Machine Learning-based Detection of Concept Drift in Business Processes

Alexander Kraus[1*] and Han van der Aa[2]

[1*]Data and Web Science Group, University of Mannheim, Germany.
[2]Faculty of Computer Science, University of Vienna, Austria.

*Corresponding author(s). E-mail(s): alexander.kraus@uni-mannheim.de;
Contributing authors: han.van.der.aa@univie.ac.at;

Abstract

Concept drift in process mining occurs when a single event log includes data from multiple versions of a process, making the detection of such drifts essential for ensuring reliable process mining results. Although many techniques have been proposed, they exhibit limitations in accuracy and scope. Specifically, their accuracy diminishes when facing noise, varying drift types, or different levels of change severity. Additionally, these techniques primarily focus on detecting sudden and gradual drifts, overlooking the automated detection of incremental and recurring drifts. To address these limitations, we present `CV4CDD-4D`, a novel approach for automated concept drift detection that can identify sudden, gradual, incremental, and recurring drifts. Our approach follows an entirely different paradigm. Specifically, it employs a supervised machine learning model fine-tuned on a large collection of event logs with known concept drifts, enabling the model to learn how drifts manifest in event logs. The possibility to train such a model has recently emerged through a tool that generates event logs with known concept drifts. However, applying supervised machine learning remains challenging due to the complexities of encoding. To address this, we propose converting an event log into an image-based representation that captures process evolution over time, enabling the use of a state-of-the-art computer vision model to detect drifts. Our experiments show that our approach, compared to existing solutions, improves the accuracy and robustness to noise of drift detection while extending coverage to a broader range of drift types, highlighting the potential of this new paradigm.

Keywords: Process mining, Concept drift detection, Machine learning, Deep learning, Computer vision, Object detection.

1 Introduction

Organizations execute various business processes to achieve their business objectives. These business processes are often supported by information systems that record data generated during the execution of process instances. *Event logs* represent snapshots of such recorded data over a specific period of time, forming the basis for *process mining*, a collection of techniques that examine how business processes are truly executed [1]. However, due to the dynamic nature of business environments, processes under analysis are often not in a steady state but are rather subject to changes [6]. These changes can result in the presence of *concept drifts* in event logs, i.e., situations in which an event log contains data from multiple versions of the same process. *Concept drift detection* aims to identify these different versions in order to prevent the contamination of process mining results that can occur from mixing them. [6].

The importance of concept drift detection in process mining has been widely acknowledged [54], resulting in a range of proposed concept drift detection techniques [47]. However, these existing techniques exhibit notable limitations concerning their accuracy and scope. With respect to their *accuracy*, the performance of existing techniques tends to significantly decline in situations where noise, varying types of drifts, or different levels of change severity are present in an event log. Such declines occur because the techniques depend on algorithmic design choices, often based on heuristic strategies and assumptions about how drifts manifest in event logs. Since, these assumptions are not applicable to all situations, algorithms based on them can be subject to issues such as a lack of robustness to noise or generally poor accuracy. With respect to their *scope*, existing techniques typically detect only a subset of the commonly-known drift types, typically just focusing on *sudden drifts*, with few techniques also considering *gradual* ones [13]. The detection of more complex drifts, such as *incremental* and *recurring drifts* [6], is generally overlooked and has not yet been tackled in an automated manner. Due to these limitations, existing concept drift detection techniques are thus unable to provide a precise and comprehensive understanding of how processes evolve over time.

To overcome this, we propose `CV4CDD-4D`, a concept drift detection approach that can detect sudden, gradual, incremental, and recurring drifts in an automated manner, with high accuracy. We achieve this by following an entirely different paradigm for drift detection in process mining. Specifically, unlike existing techniques, our approach uses a machine learning model trained on a large dataset of labeled event logs, enabling it to learn how drifts of different types manifest themselves in event logs. The possibility of training such a model has only recently emerged, thanks to a tool for generating large collections of logs with known concept drifts [15]. However, even with such data, the challenge of applying (supervised) machine learning to concept drift detection is far from straightforward. This difficulty stems from the challenge of capturing the progression of an entire process over time, in a way that it can serve as suitable for input into a machine learning model. To address this challenge, we draw inspiration from research that uses image-based representations to encode multi-faceted data about processes [42, 43]. Therefore, we first turn an event log into an image that visualizes differences in process behavior over time. This enables us to employ a state-of-the-art object detection model [28] (from the field of computer vision), fine-tuned on a large

collection of event logs with known concept drifts, to recognize where drifts occur in unseen event logs. Our experiments reveal the efficacy of this idea, showing that our approach significantly improves the state of the art in terms of accuracy, robustness to noise, and automation in detecting drifts, while covering a broader range of drift types.

This paper presents an extended and revised version of our earlier work [23] in which we introduced the first version of our computer vision-based approach for concept drift detection in process mining. The current paper extends our earlier work in two main directions. First, we broadened the scope of our approach by including the detection of complex drift types, i.e., incremental and recurring drifts, in addition to the sudden and gradual drifts already covered by the original version. Second, we considerably extended the evaluation of our approach with additional experiments, which allowed us to investigate the performance of our approach in more depth. Specifically, beyond extending the existing experiments to the broader scope of our extended approach, we included a sensitivity analysis in which we investigate the impact of the user parameters on our approach's performance, and performed a qualitative analysis and benchmark comparison on various real-world event logs.

In the remainder, we define the scope of the work in Section 2. Our `CV4CDD-4D` approach, including the input to the problem and desired output, is detailed in Section 4. Section 5 presents the evaluation of our approach against state-of-the-art techniques. Finally, Section 3 reflects on related work before Section 6 concludes the paper.

2 Problem Scope

Our work addresses the problem of detecting concept drifts in the control-flow perspective of a process based on data recorded in an event log. Conventionally, such concept drifts encompass four types: sudden, gradual, incremental, and recurring drifts [6], as illustrated in Figure 1. We pose that these four types can be sub-divided into two groups:

Simple drifts. We jointly refer to sudden and gradual drifts as *simple drifts*, since they correspond to a single change in a process from one version to another:

- *Sudden drift:* A sudden drift describes an abrupt replacement of one process version by another. For instance, Figure 1 shows the replacement of process version v_1 by version v_2 at a certain moment, i.e., the change point p_1. This means that all process instances that start after p_1 will follow process version v_2. Sudden drifts are often observed in scenarios such as emergency response planning, where airlines and airports may alter their security procedures in response to new regulations [6].
- *Gradual drift:* A gradual drift describes a situation where the replacement of process version v_1 by v_2 involves a transition period in which both versions coexist. In these cases, after an initial change point p_1 an increasing fraction of new process instances will follow version v_2, until the roll-out of that version is completed at change point p_2. Gradual drifts, for instance, will occur when an organization starts training its employees in a spread-out manner regarding a new way of

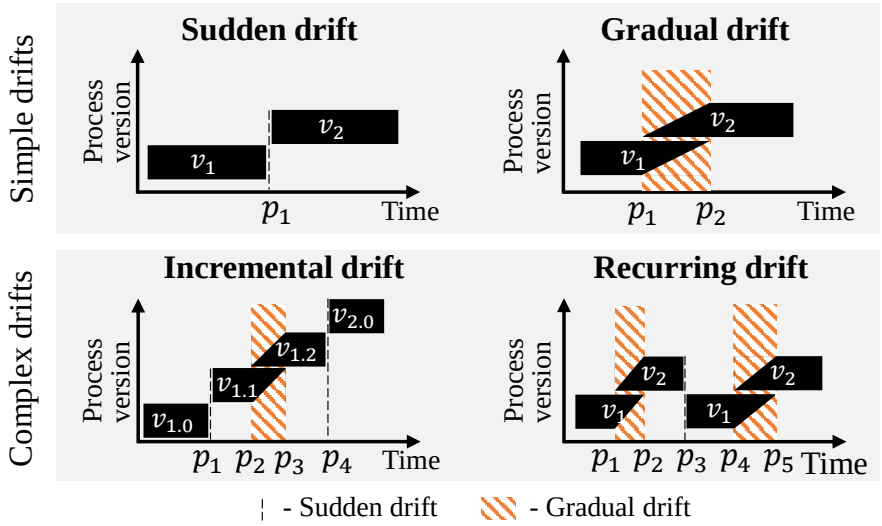


Fig. 1: Problem scope: detection of sudden, gradual, incremental, and recurring drifts.

working, so that more and more employees start following the new version during the training period.

Complex drifts. We jointly refer to incremental and gradual drifts as *complex drifts*, since they consist of several, related simple drifts:

- *Incremental drift:* An incremental drift occurs when one process version is replaced by another through a sequence of simple drifts, rather than at once. For instance, in Figure 1, process version $v_{1.0}$ is replaced by $v_{2.0}$ through a sequence of sudden and gradual drifts. This results in two intermediate versions, $v_{1.1}$ and $v_{1.2}$, along with a total of four change points. Generally, the simple drifts that make up an incremental drift must relate to the same transformation initiative and are characterized by relatively low *change severity*, which measures the extent to which process behavior is impacted by a change. A notable example of this type of drift are the process changes that arise from agile business process management methodologies [6].
- *Recurring drift:* A recurring drift is characterized by a situation when different versions of a process reoccur in an alternating fashion. For instance, Figure 1 illustrates a situation where a process switches between versions v_1 and v_2, following a sequence of sudden and gradual drifts. Recurring drifts are, for instance, common in processes with seasonal patterns.

In the following section, we consider the relevant literature within the scope of the discussed problem.

3 Related Work

Since establishing the problem and importance of concept drift detection in process mining more than a decade ago [54], various techniques have been proposed to address

this problem, as highlighted in recent literature reviews [13,47]. We first discuss existing techniques for offline concept drift detection, since these tackle the same or a subset of the task that our work addresses. Afterwards, we also consider works that focus on others types of concept drift detection, such as in online or multi-perspective settings.

Table 1: Classification of different concept drift detection techniques.

Technique	Change point detection	Drift type detection			
		Sudden	Gradual	Incr.	Rec.
Various works	✷				
Bose et al. [6]	✷	(✷)	(✷)		
Martjushev et al. [35]	✷	(✷)	(✷)		
Maaradji et al. [33]	✷	✷	✷		
Yeshchenko at el. [58]	✷	✷	(✷)	(✷)	(✷)
Our approach (`CV4CDD-4D`)	✷	✷	✷	✷	✷

Legend: "✷" - automated, "(✷)" - semi-automated.

Offline concept drift detection. Table 1 provides an overview of existing concept drift detection techniques, including their scope and automation level.

Change point detection. The majority of existing techniques (first row) just focus on detecting change points in an event log since this is a required first step in concept drift detection [2,7,14,19,21,25,30–32,34,36–38,38,46,48,55–57,60]. They use a wide range of ways to tackle this task, including statistical testing, various kinds of feature representations, windowing strategies, change and trend detection, and clustering. Many of the techniques achieve good results, as demonstrated in a recent evaluation framework [3]. However, our evaluation experiments show that our proposed `CV4CDD-4D` approach outperforms them in terms of accuracy and robustness to noise when detecting change points. Furthermore, these techniques do not go beyond the detection of change points, meaning that they are only able to recognize *when* a process's behavior significantly changed, but they do not provide insights into *what* type of drift actually occurred.[1] As a result, they do not provide information about how the process evolved over time.

Drift type detection. A few techniques [6,33,35,58] go beyond change point detection and are also able to detect drifts and their types (though often limited in scope and automation).

Bose et al. [6] introduced concept drift detection in process mining, presenting a method for automatically detecting sudden and gradual drifts using statistical testing of feature vectors. Although this approach is pioneering, it has limitations in automated drift detection. Users must specify the type of drift in advance and manually select features, relying on prior knowledge of drift characteristics. Otherwise, they may face the computational burden of testing all feature combinations. Additionally,

[1] Recent techniques [36,37] demonstrate potential for detecting all four drift types. However, their current implementations are limited to change point detection, with plans to extend their capabilities to other drift types in future versions.

users must specify a window size for drift detection, potentially missing some drift occurrences.

To address the window size limitation, Martushev et al. [35] enhanced this method by introducing adaptive windowing, which automatically adjusts the window size when searching for drifts. However, this approach still requires users to define minimum and maximum window size parameters as boundaries for automated window adaptation. Moreover, users must define the type of drift beforehand. Therefore, both techniques remain limited to detecting only sudden and gradual drifts, neglecting recurring and incremental drifts.

The work by Maaradji et al. [33] introduced an alternative technique (ProDrift) for detecting sudden and gradual drifts, addressing the main limitation of previous solutions [6,35] and representing the current state of the art in simple drift detection. Their approach involves a two-step process: first, it detects change points to identify sudden drifts and then it employs post-processing on the output of the first step to detect gradual drifts. Specifically, they analyze the behavior within the intervals between two change points by statistically assessing whether it exhibits a mixture of behavior distributions before and after these points. However, this distribution-based method has a significant drawback: in situations where noise is present in an event log, their approach experiences a notable decrease in detection accuracy, as demonstrated in our evaluation (Section 5.1).

The Visual Drift Detection (VDD) technique, introduced by Yeshchenko et al. [58], is a stand-alone solution for detecting four types of concept drifts in event logs. This technique leverages concepts such as temporal logic, DECLARE constraints [11], and time series analysis to group similar declarative behavioral constraints and automatically identify change points. To identify different types of drifts, it provides visual aids, including Drift Maps, Drift Charts, and directly-follows graphs. However, the identification of gradual, recurring, and incremental drifts remains a manual process that relies on user interpretation. As a result, recognizing drift and their types within the same event log can be challenging and subjective. Our approach overcomes this problem by detecting all four types of drifts in an automated manner.

Overall, it is evident that the comprehensive offline detection and characterization of drifts and their types in event logs have not been adequately addressed.

Other concept drift detection techniques. A variety of existing techniques for detecting concept drift address other aspects of the problem. For instance, some approaches simultaneously consider multiple process perspectives, such as time, resource involvement, and data, when identifying concept drifts [8, 22, 44, 45]. Other techniques not only aim to detect concept drifts but also provide explanations for these changes [4,16]. Various works focus on detecting concept drifts in event streams (online settings), which allows for real-time detection. These techniques can detect change points [18, 41] and offer detailed drift localization [39, 40] while dealing with challenges such as computational overhead, real-time processing requirements, and the need for continuous monitoring. Some online drift detection techniques specifically target drift type detection [5, 17, 20, 45, 51, 53], while also considering different process perspectives [52]. Finally, certain techniques operate on trace streams rather

than event streams. These techniques can identify change points [9, 24, 46, 50, 55, 59] and detect drift types [29].

In the following section, we present our approach, which overcomes existing limitations and advances the state of the art by offering a reliable and automated method for identifying all four types of drifts in event logs. Our method leverages an innovative machine learning paradigm to learn how drifts manifest in event logs, moving beyond traditional hand-crafted, unsupervised techniques.

4 Approach

This section introduces `CV4CDD-4D`, our computer vision approach for concept drift detection. As visualized in Figure 2, our approach consists of two main steps. First, we transform an event log into an image that captures the behavioral (dis)similarity of a process over time. Then, the image is passed to our fine-tuned computer vision model, which identifies if drifts are present in an event log and, if so, determines their type (sudden, gradual, incremental, recurring) and corresponding change points.

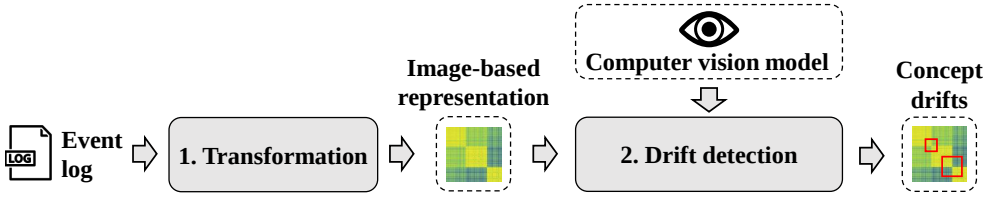


Fig. 2: Our `CV4CDD-4D` approach: overview of the main steps.

Before describing the two steps of our approach in detail, we define the approach's input and desired outcome.

Input. Our approach takes as input an *event log* L, which we define as a collection of events e recorded by an information system during process execution. Each *event* $e \in L$ needs to at least have a *case ID*, an *activity*, and a *timestamp*. We use σ to represent a *trace*, a sequence of events from L with the same case ID, ordered by their timestamps. Finally, we denote Σ_L as the ordered collection of traces, arranged according to the timestamp of their first event.

Output. Given an event log L as input, the desired output is a collection of drifts $D := \{d_1, \cdots, d_n\}$, where each drift d_i is represented by a tuple, $d_i := (type, p_{start}, p_{end})$, with $d_i.type$ specifying the drift type, and $d_i.p_{start}$ and $d_i.p_{end}$ denoting the drift's start and end change points, respectively. In case of a sudden drift, the start and end points are the same. For a complex drift, which consists of a sequence of sudden and gradual drifts, the start and end change points are defined by the start change point of the first drift and the end change point of the last drift in the sequence, respectively.

4.1 Transformation of Event Log into Image

The first approach step takes as input an event log and transforms it into an image. The image visualizes the behavioral (dis)similarity of a process over time recorded, which can be used to recognize concept drifts. The transformation includes four sub-steps, as depicted in Figure 3.

Fig. 3: First approach step: transforming an event log into an image.

Split traces into windows. The approach first splits the chronologically ordered traces in Σ_L into an ordered collection of N windows, $W := \langle w_1, \ldots, w_N \rangle$. These windows are non-overlapping and collectively cover all recorded traces in Σ_L, with each window w_i containing approximately $|\Sigma_L|/N$ traces.

During the fine-tuning of the computer vision model, we use $N = 200$ as a default value for the number of windows. For the training collection (see Table 2), we tested various options and selected 200 because it, on average, provides an effective image-based representation for event logs in the context of concept drift detection. This choice was motivated by the need for a window size that strikes a balance: it should not be too large (which could conceal gradual drifts within a single window, making them undetectable) or too small (which might capture too few process traces to adequately represent process behavior). The selection is primarily influenced by the configurations for the length of generated drift and drift-free periods (and their more complex patterns) used to generate event logs for our training collection. Therefore, using a different configuration or data generation tool would likely require re-evaluating the choice of this parameter.

We recommend using 200 windows also as the default setting for detecting concept drifts using `CV4CDD-4D` on a new event log. However, for small event logs (e.g., with fewer than 2,000 traces), decreasing the number of windows avoids having too few traces per window, as demonstrated in our sensitivity analysis in Experiment 1 (see Section 5.1). Conversely, larger event logs (especially those spanning a long time range) may benefit from having more windows, to prevent drifts occurring within the span of a single window.

Calculate behavioral representation. After establishing W, our approach computes a *behavioral representation*, $B(w_i)$ for each window w_i, which characterizes the process behavior of w_i's traces. Each $B(w_i)$ consists of a collection of two-dimensional tuples, each storing a behavioral pattern and its frequency, as visualized in Figure 4.

A common behavioral representation used in process mining is to capture the directly-follows frequencies observed during a time window [13], which we use as the default for our approach. It counts how often an activity was observed to directly succeed another one in all the traces of a given window. However, it is important to note that the choice for a behavioral representation is flexible, provided that it yields a

numeric frequency distribution over a predefined set of relations or patterns across the window. Therefore, `CV4CDD-4D` can also cover other types of relations (e.g., eventually follows), sets of relation types, such as those of a behavioral profile [49], or declarative constraints [12]. However, this would require retraining the computer vision model to learn drift patterns in the new behavior representation (see Section 4.2).

Windows	**Traces**	**Behavioral representation**	**Similarity measure**
w_i	$\langle$a,b,c$\rangle^2$ $\langle$a,c$\rangle$	$B(w_i) = \begin{pmatrix} a \to b : 2 \\ b \to c : 2 \\ a \to c : 1 \\ c \to d : 0 \end{pmatrix}$	$S[i,j] = sim(B(w_i), B(w_j))$ $= cosine(\begin{bmatrix} 2 \\ 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix})$
w_j	$\langle$a,b,c$\rangle$ $\langle$a,c,d$\rangle$	$B(w_j) = \begin{pmatrix} a \to b : 1 \\ b \to c : 1 \\ a \to c : 1 \\ c \to d : 1 \end{pmatrix}$	$= 0.83$

Fig. 4: Illustration of the similarity calculation between two windows.

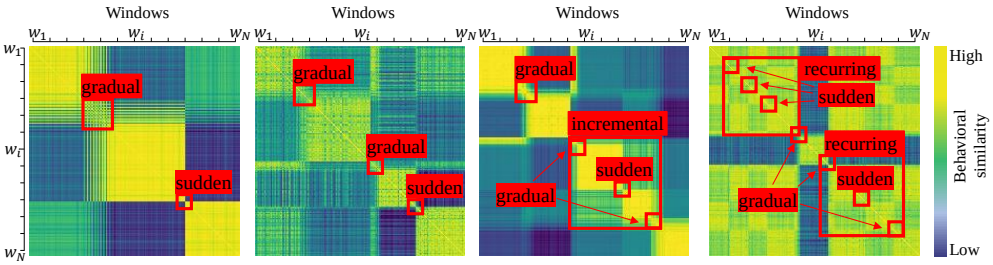
Measure similarity. Next, our approach compares the behavioral representations obtained for the different windows, quantifying their similarity. This comparison is done for each pair w_i and w_j from W, resulting in a symmetric similarity matrix denoted as S. Each entry $S[i,j]$ in this matrix shows the similarity between the behavior represented by $B(w_i)$ and $B(w_j)$.

The similarity matrix S can be established using various similarity measures. In our approach, we use cosine similarity as the default measure because it effectively detects newly introduced or removed behavior relations and is well-suited for assessing changes in the frequencies. In theory, other commonly used similarity measures, such as Earth Mover's Distance [7], could also be applied. However, this would again require retraining the computer vision model. Figure 4 illustrates the calculation of the similarity measure between two windows, using a behavioral representation based on directly-follows frequencies and the cosine similarity measure.

Transform into image. Finally, to enable image-based concept drift detection, the similarity matrix S is transformed into an image. For this transformation, our approach normalizes the matrix values to a range between 0 and 1, where the maximum similarity corresponds to 1 and the minimum similarity is 0. Each normalized value is then scaled by 255 and converted to integers, resulting in a range between 0 and 255. Finally, using the Python Imaging Library[2] and a color map, images are generated from the normalized values.

Figure 5 depicts a few examples of the outcome of this step, covering different drift scenarios. Note that the annotation that is shown is covered in Section 4.2.

[2]Available online: https://python-pillow.org



a) Gradual and sudden drifts | b) Gradual, gradual, and sudden drifts | c) Gradual and incremental drifts | d) Recurring, gradual, and recurring drifts

Fig. 5: Output of the first approach step (incl. annotations).

4.2 Drift Detection

The second step of `CV4CDD-4D` takes as input the image obtained from the previous step and applies a fine-tuned object detection model to detect concept drifts. In this section, we present an overview of the object detection task and employed object detection model, elaborate on the training data and its annotation, and clarify the training configurations.

Object detection using RetinaNet. Object detection is a fundamental task in computer vision, where the goal is to identify and locate objects within images. Deep learning methods have significantly advanced this field by directly learning features from data, leading to breakthroughs in object detection [28]. *RetinaNet* [26] is a recent addition to deep learning-based object detection models. Known for its effectiveness and reliability, RetinaNet has become widely adopted in both research and practical applications, setting new standards in object detection performance.

RetinaNet Architecture. The RetinaNet model consists of three main components [26]:

- *Backbone Network:* The backbone network identifies patterns necessary for object detection. RetinaNet uses ResNet, a deep convolutional neural network, which improves learning efficiency by allowing inputs to bypass certain layers.
- *Feature Pyramid Network (FPN):* The FPN processes multi-scale feature maps from the backbone to create a pyramid of features at different resolutions. It enhances the model's ability to detect objects of varying sizes by combining high-resolution feature maps from earlier layers with lower-resolution feature maps from deeper layers.
- *Classification and Bounding Box Subnetworks:* These subnetworks enable accurate object detection. The classification subnet predicts the probability of an object belonging to a particular class at each location on the feature map, operating at multiple levels of the FPN. The box regression subnet predicts the coordinates of bounding boxes for objects and refines their positions.

Additionally, RetinaNet uses a specialized loss function, *Focal Loss*, to address class imbalance by down-weighting easily classified background examples and focusing on difficult-to-detect objects. This makes RetinaNet effective in detecting concept drift and handling noise across various sections of an image.

Training data and annotation. We use a training set of event logs with known concept drifts. Each event log in the training set is transformed into an image using the first step of our `CV4CDD-4D` approach (Section 4.1). Then, each image is annotated based on the drift information stored in the gold standard, capturing where different drifts occur and what their types are. For the annotation, we define bounding boxes using the widely-employed COCO (Common Objects in Context) format [27].

In our case, we use these bounding boxes to capture where drifts of certain types occurred in the image, as shown for various scenarios in Figure 5. To annotate sudden drifts, characterized by a single change point p that belongs to a window w_i, we establish a bounding box around w_i that spans in total 2 windows in both directions from w_i, resulting in a total length of 5 windows. For gradual drifts, we create annotations using windows that correspond to the start and end change points. Each change point p_{start} and p_{end} is associated with a trace index, which belongs to a particular window in W. The corresponding windows w_i and w_j, allow us to link p_{start} and p_{end} to their window indices i and j. We use these indices to define a bounding box for gradual drift within an image.

Training configurations. To operationalize `CV4CDD-4D`, we specifically use the RetinaNet model from the TensorFlow Model Garden[3], based on the SpineNet backbone (ID 143). The model is pre-trained on the COCO dataset[4], a widely-used dataset for object detection. To adapt it to our specific task, we fine-tune the model on a training set and halt fine-tuning using a validation set to prevent overfitting.

Image input, batch size, anchor boxes. We use a fixed input size of 256×256 for fine-tuning the model[5] with a batch size of 128 images, taking 312 iterations per epoch. The model undergoes 500 epochs of training, with augmented images to increase diversity and robustness by scaling up to two times or down to one-tenth of their original size. We employ anchor boxes with a 1:1 aspect ratio, concentrating exclusively on square-shaped bounding boxes, since all annotations correspond to squares of different sizes along the diagonal of the image.

Optimization and learning rate. We use stochastic gradient descent with a momentum of 0.9 and clip norm of 10, known for its simplicity and efficiency in training deep learning models, especially with large datasets. Momentum aids convergence by leveraging past gradients, while gradient clipping prevents exploding gradients, ensuring stability, particularly in complex neural networks. Additionally, we employ a cosine learning rate, widely adopted for its simplicity and ability to enhance convergence and generalization. This schedule adjusts the learning rate throughout training, following a cosine-shaped function.

Confidence level. For each detected drift, the model assigns a confidence level between 0 and 1, reflecting its certainty in identifying and classifying the drift in the image. A score closer to 0 indicates that no drift is present, while a score closer to 1 means the model is confident that a drift of a certain type is present and correctly classified. We set a threshold of 50% for the detection confidence level, meaning that if the model detects a drift type with confidence above 50%, we consider the drift as detected.

[3] TensorFlow Model Garden, Available online: https://github.com/tensorflow/models
[4] COCO dataset, Available online: https://cocodataset.org/
[5] If an input image provided for inference has a different pixel size, i.e., because it was established using a different number of windows N, RetinaNet automatically rescales the image to the default size.

Using the fine-tuned RetinaNet model, our CD4CDD-4D approach can be directly applied to detect concept drifts in unseen event logs.

5 Evaluation

This section presents three experiments conducted to comprehensively evaluate the performance of our CV4CDD-4D approach for concept drift detection from multiple perspectives. In Experiment 1, we assess the accuracy of our approach in detecting change points in event logs and compare our results to various baseline techniques that address this critical task for concept drift detection. Next, in Experiment 2, we evaluate the accuracy of our approach in detecting drifts and their types and highlight its advantages over a comparable state-of-the-art technique. Finally, in Experiment 3, we apply our approach to real-life event logs and compare the insights obtained with findings from the state-of-the-art technique.

To ensure reproducibility, the data collection, implementation, configurations, and raw results are accessible in our public repository[6].

5.1 Experiment 1: Change Point Detection

In this section, we assess the ability of our approach to detect change points in event logs in comparison to existing baselines using synthetic data. We consider this problem in isolation from the detection of drifts, given its fundamental role in concept drift detection, as also evidenced by the various techniques that have been proposed to address it. In the following, we discuss the evaluation setup, obtained results, and findings from a sensitivity analysis.

5.1.1 Evaluation Setup

Below we elaborate on the details of the data collection, baselines, evaluation measures, as well as configurations used to evaluate our approach.

Data collection. Our data collection comprises two datasets, summarized in Table 2.

CDLG dataset. To train, validate, and test our drift detection approach, we require a large collection of event logs that contain known (i.e., gold-standard) concept drifts of sudden, gradual, incremental, and recurring types. Since such a collection is not publicly available, we, therefore, generated synthetic datasets using CDLG (Concept Drift Log Generator) [15], a tool for the automated generation of event logs with concept drifts, which comes with a wide range of parameters.

We used CDLG to generate 50,000 event logs, allocating 80% for training, 5% for validation, and 15% testing. The generated event logs have the following characteristics:

- The logs are generated from process trees containing between 6 and 20 activities, as well as sequential, choice, parallel, and loop operators.
- Each event log has between 1,000 and 21,000 traces (with an average of around 7,200 traces per log), and the average trace length varies from 1 to 65 events.

[6] Project repository: https://gitlab.uni-mannheim.de/processanalytics/cv4cdd.

Table 2: Characteristics of the two synthetic datasets.

Characteristics	**CDLG dataset**			**CDRIFT**
(Number of)	**Training**	**Validation**	**Test**	**dataset**
Event logs	40 000	2500	7500	115
→ without drifts	9834	586	1834	0
→ with noise	19 908	1250	3768	60
Drifts	112 660	7069	21 229	156
→ Sudden drifts	41 295	2587	7827	156
→ Gradual drifts	41 395	2615	7776	0
→ Incremental drifts	14 967	986	2823	0
→ Recurring drifts	15 003	881	2803	0
Change points	120 151	7501	22 567	156
→ Sudden	59 971	3726	11 333	156
→ Gradual	60 180	3775	11 234	0

- The event logs have 0 to 3 drifts each (with equal probability). Incremental and recurring drifts consist of 3 simple drifts (of sudden or gradual type), yielding a maximum of 18 change points per log.
- Each drift introduces changes up to 30% of the process tree elements (activities and operators) through deletion, insertion, or swapping.
- A quarter of the logs contain randomly inserted noise in 30% of the traces and another quarter in 60% of the traces. The other half are noise-free.

CDRIFT dataset. To assess the generalizability of our approach and verify that its performance is not restricted to the characteristics of the CDLG dataset, we also consider a dataset used in a recent experimental study [3], which we refer to as the CDRIFT dataset. This set consists of 115 synthetic event logs, previously employed in evaluating various concept drift detection techniques, stemming from three sources [6, 10, 41]. The logs have about 1700 traces on average and contain between 1 and 4 change points. Notably, the CDRIFT dataset only contains sudden drifts, though.

Baselines. We compare our approach to seven techniques for the detection of change points that were used in a recent benchmark study by Adams et al. [3]:

1. Bose/J by Bose et al. [6] uses non-overlapping and continuous fixed-size windowing with activity pair-based feature extraction and statistical testing.
2. Adwin/J by Martjushev et al. [35] improves the Bose/J technique by introducing adaptive windowing using the ADWIN approach.
3. ProGraphs by Seeliger et al. [48] implements non-overlapping and continuous adaptive-size windowing, uses graph-based process features alongside Heuristics Miner, and employs statistical testing.
4. ProDrift by Maaradji et al. [33] employs non-overlapping continuous fixed and adaptive-size windowing, statistical testing, and an oscillation filter.
5. Rinv by Zheng et al. [60] uses behavioral profiles, a process similarity measure, and DBSCAN clustering.

6. EMD by Brockhoff et al. [7] employs a sliding window approach with local multi-activity feature extraction and the Earth Mover's Distance.
7. LCDD by Lin et al. [25] uses both static and adaptive sliding windows, incorporates directly-follows relations, and ensures local completeness.

Evaluation measures. We report on results obtained using established evaluation measures for detecting change points [3]. Specifically, for each event log, we compare the sequences of detected $P^d = \langle p_1^d, \ldots, p_n^d \rangle$ and gold-standard $P^g = \langle p_1^g, \ldots, p_m^g \rangle$ change points ($n, m \geq 0$), where each change point is represented by the ordinal number of the first trace that started after the change.

To identify which gold-standard change points have been successfully detected, we use the linear program proposed by Adams et al. [3] to establish a pairwise mapping between the points in P^d and P^g. This program finds an optimal mapping M, assigning as many points to each other as possible, while minimizing the distance between corresponding change points. Note that no point in P^d is assigned to multiple points in P^g or vice versa. Furthermore, M will only include pairs $p_i^d \sim p_j^g$ that are within an acceptable distance from each other, which we refer to as the allowed *latency*. We define latency as a percentage of the total traces in Σ_L, i.e., it must hold that $|p_i^d - p_j^g| \leq |\Sigma_L| * \mathit{latency}$. We report on results obtained using latency levels of 1%, 2.5%, and 5%.

Due to the consideration of latency, each correspondence in M is regarded as a true positive. From this, we derive *precision* (Prc.) as $|M|/|P^d|$, i.e., the fraction of detected change points that are correct according to the gold standard, *recall* (Rec.) as $|M|/|P^g|$, i.e., the fraction of correctly detected gold-standard change points, and the F1-score as the harmonic mean of precision and recall.

Configurations. For different datasets, we use different configurations for the baselines and our approach.

Baselines. When reporting on the performance of the baseline techniques, we use the parameter settings that we found to achieve the highest F1-score. To find these settings for the CDLG dataset, we applied the experimental framework by Adams et al. [3], which assesses different parameter configurations, using the validation subset of the CDLG dataset. For the CDRIFT dataset, we ran experiments using all configurations that are tested in the Adams et al. [3] framework and report on the results obtained using the best parameter settings. The exact parameter settings used for the different techniques per dataset are detailed in our repository.

Our approach. We fine-tuned the RetinaNet model used by our approach with the training and validation subsets of the CLDG dataset and parameters described in the second step of our approach. Given such a fine-tuned model, the only parameter to set for inference is the number of windows N to be used. For the test subset of the CDLG dataset, we use the same number of windows as used during the fine-tuning ($N = 200$). For the CDRIFT dataset, similar to the baseline's parameters, we report on the results obtained for the best parameters ($N = 70$) derived from a sensitivity analysis (see Section 5.1.3).

5.1.2 Results

In the following, we present the results obtained for the two datasets, also focusing on different latency and noise levels.[7]

Table 3: Overall results for detecting change points.

Dataset	Technique	Latency 1%			Latency 2.5%			Latency 5%		
		Prc.	**Rec.**	**F1**	**Prc.**	**Rec.**	**F1**	**Prc.**	**Rec.**	**F1**
CDLG (test)	Bose/J	0.32	0.39	0.25	0.49	0.60	0.54	0.57	0.70	0.63
	Adwin/J	0.43	0.28	0.34	0.58	0.38	0.46	0.63	0.42	0.50
	ProGraphs	0.24	0.26	0.25	0.48	0.52	0.50	0.58	0.64	0.61
	ProDrift	0.55	0.22	0.32	0.74	0.30	0.43	0.76	0.31	0.44
	Rinv	0.36	0.44	0.40	0.46	0.56	0.51	0.53	0.64	0.58
	EMD	0.36	0.44	0.39	0.51	0.62	0.56	0.58	0.71	0.64
	Lcdd	0.27	0.41	0.32	0.39	0.61	0.48	0.46	0.72	0.56
	`CV4CDD-4D`	**0.87**	**0.75**	**0.81**	**0.90**	**0.77**	**0.83**	**0.90**	**0.77**	**0.83**
CDRIFT	Bose/J	0.08	0.07	0.07	0.60	0.52	0.56	0.75	0.66	0.70
	Adwin/J	0.15	0.11	0.13	0.40	0.29	0.34	0.71	0.51	0.59
	ProGraphs	0.21	0.18	0.19	0.48	0.41	0.45	0.78	0.67	0.72
	ProDrift	**0.91**	0.32	0.48	**1.00**	0.35	0.52	**1.00**	0.35	0.52
	Rinv	0.01	0.00	0.00	0.23	0.18	0.20	0.47	0.36	0.41
	EMD	0.05	0.03	0.04	0.88	0.59	0.71	0.97	0.66	0.79
	Lcdd	0.00	0.01	0.01	0.02	0.04	0.02	0.24	0.64	0.35
	`CV4CDD-4D`	0.61	**0.52**	**0.56**	**1.00**	**0.86**	**0.92**	**1.00**	**0.86**	**0.92**

Support: CDLG dataset (test): 22567 change points, CDRIFT dataset: 156 change points.

Accuracy. In the following, we describe the results for each of the two datasets.
CDLG dataset (test). For the test subset of the CDLG dataset, our `CV4CDD-4D` approach consistently outperforms the baselines, demonstrating F1-scores ranging from 0.81 at 1% latency to 0.83 at 2.5% and 5% latencies. It already reaches its peak performance at just 2.5% latency, surpassing the best baseline, EMD, by 0.27. In terms of recall, our approach outperforms the baseline scores by 0.30 and 0.15 at 1% and 2.5% latencies, respectively. At 5% latencies, our approach achieves also the highest recall of 0.77, however, the Lcdd, EMD, and Bose/J techniques achieve comparable recall scores, each exceeding 0.70. Despite this, they exhibit lower precision, resulting in significantly lower F1-scores. Finally, in terms of precision, our `CV4CDD-4D` approach surpasses the best-performing baseline, ProDrift, by margins of 0.22, 0.16, and 0.14 for 1%, 2.5%, and 5% latency, respectively.
CDRIFT dataset. We obtain overall similar results for the CDRIFT dataset. Our `CV4CDD-4D` approach outperforms all baselines across latency levels, achieving F1-scores of 0.56, 0.92, and 0.92 at 1%, 2.5%, and 5% latencies, respectively. These higher values can be attributed to the fact that the CDRIFT dataset contains only sudden drifts, which are relatively easier to detect for our approach. Only at 1% latency does

[7]Given the non-determinism involved in training deep learning models, we repeated the training and inference procedure of our approach five times. These repetitions resulted in mean standard deviations of less than 0.1 percentage points across all measures (for the test subset of the CDLG dataset). We report on the results of the first run in the remainder.

ProDrift achieve a higher precision of 0.91 compared to 0.61 for our approach. The reason for the lower precision of our approach is rather technical. It is mainly attributed to the annotation of sudden drifts using bounding boxes of 5 windows spanning around the position of an actual sudden drift in an image. In scenarios with 70 windows and a latency of just 1%, inaccuracies arise during the transformation from the coordinates of the bounding box to the corresponding window index and subsequently to the first trace within the window, leading to low precision and recall. This is supported by the correctly positioned bounding boxes in the respective images, along with the observation that accuracy sharply increases to its peak values at the next latency of 2.5%.

Noise impact. To evaluate the robustness to noise of our approach, we report the results for the event logs with different noise levels in the test subset of the CDLG dataset. We report results using 5% latency because the accuracy of several baselines drops significantly for 1% and 2.5% latencies, making it difficult to determine whether the decrease in performance is caused by noise or latency.

Table 4: Noise impact on detecting change points (5% latency).

Technique	W/o noise			With 30% noise			With 60% noise		
	Prc.	**Rec.**	**F1**	**Prc.**	**Rec.**	**F1**	**Prc.**	**Rec.**	**F1**
Bose/J	0.59	0.70	0.64	0.57	0.70	0.63	0.55	0.71	0.62
Adwin/J	0.64	0.43	0.51	0.62	0.41	0.50	0.62	0.40	0.49
ProGraphs	0.55	0.64	0.59	0.60	0.66	0.63	0.64	0.61	0.63
ProDrift	0.76	0.60	0.67	0.67	0.002	0.004	0.33	0.004	0.001
Rinv	0.63	**0.83**	0.72	0.39	0.41	0.40	0.40	0.48	0.44
EMD	0.58	0.72	0.64	0.59	0.70	0.64	0.57	0.70	0.62
Lcdd	0.63	0.76	0.69	0.35	0.77	0.48	0.36	0.60	0.45
CV4CDD-4D	**0.89**	0.78	**0.83**	**0.89**	**0.75**	**0.82**	**0.89**	**0.75**	**0.81**

Support: 22567 change points.

As summarized in Table 4, our CV4CDD approach maintains consistent performance regardless of noise, achieving the highest F1-scores from 0.83 for logs without noise to 0.82 and 0.81 for logs with 30% and 60% noisy traces, respectively. In noise-free conditions, three baselines (ProDrift, Lccd, and Rinv) come close to our results. The Rinv technique shows an outstanding recall of 0.83, while ProDrift maintains its lead across the baselines in precision. However, all three of these baselines experience a notable decline in accuracy when noise is introduced, particularly ProDrift. Conversely, baselines with relatively lower accuracy on the noise-free logs (EMD, Adwin/J, and ProGraphs) demonstrate less vulnerability to noise. This reveals that the baselines are subject to a trade-off between performance in noise-free conditions and robustness to noise, which does not apply to our approach.

5.1.3 Sensitivity Analysis

Finally, we discuss how the number of windows, N, specified by the user affects the performance of our approach. To investigate this, we conduct a sensitivity analysis that examines a range of windows and the corresponding evaluation measures for the two

datasets. Considering the average size of the event logs, we analyze windows between 100 and 300 for the test subset of the CDLG dataset and between 60 and 200 for the CDRIFT dataset.

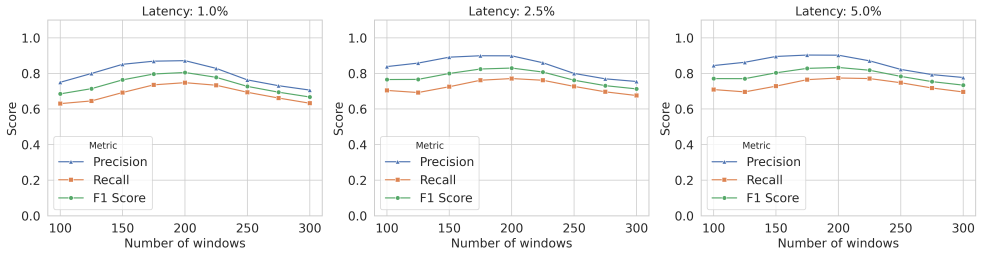


Fig. 6: Sensitivity analysis (the test subset of the CDLG dataset).

CDLG dataset (test). Figure 6 shows the effects of different number of windows on the evaluation measures across the latency levels. All three figures show the same major trend. Specifically, we observe that the F1-score remains within a corridor of ± 5 percentage points for each latency level, with a slight decline in performance at both extremes of the examined range of windows. This indicates that the approach is generally robust to the choice of the number of windows when detecting drifts in the test subset of the CDLG dataset.

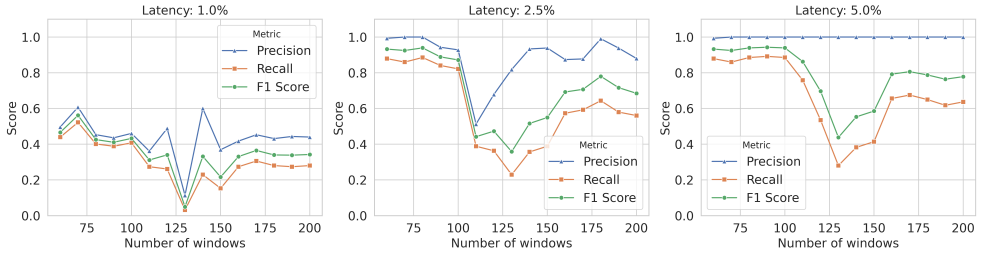


Fig. 7: Sensitivity analysis (CDRIFT dataset).

CDRIFT dataset. Figure 7 illustrates the results of our sensitivity analysis, where we can see two major findings. First, the results suggest that for event logs with relatively few traces (as in the CDRIFT dataset), it is reasonable to reduce the number of windows from the default value of 200 to 100 or fewer, allowing each window to capture more traces and better represent process behavior. Second, a noticeable decline in recall occurs between 100 and 160 windows, with the lowest performance observed at 130 windows. This outcome can be attributed to the structure of the synthetic event logs in the CDRIFT dataset, which typically feature either one change point in the middle of the log (in the majority of logs) or multiple evenly spaced change points. At 130 windows, change points fall near the center of a window, causing the distance

from the start of the window (used for the detection of change points) to exceed the allowable latency, resulting in reduced performance.

Overall, we can observe in this experiment that our approach achieves a notable performance improvement with respect to latency and shows consistent robustness to noise when it comes to the detection of change points in event logs.

5.2 Experiment 2: Concept Drift Detection

This section discusses the experiment conducted to evaluate the performance of our approach to detect drift and their types, also in comparison with the state-of-the-art technique. In the following, we discuss the evaluation setup, obtained results, and insights from a sensitivity analysis.

5.2.1 Evaluation Setup

First, we provide information regarding the baseline, data collection, configurations, and evaluation measures used in this experiment.

Baseline. We compare our approach against the Visual Drift Detection (VDD) technique proposed by Yeshchenko at el. [58]. We selected this technique because it stands out as the only existing technique that can be used to detect four types of drifts from an event log and is the only (partially) comparable solution to our approach, due to a lag in automation. In our experiments, we use the online version of the VDD technique[8].

Data collection. We use the test subset of the CDLG dataset to evaluate the accuracy of our approach in detecting concept drift, as no other collections of event logs encompass the necessary drift scenarios that include a mix of different numbers, types of drifts, noise levels, and severities of process changes. However, since the VDD technique is not fully automated and depends on user interpretation of visualizations, we showcase the advantages of our approach using a specific event log from our test subset of the CDLG dataset. Specifically, we use log number 5436, which describes a complex drift scenario with noisy behavior and serves as a proper example to demonstrate the effectiveness of our approach in detecting complex drifts, particularly in comparison to the baseline. This log contains 60% noise, 72,433 events, 10,103 traces, 3,787 trace variants, and 9 distinct activities. The left side of Table 7 indicates that the log includes two complex drifts: incremental and recurring. Both drifts consist of a sequence of three simple drifts: sudden, gradual, and sudden, leading to a total of 8 change points.

Configurations. For the baseline, we use the suggested default parameters of the online version of the tool for the selected event log: window size 330, slide size 165, cut threshold 300. For our `CV4CDD-4D` approach, we use the default value of 200 windows.

Evaluation measures. We report on precision, recall, and F1-score by comparing a collection of detected drifts D^d to the gold-standard drifts D^g. For each drift type, t, and the corresponding detected $D^d(t) \subseteq D^d$ and gold-standard drifts $D^g(t) \subseteq D^g$, a true positive (tp) occurs if there is a detected drift $d_k^d \in D^d(t)$ of which both the start and end change points correspond to those of a gold-standard drift $d_l^g \in D^g(t)$

[8] Available online: https://yesanton.github.io/Process-Drift-Visualization-With-Declare/client/build/

(given a certain latency level). However, if only the start or end point of d_i^g is detected correctly, we still count it as 0.5 of a true positive (as well as 0.5 of a false positive). If neither of the detected change points for a drift corresponds to the gold standard, it is considered a false positive (fp). Finally, the number of false negative and true positives ($fn + tp$) is given by the number of actual drifts of a given type.

Given these scores, we compute *precision* (Prc.) as $tp/(tp + fp)$, *recall* (Rec.) as $tp/(fn + tp)$, and the F1-score per drift type (and logs without drifts), as well for the overall detection (using weights to account for their different support values).

5.2.2 Results

In the following, we present the overall results of our approach with respect to different latency and noise levels. Then, we show the advantage of using our approach in comparison to the baseline.

Table 5: Concept drift detection results by latency levels.

Drift	Support	Latency 1%			Latency 2.5%			Latency 5%		
		Prc.	**Rec.**	**F1**	**Prc.**	**Rec.**	**F1**	**Prc.**	**Rec.**	**F1**
No drifts	1834	0.91	1.00	0.95	0.91	1.00	0.95	0.91	1.00	0.95
Sudden	11333	0.99	0.76	0.86	0.99	0.76	0.86	0.99	0.76	0.86
Gradual	11234	0.99	0.60	0.75	1.00	0.62	0.77	1.00	0.62	0.77
Incremental	2823	0.97	0.92	0.95	0.98	0.97	0.98	0.99	0.98	0.98
Recurring	2803	0.99	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99
Overall		**0.98**	**0.75**	**0.84**	**0.99**	**0.76**	**0.86**	**0.99**	**0.76**	**0.86**

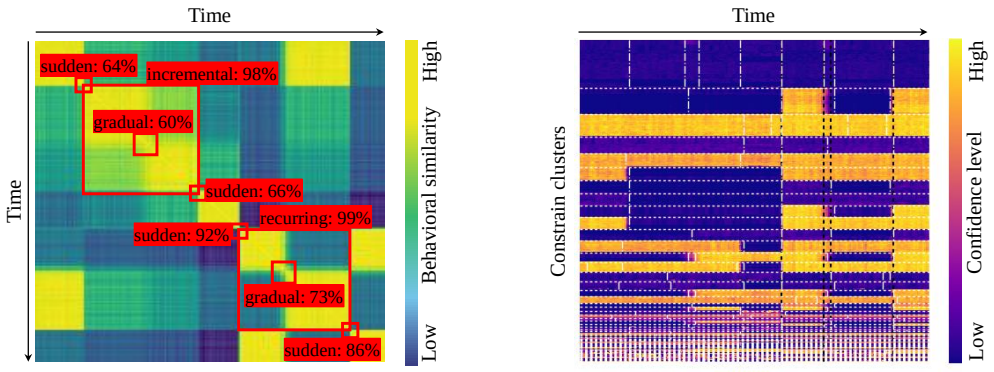
Accuracy. Table 5 presents the results obtained for different latency levels. Our `CV4CDD-4D` approach shows F1-scores ranging from 0.84 at 1% latency to 0.86 at 5% latency. For event logs without any drifts, the approach achieves a perfect recall of 1.00 and a precision of 0.91 across all latency levels. This indicates that it correctly identifies all event logs without drifts. However, in 1 out of 10 cases, the approach incorrectly detects a drift in an event log where no drift exists. For sudden and gradual drifts, precision remains high (above 0.99), but recall drops to 0.76 for sudden drifts and ranges between 0.60 and 0.62 for gradual drifts. This suggests that some actual sudden and gradual drifts, particularly gradual ones, are not detected as such. This can be attributed to the fact that our test set includes event logs with varying noise levels and process changes of different severities, making accurate detection a challenging task. Lastly, for incremental and recurring drifts, the approach achieves results above 0.92 for all measures and latency levels, indicating that it accurately detects the start and end points of these more complex drifts. Compared to simple drifts, the recall for complex drifts is notably higher. This is due to the more distinct patterns these drifts produce in an image, which makes them easier for the approach to detect. In contrast, simple drifts are more likely to be missed, particularly when the change severity is low or noise is present.

Table 6: Concept drift detection results by noise levels.

Drift	Support	W/o noise			With 30% noise			With 60% noise		
		Prc.	Rec.	F1	Prc.	Rec.	F1	Prc.	Rec.	F1
No drifts	1834	0.91	1.00	0.95	0.92	1.00	0.96	0.90	1.00	0.95
Sudden	11333	0.99	0.76	0.86	0.99	0.77	0.86	0.99	0.75	0.85
Gradual	11234	1.00	0.63	0.77	0.99	0.60	0.75	0.99	0.59	0.74
Incremental	2823	0.98	0.95	0.97	0.99	0.97	0.98	0.98	0.95	0.96
Recurring	2803	0.99	0.98	0.99	0.99	0.98	0.99	0.99	0.98	0.99
Overall		**0.99**	**0.77**	**0.86**	**0.99**	**0.76**	**0.85**	**0.98**	**0.75**	**0.84**

Noise impact. Table 6 shows the results for different noise levels. Overall, the results remain consistent across all noise levels, demonstrating that our approach is robust to noise. The only notable variation is a slight decrease in recall for gradual drifts, from 0.63 for noise-free event logs to 0.59 for logs with 60% of noise. This suggests that the relatively low recall observed in Table 5 is primarily due to the complexity of the drift scenario rather than the influence of noise.

Comparison with the baseline. Figure 8 presents the results of our approach compared to the baseline for the selected event log.



(a) Results of our CV4CDD-4D approach.

(b) Results of the VDD technique.

Fig. 8: Drift detection results for the selected log.

Our approach. Figure 8a illustrates the results of our `CV4CDD-4D` approach, highlighting the detected drifts and their corresponding confidence levels, which indicate the probability of the detected drift belonging to a specific drift type. Table 7 shows the corresponding summary of all detected drifts, including their type and respective start and end points. Based on the actual drifts and the deviations between detected and actual change points, our approach successfully identifies all drifts in this event log for a 1% latency, which allows deviations of at most 101 traces, given the log size

of 10,103 traces. The average deviation across all change points is approximately 23 traces, which is consistent with the expected deviation[9].

Table 7: Actual drifts vs. detected drifts using our approach.

Actual drifts			Detected drifts			Deviation	
Type	**Start**	**End**	**Type**	**Start**	**End**	**Start**	**End**
Incremental	1387	4668	(+) Incremental	(+) 1351	(+) 4651	−36	−17
→ Sudden	1387	1387	(+) → Sudden	(+) 1351	(+) 1351	−36	−36
→ Gradual	2986	3463	(+) → Gradual	(+) 2951	(+) 3401	−35	−62
→ Sudden	4668	4668	(+) → Sudden	(+) 4651	(+) 4651	−17	−17
Recurring	5835	9001	(+) Recurring	(+) 5801	(+) 9001	−34	0
→ Sudden	5835	5835	(+) → Sudden	(+) 5801	(+) 5801	−34	−34
→ Gradual	6947	7321	(+) → Gradual	(+) 6951	(+) 7301	4	−20
→ Sudden	9001	9001	(+) → Sudden	(+) 9001	(+) 9001	0	0

"(+)" indicates that the detected information correspond to the actual, given 1% latency.

VDD technique. Figure 8b presents the primary outcome of the baseline technique: the Drift Map. This map displays over 900 detected behavioral rules (on the y-axis), organized into 61 behavioral clusters, which are indicated by horizontal dashed white lines. The Drift Map highlights four sudden drifts indicated by vertical dashed black lines. While the first and the last drifts are correctly identified, the two other sudden drifts actually mark two change points that denote the start and end moments of a gradual drift. This gradual drift can only be identified through visual inspection of the gradual change in the confidence level in certain behavior clusters. Using the Drift Map, users can also observe that several clusters exhibit recurring behavior in the second part of the log, suggesting that detected drifts belong to a recurring drift. However, analyzing the first part of the log, which contains incremental drifts, proves to be more challenging. Although the Drift Map detects several changes across different behavioral clusters (indicated by a white dashed line within clusters), it becomes nearly impossible to conclude that these changes are part of the same incremental drift.

5.2.3 Sensitivity Analysis

Similar to the first experiment, we conclude this experiment by presenting the insights gained from the sensitivity analysis conducted on the test subset of the CDLG dataset. We consider again different number of windows between 100 and 300 and illustrate the impact of the number of windows on drift detection accuracy across three latency levels.

Figure 9 illustrates how varying the number of windows influences evaluation measures across different latency levels. Similar to the sensitivity results from Experiment 1 (for the test subset of the CDLG dataset), we again observe a slight decline in performance at both extremes of the examined number of windows. However, regardless

[9]For a log with 10,103 traces divided into 200 windows, each window contains approximately 50 traces. In cases of accurate drift detection, the expected error between the actual trace index and the first trace in the window is half the window size, i.e., 25 traces.

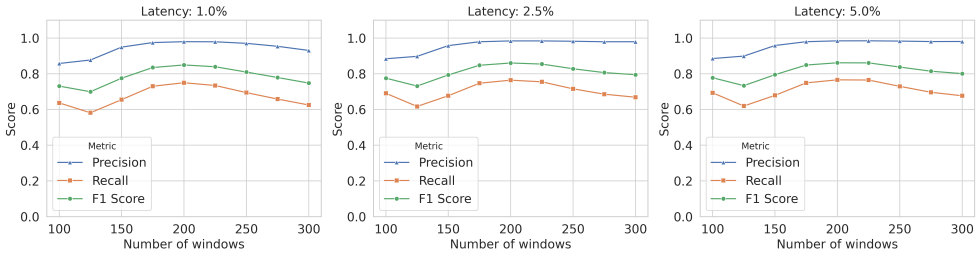


Fig. 9: Sensitivity analysis.

of latency, the evaluation measures remain within a corridor of ±5 percentage points, indicating that drift detection is also robust to the choice of the number of windows.

In summary, the evaluation results of this experiment suggest that our approach detects various types of drifts with high precision and acceptable recall across different latencies, while remaining robust to noise and choice of the number of windows. Compared to the state-of-the-art techniques, our approach represents a notable advancement towards the automated and thorough detection of concept drifts.

5.3 Experiment 3: Evaluation on Real-Life Event Logs

In this experiment, we report results obtained using our approach on real-life event logs and compare them with findings obtained in a comparable study. In the following, we discuss the evaluation setup and obtained results.

5.3.1 Evaluation Setup

We provide a summary of the characteristics of the real-life event logs and detail the configurations used in our approach.
Data collection. To allow a direct comparison of insights, we selected the same three real-life event logs used in the study by Yeshchenko et al. [58], where the authors employed their VDD technique to detect four types of drifts. The characteristics of the selected event logs are summarized in Table 8. These logs exhibit diverse characteristics in terms of the number of traces, trace variants (unique sequence of executed activities), number of events, and distinct activities.

Table 8: Characteristics of the real-life event logs.

Log name	#Traces	#Trace variants	#Events	#Unique activities
Hospital Log	1,143	981	150,291	624
Help desk Log	4,580	226	21,348	14
Sepsis Log	1,050	846	15,214	16

Configuration. To accommodate the relatively low number of traces in the selected real-life event logs, we applied our approach using an adjusted number of windows.

Specifically, we report results using 75 windows for the Hospital and Sepsis logs (which contain around 1,000 traces) and 100 windows for the Helpdesk event log (which has approximately 4,500 traces).

5.3.2 Results

Figure 10 presents the results obtained by our approach for the real-life event logs, which we compare with the findings reported using the VDD technique [58].

Hospital Log. For the Hospital event log, our method identifies a single sudden drift around June 12, 2006. This detection partially aligns with the second sudden drift reported by the VDD technique (around July 07, 2006) [58]. However, the Drift Map identified an additional sudden drift in November 2005, which our approach does not capture. Lastly, neither our approach nor the VDD tool detected any complex drifts within the log.

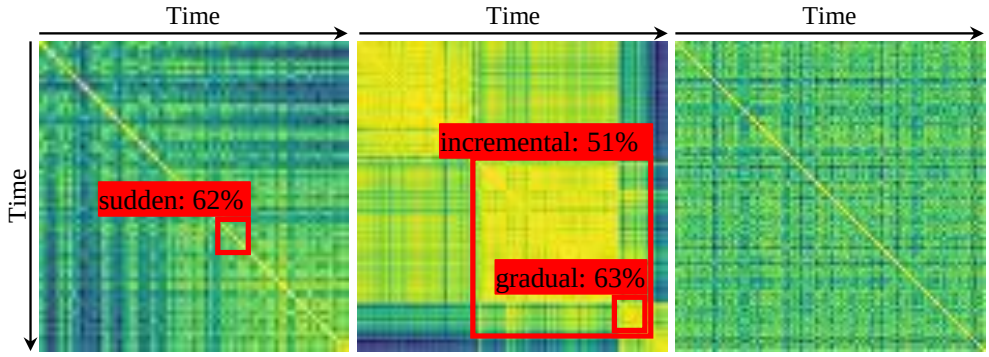


Fig. 10: Detected drifts by our `CV4CDD-4D` approach.

Help Desk Log. In the case of the Help desk event log, our approach identifies an incremental drift that begins on September 5, 2012, and concludes on July 10, 2013. Within this period, we also detect a gradual drift between June 8, 2011, and August 9, 2013, as part of the incremental drift process. The start points of these two drifts correspond to the sudden drifts detected by the VDD technique. However, according to our findings, these drifts represent change points within the detected incremental drift. For certain behavioral clusters, the VDD technique also detects gradual and incremental drifts, which are temporally aligned with the two drifts identified by our approach. Additionally, the VDD tool detects several sudden drifts in some clusters, though, as noted by the authors, these are considered outliers rather than true drifts. Our image-based representation of the event logs depicts these outliers, though our approach does not classify them as drifts.

Sepsis Log. Our approach does not detect any notable drifts in the Sepsis event log. As shown in Figure 10, the process behavior remains relatively homogeneous throughout the entire recorded period. Similarly, the VDD technique does not show

any patterns of change over time, indicating that the major clusters of behavior do not exhibit significant drifts [58]. However, the authors identify some recurring patterns in a few minor clusters, which are associated with two specific activities. While these patterns may indeed suggest recurring drifts, they could also be attributed to the specific cases within these minor clusters, as they represent only a small portion of the overall behavior.

Overall, this experiment demonstrates that our `CV4CDD-4D` approach can be effectively applied to real-life event logs, providing insights that align with and also extend the findings of the state-of-the-art technique. However, our approach automatically identifies when drifts occur and what their types are without the need of any additional interpretation of different visualizations.

6 Conclusion

In this paper, we proposed `CV4CDD-4D`, a concept drift detection approach that can detect sudden, gradual, incremental, and recurring drifts in an automated manner from event logs (offline setting). It is based on a novel idea to detect drifts in an event log using an object detection model (RetinaNet) fine-tuned with a large collection of event logs that contain known concept drifts. In the conducted experiments, we demonstrated that `CV4CDD-4D` considerably outperforms available baselines for detecting change points in event logs across several datasets, including well-established datasets commonly used to evaluate concept drift detection techniques. We also demonstrated the accuracy in detecting all four types of drifts and the robustness to noise of our approach to noise. Additionally, we highlighted its advantages in performing qualitative analysis on various real-world event logs compared to the state of the art. Finally, it is worth noting that `CV4CDD-4D` stands out not only as the first approach using techniques from computer vision for concept drift detection in process mining, but as the first approach using supervised machine learning in general.

In future work, we aim to address the limitations of our approach and enhance its capabilities. We plan to refine the annotation of sudden drifts, as the current bounding box of 5 windows may be too large for small event logs. Additionally, our goal is to develop an algorithm to determine the optimal number of windows based on event log characteristics, removing the need for user selection. We also intend to train our model using diverse data sources, moving beyond our current reliance on a single tool for generating event logs with known concept drifts. We plan to enhance our model by training it with diverse data sources and various parameter options for similarity measures and behavioral representations, thereby improving its sensitivity to drift patterns. We will enhance our capabilities in concept drift detection to encompass multiple process perspectives, including time, resources, and data. Additionally, we target including drift localization to gain insights into the changes that occur after each drift. Finally, we aim to extend further the evaluation of our approach using real-life event logs, particularly focusing on datasets with known concept drifts as they become available.

Acknowledgment

We acknowledge the work of Jonathan Kößler for conducting the initial testing of the project's idea in his master's thesis.

Statements and Declarations

Ethics approval and consent to participate

Not applicable.

Funding

No funding was received to assist with the preparation of this manuscript.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Availability of data and materials

The implementation, experimental details, and obtained raw results are available through the repository linked in the Evaluation section.

Authors' contributions

All authors contributed equally to writing and reviewing the manuscript.

References

[1] van der Aalst, W.: Process Mining: Data Science in Action. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4_1

[2] Accorsi, R., Stocker, T.: Discovering workflow changes with time-based trace clustering. In: Data-Driven Process Discovery and Analysis. pp. 154–168. Springer, Berlin, Heidelberg (2012)

[3] Adams, J.N., Pitsch, C., Brockhoff, T., van der Aalst, W.: An experimental evaluation of process concept drift detection. Proceedings of the VLDB Endowment **16**(8). https://doi.org/10.14778/3594512.3594517

[4] Adams, J.N., van Zelst, S.J., Rose, T., van der Aalst, W.: Explainable concept drift in process mining. Information Systems **114**, 102177 (2023). https://doi.org/10.1016/j.is.2023.102177

[5] Barbon Junior, S., Tavares, G.M., Da Costa, V.G.T., Ceravolo, P., Damiani, E.: A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In: Companion Proceedings of the The Web Conference (WWW '18). pp. 319–326. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3184558.3186343

[6] Bose, R.J.C., Van Der Aalst, W., Žliobaitė, I., Pechenizkiy, M.: Dealing with concept drifts in process mining. IEEE Transactions on Neural Networks and Learning Systems (TNNLS) **25**(1), 154–171 (2013). https://doi.org/10.1109/TNNLS.2013.2278313
[7] Brockhoff, T., Uysal, M.S., van der Aalst, W.: Time-aware concept drift detection using the earth mover's distance. In: 2nd International Conference on Process Mining (ICPM), Padua, Italy, October 4-9, 2020. pp. 33–40. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00016
[8] Cao, Y., Lin, L., Di, Y., Li, X., Chen, W.: Mdd: Process drift detection in event logs integrating multiple perspectives. In: 2024 IEEE International Conference on Web Services (ICWS). pp. 1125–1134 (2024). https://doi.org/10.1109/ICWS62655.2024.00135
[9] Carmona, J., Gavaldà, R.: Online techniques for dealing with concept drift in process mining. In: Advances in Intelligent Data Analysis XI: IDA 2012. pp. 90–102. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34156-4_10
[10] Ceravolo, P., Tavares, G.M., Junior, S.B., Damiani, E.: Evaluation goals for online process mining: A concept drift perspective. IEEE Transactions on Services Computing (TSC) **15**(4), 2473–2489 (2022). https://doi.org/10.1109/TSC.2020.3004532
[11] Ciccio, C.D., Mecella, M.: On the discovery of declarative control flows for artful processes. ACM Transactions on Management Information Systems **5**(4), 1–37 (2015). https://doi.org/10.1145/2629447
[12] van Der Aalst, W., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science Research and Development: CSRD 2009 **23**, 99–113 (2009). https://doi.org/10.1007/s00450-009-0057-9
[13] Elkhawaga, G., Abuelkheir, M., Barakat, S.I., Riad, A.M., Reichert, M.: CONDA-PM: a systematic review and framework for concept drift analysis in process mining. Algorithms **13**(7), 161 (2020). https://doi.org/10.3390/a13070161
[14] Gallego-Fontenla, V., Vidal, J.C., Lama, M.: A conformance checking-based approach for sudden drift detection in business processes. IEEE Transactions on Services Computing **16**(1), 13–26 (2021). https://doi.org/10.1109/TSC.2021.3120031
[15] Grimm, J., Kraus, A., van der Aa, H.: CDLG: A tool for the generation of event logs with concept drifts. In: International Conference on Business Process Management: BPM 2022 (Demos). vol. 3216, pp. 92–96. CEUR-WS (2022)
[16] Guan, W., Cao, J., Gu, Y., Qian, S.: AIMED: An automatic and incremental approach for business process model repair under concept drift. Information Systems **119**, 102285 (2023). https://doi.org/10.1016/j.is.2023.102285
[17] Hanga, K.M., Kovalchuk, Y., Gaber, M.M.: PgraphD*: Methods for drift detection and localisation using deep learning modelling of business processes. Entropy **24**(7) (2022). https://doi.org/10.3390/e24070910
[18] Hassani, M.: Concept drift detection of event streams using an adaptive window. In: 33rd International ECMS Conference on Modelling and Simulation: ECMS 2019. pp. 230–239 (Jun 2019). https://doi.org/10.7148/2019-0230

[19] Hompes, B., Buijs, J.C., van der Aalst, W., Dixit, P.M., Buurman, J.: Detecting changes in process behavior using comparative case clustering. In: Data-Driven Process Discovery and Analysis: SIMPDA 2015. pp. 54–75. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-53435-0_3
[20] Huete, J., Qahtan, A.A., Hassani, M.: PrefixCDD: Effective online concept drift detection over event streams using prefix trees. In: 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC). pp. 328–333 (2023). https://doi.org/10.1109/COMPSAC57700.2023.00051
[21] Impedovo, A., Mignone, P., Loglisci, C., Ceci, M.: Simultaneous process drift detection and characterization with pattern-based change detectors. In: 23rd International Conference on Discovery Science: DS 2020, Thessaloniki, Greece, October 19–21, 2020, Proceedings 23. pp. 451–467. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61527-7_30
[22] Klijn, E.L., Mannhardt, F., Fahland, D.: Multi-perspective concept drift detection: Including the actor perspective. In: Advanced Information Systems Engineering: CAiSE 2024. pp. 141–157. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-61057-8_9
[23] Kraus, A., van der Aa, H.: Looking for change: A computer vision approach for concept drift detection in process mining. In: International Conference on Business Process Management. pp. 273–290. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-70396-6_16
[24] Lakshmanan, G.T., Keyser, P.T., Duan, S.: Detecting changes in a semi-structured business process through spectral graph analysis. In: IEEE 27th International Conference on Data Engineering Workshops. pp. 255–260 (2011). https://doi.org/10.1109/ICDEW.2011.5767640
[25] Lin, L., Wen, L., Lin, L., Pei, J., Yang, H.: LCDD: detecting business process drifts based on local completeness. IEEE Transactions on Services Computing (TSC) **15**(4), 2086–2099 (2020). https://doi.org/10.1109/TSC.2020.3032787
[26] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2980–2988 (2017)
[27] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: Computer Vision: ECCV 2014. pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
[28] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. International Journal of Computer Vision (IJCV) **128**, 261–318 (2020). https://doi.org/10.1007/s11263-019-01247-4
[29] Liu, N., Huang, J., Cui, L.: A framework for online process concept drift detection from event streams. In: 2018 IEEE International Conference on Services Computing (SCC). pp. 105–112 (2018). https://doi.org/10.1109/SCC.2018.00021
[30] Lu, Y., Chen, Q., Poon, S.K.: A robust and accurate approach to detect process drifts from event streams. In: International Conference on Business Process Management. pp. 383–399. Springer International Publishing, Cham (2021).

https://doi.org/10.1007/978-3-030-85469-0_24
[31] Luengo, D., Sepúlveda, M.: Applying clustering in process mining to find different versions of a business process that changes over time. In: Business Process Management Workshops: BPM 2011 Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9. vol. 99, pp. 153–158. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_15
[32] Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Fast and accurate business process drift detection. In: 13th International Conference on Business Process Management: BPM 2015, Innsbruck, Austria, August 31 – September 3, 2015, Proceedings 13. vol. 9253, pp. 406–422. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_27
[33] Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Transactions on Knowledge and Data Engineering (TKDE) **29**(10), 2140–2154 (2017). https://doi.org/10.1109/TKDE.2017.2720601
[34] Manoj Kumar, M., Thomas, L., Annappa, B.: Capturing the sudden concept drift in process mining. Algorithms & Theories for the Analysis of Event Data. ATAED'15, Brussels, Belgium, June 22-23, 2015 **1371**, 132–134 (2015)
[35] Martjushev, J., Bose, R.J.C., Van Der Aalst, W.: Change point detection and dealing with gradual and multi-order dynamics in process mining. In: Perspectives in Business Informatics Research. BIR 2015. pp. 161–178. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21915-8_11
[36] Meira Neto, A.C., de Sousa, R.G., Fantinato, M., Peres, S.M.: Revisiting the transition matrix-based concept drift approach: Improving the detection task reliability through additional experimentation. SN Computer Science **5**(1), 188 (2024). https://doi.org/10.1007/s42979-023-02536-z
[37] Neto, A.C.M., de Sousa, R.G., Fantinato, M., Peres, S.M.: Towards a transition matrix-based concept drift approach: Experiments on the detection task. In: Proceedings of the 25th International Conference on Enterprise Information Systems - Volume 2: ICEIS. pp. 361–372. INSTICC, SciTePress (2023). https://doi.org/10.5220/0011843600003467
[38] Nguyen, H., Dumas, M., La Rosa, M., ter Hofstede, A.H.: Multi-perspective comparison of business process variants based on event logs. In: Conceptual Modeling. ER 2018. pp. 449–459. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00847-5_32
[39] Ostovar, A., Leemans, S.J., Rosa, M.L.: Robust drift characterization from event streams of business processes. ACM Transactions on Knowledge Discovery from Data (TKDD) **14**(3), 1–57 (2020). https://doi.org/10.1145/3375398
[40] Ostovar, A., Maaradji, A., La Rosa, M., ter Hofstede, A.H.: Characterizing drift from event streams of business processes. In: Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29. pp. 210–228. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_14
[41] Ostovar, A., Maaradji, A., La Rosa, M., ter Hofstede, A.H.M., van Dongen, B.F.V.: Detecting drift from event streams of unpredictable business processes. In:

Conceptual Modeling. ER 2016. pp. 330–346. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-46397-1_26
[42] Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D., Modugno, G.: Orange: outcome-oriented predictive process monitoring based on image encoding and cnns. IEEE Access **8**, 184073–184086 (2020). https://doi.org/10.1109/access.2020.3029323
[43] Pfeiffer, P., Lahann, J., Fettke, P.: Multivariate business process representation learning utilizing gramian angular fields and convolutional neural networks. In: International Conference on Business Process Management. pp. 327–344. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-85469-0_21
[44] Qahtan, A.A., Alharbi, B., Wang, S., Zhang, X.: A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 935–944. KDD '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2783258.2783359
[45] Richter, F., Seidl, T.: Tesseract: time-drifts in event streams using series of evolving rolling averages of completion times. In: 15th International Conference on Business Process Management: BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15. pp. 289–305. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_17
[46] Sato, D.M.V., Barddal, J.P., Scalabrin, E.E.: Interactive process drift detection framework. In: International Conference on Artificial Intelligence and Soft Computing. pp. 192–204. Springer, Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-87897-9_18
[47] Sato, D.M.V., De Freitas, S.C., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Computing Surveys **54**(9), 1–38 (2021). https://doi.org/10.1145/3472752
[48] Seeliger, A., Nolle, T., Mühlhäuser, M.: Detecting concept drift in processes using graph metrics on process graphs. In: Proceedings of the 9th Conference on Subject-Oriented Business Process Management. S-BPM ONE '17, vol. 9, pp. 1–10. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3040565.3040566
[49] Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on synthesis from well-structured behavioral profiles. International Journal of Cooperative Information Systems **21**(01), 55–83 (2012). https://doi.org/10.1142/S0218843012400035
[50] de Sousa, R.G., Peres, S.M., Fantinato, M., Reijers, H.A.: Concept drift detection and localization in process mining: an integrated and efficient approach enabled by trace clustering. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. p. 364–373. SAC '21, ACM, New York, NY, USA (2021). https://doi.org/10.1145/3412841.3441918
[51] Stertz, F., Rinderle-Ma, S.: Process histories-detecting and representing concept drifts based on event streams. In: On the Move to Meaningful Internet Systems: OTM 2018 Conferences, Valletta, Malta, October 22-26, 2018, Proceedings, Part I. pp. 318–335. Springer (2018). https://doi.org/10.1007/978-3-030-02610-3_18

[52] Stertz, F., Rinderle-Ma, S.: Detecting and identifying data drifts in process event streams based on process histories. In: Information Systems Engineering in Responsible Information Systems: CAiSE Forum 2019, Rome, Italy, June 3–7, 2019, Proceedings 31. pp. 240–252. Springer (2019). https://doi.org/10.1007/978-3-030-21297-1_21

[53] Tavares, G.M., Ceravolo, P., Turrisi Da Costa, V.G., Damiani, E., Barbon Junior, S.: Overlapping analytic stages in online process mining. In: 2019 IEEE International Conference on Services Computing. pp. 167–175 (2019). https://doi.org/10.1109/SCC.2019.00037

[54] Van Der Aalst, W., et al.: Process mining manifesto. In: Business Process Management Conference. BPM Workshop. pp. 169–194. Springer (2011). https://doi.org/10.1007/978-3-642-28108-2_19

[55] Weber, P., Bordbar, B., Tino, P.: Real-time detection of process change using process mining. In: Imperial College Computing Student Workshop. vol. DTR11-9, pp. 108–114. Citeseer, Imperial College London (2011)

[56] Xu, J., Zhang, Y., Duan, Q.: Concept drift detection and localization framework based on behavior replacement. Applied Intelligence **53**(13), 16776–16796 (2023). https://doi.org/10.1007/s10489-022-04341-2

[57] Yang, L., McClean, S., Donnelly, M., Burke, K., Khan, K.: Process duration modelling and concept drift detection for business process mining. In: IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation. pp. 653–658 (2021). https://doi.org/10.1109/SWC50871.2021.00097

[58] Yeshchenko, A., Di Ciccio, C., Mendling, J., Polyvyanyy, A.: Visual drift detection for event sequence data of business processes. IEEE Transactions on Visualization and Computer Graphics **28**(8), 3050–3068 (2021). https://doi.org/10.1109/TVCG.2021.3050071

[59] Zellner, L., Richter, F., Sontheim, J., Maldonado, A., Seidl, T.: Concept drift detection on streaming data with dynamic outlier aggregation. In: Process Mining Workshops. ICPM 2020. pp. 206–217. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-72693-5_16

[60] Zheng, C., Wen, L., Wang, J.: Detecting process concept drifts from event logs. In: On the Move to Meaningful Internet Systems: OTM 2017 Conferences. pp. 524–542. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-69462-7_33