Adaptive Bernstein Change Detector for High-Dimensional Data Streams

Marco Heyden*, Edouard Fouché, Vadim Arzamasov, Tanja Fenn, Florian Kalinke and Klemens Böhm

Karlsruhe Institute of Technology, Karlsruhe, Germany.

*Corresponding author(s). E-mail(s): marco.heyden@kit.edu;
Contributing authors: firstname.lastname@kit.edu;

Acknowledgments

This work was supported by the German Research Foundation (DFG) Research Training Group GRK 2153: *Energy Status Data — Informatics Methods for its Collection, Analysis and Exploitation.*

This version of the article has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s10618-023-00999-5.

Abstract

Change detection is of fundamental importance when analyzing data streams. Detecting changes both quickly and accurately enables monitoring and prediction systems to react, e.g., by issuing an alarm or by updating a learning algorithm. However, detecting changes is challenging when observations are high-dimensional. In high-dimensional data, change detectors should not only be able to identify when changes happen, but also in which subspace they occur. Ideally, one should also quantify how severe they are. Our approach, ABCD, has these properties. ABCD learns an encoder-decoder model and monitors its accuracy over a window of adaptive size. ABCD derives a change score based on Bernstein's inequality to detect deviations in terms of accuracy, which indicate changes. Our experiments demonstrate that ABCD outperforms its best competitor by up to 20% in F1-score on average. It can also accurately estimate changes' subspace, together with a severity measure that correlates with the ground truth.

Keywords: change detection, concept drift, data streams, high-dimensionality

1 Introduction

Data streams are open-ended, ever-evolving sequences of observations from some process. They pose unique challenges for analysis and decision-making. One crucial task is to detect changes, i.e., shifts in the observed data, that may indicate a change in the underlying process. Change detection has been an active research area. However, the high-dimensional setting, in which observations contain a large number of simultaneously measured quantities, did not receive enough attention. Yet, it may yield useful insights in environmental monitoring (de Jong and Bosman, 2019), human activity recognition (Vrigkas et al, 2015), network traffic monitoring (Naseer et al, 2020), automotive (Liu et al, 2019), predictive maintenance (Zhao et al, 2018), and biochemical engineering (Mowbray et al, 2021):

Example (Biofuel production)**.** The production of fuel from biomass is a complex process comprising many interdependent process steps. Those include pyrolysis, synthesis, distillation, and separation. Many steps rely on (by-)products of other steps as reactants, leading to a highly interconnected system with many process parameters. A monitoring system tracks the process parameters to detect failures in the plant: (i) The system must detect changes in a large (i.e., high-dimensional) vector of process parameters, which may indicate failures. (ii) The system must find out which process parameters are affected by the change to allow for a targeted reaction. Since the system is very complex and has many interconnected components, change is often evident only when considering correlations between process parameters. An example would be the correlation between temperature and concentration fluctuations. So it is insufficient to monitor each process parameter in isolation. (iii) There can exist slight changes which only require minor adjustments and more severe ones that require immediate intervention to avoid a shutdown of the plant. The monitoring system should provide an estimate of the severity of change.

The example illustrates three requirements for modern change detectors:

- **R1: Change point.** The primary task of change detectors is to identify that the data stream has changed and when it occurred.
- **R2: Change subspace.** A change may only concern a subset of dimensions — the *change subspace*. Change detectors for high-dimensional data streams should be able to identify such subspaces.
- **R3: Change severity.** Quantifying relative change severity to distinguish between changes of different importance is essential to react appropriately.

Prior works already acknowledge the relevance of the above requirements (Lu et al, 2019; Webb et al, 2018). However, fulfilling R1–R3 in combination remains challenging since they depend on each other: on the one hand, detecting changes in high-dimensional data is difficult because changes typically only affect few dimensions. Unaffected dimensions "dilute" a change (i.e., a change occuring in a subspace appears to be less severe in the full space).

This might make changes harder to detect in all dimensions. On the other hand, detecting the change subspace should occur *after* detecting a change, since monitoring all possibles subspaces is intractable. Last, one should restrict computation of change severity to the change subspace to eliminate dilution.

Existing methods for change detection, summarized in Table 1, either are univariate (UV), multivariate (MV), or specifically designed for high-dimensional data (HD); the latter claim efficiency w.r.t. high-dimensionality or resilience against the "curse of dimensionality". However, they do not fulfill R1-R3 in combination sufficiently well as Section 2 describes.

Thus, we propose the Adaptive Bernstein Change Detector (ABCD), which addresses R1-R3 in combination. We articulate our contributions as follows:

(i) Problem Definition: We formalize the problem of detecting changes in high-dimensional data streams such that R1-R3 can be tackled in combination. **(ii) Adaptive Bernstein Change Detector:** We present ABCD, a change detector for high-dimensional data, that satisfies R1–R3. It monitors the loss of an encoder-decoder model using an adaptive window size and statistical testing. Adaptive windows enable ABCD to detect severe changes quickly and, over a longer period, identify hard-to-detect changes that would typically require a large window size. **(iii) Bernstein change score:** Our approach applies a statistical test based on Bernstein's inequality. This limits the probability of false alarms. **(iv) Online computation:** We propose an efficient method for computing the change score in adaptive windows and discuss design choices leading to constant time and memory. **(v) Benchmarking:** We conduct experiments on 10 data streams based on real-world and synthetic data with many dimensions and compare ABCD with recent approaches. The results indicate that ABCD outperforms its competitors consistently w.r.t. R1–R3, is robust to high-dimensional data and is useful in domains including human activity recognition, gas detection, and image processing. We also study ABCD's parameter sensitivity. Our code[1] follows the popular Scikit-Multiflow API (Montiel et al, 2018), so it is easy to use in future research.

2 Related work

2.1 Change detector types

Most existing change detectors are *supervised*, i.e., they focus on detecting changes in the relationship between input data and a target variable (Iwashita and Papa, 2019). However, class labels are rarely available in reality, which limits their applicability. On the contrary, the *unsupervised* change detectors aim to detect changes only in the input data. Our approach belongs to this category, so we restrict our review to unsupervised approaches.

Most existing approaches detect changes whenever a measure of discrepancy between newer observations (the current window) and older observations (the reference window) exceeds a threshold. Some approaches, e.g.,

[1] https://github.com/heymarco/AdaptiveBernsteinChangeDetector

Table 1: Related work.

Approach	Reference	Type	R1	R2	R3
ADWIN	Bifet and Gavaldà (2007)	UV	✓	–	–
SeqDrift2	Pears et al (2014)	UV	✓	–	–
kdq-Tree	Dasu et al (2006)	MV	✓	–	✓
PCA-CD	Qahtan et al (2015)	MV	✓	–	✓
IKS	dos Reis et al (2016)	MV	✓	✓	–
LDD-DSDA	Liu et al (2017)	MV	✓	–	–
AdwinK	Faithfull et al (2019)	MV	✓	✓	–
D3	Gözüaçık et al (2019)	MV	✓	–	✓
ECHAD	Ceci et al (2020)	MV	✓	–	✓
IBDD	de Souza et al (2020)	HD	✓	–	✓
WATCH	Faber et al (2021)	HD	✓	–	✓
ABCD	this work	HD	✓	✓	✓

D3 (Gözüaçık et al, 2019) or PCA-CD (Qahtan et al, 2015), implement the reference and current window as two contiguous sliding windows. Other approaches, such as IBDD (de Souza et al, 2020), IKS (dos Reis et al, 2016) or WATCH (Faber et al, 2021) use a fixed reference window. A major problem is to choose the appropriate size for the window; thus (Bifet and Gavaldà, 2007) propose windows of adaptive size, that grow while the stream remains unchanged and shrink otherwise. Several work leverage this principle, e.g. (Sun et al, 2016; Khamassi et al, 2015; Fouché et al, 2019; Suryawanshi et al, 2022). We also use adaptive windows to lower the number of parameters of ABCD.

2.2 Univariate change detection

There exist many approaches for change detection in univariate (UV) data streams. Two of them, Adaptive Windowing (ADWIN) (Bifet and Gavaldà, 2007) and SeqDrift2 (Pears et al, 2014), share some similarity with our approach. Like ADWIN, ABCD relies on an adaptive window. Like SeqDrift2, it uses Bernstein's inequality (Bernstein, 1924). But unlike ADWIN and SeqDrift2, ABCD can handle high-dimensional data while fulfilling R1-R3.

2.3 Multivariate change detection

To detect changes in multivariate (MV) data, some approaches apply univariate algorithms in each dimension of the stream. Faithfull et al (2019) propose to use one ADWIN detector per dimension (with k dimensions). They declare a change whenever a certain fraction of the detectors agree. We call this approach AdwinK later on. Similarly, IKS (dos Reis et al, 2016) uses an incremental variant of the Kolmogorov-Smirnov test deployed in each dimension. Unlike AdwinK, IKS issues an alarm if at least one dimension changes.

There also exist approaches specifically designed for multivariate (Jaworski et al, 2020; Ceci et al, 2020; Qahtan et al, 2015; Gözüaçık et al, 2019; Dasu et al, 2006), or even high-dimensional (HD) data (Faber et al, 2021; de Souza et al, 2020). Similar to ABCD, Jaworski et al (2020) and Ceci et al (2020)

use dimensionality-reduction methods to capture the relationships between dimensions. However, our approach is computationally more efficient, limits the probability of false alarms, identifies change subspace, and estimates change severity. D3 (Gözüaçık et al, 2019) uses the AUC-ROC score of a discriminative classifier that tries to distinguish the data in two sliding windows. It reports a change if the AUC-ROC score exceeds a pre-defined threshold. PCA-CD (Qahtan et al, 2015) first maps observations in two windows to fewer dimensions using PCA. Then the approach estimates the KL-divergence between both windows for each principal component. PCA-CD detects a change if the maximum observed KL-divergence exceeds a threshold. However, (Goldenberg and Webb, 2019) point out that this technique is limited to linear transformations and ignores combined change in multiple dimensions. LDD-DSDA (Liu et al, 2017) measures the degree of local drift that describes regional density changes in the input data. The approach proposed by (Dasu et al, 2006) structures observations from two windows (sliding or fixed) in a kdq-tree. For each node, they measure the KL-divergence between observations from both windows. However, (Qahtan et al, 2015) show experimentally that this approach is not suitable for high-dimensional data.

IBDD (de Souza et al, 2020) and WATCH (Faber et al, 2021) specifically address challenges arising from high-dimensional data. The former monitors the mean squared deviation between two equally sized windows. The latter monitors the Wasserstein distance between a reference and a sliding window. However, both cannot detect change subspaces or measure severity.

2.4 Offline change point detection

Offline change point detection, also known as signal segmentation, divides time series of a given length into K homogeneous segments (Truong et al, 2020). Many of the respective algorithms are not suitable for data streams: Some require specifying K a priori (Bai and Perron, 2003; Harchaoui and Cappe, 2007; Lung-Yut-Fong et al, 2015); others (Killick et al, 2012; Lajugie et al, 2014; Matteson and James, 2014; Chakar et al, 2017; Garreau and Arlot, 2018) scale superlinearly with time. WATCH (Faber et al, 2021), discussed above, is the state of the art extension of offline change point detection to data streams.

2.5 Change subspace

The notion of a *change subspace* is different from the existing notion of *change region* (Lu et al, 2019). The former describes a subset of dimensions that changed, the latter identifies density changes in some local region, e.g., a hyperrectangle or cluster (Liu et al, 2017). Our definition of change subspaces is related to *marginal change magnitude* (Webb et al, 2018), but is more general since it can also accomodate changes in a subspace's joint distribution.

Because high-dimensional spaces are typically sparse (due to the curse of dimensionality), identifying density changes in them is not effective. On the other hand, knowing that a change affected a specific set of dimensions can

help identify the cause of the change, as we have motivated in our introductory example. Thus, we focus on detecting change subspaces in this work.

In the domain of statistical process control, some approaches extend well-known methods, such as Cusum (Page, 1954) or Shewhart charts (Shewhart, 1930), to multiple dimensions. They address the problem of identifying change subspaces to some extent, however, they often make unrealistic assumptions: they focus on Gaussian or sub-Gaussian data (Chaudhuri et al, 2021; Xie et al, 2020), require that different dimensions are initially independent (Chaudhuri et al, 2021), require subspace changes to be of low rank (Xie et al, 2020), or assume that the size of the change subspace is known a priori (Jiao et al, 2018).

From the approaches reviewed in Section 2.3 only AdwinK and IKS identify the corresponding change subspace. However, both approaches do not find changes that hide in subspaces, e.g., correlation changes, because they monitor each dimension in isolation. In contrast, our approach aims to learn the relationships between different dimensions so that it can detect such changes. Next, AdwinK cannot identify subspaces with fewer than k dimensions.

2.6 Change severity

According to (Lu et al, 2019), change severity is a positive measure of the discrepancy between the data observed before and after the change. One can either measure the divergence between distributions directly, as done by kdq-Tree (Dasu et al, 2006), LDD-DSDA (Liu et al, 2017), and WATCH (Faber et al, 2021), or indirectly with a score that correlates with change severity, as done by D3 (Gözüaçık et al, 2019). Following this reasoning, an approach that satisfies R3 should compute a score that depends on the change severity (Gözüaçık et al, 2019; Dasu et al, 2006; de Souza et al, 2020; Qahtan et al, 2015; Faber et al, 2021), i.e., the higher the score, the higher the severity. Finally, hypothesis-testing-based approaches, such as ADWIN (Bifet and Gavaldà, 2007), SeqDrift2 (Pears et al, 2014), AdwinK (Faithfull et al, 2019), or IKS (dos Reis et al, 2016), do not quantify change severity: a slight change observed over a longer time can lead to the same p-value as a severe change observed over a shorter time, hence p is not informative about change severity.

2.7 Pattern based change detection

A related line of research, pattern-based change detection, deals with identifying changes in temporal graphs (Loglisci et al, 2018; Impedovo et al, 2019, 2020a,b). In particular, Loglisci et al (2018) detect changes in the graph, identify the affected subgraphs, and quantify the amount of change for these subgraphs. This is similar to our methodology. However, these methods work well with graph data, but we are dealing with vector data. To apply these methods in our context, one would need to create a graph, e.g., by representing each dimension as a node and indicating pairwise correlations with edges. However, constructing such a graph becomes impractical for high-dimensional observations because of the exponentially growing number of subspaces.

2.8 Competitors

In our experiments, we compare to AdwinK, IKS, D3, IBDD, and WATCH. IBDD, WATCH, and D3 are recent change detectors for multivariate and high-dimensional data that fulfill R3. AdwinK extends the ADWIN algorithm to the multivariate case and fulfills R2. Finally, IKS is the only approach employing a non-parametric two-sample test for change detection while also satisfying R2.

3 Preliminaries

We are interested in finding changes in the last t observations $S = (x_1, x_2, \ldots, x_t)$ from a stream of data. Each x_i is a d-dimensional vector independently drawn from a (unknown) distribution F_i. We assume without loss of generality that each vector coordinate is bounded in $[0, 1]$, i.e., $x_i \in [0, 1]^d$.

Definition 1 (Change)**.** *A change occurs at time point t^* if the data-generating distribution changes after t^*: $F_{t^*} \neq F_{t^*+1}$.*

In high-dimensional data, changes typically affect only a subset of dimensions, which we call the *change subspace*. Let $D = \{1, 2, \ldots, d\}$ be the set of dimensions and $F_i^{D'}$ be the joint distribution of F_i observed in the subspace $D' \subseteq D$ at time step i. We define the change subspace as follows:

Definition 2 (Change subspace)**.** *The change subspace D^* at time t^* is the union of all $D' \subseteq D$ in which the joint distribution $F^{D'}$ changed and which does not contain a subspace D'' for which $F_{t^*}^{D''} \neq F_{t^*+1}^{D''}$.*

If the dimensions in D^* are uncorrelated, then changes will be visible on the marginal distributions, i.e., all D' are of size 1. However, changes may only be detectable w.r.t the joint distribution of D^* or the union of its subspaces of size greater than 1, which our definition accommodates. Note that the definition can also handle multiple co-occurring changes and considers them as one single change. Last, change severity measures the difference between $F_{t^*}^{D^*}$ and $F_{t^*+1}^{D^*}$:

Definition 3 (Change severity)**.** *The severity of a change is a positive function Δ of the mismatch between $F_{t^*}^{D^*}$ and $F_{t^*+1}^{D^*}$.*

Since we do not know the true distributions F_{t^*} and F_{t^*+1}, the best we can do is detecting changes and their characteristics based on the observed data.

4 Approach

4.1 Principle of ABCD

Direct comparison of high-dimensional distributions is impractical as it requires many samples (Gretton et al, 2012). Yet the number of variables required

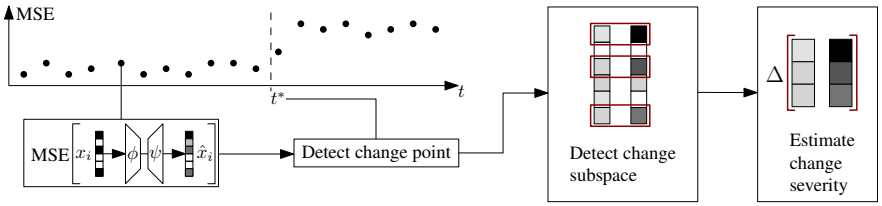


Figure 1: Overview of ABCD.

to describe such data with high accuracy is often much smaller than d (Lee and Verleysen, 2007). Dimensionality reduction techniques let us *encode* observations in fewer dimensions. The more information encodings retain, the better one can reconstruct (*decode*) the original data. However, if the distribution changes, the reconstruction will degrade and produce higher errors.

We leverage this principle in ABCD by monitoring the reconstruction loss of an encoder-decoder model $\psi \circ \phi$ for some encoder function ϕ and decoder function ψ. Figure 1 illustrates this. Specifically, we first learn $\phi : [0,1]^d \to [0,1]^{d'}$ with $d' = \lfloor \eta d \rfloor < d$, $\eta \in (1/d, 1)$, mapping the data to fewer dimensions, and $\psi : [0,1]^{d'} \to [0,1]^d$. Then, we monitor the loss between each x_t and its reconstruction $\hat{x}_t = \psi \circ \phi(x_t) = \psi(\phi(x_t))$:

$$L_t = MSE(x_t, \hat{x}_t) = \frac{1}{d}\sum_{j=1}^{d}(x_{t,j} - \hat{x}_{t,j})^2 = \frac{1}{d}\sum_{j=1}^{d} L_{t,j} \tag{1}$$

We hypothesize that distribution changes lead to outdated encoder-decoder models — see for example (Jaworski et al, 2020) for empirical evidence. Hence, we assume that changes in the reconstruction affect the *mean* μ_{t^*+1} of the loss, because the model can no longer accurately reconstruct the input:

$$F_{t^*} \neq F_{t^*+1} \implies \mu_{t^*} \neq \mu_{t^*+1} \tag{2}$$

We can now replace the definition of change in high-dimensional data with an easier-to-evaluate, univariate proxy:

$$\exists t^* \in [1, \ldots, t] : \mu_{t^*} \neq \mu_{t^*+1} \tag{3}$$

It allows detecting arbitrary changes in the original (high-dimensional) distribution as long as they affect the average reconstruction loss of the encoder-decoder. Since the true μ_{t^*} and μ_{t^*+1} are unknown, we estimate them from the stream:

$$\hat{\mu}_{1,t^*} = \frac{1}{t^*}\sum_{i=1}^{t^*} L_i, \quad \hat{\mu}_{t^*+1,t} = \frac{1}{t-t^*}\sum_{i=t^*+1}^{t} L_i. \tag{4}$$

4.2 Detecting the change point

ABCD detects a change at t^* if $\hat{\mu}_{1,t^*}$ differs *significantly* from $\hat{\mu}_{t^*+1,t}$. To quantify this, we derive a test based on Bernstein's inequality (Bernstein, 1924). It is often tighter than more general alternatives like Hoeffding's inequality (Boucheron et al, 2013). Let $\hat{\mu}_1, \hat{\mu}_2$ be the averages of two independent samples from two univariate random variables. One wants to evaluate if both random variables have the same expected values: The null hypothesis H_0 is $\mu_1 = \mu_2$. Based on the two samples, one rejects H_0 if $\Pr(|\hat{\mu}_1 - \hat{\mu}_2| \geq \epsilon) \leq \delta$ where δ is a preset significance level. The following theorem allows evaluating Equation (3) based on Bernstein's inequality.

Theorem 1 (Bound on $\Pr(|\hat{\mu}_1 - \hat{\mu}_2| \geq \epsilon)$)**.** *Given two independent samples X_1, X_2 of size n_1 and n_2 from two random variables with unknown expected values μ_1, μ_2 and variances σ_1^2, σ_2^2. Let $\hat{\mu}_1, \hat{\mu}_2$ denote the sample means and let $|\mu_1 - x_i| < M$ for all $x_i \in X_1$ and $|\mu_2 - x_i| < M$ for all $x_i \in X_2$ respectively. Assuming $\mu_1 = \mu_2$, we have:*

$$\Pr(|\hat{\mu}_1 - \hat{\mu}_2| \geq \epsilon) \leq 2\exp\left\{-\frac{n_1(\kappa\epsilon)^2}{2\left(\sigma_1^2 + \frac{1}{3}\kappa M\epsilon\right)}\right\} + 2\exp\left\{-\frac{n_2((1-\kappa)\epsilon)^2}{2\left(\sigma_2^2 + \frac{1}{3}(1-\kappa)M\epsilon\right)}\right\} \in (0,4] \quad \forall\kappa \in [0,1]. \tag{5}$$

Proof We follow the same steps as in (Bifet and Gavaldà, 2007; Pears et al, 2014).

Recall Bernstein's inequality: Let $x_1, \ldots, x_n$ be independent random variables with sample mean $\hat{\mu} = 1/n \sum x_i$ and expected value μ s.th. $\forall x_i : |x_i - \mu| \leq M$. Then, for all $\epsilon > 0$,

$$\Pr(|\hat{\mu} - \mu| \geq \epsilon) \leq 2\exp\left\{-\frac{n\epsilon^2}{2\left(\sigma^2 + \frac{1}{3}M\epsilon\right)}\right\}. \tag{6}$$

We apply the union bound to $\Pr(|\hat{\mu}_1 - \hat{\mu}_2| \geq \epsilon)$. For all $\kappa \in [0,1]$, we have:

$$\Pr(|\hat{\mu}_1 - \hat{\mu}_2| \geq \epsilon) \quad \leq \quad \Pr(|\hat{\mu}_1 - \mu_1| \geq \kappa\epsilon) \quad + \quad \Pr(|\hat{\mu}_2 - \mu_2| \geq (1-\kappa)\epsilon) \tag{7}$$

Substituting above with Bernstein's inequality completes the proof. □

With regard to change detection, one can use Equation (5) to evaluate for a time point k if a change occurred. The question is, however, how to choose ϵ to limit the probability of false alarm at any time t to a maximum δ.

Our approach is to set ϵ to the observed $|\hat{\mu}_{1,k} - \hat{\mu}_{k+1,t}|$ and to set $n_1 = k$, $n_2 = t - k$. The result bounds the probability of observing $|\hat{\mu}_{1,k} - \hat{\mu}_{k+1,t}|$ between two independent samples of sizes k and $t - k$ under H_0. If this probability is very low, the distributions must have changed at k. Then, we search for changes at multiple time points k in the current window. Hence, we obtain multiple such probability estimates; our change score is their minimum:

$$p = \min_k \Pr(|\hat{\mu}_1 - \hat{\mu}_2| \geq |\hat{\mu}_{1,k} - \hat{\mu}_{k+1,t}|) \tag{8}$$

The corresponding change point t^* splits $(L_1, L_2, \ldots, L_t)$ into the two subwindows with the statistically most different mean.

4.2.1 Choice of parameter κ

The bound in Equation (5) holds for any $\kappa \in [0, 1]$. A good choice, however, provides a tighter estimate, resulting in faster change detection for a given rate of allowed false alarms δ. (Bifet and Gavaldà, 2007) suggest to choose κ s.th. $Pr(|\hat{\mu}_1 - \mu_1| \geq \kappa\epsilon) \approx Pr(|\hat{\mu}_2 - \mu_2| \geq (1-\kappa)\epsilon)$, that approximately minimizes the upper bound. Substituting both sides with Bernstein's inequality, we get

$$\frac{n_1(\kappa\epsilon)^2}{\sigma_1^2 + \frac{\kappa M\epsilon}{3}} = \frac{n_2(1-\kappa)^2\epsilon^2}{\sigma_2^2 + \frac{(1-\kappa)M\epsilon}{3}}. \tag{9}$$

Setting $n_1 = rn_2$ and simplifying, we have

$$\frac{3\sigma_1^2 + \kappa M\epsilon}{r\kappa^2} = \frac{3\sigma_2^2 + (1-\kappa)M\epsilon}{(1-\kappa)^2}. \tag{10}$$

To solve for κ, note that $|\hat{\mu}_{1,k} - \hat{\mu}_{k+1,t}| \approx 0$ for large enough k and $t - k$ while there is no change. This leads to a change score $p \gg \delta$ for any choice of κ. Hence, choosing κ optimal is irrelevant while there is no change.

In contrast, if a change occurs, the change in the model's loss dominates the variance in both subwindows, leading to $M\epsilon \gg \sigma_1^2, \sigma_2^2$. In that case, the influence of σ_1^2, σ_2^2 is negligible for sufficiently large κ and $1 - \kappa$:

$$\frac{\kappa M\epsilon}{r\kappa^2} = \frac{(1-\kappa)M\epsilon}{(1-\kappa)^2}. \tag{11}$$

Solving Equation (11) for κ results in our recommendation for κ (Equation (12) which we restrict to $[\kappa_{min}, 1 - \kappa_{min}]$ with $\kappa_{min} = 0.05$.

$$\kappa = \frac{1}{1+r} = \frac{n_2}{n_1 + n_2} \tag{12}$$

4.2.2 Minimum sample sizes and outlier sensitivity

This section investigates the conditions under which ABCD detects changes.

We derive a minimum size of the first window above which ABCD detects a change. It bases on the fact that the number of observations before an evaluated time point k remains fixed while the number of observations after k grows with t. Those counts are $n_1 = k$ and $n_2 = t - k$ in Equation (5). Also, since we consider bounded random variables, their variance is bounded as well. Hence, the second term in Equation (5) approaches 0 for any $\epsilon > 0$. With this, solving

Equation (5) for n_1 yields:

$$n_1 \geq \left\lceil 2 \log\left(\frac{2}{\delta}\right)\left(\frac{\sigma_1^2}{(\kappa\epsilon)^2} + \frac{M}{3\kappa\epsilon}\right)\right\rceil. \tag{13}$$

By setting $\epsilon = |\hat{\mu}_1 - \hat{\mu}_2|$ we see that the required size of the first window decreases the larger the change in the average reconstruction error. For example, with $M = 1$, $\epsilon = \sigma_1 = 0.1$, and $\delta = 0.05$ our approach requires $n_1 \geq 32$.

Since ABCD detects changes in the average reconstruction loss of a bounded vector, it is stable with respect to outliers as long as they are reasonably rare. To see this, assume w.l.o.g. that window 1 contains n_{out} outliers and that $\epsilon > 0$. One can show that the average of the outliers, $\hat{\mu}_{out}$, must exceed the average of the remaining inliers, $\hat{\mu}_{in}$, by $n_1\epsilon/n_{out}$. In the example above, a single outlier would thus have to exceed $\hat{\mu}_{in}$ by $n_1\epsilon = 3.2$. This, however, is impossible because $M = 1$ bounds the reconstruction loss.

4.3 Detecting the change subspace

After detecting a change, we identify the change subspace. Restricting the encoding size to $d' < d$ forces the model to learn relationships between different input dimensions. As a result, the loss observed for dimension j contains not only information about the change in that dimension (i.e., the marginal distribution in j changes), but also about correlations influencing dimension j. Hence, we can detect changes in the marginal- and joint-distributions by evaluating in which dimensions the loss changed the most.

Algorithm 1 describes how we identify change subspaces. For each dimension j, we compute the average reconstruction loss (the squared error in dimension j) before and after t^*, denoted $\hat{\mu}^j_{1,t^*}, \hat{\mu}^j_{t^*+1,t}$ (lines 5 and 6), and the standard deviation $\sigma^j_{1,t^*}, \sigma^j_{t^*+1,t}$ (lines 6 and 7). We then evaluate Equation (5), returning an upper bound on the p-value in the range $(0, 4]$ for dimension j (line 9). If $p_j < \tau \in [0, 4]$, an external parameter for which we give a recommendation later on, we add j to the change subspace (lines 10 and 11).

4.4 Quantifying change severity

ABCD provides a measure of change severity in the affected subspace, based on the assumption that the loss in the change subspace increases with severity. Hence, we compute the average reconstruction loss observed in D^* before and after the change,

$$\hat{\mu}^{D^*}_{1,t^*} = \frac{1}{|D^*|t^*} \sum_{i=1}^{t^*} \sum_{j \in D^*} L_{i,j}, \quad \hat{\mu}^{D^*}_{t^*+1,t} = \frac{1}{|D^*|(t-t^*)} \sum_{i=t^*+1}^{t} \sum_{j \in D^*} L_{i,j} \tag{14}$$

Algorithm 1 Identification of change subspaces.

Require: $(x_1, \hat{x}_1), \ldots, (x_t, \hat{x}_t)$, t^*

procedure SUBSPACE
  $D^* \leftarrow \emptyset$
  for all $j \in 1, \ldots, d$ **do**
    $s \leftarrow \left((x_{i,j} - \hat{x}_{i,j})^2 \ \forall i \in 1, \ldots, t\right)$
    $\hat{\mu}^j_{1,t^*} = \frac{1}{t^*} \sum_{i=1}^{t^*} s_i, \quad \hat{\mu}^j_{t^*+1,t} = \frac{1}{t-t^*} \sum_{i=t^*+1}^{t} s_i$
    $\sigma^j_{1,t^*} = \sqrt{\frac{1}{t^*} \sum_{i=1}^{t^*} \left(s_i - \hat{\mu}^j_{1,t^*}\right)^2}$
    $\sigma^j_{t^*+1,t} = \sqrt{\frac{1}{t-t^*} \sum_{i=t^*+1}^{t} \left(s_i - \hat{\mu}^j_{t^*+1,t}\right)^2}$
    $p_j \leftarrow$ Evaluate Equation (5) ▷ Bernstein score
    if $p_j < \tau$ **then**
      $D^* \leftarrow D^* \cup \{j\}$
  Return D^*

and the standard deviation observed before the change:

$$\sigma^{D^*}_{1,t^*} = \sqrt{\frac{1}{t^*} \sum_{i=1}^{t^*} \left(\hat{\mu}^{D^*}_i - \hat{\mu}^{D^*}_{1,t^*}\right)^2} \text{ with } \hat{\mu}^{D^*}_i = \frac{1}{|D^*|} \sum_{j \in D^*} L_{i,j} \tag{15}$$

We then standard-normalize the average reconstruction loss $\hat{\mu}^{D^*}_{t^*+1}$ observed after the change:

$$\Delta = \frac{\left|\hat{\mu}^{D^*}_{t^*+1,t} - \hat{\mu}^{D^*}_{1,t^*}\right|}{\sigma^{D^*}_{1,t^*}} \in \mathbb{R}^+ \tag{16}$$

Intuitively, Δ is the standard deviation of model's loss on the new distribution.

4.5 Working with windows

In comparison to most approaches, ABCD evaluates multiple possible change points within an adaptive time interval $[1, \ldots, t]$. This frees the user from choosing the window size a-priori and allows to detect changes at variable time scales. Next, we discuss how to efficiently evaluate those time points.

4.5.1 Maintaining loss statistics online

To avoid recomputing average reconstruction loss values and their variance for multiple time points every time new observations arrive, we store Welford aggregates $A_{1,k}$ summarizing the stream in the interval $[1, \ldots, k]$. Each aggregate $A_{1,k}$ is a tuple containing the average reconstruction loss $\hat{\mu}_{1,k}$ and the sum of squared differences $ssd_{1,k} = k^{-1} \sum_{j=1}^{k} L_j$. We store these aggregates for the time interval $[1, \ldots, t]$.

Creating a new aggregate. Every time a new observation with loss L_t arrives, we create a new aggregate based on the previous aggregate $A_{1,t-1} = (\hat{\mu}_{1,t-1}, ssd_{1,t-1})$ in $\mathcal{O}(1)$ using Welford's algorithm (Knuth, 1997):

$$\hat{\mu}_{1,t} = \hat{\mu}_{1,t-1} + \frac{1}{t}(L_t - \hat{\mu}_{1,t-1}) \tag{17}$$

$$ssd_{1,t} = ssd_{1,t-1} + (L_t - \hat{\mu}_{1,t-1})\,(L_t - \hat{\mu}_{1,t}) \tag{18}$$

Computing the statistics. Two aggregates $A_{1,k}$ and $A_{1,t}$, $t > k$ overlap in the time interval $[1, \ldots, k]$. We leverage this overlap to derive an aggregate $A_{k+1,t} = (\hat{\mu}_{k+1,t}, ssd_{k+1,t})$ representing the time interval $[k+1, \ldots, t]$. Equation (19) and Equation (20) are based on Chan's method for combining variance estimates of non-overlapping samples (Chan et al, 1982).

$$\hat{\mu}_{k+1,t} = \frac{1}{t-k}(t\hat{\mu}_{1,t} - k\hat{\mu}_{1,k}) \tag{19}$$

$$ssd_{k+1,t} = ssd_{1,t} - ssd_{1,k} - \frac{k(t-k)}{t}\,(\hat{\mu}_{1,k} - \hat{\mu}_{k+1,t})^2 \tag{20}$$

From $ssd_{1,k}$ and $ssd_{k+1,t}$ we can compute the sample variances as follows:

$$\sigma^2_{1,k} = \frac{ssd_{1,k}}{k-1}, \quad \sigma^2_{k+1,t} = \frac{ssd_{k+1,t}}{t-k-1} \tag{21}$$

Derivation. Given two non-overlapping samples $A = \{x_1, \ldots, x_m\}$ and $B = \{x_1, \ldots, x_n\}$ of a real random variable. Let $T_A = \sum_{i=1}^{m} x_i$ and $T_B = \sum_{i=1}^{n} x_i$ be the sums of the samples and $ssd_A = \sum_{i=1}^{m}(x_i - m^{-1}T_A)^2$ and $ssd_B = \sum_{i=1}^{n}(x_i - n^{-1}T_B)^2$ be the sums of squared distances from the mean.

For the union of both sets $AB = A \cup B$ we have $T_{AB} = T_A + T_B$, which is equivalent to $(m+n)\hat{\mu}_{AB} = m\hat{\mu}_A + n\hat{\mu}_B$. Solving for $\hat{\mu}_B$ gives

$$\hat{\mu}_B = \frac{m+n}{n}\hat{\mu}_{AB} - \frac{m}{n}\hat{\mu}_A. \tag{22}$$

Substituting $n = t-k$, $m = k$, $\hat{\mu}_A = \hat{\mu}_{1,k}$, $\hat{\mu}_B = \hat{\mu}_{k+1,t}$, and $\hat{\mu}_{1,t} = \hat{\mu}_{AB}$ gives Equation (19); next we derive Equation (20). Chan et al (1982) state:

$$ssd_{AB} = ssd_A + ssd_B + \frac{m}{n(m+n)}\left(\frac{n}{m}T_A - T_B\right)^2, \tag{23}$$

which is equivalent to

$$ssd_{AB} = ssd_A + ssd_B + \frac{m}{n(m+n)}\left(n\underbrace{\left(\frac{1}{m}T_A - \frac{1}{n}T_B\right)}_{=\hat{\mu}_A - \hat{\mu}_B}\right)^2. \tag{24}$$

Solving for ssd_B, applying the former substitutions, and setting $ssd_A = ssd_{1,k}$, $ssd_B = ssd_{k+1,t}$, and $ssd_{1,t} = ssd_{AB}$ results in Equation (20).

4.6 Implementation

Algorithm

One can implement ABCD as a recursive algorithm, see Algorithm 2, which restarts every time a change occurs. We keep a data structure W that contains the aggregates, instances, and reconstructions. W can either be empty, or, in the case of a recursive execution, already contain data from the previous run.

Prior to execution, our algorithm must first obtain a model of the current data from an initial sample of size n_{min}. If necessary, ABCD allows enough instances to arrive (lines 5–7). Larger choices of n_{min} allow for better approximations of the current distribution but delay change detection. Hence our recommendation is to set n_{min} as small as possible to still learn the current distribution; a default of $n_{min} = 100$ has worked well for us.

Afterwards, the algorithm trains the model using the instances in W (lines 8–9). ABCD can in principle work with various encoder-decoder models; thus we deal with tuning the model only on a high level. Nonetheless, we give recommendations in our sensitivity study later on.

After model training, ABCD detects changes. It reconstructs each new observation x_{t+1} (line 11), creates a new aggregate $A_{1,t+1}$ (line 12), and adds $w_{t+1} \coloneqq (A_{1,t+1}, \hat{x}_{t+1}, x_{t+1})$ to W (lines 13–14). Our approach then computes change score p and change point t^* (lines 15–16). If $p < \delta$, it detects a change.

Once ABCD detects a change, it identifies the corresponding subspace and evaluates its severity (lines 21–22). Then it adapts W by dropping the outdated part of the window (line 23), including all information obtained with the outdated model. At last, we restart ABCD with the adapted window (line 24).

Discussion

In the worst case our approach consumes linear time and memory because W grows linearly with t. However, we can simply restrict the size of W to n_{max} items for constant memory or evaluate only k_{max} window splits for constant runtime. In the latter case we split W at every t/k_{max}th time point. Regarding n_{max}, it is beneficial that the remaining aggregates still contain information about all observations in $(1, \ldots, t)$. Hence, ABCD considers the *entire* past since the last change even though one restricts the size of W.

ABCD can work with any encoder-decoder model, such as deep neural networks. However, handling a high influx of new observations faster than the model's processing capability can be challenging. Assuming that $\psi \circ \phi \in \mathcal{O}(g(d))$ for some function g of dimensionality d, the processing time of a single instance during serial execution is in $\mathcal{O}\left(g(d) + k_{max}\right)$. Nevertheless, both the deep architecture components and the computation of the change score (cf. Equation 8) can be executed in parallel using specialized hardware.

Algorithm 2 Adaptive Bernstein Change Detector (ABCD)

Require: The model $\psi \circ \phi$, threshold δ, threshold τ

procedure ABCD(W)
  $\psi \circ \phi \leftarrow Null$; $t \leftarrow |W|$ ▷ Model not yet trained
  while new instance x_{t+1} **do**
    if $t < n_{min}$ **then** ▷ Warm up
      $w_{t+1} \leftarrow (-, -, x_{t+1})$
      $W \leftarrow W \mid\mid w_{t+1}$
    else if $\psi \circ \phi = Null$ **then**
      $\psi \circ \phi \leftarrow$ TrainModel(W)
    else
      $\hat{x}_{t+1} \leftarrow \psi(\phi(x_{t+1}))$ ▷ Reconstruct
      $A_{t+1} \leftarrow$ update aggregate A_t with L_{t+1}
      $W \leftarrow W \mid\mid (A_{t+1}, \hat{x}_{t+1}, x_{t+1})$
      $p \leftarrow$ Equation (8) ▷ Bernstein score
      $t^* \leftarrow \text{argmin}_k$ of Equation (8)
      if $p < \delta$ **then** ▷ A change occurred
        $D^* \leftarrow$ Subspace(W, t^*, τ)
        $\Delta \leftarrow$ Severity(W, t^*, D^*)
        $W \leftarrow \{(-, -, x_i)\ \forall w_i \in W : i > t^*\}$
        ABCD(W) ▷ Restart

Dimensionality reduction techniques are often already present in data stream mining pipelines, for example as a preprocessing step to improve the accuracy of a classifier (Yan et al, 2006). Reusing an existing dimensionality reduction model makes it is easy to integrate ABCD into an existing pipeline.

Bernstein's inequality holds for zero-centered bounded random variables that take absolute values of at maximum M almost surely. While $M = 1$ serves as a theoretical upper limit of the zero-centered reconstruction error $L_t - \mathbb{E}[L_t]$ for $x_t \in [0, 1]^d$, we observe that this theoretical limit is very conservative in practice (cf. Appendix A.3). In fact, observing an error of 1 corresponds to an instance and reconstruction of $x = [0]^d$ and $\hat{x} = [1]^d$. This leads us to use $M = 0.1$ in our experiments.

5 Experiments

This section describes our experiments and results. We first describe the experimental setting (Section 5.1). Then we analyze ABCD's change detection performance (Section 5.3), its ability to find change subspaces and quantify change severity (Section 5.4), and its parameter sensitivity (Section 5.5).

5.1 Algorithms

We evaluate ABCD with different encoder-decoder models: (1) Principal Component Analysis (PCA) ($d' = \eta d$), (2) Kernel-PCA ($d' = \eta d$, RBF-kernel), and

Table 2: Evaluated approaches and their parameters.

Algorithm	Parameter	Values
ABCD	model	PCA, Kernel PCA, Autoencoders
	δ	0.05
	η	$0, 3, 0.5, 0.7$
	$E^{\dagger}$	$20, 50, 100$
	n_{min}; k_{max}; τ	100; 20; 2.5
AdwinK	k	$0.01^{*}, 0.05^{*}, 0.1^{*}, 0.2^{*}, 0.3^{*}, 0.4^{*}, 0.5^{*}$
	δ	0.05
D3	ω	$100^{*}, 250^{*}, 500^{*}$
	ρ	$0.1^{*}, 0.2^{*}, 0.3^{*}, 0.4^{*}, 0.5^{*}$
	τ	$0.6^{*}, 0.7^{*}, 0.8^{*}, 0.9^{*}$
	model	Logistic Regression*, Decision Tree
	tree depth	1, 3, 5
IBDD	ω	$100, 200, 300$
	m	$10, 20, 50, 100$
IKS	W	$100^{*}, 200, 500^{*}$
	δ	0.05
WATCH‡	ω	$500, 1000$
	κ	100
	ϵ	$2, 3$
	μ	$1000, 2000$

* used or recommended in the respective papers
† only relevant for autoencoders
‡ authors did not recommend parameters for their approach

(3) a standard fully-connected autoencoder model with one hidden ReLU layer ($d' = \eta d$) and an output layer with sigmoid activation. For (1) and (2), we rely on the default scikit-learn implementations. We implement the autoencoder (3) in pytorch and train it through gradient descent using E epochs and an Adam optimizer with default parameters according to Kingma and Ba (2015); see Appendix A.1 for pseudocode of the autoencoder training procedure.

We compare ABCD with AdwinK, IKS, IBDD, WATCH, and D3 (c.f. Section 2). We evaluate for each approach a large grid of parameters, shown in Table 2. Whenever possible, the evaluated grids of hyperparameters for competitors base on recommendations in respective papers. Otherwise, we choose them based on preliminary experiments. For ABCD, we evaluate larger and smaller values for δ, η and E to observe our approach's sensitivity to those parameters. The choice of $\tau = 2.5$ is our recommended default based on our sensitivity study in Section 5.5. Last, we set $n_{min} = 100$ and $k_{max} = 20$, minimum values that have worked well in preliminary experiments.

5.2 Datasets

There are not many public benchmark data streams for change detection. Thus we generate our own from seven real-world (rw) and synthetic (syn) classification datasets, similar to (Faber et al, 2021; Faithfull et al, 2019). We simulate changing data streams[2] by sorting the data by label, unless stated otherwise. If the label changes, a change has occurred. In real-world data streams, the number of observations between changes depends on each dataset, reported below. In the synthetic streams, we introduce changes every 2000 observations, which is a relatively large interval, to assess whether some approaches generate many false alarms. The generators base on the following datasets:

- **HAR (rw):** The dataset *Human Activity Recognition with Smartphones* (Anguita et al, 2013) ($d = 561$) bases on smartphone accelerometer and gyroscope readings for different actions a person performs. A change occurs on average every 1768 observations.
- **GAS (rw):** This data set (Vergara et al, 2011) ($d = 128$) contains data from 16 sensors exposed to 6 gases at various concentrations. A change occurs on average every 2265 observations.
- **LED (syn):** The LED generator samples instances representing a digit on a seven segment display. It contains 17 additional random dimensions. We add changes by varying the probability of bit-flipping in the relevant dimensions.
- **RBF (syn):** The RBF generator (Bifet et al, 2010) starts by drawing a fixed number of centroids. For each new instance, the generator chooses a centroid at random and adds Gaussian noise. To create changes, we increment the seed of the generator resulting in different centroids. We then use samples from the new generator in a subspace of random size.
- **MNIST, FMNIST, and CIFAR (syn):** Those data generators sample from the image recognition datasets MNIST (LeCun et al, 1998), Fashion MNIST (FMNIST) (Xiao et al, 2017) ($d = 784$), and CIFAR (Krizhevsky et al, 2009) ($d = 1024$, grayscale).

Changes can occur rapidly ("abrupt" or "sudden") or in time intervals ("gradual" or "incremental"). The shorter the interval, the more sudden the change. We vary the interval size between 1 and 300 unless stated otherwise. Real-world and image data do not have a ground truth for change subspaces and severity. Thus we generate three additional data streams:

- **HSphere (syn):** This generator draws from a d^*-dimensional hypersphere bound to $[0, 1]$ and adds $d - d^*$ random dimensions. We vary the radius and center of the hypersphere to introduce changes. The change subspace contains those dimensions that define the hypersphere.
- **Normal-M/V (syn):** These generators sample from a d^*-dimensional normal distribution and add $d - d^*$ random dimensions. For type **M**, changes affect the distribution's mean, for **V** we change the distribution's variance.

[2] Available at https://github.com/heymarco/AdaptiveBernsteinChangeDetector

5.3 Change point detection

We use precision, recall, and F1-score to evaluate the performance of the approaches at detecting changes. We define true positives (TP), false positives (FP) and false negatives (FN) as follows:

- **TP:** A change was detected before the next change.
- **FN:** A change was not detected before the next change.
- **FP:** A change was detected although no change occurred.

Also, we report the mean time until detection (MTD) indicating the average number of instances until a change is detected.

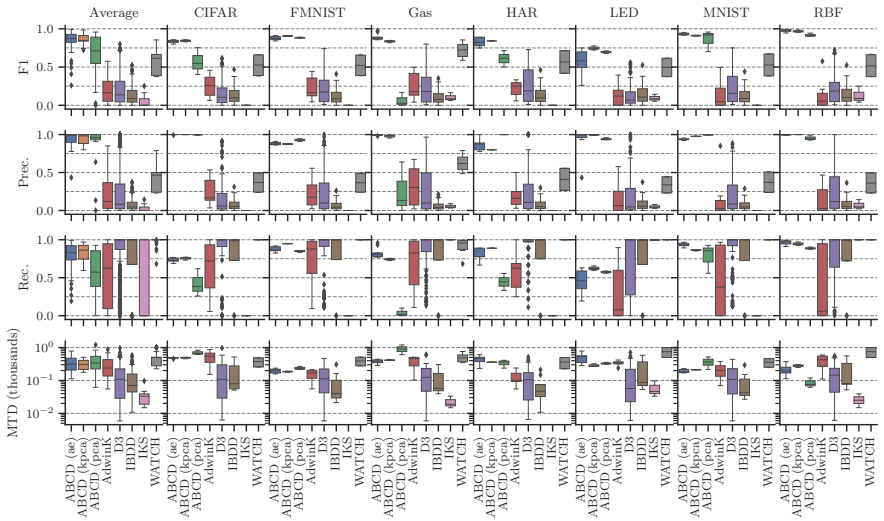


Figure 2: Change Point Detection: Results for different algorithms and datasets; each box contains the results for the evaluated grid of parameters.

Figure 2 shows F1-score, precision, recall, and MTD for all datasets and algorithm, as well as a column "Average" that summarizes across datasets. Each box contains the results for the grid of hyperparameters shown in Table 2. We see that our approach outperforms its competitors w.r.t. F1-score and precision. It also is competitive in terms of recall, though it loses against IKS, IBDD, and WATCH. These approaches seem overly sensitive. The results also indicate that ABCD works well for a wide range of hyperparameters. One reason is that ABCD uses adaptive windows, thereby eliminating the effect of a window size parameter (demonstrated in Section 5.6). Another reason is that ABCD detects changes in reconstruction loss irrespective of the actual quality of the reconstructions. For instance, Kernel PCA and PCA produce reconstructions of different accuracy in our experiments. However, for both models, the average accuracy changes when the stream changes, which is what our algorithm

Table 3: Results of approaches with their best hyperparameter configuration w.r.t. F1 score averaged over all data sets.

Approach	F1	Prec.	Rec.	MTD
ABCD (ae)	0.90	0.96	0.87	250
ABCD (kpca)	0.88	0.95	0.84	312
ABCD (pca)	0.73	0.93	0.65	442
AdwinK	0.46	0.48	0.57	400
D3	0.70	0.63	0.82	251
IBDD	0.45	0.30	0.97	396
IKS	0.08	0.04	0.43	24
WATCH	0.69	0.54	1.00	626

detects. Refer to Appendix A.3 for an illustration of the models' reconstruction loss over time. Hence, our reported results do not yield information about the actual accuracy of the underlying encoder-decoder models.

ABCD has a higher MTD than D3, IBDD, and IKS, i.e., it requires more data to detect changes. However, those competitors are much less conservative and detect many more changes than exist in the data. Hence they have low precision but high recall — this leads to a lower MTD.

Table 3 reports the results of all approaches with their best hyperparameters. WATCH and D3 achieve relatively high F1-score and precision. In fact, those approaches are our strongest competitors although we still outperform them by at least 3 %. Further, WATCH has an MTD of 626, which is more than ABCD while D3 and ABCD have a comparable MTD.

ABCD has much higher precision than its competitors. We assume this is because ABCD (1) leverages the relationships between dimensions, in comparison to AdwinK, IKS, or IBDD, and (2) learns those relationships more effectively than, say, D3 or WATCH. For example, we observed in our experiments that WATCH was frequently unable to accurately approximate the Wasserstein distance in high-dimensional data.

ABCD has lower recall than most competitors, partly due to their oversensitivity. In this regard, our approach might benefit from application-specific encoder-decoder models that leverage structure in the data, such as spacial relationships between the pixels of an image, more effectively.

5.4 Change subspace and severity

We now evaluate change subspace identification and change severity estimation. We set $d = \{24, 100, 500\}$ and vary the change subspace size d^* randomly in $[1, d]$ (except for LED, here the subspace always contains dimensions 1–7). We set the ground truth for the severity to the absolute difference between the parameters that define the concepts, e.g., the hypersphere-radius in HAR before and after the change. We report an approach's subspace detection accuracy (SAcc.), where true positives (true negatives) represent those dimensions that were correctly classified as being member (not being member) of the change subspace. We use Spearman's correlation between the detected severity and the ground truth. We also report the F1-score for detecting change points.

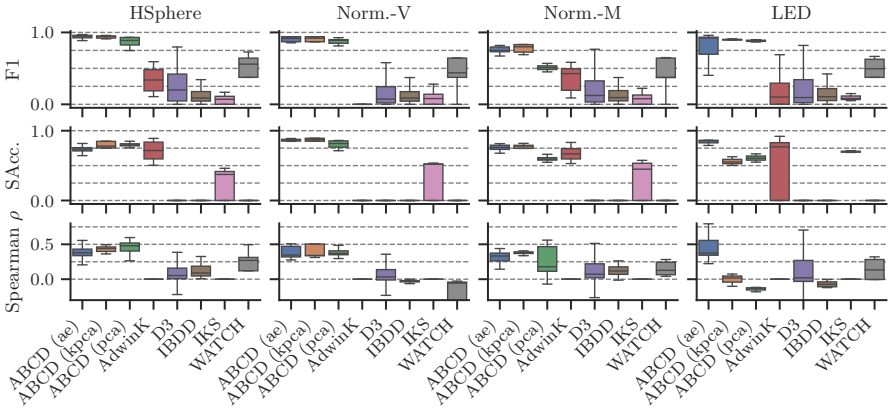


Figure 3: Results for evaluating change subspace and severity.

Figure 3 shows our results. As before, each box summarizes the results for the grid of evaluated hyperparameters. Comparing the two approaches, AdwinK and IKS, that monitor each dimension separately, we see that the former can only detect changes that affect the mean of the marginal distributions (i.e., on Norm-M, LED). At the same time, the latter can also detect other changes (e.g., changes in variance). This is expected since AdwinK compares the mean in two windows while IKS compares the empirical distributions.

Regarding subspace detection, our approach achieves an accuracy of 0.72 for PCA, 0.78 for autoencoders, and 0.79 for Kernel PCA. AdwinK performs similarly well when changes affect the mean of the marginal distributions. Except on LED, IKS performs worse than ABCD and AdwinK, presumably because IKS issues an alarm as soon as a single dimension changed.

The estimates of our approach correlate more strongly with the ground truth than those of competitors, with an average of 0.31 for PCA, 0.36 for Kernel PCA and 0.37 for Autoencoders. However, we expect more specialized models to better than our tested models. On LED, PCA-based models appear to struggle to separate patterns from noise, resulting in poor noise level estimates and low correlation scores.

5.5 Parameter sensitivity of ABCD

Sensitivity to η

Figure 4a plots F1 for different datasets over η. We observe that the size of the bottleneck does not significantly impact the change detection performance of ABCD (ae) and ABCD (kpca). For PCA, however, too large bottlenecks seem to inhibit change detection on CIFAR, Gas, and MNIST. For those datasets, we assume that the change occurs along the retained main components, rendering it undetectable; see Appendix A.2 for an illustration. Figure 4b shows the subspace detection accuracy and Spearman's ρ. The influence of η on both

metrics is low. As mentioned earlier, we assume that a *change* in reconstruction loss, rather than the quality of reconstruction itself, is crucial for ABCD. An exception is the LED dataset, on which PCA and Kernel-PCA are unable to provide a measure that positively correlates with change severity. We hypothesize that those methods struggle to separate patterns from noise, resulting in poor noise level estimates and low correlation scores.

Sensitivity to E

Figure 4c plots our approach's performance for different choices of E. Overall, our approach seems to be robust to the choice of E. On LED, however, larger choices of E lead to substantial improvements in F1-score. The reason may be that the autoencoder does not converge to a proper representation of the data for small E. To avoid this, we recommend choosing $E \geq 50$ and to increase the value if one observes that the model has not yet converged sufficiently.

Sensitivity to τ

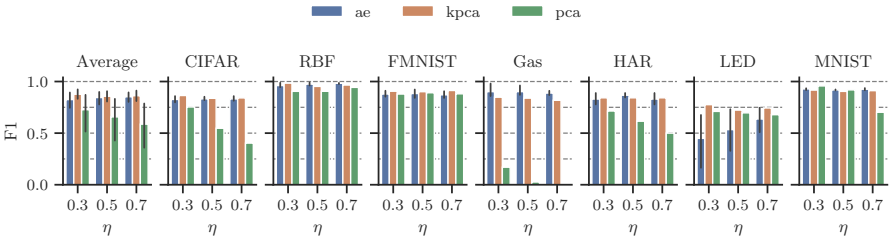
Figure 4d investigates how the choice of τ affects the performance of ABCD at detecting subspaces. Since the change score in Equation (5) provides an upper bound on the probability that a change occurred, the function can return values greater than 1, i.e,. in the range $(0, 4]$. Hence we vary τ in that range and record the obtained subspace detection accuracy. For all approaches we achieve optimal accuracy at $\tau \approx 2.5$. This is probably because some dimensions could change more severely than others, resulting in variations of the change scores observed in the different dimensions of the change subspace. Based on our findings we recommend $\tau = 2.5$ as default.

5.6 Ablation study on window types

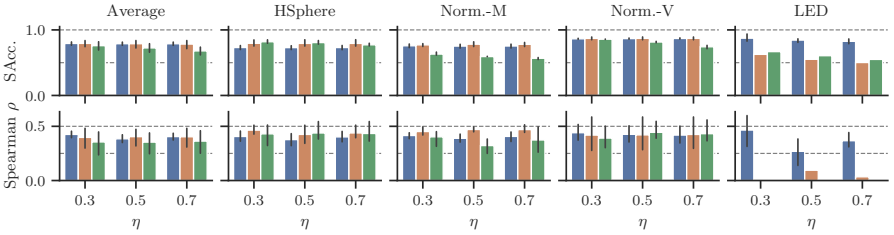
Next, we investigate the effect of different window types on change detection performance. We evaluate those commonly found in change detection literature (and in our competitors) and couple them with encoder-decoder models and the probability bound in Equation (5). In particular, we compare: (1) Adaptive windows (AW), as in ADWIN, AdwinK, and our approach, (2) fixed reference windows (RW), as in IKS, (3) sliding windows (SW), as in WATCH, and (4) jumping windows (JW), as in D3. The latter "jump" every $\rho|W|$ instances.

We evaluate the hyperparameters mentioned in Table 2. For example, because D3 uses jumping windows, we include the evaluated hyperparameters for D3 in our evaluation of jumping windows. In addition, we extend the grid with other reasonable choices since we already preselected those in Table 2 for our competitors in a preliminary study. For ABCD we use $\eta = 0.5$ and $E = 50$.

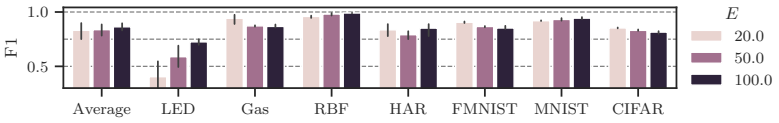
Table 4 reports the average over all hyperparameter combinations. AWs yield higher F1-score and recall than other techniques, while precision remains high (≥ 0.95). SWs have a lower MTD than AWs and hence seem to require a fewer instances until they detect a change. This is expected: in contrast to sliding windows, adaptive windows allow the detection of even slight changes after a longer period of time, resulting in both higher MTD and recall.



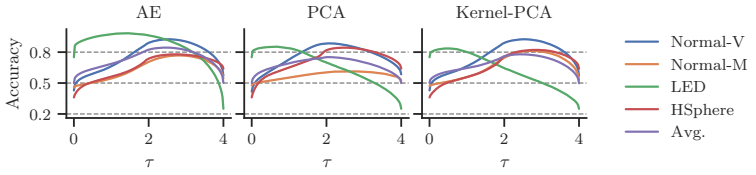
(a) Influence of η on the identification of changes.



(b) Influence of η on the estimation of change subspaces and severity.



(c) Change detection performance of ABCD (ae) depending on E.



(d) Subspace detection accuracy of ABCD depending on τ.

Figure 4: Sensitivity of our approach to its hyperparameters.

5.7 Runtime analysis

5.7.1 Comparison with competitors

Figure 5a shows the mean time per observation (MTPO) of ABCD and its competitors for $d \in \{10, 100, 1000, 10,000\}$ running single-threaded. The results are averaged over all evaluated parameters (Table 2). ABCD (id) replaces the encoder-decoder model with the identity which does not cause overhead. This allows measuring how much the encoder-decoder model influences ABCD's

Table 4: Ablation: Using encoder-decoder models with different window types.

Model	Window	F1	Prec.	Rec.	MTD
AE	AW	**0.83**	0.95	**0.78**	455.6
	RW	0.53	**1.00**	0.21	403.6
	SW	0.62	**1.00**	0.40	**207.2**
	JW	0.52	0.79	0.46	239.1
KPCA	AW	**0.83**	0.99	**0.75**	309.0
	RW	0.56	**1.00**	0.23	456.3
	SW	0.68	**1.00**	0.49	**202.8**
	JW	0.50	0.77	0.33	266.2
PCA	AW	**0.72**	0.98	**0.55**	355.3
	RW	0.36	**1.00**	0.09	400.0
	SW	0.53	**1.00**	0.33	**206.7**
	JW	0.46	0.75	0.20	239.9

runtime. The results confirm that the runtime of ABCD alone, i.e, without the encoding-decoding-process, remains unaffected by a stream's dimensionality.

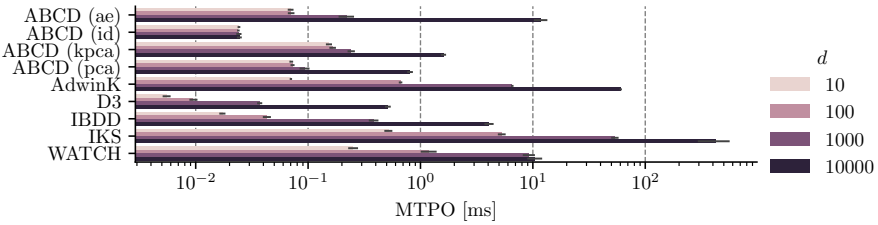
We observe that our approach is able to process around 10,000 observations per second for $d \leq 100$. This is more than IKS, WATCH and AdwinK (except at $d = 10$) but slower than D3 and IBDD. The reason is that our approach evaluates k_{max} possible change points in each time step. In high-dimensional data, our competitors' MTPO grows faster than ABCD with PCA or KPCA; in fact, ABCD (pca) is second fastest after D3 for $d \geq 1000$. An exception is WATCH at $d = 10000$. This is due to an iteration cap for approximating the Wasserstein distance restricting the approach's MTPO.

5.7.2 Runtime depending on window size

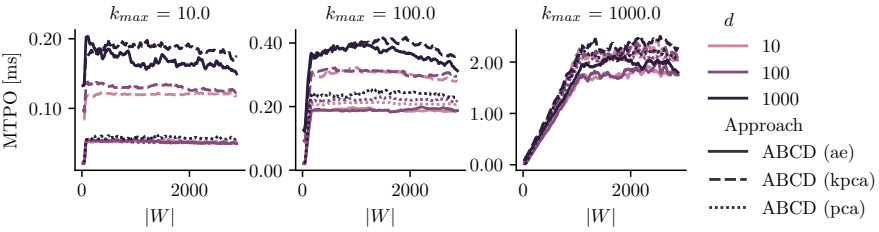
Next, we investigate ABCD's runtime for different choices of k_{max} and η. We run this experiment on a single CPU thread. For all three evaluated models, the encoding-decoding of an observation has a time complexity of $\mathcal{O}(\eta d^2)$; hence, ABCD's processing time of one instance is in $\mathcal{O}(\eta d^2 + k_{max})$. We therefore expect a quadratic increase in execution time with dimensionality and a linear increase with η and k_{max} when running on a single core.

The results in Figure 5b show the influence of k_{max} on the execution time: k_{max} effectively restricts the MTPO as soon as $|W| = k_{max}$. Afterwards, MTPO remains unaffected by $|W|$. This also confirms that one can evaluate different possible change points in constant time using the proposed aggregates.

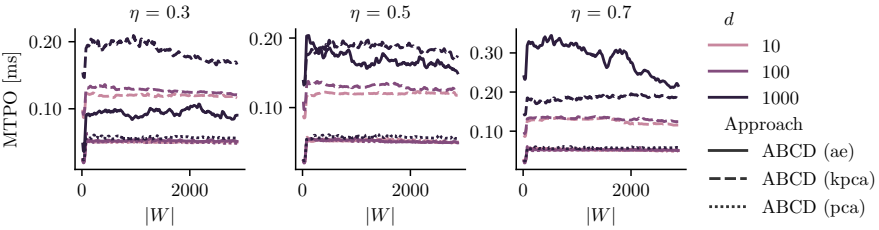
We show the runtime for different choices of bottleneck-size η in Figure 5c. η has little influence on the runtime of ABCD with PCA and Kernel-PCA . However, coupled with an autoencoder (implemented in pytorch) we observe the expected linear increase in execution time from 0.1 ms for $\eta = 0.3$ to 0.3 ms for $\eta = 0.7$. Considering that change detection performance has shown to remain stable even for smaller choices of η, we recommend $\eta \leq 0.5$ as default.



(a) Mean time per observation in milliseconds.



(b) MTPO of ABCD over $|W|$ using $E = 50$ and $\eta = 0.5$ varying k_{max}.



(c) MTPO of ABCD over $|W|$ using $E = 50$ and $k_{max} = 100$ varying η.

Figure 5: Runtime analysis of ABCD.

6 Conclusion

We presented a change detector for high-dimensional data streams, called ABCD, that monitors the reconstruction loss of an encoder-decoder-model in an adaptive window with a change score based on Bernstein's inequality. Our approach identifies changes and change subspaces, and provides a severity measure that correlates with the ground truth. Since encoder-decoder models are already used in many domains (Rani et al, 2022), our approach is widely applicable. In the future, it would thus be interesting to test ABCD with application or data specific encoder-decoder models. For example, one might observe even better performance on streams of image data when applying convolutional autoencoders. Last, ABCD could also benefit from a theoretical analysis of the relationship between changes in data distribution and the loss of different encoder-decoder models.

References

Anguita D, Ghio A, Oneto L, et al (2013) A public domain dataset for human activity recognition using smartphones. In: ESANN, URL https://www.esann.org/sites/default/files/proceedings/legacy/es2013-84.pdf

Bai J, Perron P (2003) Critical values for multiple structural change tests. The Econometrics Journal 6(1):72–78. https://doi.org/https://doi.org/10.1111/1368-423X.00102

Bernstein SN (1924) On a modification of Chebyshev's inequality and of the error formula of Laplace. Ann. Sci. Inst. Savantes Ukraine, Sect. Math.

Bifet A, Gavaldà R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the Seventh SIAM International Conference on Data Mining. SIAM, pp 443–448, https://doi.org/10.1137/1.9781611972771.42

Bifet A, Holmes G, Pfahringer B (2010) Leveraging bagging for evolving data streams. In: ECML PKDD, Lecture Notes in Computer Science, vol 6321. Springer, pp 135–150, https://doi.org/10.1007/978-3-642-15880-3_15

Boucheron S, Lugosi G, Massart P (2013) Concentration Inequalities: A Nonasymptotic Theory of Independence. Oxford University Press, https://doi.org/10.1093/acprof:oso/9780199535255.001.0001

Ceci M, Corizzo R, Japkowicz N, et al (2020) ECHAD: Embedding-based change detection from multivariate time series in smart grids. IEEE Access 8:156,053–156,066. https://doi.org/10.1109/ACCESS.2020.3019095

Chakar S, Lebarbier E, Lévy-Leduc C, et al (2017) A robust approach for estimating change-points in the mean of an AR(1) process. Bernoulli 23(2):1408 – 1447. https://doi.org/10.3150/15-BEJ782

Chan TF, Golub GH, LeVeque RJ (1982) Updating formulae and a pairwise algorithm for computing sample variances. Tech. rep., Heidelberg

Chaudhuri A, Fellouris G, Tajer A (2021) Sequential change detection of a correlation structure under a sampling constraint. In: ISIT, pp 605–610, https://doi.org/10.1109/ISIT45174.2021.9517736

Dasu T, Krishnan S, Venkatasubramanian S, et al (2006) An information-theoretic approach to detecting changes in multi-dimensional data streams. In: Proc. Symposium on the Interface of Statistics, Computing Science, and Applications (Interface)

Faber K, Corizzo R, Sniezynski B, et al (2021) WATCH: Wasserstein change point detection for high-dimensional time series data. In: Big Data. IEEE,

pp 4450–4459, https://doi.org/10.1109/BigData52589.2021.9671962

Faithfull WJ, Diez JJR, Kuncheva LI (2019) Combining univariate approaches for ensemble change detection in multivariate data. Inf Fusion 45:202–214. https://doi.org/10.1016/j.inffus.2018.02.003

Fouché E, Komiyama J, Böhm K (2019) Scaling multi-armed bandit algorithms. In: SIGKDD. ACM, pp 1449–1459, https://doi.org/10.1145/3292500.3330862

Garreau D, Arlot S (2018) Consistent change-point detection with kernels. Electronic Journal of Statistics 12(2):4440–4486. URL https://hal.science/hal-01416704

Goldenberg I, Webb GI (2019) Survey of distance measures for quantifying concept drift and shift in numeric data. Knowl Inf Syst 60(2):591–615. https://doi.org/10.1007/s10115-018-1257-z

Gözüaçık O, Büyükçakır A, Bonab H, et al (2019) Unsupervised concept drift detection with a discriminative classifier. In: CIKM. ACM, p 2365–2368, https://doi.org/10.1145/3357384.3358144

Gretton A, Borgwardt KM, Rasch MJ, et al (2012) A kernel two-sample test. J Mach Learn Res 13:723–773. https://doi.org/10.5555/2503308.2188410

Harchaoui Z, Cappe O (2007) Retrospective mutiple change-point estimation with kernels. In: IEEE/SP 14th Workshop on Statistical Signal Processing, pp 768–772, https://doi.org/10.1109/SSP.2007.4301363

Impedovo A, Ceci M, Calders T (2019) Efficient and accurate non-exhaustive pattern-based change detection in dynamic networks. Lecture notes in computer science, vol 11828. Springer, pp 396–411, https://doi.org/10.1007/978-3-030-33778-0_30

Impedovo A, Loglisci C, Ceci M, et al (2020a) Condensed representations of changes in dynamic graphs through emerging subgraph mining. Engineering Applications of Artificial Intelligence 94:103,830. https://doi.org/10.1016/j.engappai.2020.103830

Impedovo A, Mignone P, Loglisci C, et al (2020b) Simultaneous process drift detection and characterization with pattern-based change detectors. Lecture notes in computer science, vol 12323. Springer, pp 451–467, https://doi.org/10.1007/978-3-030-61527-7_30

Iwashita AS, Papa JP (2019) An overview on concept drift learning. IEEE Access 7:1532–1547. https://doi.org/10.1109/ACCESS.2018.2886026

Jaworski M, Rutkowski L, Angelov P (2020) Concept drift detection using autoencoders in data streams processing. In: ICAISC. Springer-Verlag, p 124–133, https://doi.org/10.1007/978-3-030-61401-0_12

Jiao Y, Chen Y, Gu Y (2018) Subspace change-point detection: A new model and solution. IEEE Journal of Selected Topics in Signal Processing 12(6):1224–1239. https://doi.org/10.1109/JSTSP.2018.2873147

de Jong KL, Bosman AS (2019) Unsupervised change detection in satellite images using convolutional neural networks. In: IJCNN 2019. IEEE, pp 1–8, https://doi.org/10.1109/IJCNN.2019.8851762

Khamassi I, Sayed Mouchaweh M, Hammami M, et al (2015) Self-adaptive windowing approach for handling complex concept drift. Cogn Comput 7(6):772–790. https://doi.org/10.1007/s12559-015-9341-0

Killick R, Fearnhead P, Eckley IA (2012) Optimal detection of changepoints with a linear computational cost. Journal of the American Statistical Association 107(500):1590–1598. https://doi.org/10.1080/01621459.2012.737745

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: ICLR, URL http://arxiv.org/abs/1412.6980

Knuth DE (1997) The Art of Computer Programming: Seminumerical Algorithms, vol 2. Addison-Wesley

Krizhevsky A, Hinton G, et al (2009) Learning multiple layers of features from tiny images. Tech. rep.

Lajugie R, Bach FR, Arlot S (2014) Large-margin metric learning for constrained partitioning problems. In: ICML, JMLR Workshop and Conference Proceedings, vol 32. JMLR.org, pp 297–305, URL http://proceedings.mlr.press/v32/lajugie14.html

LeCun Y, Cortes C, Burges C (1998) The MNIST database of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist/

Lee JA, Verleysen M (2007) Nonlinear Dimensionality Reduction. Springer, https://doi.org/10.1007/978-0-387-39351-3

Liu A, Song Y, Zhang G, et al (2017) Regional concept drift detection and density synchronized drift adaptation. In: IJCAI. ijcai.org, pp 2280–2286, https://doi.org/10.24963/ijcai.2017/317

Liu P, Wang J, Wang Z, et al (2019) High-dimensional data abnormity detection based on improved Variance-of-Angle (VOA) algorithm for electric vehicles battery. In: 2019 IEEE energy conversion congress and exposition

(ECCE), pp 5072–5077, https://doi.org/10.1109/ECCE.2019.8912777

Loglisci C, Ceci M, Impedovo A, et al (2018) Mining microscopic and macroscopic changes in network data streams. Knowl Based Syst 161:294–312. https://doi.org/10.1016/j.knosys.2018.07.011

Lu J, Liu A, Dong F, et al (2019) Learning under concept drift: A review. IEEE Trans Knowl Data Eng 31(12):2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

Lung-Yut-Fong A, Lévy-Leduc C, Cappé O (2015) Homogeneity and change-point detection tests for multivariate data using rank statistics. Journal de la Société Française de Statistique 156(4):133–162

Matteson DS, James NA (2014) A nonparametric approach for multiple change point analysis of multivariate data. Journal of the American Statistical Association 109(505):334–345. https://doi.org/10.1080/01621459.2013.849605

Montiel J, Read J, Bifet A, et al (2018) Scikit-multiflow: A multi-output streaming framework. JMLR 19(1):2915–2914. URL http://jmlr.org/papers/v19/18-251.html

Mowbray M, Savage T, Wu C, et al (2021) Machine learning for biochemical engineering: A review. Biochemical Engineering Journal 172:108,054. URL https://www.sciencedirect.com/science/article/pii/S1369703X21001303

Naseer S, Ali RF, Dominic PDD, et al (2020) Learning representations of network traffic using deep neural networks for network anomaly detection: A perspective towards oil and gas IT infrastructures. Symmetry 12(11):1882. https://doi.org/10.3390/sym12111882

Page ES (1954) Continuous inspection schemes. Biometrika 41(1-2):100–115. https://doi.org/10.1093/biomet/41.1-2.100

Pears R, Sakthithasan S, Koh YS (2014) Detecting concept change in dynamic data streams. Machine Learning 97(3):259–293. https://doi.org/10.1007/s10994-013-5433-9

Qahtan AA, Alharbi B, Wang S, et al (2015) A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In: SIGKDD. ACM, New York, NY, USA, p 935–944, https://doi.org/10.1145/2783258.2783359

Rani R, Khurana M, Kumar A, et al (2022) Big data dimensionality reduction techniques in IoT: review, applications and open research challenges. Cluster Computing 25(6):4027–4049. https://doi.org/10.1007/s10586-022-03634-y

dos Reis DM, Flach PA, Matwin S, et al (2016) Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test. In: SIGKDD. ACM, pp 1545–1554, https://doi.org/10.1145/2939672.2939836

Shewhart WA (1930) Economic Quality Control of Manufactured Product, vol 9. https://doi.org/https://doi.org/10.1002/j.1538-7305.1930.tb00373.x

de Souza VMA, Chowdhury FA, Mueen A (2020) Unsupervised drift detection on high-speed data streams. In: BigData. IEEE, pp 102–111, https://doi.org/10.1109/BigData50022.2020.9377880

Sun Y, Wang Z, Liu H, et al (2016) Online ensemble using adaptive windowing for data streams with concept drift. Int J Distributed Sens Networks 12(5):4218,973:1–4218,973:9. https://doi.org/10.1155/2016/4218973

Suryawanshi S, Goswami A, Patil P, et al (2022) Adaptive windowing based recurrent neural network for drift adaption in non-stationary environment. Journal of Ambient Intelligence and Humanized Computing https://doi.org/10.1007/s12652-022-04116-0

Truong C, Oudre L, Vayatis N (2020) Selective review of offline change point detection methods. Signal Process 167. https://doi.org/10.1016/j.sigpro.2019.107299

Vergara A, Huerta R, Ayhan T, et al (2011) Gas sensor drift mitigation using classifier ensembles. In: Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data. ACM, SensorKDD '11, p 16–24, https://doi.org/10.1145/2003653.2003655

Vrigkas M, Nikou C, Kakadiaris IA (2015) A review of human activity recognition methods. Frontiers Robotics AI 2:28. https://doi.org/10.3389/frobt.2015.00028

Webb GI, Lee LK, Goethals B, et al (2018) Analyzing concept drift and shift from sample data. Data Min Knowl Discov 32(5):1179–1199. https://doi.org/10.1007/s10618-018-0554-1

Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747. URL http://arxiv.org/abs/1708.07747

Xie L, Xie Y, Moustakides GV (2020) Sequential subspace change point detection. Sequential Analysis 39(3):307–335. https://doi.org/10.1080/07474946.2020.1823191

Yan J, Zhang B, Liu N, et al (2006) Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. IEEE Transactions on Knowledge and Data Engineering 18(3):320–333. https://doi.org/10.1109/TKDE.2006.45

Zhao X, Wu J, Shi Y, et al (2018) Fault diagnosis of motor in frequency domain signal by stacked de-noising auto-encoder. Computers, Materials & Continua 57(2):223–242. URL http://www.techscience.com/cmc/v57n2/22968

A Appendix

A.1 Training of autoencoder

Algorithm 3 describes the training of the autoencoder model as done in our experiments. First, we collect the training data from the current window W (line 2). Afterwards we perform gradient descent on X_{train} for E epochs at a learning rate of lr.

Algorithm 3 Autoencoder training

Require: W, learning rate lr, number of training epochs E

1: **procedure** TrainAE(W, lr, E)
2: $X_{train} \leftarrow \{x_i \; \forall \; (-, -, x_i) \in W\}$
3: $\phi, \psi \leftarrow$ NewEncoder(), NewDecoder()
4: **for all** epochs E **do**
5: GradientDescent($\psi \circ \phi, X_{train}, lr$)
6: Return ϕ, ψ

A.2 Detectable and undetectable change for ABCD (pca)

This section illustrates under which conditions one can use principal component analysis to detect change. Figure 6 shows data from two distributions: black points (e.g., before the change) plus the associated main principle component, and blue points (e.g., after the change). On the left, the change affects the correlation between Dim. 1 and Dim. 2. This leads to an increased reconstruction error for the points highlighted in blue. On the right, the change occurs along the main principle component. I.e., the variance along the main principle component has increased. Such kind of change is undetectable by ABCD (pca) as the reconstruction error remains unchanged.

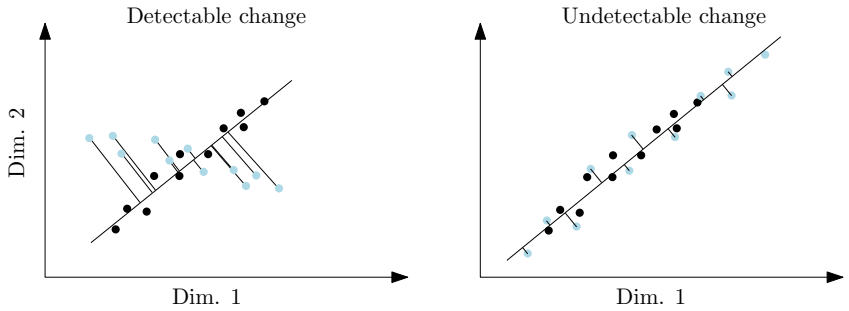


Figure 6: Illustration of detectable and undetectable change using PCA.

A.3 Reconstruction loss over time

Figure 7 shows the reconstruction loss of the evaluated encoder-decoder models over the length of the stream. We observe that indeed the reconstruction loss decreases with increasing bottleneck size (controlled by η), and with increasing number of training epochs E (first three columns). Further, we see that regardless of E, η, or the type of model, the reconstruction loss typically changes after a change point. After the change was detected, ABCD learns the new concept, which mostly leads to a decrease in reconstruction loss. Last, we observe that the theoretical limit of $M = 1$ for the absolute difference between the reconstruction loss and its expected value is overly conservative. A value of $M = 0.1$ seems to be a more realistic choice.

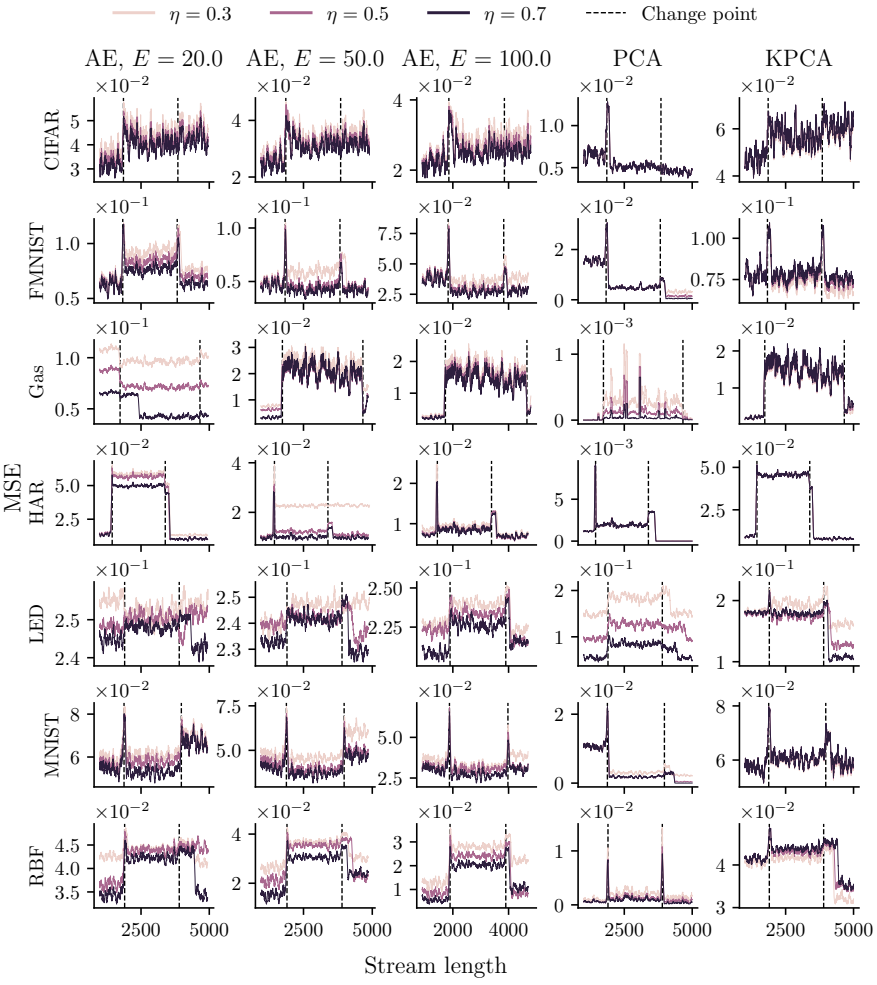


Figure 7: Reconstruction loss over the length of the stream.