OPTWIN: Drift Identification with Optimal Sub-Windows

1st Mauro D. L. Tosi
Department of Computer Science
University of Luxembourg
Esch-sur-Alzette, Luxembourg
https://orcid.org/0000-0002-0218-2413

2nd Martin Theobald
Department of Computer Science
University of Luxembourg
Esch-sur-Alzette, Luxembourg
https://orcid.org/0000-0003-4067-7609

***Abstract*—Online Learning (OL) is a subfield of Machine Learning (ML) that is increasingly gaining attention in academia and industry. A long-standing challenge in OL is the presence of *concept drifts*, which are commonly defined as unforeseeable changes in the statistical properties of an incoming data stream over time. State-of-the-art concept-drift detectors however still exhibit *high false-positive rates* in their drift identification, which then leads to an undue amount of computational resources spent by the underlying ML algorithm on retraining its model.**

In this paper, we propose OPTWIN, our "OPTimal WINdow" concept drift detector suited for both classification and regression problems. The novelty of OPTWIN lies in identifying where to split a sliding window W of error rates produced by an ML model into two provably optimal sub-windows, such that the split occurs at the earliest event at which a statistically significant difference according to either the t- or the f-tests occurred. Specifically, OPTWIN reaches this result by (1) considering *both the mean and the variance* of the error rates, and (2) improves the cost of detecting this optimal split from $\mathcal{O}(\log |W|)$ to $\mathcal{O}(1)$ per iteration. We assessed OPTWIN over the MOA framework, using ADWIN, DDM, EDDM, STEPD, and ECDD as baselines over 12 synthetic and real-world datasets with both sudden and gradual concept drifts. In our experiments, OPTWIN surpasses the F1-score of the baselines in a statistically significant manner while maintaining a lower detection delay and saving up to 21% of time spent on retraining the models.

***Index Terms*—concept drift, drift detection, data streams**

I. Introduction

Online Learning (OL) is an area of research which gained an increasing amount of attention in academia and industry over the past years. OL is a sub-field of Machine Learning (ML) in which an underlying ML technique aims to constantly update its model's parameters from an incoming data stream under limited time and space constraints. Ideally, OL methods are trained in real-time and thereby aim to maximize the prediction accuracy of their models based on bounded windows of data instances they have previously seen and which they may see in the near future [1]. Use-cases of OL are varied and include online video segmentation [2], spam detection [3], fraud detection [4], and many others [5]–[7].

The presence of *concept drifts* is one of the numerous challenges when processing data streams in real-time. Concept drifts are commonly defined as unforeseeable changes in the statistical properties of the incoming data stream over time [8]. Such a change may have different sources and types, and it may involve different adaptation strategies (cf. Section II). In practice, concept drift is the key phenomenon that impairs the performance of pre-trained ML models over data streams.

The most common form of drift detection is the *error rate-based* one. This type of drift detection uses the difference between the number or magnitude of historical prediction errors and new prediction errors to determine if a concept drift occurred [8]. The most popular error rate-based algorithms are ADWIN [9], DDM [10], and their variations as [11], [12]. Those and newer drift detectors [13], [14] are known for being lightweight and identifying concept drifts with an outstanding true-positive rate. However, those algorithms still have a high false-positive rate. Additionally, most of them work with classification problems only, with few being suited for regression problems. Therefore, as concluded by Bayram et al. [15], the development of drift detectors, which return a low false-positive rate and are suited for both classification and regression, is an open research problem in this area.

As opposed to previous approaches, which track concept drifts based on the *means* of past prediction errors, we argue that also changes in the *standard deviations* of those errors should trigger concept drifts. Take, as a simple example, a regressor that outputs the following errors at window W_0:

$$W_0 = \langle 0.3; 0.7; 0.7; 0.3; 0.3; 0.7; 0.5; 0.5 \rangle$$

While, at window W_1, it outputs the following errors:

$$W_1 = \langle 0.0; 1.0; 1.0; 0.0; 1.0; 0.0; 0.0; 1.0 \rangle$$

Current drift detectors like ADWIN would not consider this as a concept drift, as the error means μ_{W_0}, μ_{W_1} over both windows are equal to 0.5, whereas we—intuitively—understand that something changed in the error stream and, therefore, a concept drift should be flagged.

In this paper, we propose the "OPTimal WINdow" drift detector (OPTWIN). It is a sliding-window algorithm that analyzes the error rates (either binary or real-valued) produced by an underlying ML algorithm. It calculates the *optimal cut* of a sliding window W to divide it into two sub-windows W_{hist} and W_{new}. It then performs the well-known t- (for means) and f-tests (for standard deviations) to determine whether the

This work is funded by the Luxembourg National Research Fund under the PRIDE program (PRIDE17/12252781).

two sub-windows exhibit a statistically significant difference in either their means or standard deviations, respectively. OPTWIN has the following two main features: (1) it can calculate the *optimal cut* based only on the length of W in $\mathcal{O}(1)$; (2) it uses both the errors' means and standard deviations to flag drifts. Furthermore, we provide detailed and rigorous guarantees of OPTWIN's performance in terms of its true-positive (TP), false-positive (FP), and false-negative (FN) rates, as well as in its drift-identification delay.

To assess OPTWIN, we used the popular MOA framework [16] and compared OPTWIN to the most studied drift-detection frameworks in the literature: ADWIN [9], DDM [10], EDDM [11], STEPD [17] and ECDD [18]. Using MOA, we compared the *precision*, *recall* and *F1-score*, as well as the *delay* of all drift detectors over both sudden and gradual drifts. Furthermore, we also trained a Naive Bayes (NB) classifier on MOA that adapts itself based on the drift detectors and compared its results on various synthetic datasets (STAGGER [19], RANDOM RBF [20], and AGRAWAL [21]) and real-world ones (Covertype and Electricity) [22], [23]. Additionally, we likewise assessed OPTWIN on a Neural Network (NN) use-case. Specifically, we pre-trained a Convolutional NN (CNN) with the CIFAR-10 [24] image dataset and simulated an end-to-end OL scenario with concept drifts by using OPTWIN and ADWIN as drift detectors.

Our results show that OPTWIN reliably identifies both sudden and gradual drifts in classification and regression problems. Furthermore, OPTWIN is the drift detector with the best F1-score when compared to the baselines. This is due to its higher precision, which indicates a low FP rate. With respect to the drift-identification delay, there was no drift detector that was superior to the others in a majority of the datasets. In terms of runtime, OPTWIN in combination with the CNN training pipeline is 21% faster than a similar ADWIN pipeline due to OPTWIN's lower FP rate, which leads to a significantly reduced amount of re-training iterations. Therefore, our experiments indicate that OPTWIN identifies concept drifts with a similar delay as other drift detectors but maintains a higher precision and recall in the drift detection, which reduces the overall runtime of OL pipelines that trigger a re-training of their models upon each detected concept drift.

II. Background

A concept drift may exhibit different characteristics, whose understanding is essential for a fast and reliable drift detection. Consider an unbounded data stream that receives *features* $x_i \in X$ and *labels* $y_i \in Y$ in the form of pairs of instances (x_i, y_i) used for training an underlying ML model. At *time* t, these instances follow a certain distribution $P_t(X, Y)$. A concept drift is the change of this distribution over time. Formally, a concept drift occurs at time $t+1$ iff $P_t(X, Y) \neq P_{t+1}(X, Y)$ [8], [25], [26]. The underlying data distribution $P_t(X, Y)$ may also be decomposed and expressed as a product of probabilities $P_t(X, Y) = P_t(X) \times P_t(Y|X)$ via the well-known *chain rule of probability*. Thus, a concept drift may come from two *sources*: (1) $P_t(X) \neq P_{t+1}(X)$, which is known as a *virtual drift* because it does not impact the decision boundaries of the learner; and (2) $P_t(Y|X) \neq P_{t+1}(Y|X)$, which is known as an *actual drift* because it directly impacts the learner's accuracy [8], [25], [26]. It is also possible to have both drift sources simultaneously [8].

Another important characteristic of a concept drift is its *type*. Concept drifts can be (1) *sudden*; (2) *incremental*; (3) *gradual*; and (4) *reoccurring* (see Figure 1) [8], [26]. Sudden drifts occur when the probability distribution changes completely within a single step. Incremental drifts occur when the distribution $P_t(X, Y)$ changes incrementally until its convergence. Gradual drifts occur when the new distribution gradually replaces the old one. Reoccurring drifts occur when distributions can reoccur after some time.

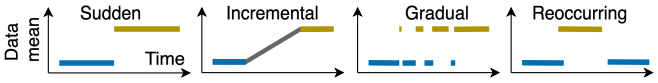

Fig. 1. Concept drift types.

To recover from concept drifts, one may adapt the learner according to different strategies. The selection of the appropriate strategy depends not only on the type and source of the drifts but also on the use-case and the adopted learner. A common strategy is to train a new model with the latest data points whenever a drift is identified. Another strategy is to adjust the current learner instead of training a new one from scratch. Moreover, it is also possible to have an ensemble of learners, which is useful for adapting to reoccurring drifts [8].

The most popular procedure to evaluate learning algorithms that handle concept drifts is the *prequential procedure*. Prequential is an evaluation scheme in which each data point is used for testing before training the learning algorithm. Therefore, it is not necessary to know when a drift occurred to perform the evaluation, which is useful when using real-world datasets that do not have labeled drifts [8]. Regarding the metrics used to evaluate the drift detector, one may refer to the following ones: TPs, FPs and FNs (and hence *precision*, *recall* and *F1-score*) in the detection rates, as well as the *delay* of the drift detection [8], [27].

III. OPTWIN – Optimal Window Concept Drift Detector

OPTWIN is the new concept-drift detector which we propose in this paper. It is an error rate-based drift detector, which tracks the error rates produced by an OL learner in a sliding window W. Its name stands for "OPTimal WINdow" due to its calculation of the optimal split of the sliding window. We consider this split "optimal" because it is detected for the first element in W at which a statistically significant difference between either the means or the variances of the error rates (based on the common t- and the f-tests) among two sub-windows of W occurs. Specifically, W keeps growing until either a concept drift is detected or a maximum user-defined size w_{max} is reached. Based on the sliding-window size $|W|$ and a pre-defined confidence level δ, OPTWIN calculates the

optimal point to divide W into "historical" (W_{hist}) and "new" (W_{new}) data points.

A. Setup & Parameters

Without loss of generality, we consider a typical OL scenario, in which OPTWIN receives as input a sequence of real numbers $x_1, x_2, ..., x_i, ...$ from a possibly unbounded data-stream X. Thus, the algorithm is assumed to access only one element x_i at time i from the stream, and it buffers previously seen elements in a *sliding window* $W \subset X$ of consecutive events. Moreover, OPTWIN requires as parameters (1) δ, the *confidence level* for the concept-drift detection; (2) w_{max}, the *maximum size* of the sliding window W; and (3) ρ, a parameter we refer to as *robustness*, which we define as the minimum ratio by which $\mu_{W_{new}}$ has to vary in relation to $\sigma_{W_{hist}}$ in order for OPTWIN to consider this variation as a concept drift. The intuition behind ρ is to enable users to specify the expected magnitude of concept drifts relative to the data's standard deviation instead of its raw value, simplifying its definition and reducing dependency on problem types and data magnitude.

B. Assumptions

- There is no concept drift within the first w_{min} data instances from X (which is needed to initialize OPTWIN).
- A concept drift occurs when the means or the standard deviations within two sub-windows of W are statistically different.
- Within any two sub-windows of W, the values produced by the test statistic of the unequal-variance t-test [28] (henceforth called "t_value") follow a t-distribution.
- Within any two sub-windows of W, the values produced by the test statistic of the f-test [29] (henceforth called "f_value") follow an f-distribution.

Our first assumption mitigates the negative impact of outliers when only a few data points are available. The second assumption points out when we expect OPTWIN to identify concept drifts. The third and the fourth assumptions are generally required to guarantee the validity of the t- and f-tests, respectively, but do not impose limitations on the values ingested from the data stream in practice.

C. Algorithm

As seen in Algorithm 1, OPTWIN first needs to be initialized by creating an empty sliding window W. Analogously to other drift detectors, it collects w_{min} many first elements from the stream (usually within 30 to 50) to guarantee that it has a minimum amount of data to output statistically relevant drift detections. It also defines a small constant $\eta = 1e^{-5}$, which avoids a division by 0 when added to the standard deviations during the calculation of the f-test.

Second, the ADDELEMENT procedure is called once to process each data element received from the data stream. The procedure starts by inserting the most recent data element x_i into W and by checking whether W has already reached the minimum amount of elements to detect a possible concept drift. If so, it checks if W increased above w_{max} and removes the oldest element from W if necessary. Up to this point, OPTWIN performs standard queuing operations to maintain its sliding window bounded between w_{min} and w_{max} elements.

Algorithm 1 OPTWIN

Input parameters:
- δ – confidence level
- ρ – robustness
- w_{max}– max window size

Global variables:
$W = \langle\rangle$ – sliding window
$w_{min} = 30$ – min window size
$\eta = 1e^{-5}$ – avoids division by 0

```
1:  procedure ADDELEMENT(x_i)
2:      W ← W ∪ x_i
3:      if |W| < w_min then
4:          return False
5:      else if |W| ≥ max_lenght then
6:          W ← W − W_0
7:      ν ← OPTIMALCUT(|W|, ρ, δ^(1/4))                cf. Equation (1)
8:      ν_split ← ⌊ν |W|⌋
9:      W_hist ← W_{0:ν_split}
10:     W_new ← W_{ν_split:|W|−1}
                                                       // f-test
11:     if (σ_Wnew + η)^2 / (σ_Whist + η)^2 > f_ppf(δ^(1/4), ν|W| − 1, (1 − ν)|W| − 1)
        then
12:         reset()
13:         return True
                                                       // t-test
14:     else if t_value(W_hist, W_new) > t_ppf(δ^(1/4), df) then
15:         reset()
16:         return True
```

Then, OPTWIN calculates ν, which is the optimal splitting point of W into W_{hist} and W_{new}, by solving Equation 1 for the highest value of ν in terms of δ', ρ and $|W|$:

$$\rho = t\_ppf(\delta', df)\sqrt{\frac{1}{\nu|W|} + \frac{f\_ppf(\delta', \nu|W|-1, (1-\nu)|W|-1)}{(1-\nu)|W|}} \tag{1}$$

where ρ is the aforementioned, user-defined robustness parameter. During this calculation, OPTWIN uses t_ppf and f_ppf, which are the Probability Point Functions (PPF) of the t- and the f-distributions, respectively. To calculate t_ppf, we apply a confidence of $\delta' = \delta^{\frac{1}{4}}$, which considers the application of the two tests to calculate ν in Equation 1 and the two tests on Lines 11 and 14 in Algorithm 1.

Finally, OPTWIN performs the t- and f-tests (in any order) to compare both sub-windows and determine if their means and standard deviations, respectively, belong to the same distribution. If not, a concept drift is flagged and the algorithm is reset. Otherwise, the ADDELEMENT method is called for the next element from the data stream in an iterative manner.

The usage of the t- and f-tests to identify significant changes in the means and standard deviations among series of data values is well-established. However, OPTWIN's novelty comes from the combination of both tests to calculate ν, and thereby identify where to optimally divide this series of values to perform both tests. In short, by solving Equation 1 in terms of ν, we determine the minimum size of W_{new} (the optimal splitting point of W) that statistically guarantees the identification of any concept drift with a robustness of at least ρ when using the t- and f-tests, thereby achieving lower

drift-detection delays than other approaches. To calculate ν, other than the confidence level δ, the only values that shall be inputted by the user are w_{max} and ρ. Regarding ρ, when selecting a small value, one may expect to identify smaller drifts in exchange of a higher drift-detection delay. On the other hand, with higher values of ρ, one can expect a smaller detection delay in exchange of missing smaller drifts. In practice, ρ shall be set based on the magnitude and frequency of drifts expected. However, determining the correct ρ to each use case should not be a difficult task, since different ρ's tend to produce similar results (as seen in Section IV). Regarding w_{max}, with a higher value, one may expect smaller drift-detection delays in exchange for more memory usage. In practice, when using $\rho = 0.1$, we did not observe a significant variation in $|W_{new}|$ even when increasing w_{max} to more than 25,000 elements.

One important point to note is that we can only calculate ν as the optimal splitting point if $|W| \geq w_{proof}$, with w_{proof} representing the minimum size of W in which there exists a ν that solves Equation 1. Otherwise, we set ν to the middle of W until it grows to the minimum size needed to solve Equation 1. Thus, if it exists, it is defined as the highest root of Equation 1. Otherwise, it is set to $\nu = 0.5$.

Below, we present Theorem 3.1, our main theoretical result. It gives guarantees in terms of OPTWIN's false positive (FP) and false negative (FN) bounds for identifying concept drifts. Theorem's 3.1 proof is available in the extended version of this paper [30].

Theorem 3.1:

- **False Positive Bound.** At every step, if μ_W and σ_W^2 remain constant within W, OPTWIN will flag a concept drift at this step with a confidence of at most 1-δ.
- **False Negative Bound** (for mean drift with large enough W). For any partitioning of W into two sub-windows $W_{hist}\,W_{new}$, with $|W| \geq w_{proof}$ and W_{new} containing the most recent elements, if $\mu_{hist} - \mu_{new} > \rho\,\sigma_{hist}$, then, with confidence δ, OPTWIN flags a concept drift in at most $|W| - \nu_{split}$ steps.
- **False Negative Bound** (for mean drift with small W). For any partitioning of W into two sub-windows $W_{hist}\,W_{new}$, with $w_{min} \leq |W| < w_{proof}$ and W_{new} containing the most recent elements, if $\mu_{hist} - \mu_{new} > \rho_{temp}\,\sigma_{hist}$, then, with confidence δ, OPTWIN flags a concept drift in at most $\frac{|W|}{2}$ steps.
- **False Negative Bound** (for standard-deviation drift with any W). For any partitioning of W into two sub-windows $W_{hist}\,W_{new}$, with $|W| \geq w_{min}$ and W_{new} containing the most recent elements, if $\frac{\sigma^2_{new}}{\sigma^2_{hist}} > f\_ppf(\delta', \nu\,|W| - 1, (1-\nu)\,|W| - 1)$, then, with confidence δ, OPTWIN flags a concept drift in at most ν_{split} steps.

D. Implementation & Analysis

We implemented OPTWIN via a combination of Java and Python scripts as extensions of the MOA [16] (Java) and the River [31] (Python) libraries (both available on Github[1]). We pre-calculated the values of ν, t_ppf, and f_ppf based on a fixed confidence value of $\delta = 0.99$ and for $w_{max} = 25{,}000$, thus $30 \leq |W| \leq 25{,}000$. All pre-calculated variables were stored in lists indexed by the window sizes $|W|$ from which they were calculated. This is possible because those variables, as seen in Equation 1, do not depend on the actual error distribution. Thus, it is not necessary to calculate them in real-time. On the other hand, we need to store the sliding window W plus the other three lists of floating point values of size up to w_{max} in memory, which takes a maximum of $w_{max} * 4 * 4$ bytes. Thus, for $w_{max} = 25{,}000$, OPTWIN would require only around 390 KB of memory.

Furthermore, the full calculation of the means and standard deviations from W_{hist} and W_{new} can be avoided. Instead of calculating them from scratch, one only needs to update them incrementally. Moreover, as W is bounded by w_{max}, we can use a circular array to make insertions at the end of the array, deletions from the beginning of the array, and then look up each value in $\mathcal{O}(1)$ time per iteration. Therefore, the ADDELEMENT procedure has an overall computational complexity of $\mathcal{O}(1)$ per element ingested from the data stream (assuming a constant cost for the numerical operations involved in resolving Equation 1 to ν). Our analytical approach thus provides great potential runtime gains over the iterative search procedure applied, e.g., by ADWIN, which requires $\mathcal{O}(\log |W|)$ computations per iteration.

By default, OPTWIN's Algorithm 1 tracks concept drifts that either increase or decrease the means and standard deviations of the variables tracked. However, in an OL scenario, we usually want to update the learner only when the number of errors increases. Therefore, in our implementation, we check if also $\mu_{new} \geq \mu_{hist}$ along with the statistical tests on Lines 11 and 14 of Algorithm 1. In doing so, we consider that a drift occurred only if μ_{new} is higher than μ_{hist} (i.e., when the learner actually decreased in performance).

IV. EXPERIMENTS & RESULTS

In this section, we report our detailed experiments to assess OPTWIN. We compared OPTWIN to the most commonly used baselines for concept-drift detection: ADWIN, DDM, EDDM, STEPD, and ECDD. We additionally investigated more recent techniques [12]–[14] but found them transitively compared to the classical techniques mentioned above. That is because they address specific frailties of the classical drift detectors like recurrent drifts and decreased sensitivity when concepts are long, often without statistical improvements over them.

We performed most of the experiments using the common MOA [16] framework which is a Java-based data stream simulator. In addition, we performed experiments on Python to assess OPTWIN on a Neural Network (NN) use case, which would not be possible via MOA.

[1]https://github.com/maurodlt/optwin

Detector	Avg. Delay	Avg. FP	Avg. F1
ADWIN	132.19	4.39	67%
DDM	569.37	0.73	86%
EDDM	1127.56	8.22	49%
STEPD	196.97	34.93	30%
ECDD	131.84	67.22	37%
OPTWIN$_{\rho=0.1}$	156.13	**0.17**	**95%**
OPTWIN$_{\rho=0.5}$	**120.98**	0.19	**95%**
OPTWIN$_{\rho=1.0}$	414.29	0.32	89%

TABLE I
AVERAGE STATISTICS OF DRIFT IDENTIFICATION ON THE FOLLOWING SYNTHETIC SETTINGS: GRADUAL BINARY DRIFT, GRADUAL NON-BINARY DRIFT, SUDDEN BINARY DRIFT, SUDDEN NON-BINARY DRIFT, SUDDEN STAGGER, SUDDEN RANDOM RBF, AND SUDDEN AGRAWL.

A. MOA Experiments

We compared 3 configurations of OPTWIN with the default configurations of our baselines. All OPTWIN configurations had $\delta = 0.99$, $w_{max} = 25{,}000$, and pre-computed values for ν, t_ppf, and f_ppf (as described on Section III-D). The difference among the OPTWIN configurations is only on ρ, which we varied between 0.1, 0.5, and 1.0 to better understand how our robustness parameter affects OPTWIN in practice.

We performed two types of experiments on MOA. The first one uses the "Concept Drift" interface, in which MOA creates a stream of data points (either binary or non-binary) and produces a concept drift that can be sudden or gradual. We later refer to those experiments according to their data input and their drift type. The second type of experiment uses the "Classification" interface. It generates data streams based on both synthetic datasets (STAGGER, RANDOM RBF, and AGRAWL) [19]–[21] and real-world ones (Electricity and Covertype) [22], [23]. The idea of these experiments is to train a classifier that is reset every time a concept drift is detected by a drift detector. We chose MOA's built-in Naive Bayes (NB) classifier for its simplicity, which facilitates the analysis of the drift-detection results. For the synthetic datasets, we generate data streams with 100,000 data points with drifts occurring every 20,000 data points (either sudden or gradual). For the real-world data sets, the drifts are already present and have an unknown location on the stream.

Table I presents the average results comparison among drift detectors on the above-mentioned configurations. We repeated each experiment 30 times and compared their average TP, FP and FN rates to compute their micro-average precision, recall, and F1-score, along with their average drift-detection delay (in terms of the number of streamed elements between the occurrence of a known concept drift and its identification by the detector). Note that we did not include in Table I the "Classification" experiments on real-world datasets nor the ones with gradual concept drifts. For the real-world datasets, it is not possible to calculate the above-mentioned metrics without knowing the drifts' locations. For the gradual drifts, we observed a divergence between the starting and ending points of those drifts in the MOA documentation and in practice. Therefore, we did not include those in our comparison.

Based on the results in Table I, we can observe a higher F1-score for OPTWIN when compared to the baselines. We assert this due to OPTWIN's low FP rate which produces a high precision and F1-score, respectively. We further compared the F1-scores of all configurations of OPTWIN with the two drift detectors that can be used for regression problems (ADWIN and STEPD), which showed OPTWIN to be superior based on a one-tailed Wilcoxon signed-rank test with $\alpha = 0.05$ in a statistically significant manner.

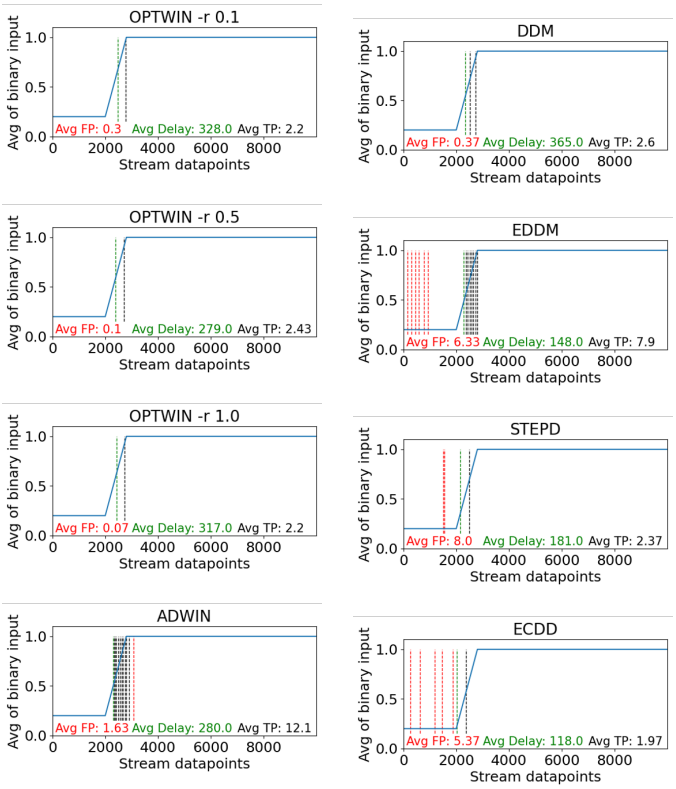

Fig. 2. Gradual binary drift detection with average TP and FP rates compared to drift-detection delays.

Regarding the drift-detection delay, OPTWIN with $\rho = 0.5$ is the drift detector with the smallest delay, taking on average 121 iterations to detect a concept drift. Observe that, the higher the FP rate of a drift detector, the higher are also its chances of identifying a drift earlier. Therefore, when analyzing ECDD's average drift delay of 131 iterations, we have to consider that it had an average of 67 FPs per run, in contrast to an average of less than 0.4 for any of the OPTWIN configurations. Furthermore, considering that DDM, EDDM, and ECDD depend on binary data, they were not included in the experiments using "non-binary" datasets.

We can better visualize the difference between the drift detectors in Figure 2, which represents one of the 30 runs of the "gradual binary drift" configuration (we selected the run with results closest to the average ones). In Figure 2, we can see the high FP rate of the EDDM, STEPD and ECDD detectors when compared to OPTWIN, DDM and ADWIN.

B. Classification Experiments

In the "Classification" experiments (still using MOA as platform), we can analyze the average accuracy achieved by the NB classifier when varying the drift detectors (cf. Table II). The idea is that the better a drift detector's performance, the better the classifier can adapt to the concept drifts, thus generating a higher prediction accuracy. However, the fast and accurate detection of the drifts did not always traverse to a better accuracy for the classifier. We can observe this by comparing Tables I and II, in which experiments using drift detectors that produced a low F1-score in Table I achieved a good accuracy in Table II. Moreover, the drift detectors with the best accuracy on real-world data sets were the ones that detected more drifts, which is why ECDD achieved such good accuracy (being the detector with the higher amount of FPs). For example, ECDD detected 426 drifts on the Electricity dataset—over twice the amount of other algorithms.

Drift Detector	Synthetic	Real-world
None	60.98	66.94
ADWIN	79.67	81.25
DDM	77.72	84.60
EDDM	77.06	85.45
STEPD	79.59	86.01
ECDD	78.28	88.46
OPTWIN$_{\rho=0.1}$	79.65	81.64
OPTWIN$_{\rho=0.5}$	79.48	84.46
OPTWIN$_{\rho=1.0}$	79.25	84.64

TABLE II
AVERAGE ACCURACY OF NB ON SYNTHETIC'S SUDDEN AND GRADUAL (STAGGER, RANDOM RBF, AND AGRAWL) AND REAL-WORLD (ELECTRICITY AND COVERTYPE) DATASETS.

C. Regression Experiments on Neural Networks

To further explore OPTWIN's behavior in a regression scenario, we compared it with ADWIN for identifying drifts from the loss of a CNN. We chose ADWIN as our baseline because it was the drift detector with the best F1-score and smallest drift-identification delay among the ones that do not require binary inputs (thus excluding DDM, EDDM, and ECDD). To simplify the reader's understanding of the generation of the concept drift, instead of training a regression problem on a CNN, we here focused on image classification by swapping the labels of the images. Thus, we provoked 4 concept drifts by swapping the labels of two classes of images every 20% of the simulated data stream. For example, after 62,480 iterations, we swapped the labels between images from "cats" to "horses". Nevertheless, as the drift detectors track the loss of the CNN, the type of the problem should not affect the experiment.

We pre-trained an image classification model [32] using the CIFAR-10 [24] data set during 100 epochs, achieving an average of 89% training accuracy over 3 different runs. Then, we simulated an OL scenario with concept drifts. Our data stream was formed by batches of 32 images from the CIFAR-10 dataset; with a total of 312,400 data points (equivalent to 100 epochs). We simulated our 4 concept drifts by swapping the labels of two classes every 62,480 iterations (the equivalent of 20 epochs). At every iteration, the model classified the 32 images and outputted the loss of the batch. We inputted this loss into the drift-detection algorithm. If a drift was detected, the next 9,372 batches of images (the equivalent of 3 epochs) were used for fine-tuning the model (thus adapting it to the concept drift). Therefore, the goal was for the drift detector to identify the 4 concept drifts that we simulated, triggering the fine-tuning of the model for a total of 12 epochs.

In Figure 3, we compare OPTWIN and ADWIN under the setting described above. First, we note that ADWIN's high FP rates made it re-train the model for much longer than OPTWIN. In comparison, ADWIN identified 15 concept drifts (with 11 FPs), thus triggering model fine-tuning for 61,562 iterations which took in total 945 seconds. In contrast, OPTWIN identified just 5 drifts (with 1 FP), thus triggering the model fine-tuning for 23,430 iterations which took 781 seconds. We note that OPTWIN's running time per iteration is superior to ADWIN, $1e^{-5}$ against $6e^{-6}$ seconds. However, OPTWIN still ends up speeding up the OL pipeline whenever re-training is triggered by the concept-drift detection (21% faster in this use-case). This is because the training of a learner is usually computationally more expensive than the drift detection. Thus, with fewer FPs, the total training time can be reduced substantially.

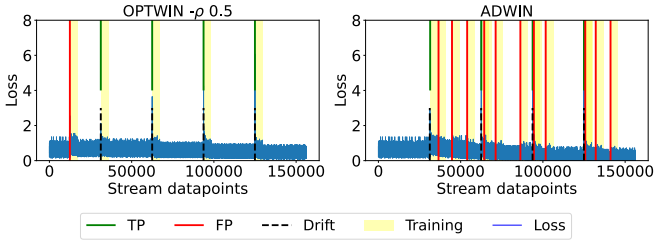

Fig. 3. Sudden drift detection over the loss of a CNN.

V. CONCLUSION

In this paper, we presented OPTWIN, a novel concept-drift detector that uses a sliding window of errors to identify concept drifts in both classification and regression problems with a low FP rate. OPTWIN's novelty relies on the assumption that changes also in the variances of the elements ingested from a data stream can be an indication of concept drift. This assumption lets it optimally divide its sliding window in $\mathcal{O}(1)$ time by applying the t- and f-tests to determine whether a concept drift occurred. To assess OPTWIN, we compared it with 5 popular drift detectors in 11 different experiments. As a result, OPTWIN achieved higher F1-scores in most of our experiments. In fact, OPTWIN had the best F1-score (with statistical significance) while maintaining a similar drift-detection delay compared to the drift detectors suited for both classification and regression problems. Moreover, OPTWIN could speed up the overall OL pipeline of a CNN by 21% (compared to ADWIN) due to its low FP rate. To conclude, we conjecture that OPTWIN's characteristics can enable even higher speed-ups in scenarios where the drift identification triggers the re-training of complex models.

References

[1] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[2] R. Koner, T. Hannan, S. Shit, S. Sharifzadeh, M. Schubert, T. Seidl, and V. Tresp, "InstanceFormer: An Online Video Instance Segmentation Framework," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, pp. 1188–1195, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25201

[3] F. Fdez-Riverola, E. L. Iglesias, F. Díaz, J. R. Méndez, and J. M. Corchado, "Applying lazy learning algorithms to tackle concept drift in spam filtering," *Expert Systems with Applications*, vol. 33, no. 1, pp. 36–48, 2007.

[4] T. Paladini, M. Bernasconi de Luca, M. Carminati, M. Polino, F. Trovò, and S. Zanero, "Advancing Fraud Detection Systems Through Online Learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 275–292.

[5] K. Wang, L. Xu, A. Taneja, and M. Tambe, "Optimistic Whittle Index Policy: Online Learning for Restless Bandits," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, pp. 10 131–10 139, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26207

[6] G. Li, P. Wang, Q. Luo, Y. Liu, and W. Ke, "Online Noisy Continual Relation Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, pp. 13 059–13 066, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26534

[7] D. Mandal, G. Radanovic, J. Gan, A. Singla, and R. Majumdar, "Online Reinforcement Learning with Uncertain Episode Lengths," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, pp. 9064–9071, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26088

[8] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

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

[10] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[11] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.

[12] R. S. Barros, D. R. Cabral, P. M. Gonçalves Jr, and S. G. Santos, "RDDM: Reactive drift detection method," *Expert Systems with Applications*, vol. 90, pp. 344–355, 2017.

[13] O. Wu, Y. S. Koh, G. Dobbie, and T. Lacombe, "Nacre: Proactive recurrent concept drift detection in data streams," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[14] J. I. G. Hidalgo, L. M. P. Mariño, and R. S. M. de Barros, "Cosine similarity drift detector," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV*. Springer, 2019, pp. 669–685.

[15] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, p. 108632, 2022.

[16] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, 2010. [Online]. Available: https://dl.acm.org/doi/10.5555/1756006.1859903

[17] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *International conference on discovery science*. Springer, 2007, pp. 264–269.

[18] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern recognition letters*, vol. 33, no. 2, pp. 191–198, 2012.

[19] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Machine learning*, vol. 1, no. 3, pp. 317–354, 1986.

[20] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 139–148.

[21] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE transactions on knowledge and data engineering*, vol. 5, no. 6, pp. 914–925, 1993.

[22] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[23] M. Harries, "Splice-2 comparative evaluation: electricity pricing (Technical Report UNSW-CSE-TR-9905)," *Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney*, vol. 2052, 1999.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[25] N. Lu, J. Lu, G. Zhang, and R. L. De Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence*, vol. 230, pp. 108–133, 2016.

[26] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[27] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006, pp. 1–24.

[28] G. D. Ruxton, "The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test," *Behavioral Ecology*, vol. 17, no. 4, pp. 688–690, 2006.

[29] M. Mahbobi and T. K. Tiemann, *Introductory business statistics with interactive spreadsheets-1st Canadian edition*. Campus Manitoba, 2016.

[30] M. D. L. Tosi and M. Theobald, "OPTWIN: Drift identification with optimal sub-windows," *arXiv preprint arXiv:2305.11942*, 2023.

[31] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem *et al.*, "River: machine learning for streaming data in python," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 4945–4952, 2021.

[32] TensorFlow, "Convolutional Neural Network (CNN); Tensorflow Core," Dec 2022. [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn