Finally, MMQT relies on the assumption that each batch W is drawn from a single modality $\phi_{0,m}$. When this assumption is violated, we have two possible scenarios. First, W might contain both pre- and post-change samples, and MMQT would correctly detect a change, as confirmed by our experiment on novel class detection. However, the detection is expected to be delayed with respect to batches entirely drawn from $\phi_{1,m}$. Second, a single batch W might include a transition among two stationary modalities $\phi_{0,m}$ and $\phi_{0,m'}$. In this case, MMQT is expected to raise a false alarm. This is however rather unlikely in high-throughput datastreams (like predictive maintenance), where batches are acquired over a short time interval. Here, it is reasonable to assume that the single-modality assumption holds for most batches, and it is possible to validate each detection over an additional batch to discard false positives due to modality transition.

VII. Conclusion

We presented MMQT, the first algorithm performing batch-wise CD in a multimodal scenario. These CD settings have never been considered in the literature but represent an extension of the classical CD setting. MMQT solves this problem by taking advantage of the efficient QuantTree scheme together with a rule to map each batch to a reference modality. We present an effective procedure for estimating the number of modalities that, for the first time in the literature, relies on false positive control rather than other clustering validation measures. Moreover, we introduce a theoretically sound bin probabilities calibration procedure that demonstrates to be very effective for small training sets.

Our experiments demonstrate that MMQT achieves compelling detection performance, guaranteeing a controlled FPR. Moreover, we show how MMQT can be employed to detect both novel classes and concept drifts, thus triggering adaptation for classifiers in Stream Learning problems in NSE.

In future work, we plan to combine the ideas of MMQT with QT-EWMA [38], a sequential extension of QuantTree for streaming data. Specifically, we are studying modality-specific statistics and smooth modality transitioning as if the bin-probability vectors of stationary batches follow a continuous trajectory in $\mathbb{R}^K$. We are also investigating further applications of MMQT to design superior active classifiers for non-stationary datastreams, where data distribution is multimodal or recurrent. Here, recurrent concepts can be represented using our histogram-based embedding, enabling the identification of a previously encountered concept and the recovery of historical information.

References

[1] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[2] D. H. Green, A. W. Langham, R. A. Agustin, D. W. Quinn, and S. B. Leeb, "Adaptation for automated drift detection in electromechanical machine monitoring," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 23, 2022, doi: 10.1109/TNNLS.2022.3184011.

[3] H. Tian, N. L. D. Khoa, A. Anaissi, Y. Wang, and F. Chen, "Concept drift adaption for online anomaly detection in structural health monitoring," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2813–2821.

[4] S. K. Perepu and K. Dey, "CDDM: A method to detect and handle concept drift in dynamic mobility model for seamless 5G services," in *Proc. IEEE Globecom Workshops (GC Wkshps*, Dec. 2020, pp. 1–6.

[5] S. Singhal, U. Chawla, and R. Shorey, "Machine learning & concept drift based approach for malicious website detection," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 582–585.

[6] A. Liu, J. Lu, and G. Zhang, "Concept drift detection via equal intensity k-means space partitioning," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3198–3211, Jun. 2021.

[7] L. I. Kuncheva, "Change detection in streaming multivariate data using likelihood detectors," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 1175–1180, May 2013.

[8] L. I. Kuncheva and W. J. Faithfull, "PCA feature extraction for change detection in multidimensional unlabeled data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 69–80, Jan. 2014.

[9] C. Alippi, G. Boracchi, D. Carrera, and M. Roveri, "Change detection in multivariate datastreams: Likelihood and detectability loss," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 2, 2016, pp. 1368–1374.

[10] G. Boracchi, D. Carrera, C. Cervellera, and D. Macciò, "QuantTree: Histograms for change detection in multivariate data streams," in *Proc. ICML*, vol. 80, 2018, pp. 639–648.

[11] E. L. Lehmann and J. P. Romano, *Testing Statistical Hypotheses*. New York, NY, USA: Springer, 2006.

[12] M. F. Schilling, "Multivariate two-sample tests based on nearest neighbors," *J. Amer. Stat. Assoc.*, vol. 81, no. 395, pp. 799–806, Sep. 1986.

[13] S. Li, Y. Xie, H. Dai, and L. Song, "M-statistic for kernel change-point detection," in *Advances in Neural Information Processing Systems*, vol. 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015, pp. 3366–3374.

[14] Z. Harchaoui, E. Moulines, and F. Bach, "Kernel change-point analysis," in *Advances in Neural Information Processing Systems*, vol. 21, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Red Hook, NY, USA: Curran Associates, 2009, pp. 609–616.

[15] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Netw.*, vol. 43, pp. 72–83, Jul. 2013.

[16] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 246–258, Feb. 2017.

[17] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 935–944.

[18] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2007, pp. 667–676.

[19] J. Shao, Z. Ahmadi, and S. Kramer, "Prototype-based learning on concept-drifting data streams," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 412–421.

[20] F. Gu, G. Zhang, J. Lu, and C.-T. Lin, "Concept drift detection based on equal density estimation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 24–30.

[21] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *Proc. Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1–7.

[22] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 324–334, Feb. 2018.

[23] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *Proc. Symp. Interface Statist., Comput. Sci., Appl.* Princeton, NJ, USA: Citeseer, 2006, pp. 1–24.

[24] G. Boracchi, C. Cervellera, and D. Macciò, "Uniform histograms for change detection in multivariate data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 1732–1739.

[25] A. Criminisi, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Found. Trends Comput. Graph. Vis.*, vol. 7, nos. 2–3, pp. 81–227, 2011.
[26] L. Frittoli, D. Carrera, and G. Boracchi, "Nonparametric and online change detection in multivariate datastreams using QuantTree," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8328–8342, Aug. 2023.
[27] D. Stucchi, P. Rizzo, N. Folloni, and G. Boracchi, "Kernel QuantTree," in *Proc. 40th Int. Conf. Mach. Learn. (ICML)*, 2023, pp. 1–21.
[28] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 620–634, Apr. 2013.
[29] Y. Song, J. Lu, A. Liu, H. Lu, and G. Zhang, "A segment-based drift adaptation method for data streams," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4876–4889, Sep. 2022.
[30] Z. Yang, S. Al-Dahidi, P. Baraldi, E. Zio, and L. Montelatici, "A novel concept drift detection method for incremental learning in nonstationary environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 309–320, Jan. 2020.
[31] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 1–7.
[32] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 481–492.
[33] X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou, "Streaming classification with emerging new class by class matrix sketching," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.
[34] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Brazilian Symp. Artif. Intell.* São Luís, Brazil: Springer, 2004, pp. 286–295.
[35] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit. Lett.*, vol. 33, no. 2, pp. 191–198, Jan. 2012.
[36] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 1181–1190.
[37] E. Yu, Y. Song, G. Zhang, and J. Lu, "Learn-to-adapt: Concept drift adaptation for hybrid multiple streams," *Neurocomputing*, vol. 496, pp. 121–130, Jul. 2022.
[38] L. Frittoli, D. Carrera, and G. Boracchi, "Change detection in multivariate datastreams controlling false alarms," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Bilbao, Spain: Springer, 2021, pp. 421–436.
[39] D. Stucchi, L. Frittoli, and G. Boracchi, "Class-distribution monitoring for concept drift detection via QT-EWMA," in *Proc. Int. Joint Conf. Neural Netw.*, 2022, pp. 1–8.
[40] D. Neupane and J. Seok, "Bearing fault detection and diagnosis using Case Western Reserve University dataset with deep learning approaches: A review," *IEEE Access*, vol. 8, pp. 93155–93178, 2020.
[41] R. B. Randall and J. Antoni, "Rolling element bearing diagnostics a tutorial," *Mech. Syst. Signal Process.*, vol. 25, no. 2, pp. 485–520, 2011.
[42] C. W. Chiu and L. L. Minku, "A diversity framework for dealing with multiple types of concept drift based on clustering in the model space," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1299–1309, Mar. 2022.
[43] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.
[44] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[45] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.
[46] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Statist., Simul. Comput.*, vol. 3, no. 1, pp. 1–27, 1974.
[47] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2, pp. 107–145, Dec. 2001.
[48] C. Alippi, G. Boracchi, and D. Carrera, "CCM: Controlling the change magnitude in high dimensional data," in *Proc. INNS Conf. Big Data*. Thessaloniki, Greece: Springer, 2016, pp. 216–225.
[49] *Case Western Reserve University Bearing Data Center*. Accessed: Sep. 7, 2021. [Online]. Available: http://csegroups.case.edu/bearingdatacenter/home
[50] V. M. A. Souza, D. M. Reis, A. G. Maletzke, and G. E. A. P. A. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Mining Knowl. Discovery*, vol. 34, pp. 1805–1858, Jul. 2020.
[51] D. V. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. Symp. Discrete Algorithms*, 2007, pp. 1027–1035.
[52] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 1, pp. 1–58, Mar. 2009.

Diego Stucchi received the master's degree in mathematics from the Università degli studi di Milano-Bicocca, Milan, Italy, in 2018. He is currently pursuing the Ph.D. degree with the Politecnico di Milano, Milan.

He is also actively collaborating with Tampere University, Tampere, Finland, on multimodal images registration and inverse problems. He holds an industrial Ph.D. grant focused on the design of algorithms to address real world AD problems. His research interests include computer vision and anomaly detection problems.

Luca Magri received the master's degree in mathematics and the Ph.D. degree from the University of Milan, Milan, Italy, in 2012 and 2015, respectively, with a thesis in computer vision.

From 2015 to 2018, he has been a Post-Doctoral Researcher with the University of Verona, Verona, Italy, and the University of Udine, Udine, Italy, working on 3-D reconstruction. He is currently a Senior Researcher (RTD-b) with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan.

Diego Carrera received the master's degree in mathematics from the Università degli Studi di Milano, Milan, Italy, in 2013, and the Ph.D. degree in information technology from the Politecnico di Milano, Milan, Italy, in 2018.

In 2015, he has been a Visiting Researcher with the Tampere University of Technology, Tampere, Finland. He is currently an Application Development Engineer with STMicroelectronics, Agrate Brianza, Italy, developing quality inspection systems to monitor the wafer production. His research interests include unsupervised learning algorithms, in particular change detection in high dimensional datastreams, anomaly detection in signal and images, and domain adaptation.

Giacomo Boracchi (Member, IEEE) received the degree master's in mathematics from the Universitá Statale di Milano, Milan, Italy, in 2004, and the Ph.D. degree in information technology from the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, in 2008.

He is currently an Associate Professor of computer engineering with DEIB, Politecnico di Milano. Since 2015, he has leading industrial research projects concerning outlier detection, X-ray systems, and automatic quality inspection systems. He has published more than 90 papers in international conferences and journals. His research interests include machine learning and image processing, in particular change/anomaly detection, domain adaptation, and image restoration and analysis.

Dr. Boracchi received the IBM Faculty Award in 2015, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS Outstanding Paper Award in 2016, the Nokia Visiting Professor Scholarship in 2017, and the NVIDIA Applied Research Grant in 2021. He has held tutorials in major IEEE conferences: ICIP 2020, ICASSP 2018, and IJCNN 2017 and 2019. From 2019 to 2020, he served as an Associate Editor for *IEEE Computational Intelligence Magazine*. He is also an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement