Concept Drift Detection via Boundary Shrinking

Yoshihiro Okawa
Artificial Intelligence Laboratory
Fujitsu Laboratories Ltd.
4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, Kanagawa 211-8588, Japan
okawa.y@fujitsu.com

Kenichi Kobayashi
Artificial Intelligence Laboratory
Fujitsu Laboratories Ltd.
4-1-1 Kamikodanaka, Nakahara-ku,
Kawasaki, Kanagawa 211-8588, Japan
kenichi@fujitsu.com

***Abstract*—A change in data distribution, called "concept drift," often degrades performance of machine-learning models trained beforehand. However, detecting concept drift and identifying where it occurs by hand is a quite costly task and it sometimes overlooks drift until the performance is significantly degraded during operation. In this study, we present an unsupervised concept-drift-detection method with an ensemble of inspector models called "Concept-Drift Detection via Boundary Shrinking (CDDBS)". To reduce delays in drift detection, we train the inspector models used in CDDBS so that their decision boundaries would be intentionally shrunk in a classification region of a certain class. These shrunk decision boundaries also make it possible to identify where the drift occurs without using true labels because they react individually and sensitively to drift occurring in their corresponding classes. We evaluated the effectiveness of CDDBS by using a simple numerical dataset, several public synthetic benchmark datasets and high-dimensional real images in the CIFAR-10 dataset with synthetic drift. CDDBS outperformed other drift-detection methods in sensitivity of drift detection without increasing the number of false alarms and successfully identified drift-occurring classes without using true labels.**

***Index Terms*—concept drift, distribution shift, unsupervised drift detection, ensemble learning**

I. Introduction

Dealing with changes in properties of data is a critical issue to ensure the quality of machine-learning technologies at a high level during operation. In particular, a change in data distribution is called "concept drift" [1], and this phenomena often deteriorates the performance of machine-learning models. A recent empirical study [2] showed that such distribution shifts degrade the accuracy of machine-learning models trained with different algorithms in classification problems. This is because such models are trained to minimize their specific loss in accordance with the distribution of non-drifting training examples. In many cases, when and how concept drift occurs are unpredictable before starting operation.

A number of useful and effective concept-drift-detection methods have been studied over the last 30 years. According to the literature [3], such methods can be divided into two categories on the basis of their reliance on the use of labeled input examples: *explicit/supervised* drift detection and *implicit/unsupervised* drift detection. Explicit/supervised methods use true labels of input examples to compute performance metrics (e.g., accuracy) and monitor them online over time. They detect concept drift when those performance metrics drop or change significantly. Implicit/unsupervised methods use the properties of feature values of unlabeled input examples to detect concept drift. Since these methods enable us to detect concept drift automatically without labeling unlabeled input examples by hand, they are effective when their true labels are difficult to obtain or are not all available immediately during operation.

Drift identification or drift understanding [4], i.e., not only to detect concept drift but also to identify where, how, and when it occurs is another significant property of concept drift detection. This property is quite useful for retraining or updating models to recover performance since it tells us which examples and/or classes should be carefully investigated after drift is detected. For this problem, a dataset shift-detection method was proposed in [5] that identifies exemplars typifying the shift without using their true labels. An ensemble method using a number of classifiers or models is another useful method of identifying concept drift, as shown in [6]. However, these methods do not focus on satisfying both this identifying property and sensitivity in drift detection. Furthermore, most of them are evaluated with only low-dimensional numerical datasets, while the number of applications using machine learning technologies for classification problems on high-dimensional input examples (e.g., colored images) has been increasing.

To acquire the above-mentioned properties in drift detection, we propose an unsupervised concept-drift-detection method with an ensemble of inspector models called "Concept-Drift Detection via Boundary Shrinking (CDDBS)". The main contributions of our study are as follows:

1) We propose CDDBS using an ensemble of inspector models that both detects and identifies concept drift in classification problems without using true labels of unlabeled input examples.
2) We introduce a specific design procedure of CDDBS. In particular, we describe how to train the inspector models used in CDDBS, the decision boundaries of which are shrunk in a certain classification region. We also introduce a drift-detection and identification algorithm suitable for CDDBS using classification scores obtained from the inspector models in CDDBS.
3) We evaluated the effectiveness of CDDBS by using

a simple numerical dataset, several public synthetic benchmark datasets and high-dimensional real images in the CIFAR-10 dataset with synthetic drift.

The rest of this paper is organized as follows: In Section II, we examine related work. In Section III, we introduce the problem formulation of our study and concept drift discussed in this paper. In Section IV, we present CDDBS with its specific design procedure and a drift-detection and identification algorithm suitable for this method. In Section V, we discuss the evaluation of the effectiveness of CDDBS. Finally, we conclude our study in Section VI.

II. Related work

Various concept-drift-detection methods have been proposed over the last 30 years [1], [3], [4]. We focus on ensemble drift-detection methods using a number of classifiers or detectors as those most related to CDDBS. In general, ensemble drift-detection methods (hereafter, ensemble methods) use outputs or scores obtained from their classifiers or detectors to detect and/or adapt concept drift in accordance with their strategies. Bagging [7] and Boosting [8] algorithms are typical strategies and a recent study [9] showed that combining the advantages of Bagging-ensembles with the Self-Adjusting Memory (SAM) algorithm improved its performance under heterogeneous types of concept drift. Another recent study [10] proposed an adaptation of eXtreme Gradient Boosting algorithm (XGB) for classification of evolving data streams. Further, ensemble methods for data stream regression problems were also discussed and evaluated in [11].

Unsupervised or semi-supervised ensemble methods for drift detection and adaptation have also been proposed. For example, SAND [12] used confidence scores obtained from an ensemble of classifier models to detect concept drift and reused them to select examples to be labeled for model updating. DyDaSL [13] also used the confidence scores to update a classifier ensemble with the self-training algorithm. In addition, EDFS [14] encapsulated multiple detectors in ensembles that are specialized in diversified feature subspaces for unsupervised concept-drift detection. However, the conventional methods listed above do not use their ensembles to satisfy both sensitivity in drift detection and identifying property to let us know where the detected drift occurs in the feature space of the operated machine-learning models. Further, these methods are evaluated with only low-dimensional numerical datasets.

Compared with these conventional methods, CDDBS both detects concept drift sensitively without using true labels of input examples and identifies where the detected drift occurs at the same time. Further, this method works effectively regardless of the dimensions of input examples.

III. Preliminary

As a preliminary, in this section, we describe a problem formulation of our study and introduce concept drift discussed in this paper.

Let $X \in \mathbb{R}^p$ be a p-dimensional input example and $y \in \mathbb{R}$ be its corresponding class label. In general classification problems using machine learning, we train a model $f : \mathbb{R}^p \to \mathbb{R}$ so that $f(X) = y$ with a number of training examples $X_{tr} \in \mathbb{R}^p$ and their true labels $y_{tr} \in \mathbb{R}$. We let $c \in [2, \infty)$ be the number of classes, $\mathcal{C} = \{0, 1, \ldots, c-1\}$ be a set of those classes, and $\mathcal{R}_i$, $i \in \mathcal{C}$ be the classification regions where $f(X) = i$. In addition, let $\mathcal{T}_{tr}$ and n_{tr} be a set and the number of training examples and their labels, respectively, i.e., $\mathcal{T}_{tr} = \{(X_{tr}, y_{tr})_1, \ldots, (X_{tr}, y_{tr})_{n_{tr}}\}$.

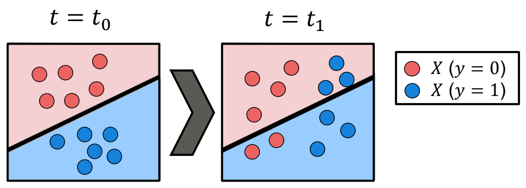


Fig. 1. Example of concept drift between time points t_0 and t_1. Red and blue dots represent input examples, classes of which are 0 and 1, respectively. Due to distribution change of input examples, some examples are misclassified with decision boundary of trained model shown with solid line.

Concept drift is a change in data distribution [1] and its example between time points t_0 and t_1 is illustrated in Fig. 1. The red and blue dots represent X mapped to a two-dimensional feature space with $y = 0$ and 1, respectively. The solid line represents a decision boundary of f trained beforehand through a certain machine-learning algorithm. As shown in Fig. 1, the distribution of X changes between t_0 and t_1. As a result, some input examples cross the decision boundary at t_1, thus, those examples are misclassified with f. This indicates that concept drift potentially degrades the performance (e.g., prediction accuracy) of the trained model.

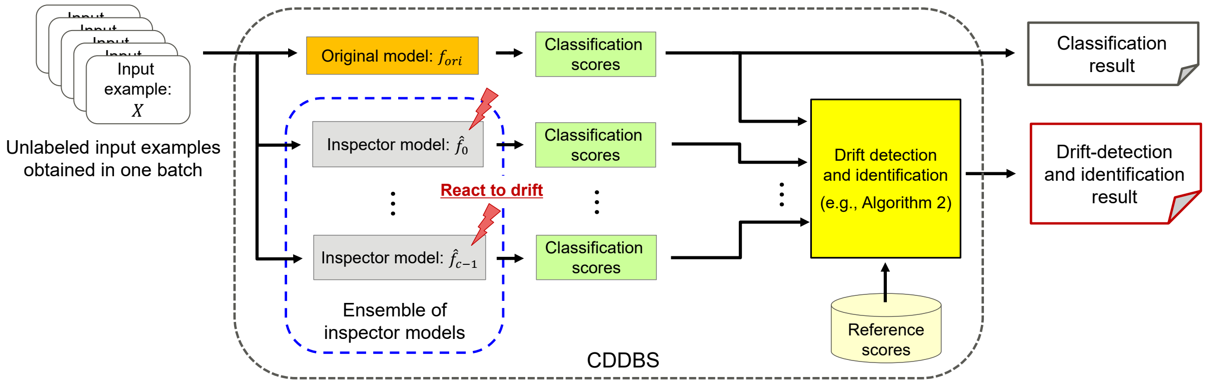
Many types of concept-drift-detection methods and algorithms using true labels of unlabeled input examples have been proposed. However, those true labels are not always available immediately during operation; thus, these methods are unable to detect concept drift online in such a situation. On the other hand, from the viewpoint of system management including model retraining or updating after drift detection, it is quite useful to know where and how input examples are drifted. To address the issues described thus far, CDDBS not only detects concept drift sensitively but also identifies where the detected drift occurs in the feature space of the model trained beforehand without using true labels of input examples.

IV. Drift detection and identification using boundary-shrunken inspector models

In this section, we introduce CDDBS and its drift-detection mechanism in Section IV-A. In Section IV-B, we introduce a specific design procedure to train the inspector models used in CDDBS, the decision boundaries of which are shrunk in a classification region of a certain class. Finally, we introduce a drift-detection and identification algorithm suitable for CDDBS in Section IV-C.

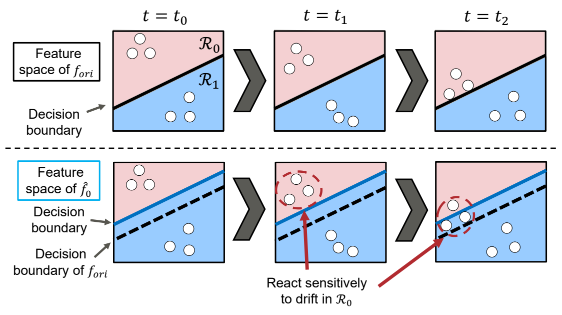
A. Concept-drift detection with ensemble of inspector models

An overview of CDDBS is illustrated in Fig. 2. It consists of an original model $f_{ori} : \mathbb{R}^p \to \mathbb{R}$ trained with $(X_{tr}, y_{tr}) \in \mathcal{T}_{tr}$



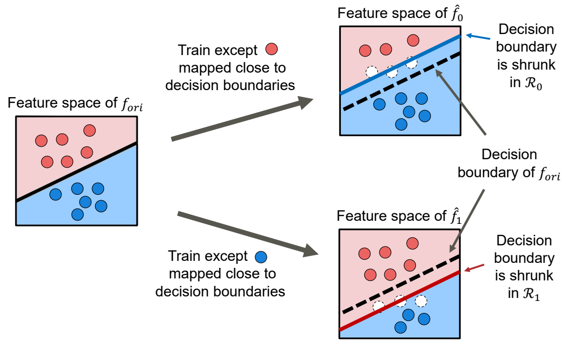
Input example: X
Unlabeled input examples obtained in one batch
Original model: f_{ori}
Classification scores
Inspector model: $\hat{f}_0$
Inspector model: $\hat{f}_{c-1}$
React to drift
Ensemble of inspector models
Drift detection and identification (e.g., Algorithm 2)
Reference scores
CDDBS
Classification result
Drift-detection and identification result

Fig. 2. Overview of CDDBS. It consists of an original model f_{ori} and an ensemble of inspector models $\hat{f}_i$. Drift detection and identification are executed using classification scores obtained from those models and their reference scores.



$t = t_0$
$t = t_1$
$t = t_2$
Feature space of f_{ori}
$\mathcal{R}_0$
$\mathcal{R}_1$
Decision boundary
Feature space of $\hat{f}_0$
Decision boundary
Decision boundary of f_{ori}
React sensitively to drift in $\mathcal{R}_0$

Fig. 3. Example of concept-drift detection and identification with inspector models in CDDBS. White dots represent unlabeled input examples, and red and blue shaded areas are classification regions $\mathcal{R}_0$ and $\mathcal{R}_1$, respectively. Decision boundary of inspector model $\hat{f}_0$ is shrunk in $\mathcal{R}_0$, thus, this inspector model reacts sensitively to drift occurring in $\mathcal{R}_0$.



Feature space of $\hat{f}_0$
Train except ● mapped close to decision boundaries
Decision boundary is shrunk in $\mathcal{R}_0$
Feature space of f_{ori}
Decision boundary of f_{ori}
Feature space of $\hat{f}_1$
Train except ● mapped close to decision boundaries
Decision boundary is shrunk in $\mathcal{R}_1$

Fig. 4. Design procedure of inspector models having shrunk decision boundaries. Each inspector model is trained with training examples except those mapped close to decision boundaries of original model.

and an ensemble of inspector models $\hat{f}_i : \mathbb{R}^p \to \mathbb{R}$, $i \in \mathcal{C}$. The decision boundaries of $\hat{f}_i$ are shrunk in a classification region of its corresponding class i. The flow of drift detection and identification through CDDBS is as follows: Once CDDBS obtains a number of unlabeled input examples as one batch, f_{ori} and $\hat{f}_i$ individually calculate their own classification scores. Then, by comparing those calculated classification scores and their reference ones, CDDBS returns a result of drift detection and identification regarding the obtained input examples. Note that, since the above-mentioned classification scores are calculated without using true labels of input examples, CDDBS is an *implicit/unsupervised* method in accordance with [3].

Next, we explain the mechanism of drift detection and identification with the inspector models used in CDDBS by using a simple binary classification problem. In Fig. 3, the top three figures show two-dimensional feature spaces with a decision boundary of f_{ori}, while the bottom three figures show those with a decision boundary of $\hat{f}_0$. The white dots represent unlabeled input examples, and the red and blue shaded areas represent classification regions $\mathcal{R}_0$ and $\mathcal{R}_1$, respectively. As shown in the bottom three figures, $\hat{f}_0$ has a decision boundary that is shrunk in $\mathcal{R}_0$ of f_{ori}.

In general classification problems using machine learning, class labels of unlabeled input examples are induced or predicted by using a certain type of classification score (e.g., confidence) obtained through trained models. These classification scores become lower as the input examples are mapped close to the decision boundaries of the trained model. Due to this property, the classification scores obtained through $\hat{f}_0$ change sensitively in accordance with the distribution change of input examples occurring in $\mathcal{R}_0$ since its decision boundaries are shrunk in that region. That is, $\hat{f}_0$ reacts sensitively to concept drift occurring in class 0 and tells us its occurrence. In the same manner, we can detect and identify drift occurring in $\mathcal{R}_1$ sensitively by using $\hat{f}_1$, the decision boundaries of which are shrunk in $\mathcal{R}_1$.

B. Design procedure of inspector models with boundary shrinking

Figure 4 shows the design procedure to train the inspector models used in CDDBS. We train each inspector model except training examples that are mapped close to the decision boundaries of f_{ori} or those classified incorrectly with f_{ori} so that their decision boundaries would be shrunk in the classification regions of their corresponding classes. We use classification scores to determine which and how many examples to be removed from training examples.

Confidence is a possible classification score to measure how close the examples are mapped to the decision boundaries in a feature space, which is often defined as the output values of the final layer in a neural network. In fact, confidence has been used to detect concept drift instead of using their true labels, as proposed in [15] and [16]. However, confidence obtained from well-trained machine-learning models with a deep neural network often becomes almost 0 or 1 [17], especially when sigmoid functions are used as activation functions in the final layer of the network. This means that confidence does not have linearity in a feature space of well-trained deep neural networks. Therefore, it is not appropriate for us to use confidence as the classification score that determines which and how many examples to be removed from the training examples of the inspector models. To address this issue, we introduce a novel metric called "sureness" that tells us how far input examples are mapped from the decision boundaries with linearity in the feature space of the trained models.

1) Sureness: We define sureness regarding X as follows:

$$\mathrm{sur}(X) := \phi\left(s_{\mathrm{1st}}(X)\right) - \phi\left(s_{\mathrm{2nd}}(X)\right), \tag{1}$$

where $\phi : \mathbb{R} \to \mathbb{R}$. In addition, $s_{\mathrm{1st}}(X) \in \mathbb{R}$ and $s_{\mathrm{2nd}}(X) \in \mathbb{R}$ are the first and second highest classification scores among all classes regarding X. To let the sureness defined in (1) have linearity in the feature space of the trained machine-learning model, the difference between $\phi\left(s_{\mathrm{1st}}(X)\right)$ and $\phi\left(s_{\mathrm{2nd}}(X)\right)$ has to be equivalent anywhere in that space. A suitable approach to satisfy this property is calculating those differences with a loss function that is used to train machine-learning models. In particular, if we train a machine-learning model by minimizing log-losses with a logarithmic function, the loss regarding X is denoted as $-\log(\eta_i(X))$, where $\eta_i(X)$ is probability to predict its label as class i. In such a case, $\log(\eta_i(X))$ has additivity; thus, their differences are equivalent anywhere in a feature space of the trained model. Therefore, sureness has linearity in the feature space of the trained model if it is calculated with $\eta_i(X)$ whose values are outputs of the trained models as scores $s_{\mathrm{1st}}(X)$ and $s_{\mathrm{2nd}}(X)$, and a logarithmic function as ϕ in (1).

We summarize the design procedure to train f_{ori} and $\hat{f}_i$, $i \in \mathcal{C}$ used in CDDBS in Algorithm 1: We first train f_{ori} with all the training examples and their true labels in Line 1. Next, we make sets $\mathcal{T}_{tr}^i$ of training examples and labels to train $\hat{f}_i$, $i \in \mathcal{C}$ in Line 3. Specifically, we add (X_{tr}, y_{tr}) to $\mathcal{T}_{tr}^i$ if y_{tr} are not i or if both y_{tr} and its predicted labels $f(X_{tr})$ are i and its sureness $\mathrm{sur}(X_{tr})$ is equal to or greater than its threshold $\rho_i \in \mathbb{R}$. Then, in Line 4, we train $\hat{f}_i$, $i \in \mathcal{C}$ with $(X_{tr}, y_{tr}) \in \mathcal{T}_{tr}^i$.

As mentioned above, CDDBS needs to train multiple inspector models. In this design procedure, fine tuning is useful to reduce its computation cost. In detail, the inspector models are available in CDDBS where parameters in only the last few layers of the trained original model are retrained instead of training all parameters in a network from scratch.

Algorithm 1 Training procedure of original model and ensemble of inspector models

```
Input: $\mathcal{T}_{tr} : \{(X_{tr}, y_{tr})\},\ \mathcal{C}$
Parameter: $\rho_i,\ i \in \mathcal{C}$
Output: $f_{ori},\ \hat{f}_i,\ i \in \mathcal{C}$
1: Train $f_{ori}$ with $(X_{tr}, y_{tr}) \in \mathcal{T}_{tr}$
2: for $i \in \mathcal{C}$ do
3:     $\mathcal{T}_{tr}^i \leftarrow \{(X_{tr}, y_{tr}) \in \mathcal{T}_{tr} | y_{tr} \neq i \vee (y_{tr} = i, f_{ori}(X_{tr}) = y_{tr}, \mathrm{sur}(X_{tr}) \geq \rho_i)\}$
4:     Train $\hat{f}_i$ with $(X_{tr}, y_{tr}) \in \mathcal{T}_{tr}^i$
5: end for
6: return $f_{ori},\ \hat{f}_i,\ i \in \mathcal{C}$
```

Algorithm 2 Drift-detection and identification algorithm suitable for CDDBS

```
Input: $\mathcal{X} : \{X\},\ \hat{f}_i,\ \mathcal{S}_i^{ref},\ i \in \mathcal{C}$
Parameter: $\alpha$
Output: $\mathcal{C}_{dri}$
1: Let $\mathcal{C}_{dri} = \emptyset,\ \mathcal{S}_i = \emptyset,\ i \in \mathcal{C}$.
2: for $X \in \mathcal{X}$ do
3:     for $i \in \mathcal{C}$ do
4:         $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{\mathrm{score}(X; \hat{f}_i)\}$
5:     end for
6: end for
7: $\mathcal{C}_{dri} \leftarrow \{i \in \mathcal{C} | \mathrm{KL}(\mathcal{S}_i, \mathcal{S}_i^{ref}) > \alpha\}$
8: return $\mathcal{C}_{dri}$
```

C. Drift-Detection and Identification Algorithm

Algorithm 2 describes a drift-detection and identification algorithm suitable for CDDBS. Through this algorithm, we obtain a set of classes $\mathcal{C}_{dri}$ as a result of drift detection and identification at each time point where the class index $i \in \mathcal{C}$ is in $\mathcal{C}_{dri}$ if $\hat{f}_i$ detects drift. The detailed procedure of this algorithm is as follows. We first obtain a set of classification scores, denoted as $\mathcal{S}_i$, $i \in \mathcal{C}$, from each $\hat{f}_i$ regarding $X \in \mathcal{X}$, where $\mathcal{X}$ is a set of unlabeled input examples obtained at each time point. Next, for each class, we calculate the Kullback−Leibler (KL) divergence to compare the obtained classification scores with their reference values in a set $\mathcal{S}_i^{ref}$ that are obtained with non-drifted examples beforehand. Then, if the calculated KL divergence, denoted as $\mathrm{KL}(\mathcal{S}_i, \mathcal{S}_i^{ref})$, is greater than its threshold α, the class index i is added to $\mathcal{C}_{dri}$. Finally, this algorithm returns $\mathcal{C}_{dri}$ as a result of drift detection and identification.

The output $\mathcal{C}_{dri}$ obtained through Algorithm 2 provides not only detection results, i.e., "whether drift occurs or not," but also their identification results, i.e., "in which class the drift is occurring". Throughout this algorithm, none of the true labels of input examples is used for drift detection and identification. Algorithm 2 is just one example of a drift-detection and identification algorithm suitable for CDDBS. In other words, other types of drift-detection and identification algorithms are also suitable for CDDBS such as those with statistical hypothesis testing, e.g., the Kolmogorov−Smirnov

(KS) test as used in [18].

V. Experimental evaluation

We evaluated the effectiveness of CDDBS in the following three scenarios:

1. with a simple numerical dataset
2. with public synthetic benchmark datasets
3. with high-dimensional real images in CIFAR-10

A. Scenario 1: with a simple numerical dataset

We first evaluated CDDBS with a simple four-class classification problem.

1) Evaluation conditions: In this scenario, we used two-dimensional training examples with four classes as illustrated in the left side of Fig. 5. The number of examples in each class was 200, and we generated those examples in accordance with normal distributions, the means of which are $(0, 0)$, $(1, 0)$, $(1, 1)$, and $(0, 1)$ for each class, and standard deviations (SDs) are 0.15 for all classes. We used a three-layer neural network with 25 neurons in its hidden layer as f_{ori}. We also illustrate the decision boundaries and classification regions of the trained f_{ori} in the left side of Fig. 5 that are mapped to the input feature space to clarify the training results.

2) Design of inspector models: To train the inspector models used in CDDBS, we remove some of the training examples that are mapped close to the decision boundaries of f_{ori}. Specifically, in this scenario, we first empirically set a threshold ρ_0 in Algorithm 1 so that 75 to 85% of training examples in $\mathcal{X}_{tr}^0$ would be removed where $\mathcal{X}_{tr}^i := \{X_{tr} | (X_{tr}, y_{tr}) \in \mathcal{T}_{tr}, y_{tr} = i, f_{ori}(X_{tr}) = i\}$, $i \in \{0, 1, 2, 3\}$. Then, we train $\hat{f}_0$ with the remaining training examples and their true labels. The training examples in $\mathcal{T}_{tr}^0$, the decision boundaries of f_{ori} and $\hat{f}_0$, and the classification regions of $\hat{f}_0$ are shown in the right side of Fig. 5. Dotted lines in this figure represent the decision boundaries of f_{ori} for ease of comparison. By comparing two figures in Fig. 5, training examples in class 0 were removed if they were mapped close to the decision boundaries. As a result, the decision boundaries of $\hat{f}_0$ trained with the remaining examples shrank in the classification region $\mathcal{R}_0$ shown with the blue shaded area. In the same manner, we train $\hat{f}_1$, $\hat{f}_2$ and $\hat{f}_3$ so that their decision boundaries would be shrunk in the classification regions of their corresponding classes.

3) Evaluation results: We evaluated drift detection and identification of CDDBS with four different types of (gradual) drift. Detailed conditions of each type of drift, and their drifted input examples are listed in Table I. We used Algorithm 2 for drift detection and identification in CDDBS and evaluated the results under the following three metrics:

- Detection delay (**Delay**): Difference between when drift is detected and occurs.
- Accuracy degradation (**Acc. degradn.**): Difference between accuracy when drift is detected and mean value of accuracy when drift does not occur.
- Number of false alarms (**# False alarms**): The number of false detection when drift does not occur.

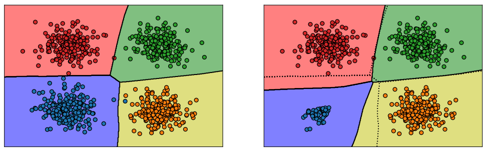
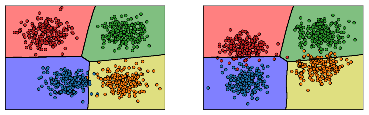


Fig. 5. Training examples and trained decision boundaries of f_{ori} (left) and $\hat{f}_0$ (right). Blue, yellow, green and red represent class 0, 1, 2 and 3, respectively. Training examples mapped close to the decision boundaries of f_{ori} are removed, then decision boundaries of $\hat{f}_0$ are shrunk in $\mathcal{R}_0$.

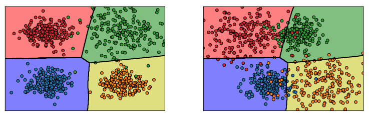
TABLE I
Drift conditions and drifted input examples in Scenario 1

Drift type	Drift-occurring classes	Drifting parameters
I	$\{0\}$	mean
II	$\{1, 3\}$	mean
III	$\{2\}$	SD
IV	$\{0, 1, 2, 3\}$	mean, SD

Input examples in each type of drift



(a) Type I (b) Type II



(c) Type III (d) Type IV

We compared the results of our CDDBS with those of the state-of-the-art unsupervised drift-detection method "CDBD [15]", which also uses changes in KL divergence regarding confidence scores to detect concept drift. The threshold for detection used in CDDBS is set to the mean plus two SDs, while that in CDBD is set to the mean plus one SD as used in [15]. Both thresholds are calculated with their reference confidence scores. We also used three detection rules for each detection method; 1/1 trigger, 2/3 trigger, and 3/5 trigger. Specifically, drift is detected with a p/q trigger if the KL-divergence is greater than the threshold at least p times in successive q time points.

The detection results from CDDBS and CDBD under the above-mentioned detection rules are listed in Table II. These results are overall mean values of all types of drift with 100 trials including validation with 20 different sets of non-drifting input examples before drift occurred in each trial. Their detailed results are listed in the top tables of Table VI at the end of this paper. From Table II, CDDBS outperformed CDBD for both "Delay" and "Acc. degradn." regardless of the detection rule, while "# False alarms" by using CDDBS was almost the same or less than that by using CDBD.

We also evaluated the identification property of CDDBS. We show mean values of the detection delays of $\hat{f}_i$, $i \in \mathcal{C}$ used in CDDBS with 2/3 trigger in Table III. The values are marked with † in this table if they are the results regarding

TABLE II
RESULTS ON A NUMERICAL DATASET IN SCENARIO 1

Detection method and rule		Delay	Acc. degradn. (%)	# False alarms
CDDBS	1/1	**1.64**	**−0.14**	2.96
CDBD	1/1	2.25	−0.15	**2.95**
CDDBS	2/3	**3.73**	**−0.30**	**0.22**
CDBD	2/3	4.49	−0.39	0.81
CDDBS	3/5	**4.87**	**−0.47**	**0.06**
CDBD	3/5	5.99	−0.58	0.41

TABLE III
RESULTS OF DRIFT IDENTIFICATION WITH DETECTION DELAY IN INSPECTOR MODELS. THE RESULTS REGARDING DRIFT-OCCURRING CLASSES ARE MARKED WITH †

Drift type	Delay			
	$\hat{f}_0$	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$
I	4.07†	17.59	9.29	9.92
II	6.83	4.83†	7.07	3.98†
III	7.21	12.84	5.33†	13.83
IV	2.65†	3.98†	3.40†	4.17†

the drift-occurring classes in each type of drift. By comparing the results shown in this table with the drift-occurring classes listed in Table I, the delays of the inspector models became shorter when the drift occurred in their corresponding classes. These results indicate that the inspector models with shrunk decision boundaries react sensitively to drift occurring in their corresponding classification regions, telling us where the detected drift occurred as a result of drift identification.

B. Scenario 2: with public synthetic benchmark datasets

Next, we evaluated CDDBS with selected public synthetic benchmark datasets in nonstationary environments.

1) Simulation conditions: In this scenario, we used 9 synthetic benchmark datasets provided by [19]: 2CDT, 2CHT, 4CR, 4CRE-V1, 4CRE-V2, UG_2C_2D, MG_2C_2D, UG_2C_3D, and UG_2C_5D. We selected these datasets if the classification accuracy of f_{ori} trained with half of the examples in the first concept, i.e., a set of a fixed number of examples, becomes lower than 70% when drift occurred using the rest of the examples, where f_{ori} is a five-layer neural network that has 15, 25 and 25 neurons in its respective hidden layers.

2) Design of inspector models: In the same manner as the design procedure described in Scenario 1, we empirically set thresholds of sureness ρ_i, $i \in \mathcal{C}$ for each dataset so that 75 to 80% of training examples in each class would be removed. Then, we train $\hat{f}_i$, $i \in \mathcal{C}$ with the rest of the training examples and their true labels in accordance with Algorithm 1, the network architectures of which are the same as that of f_{ori}.

3) Evaluation results: We evaluated the effectiveness of CDDBS for each dataset with the metrics defined in Scenario 1. We first randomly selected a quarter of examples from the latter half of examples in the first concept 20 times. We then randomly selected a quarter of examples from the examples in the rest of the concepts. We repeated this evaluating trial 50 times. In this scenario, we used the KS test with confidence and sureness of the input examples in CDDBS for drift detection, which are denoted as CDDBS$_{ks_c}$ and CDDBS$_{ks_s}$, respectively. In addition, we set $\alpha = 0.01$ in both tests. We also evaluated and compared CDDBS with several state-of-the-art unsupervised drift-detection methods: CDBD and HDDDM [20], and supervised drift-detection methods: DDM [21], EDDM [22] and ADWIN [23]. The hyper-parameters of these methods were set to be those recommended in their literature or their default ones used in open source code. In this and the following scenarios, we used 1/1 trigger as a detection rule in each method.

TABLE IV
AVERAGE RANK OF RESULTS ON BENCHMARK DATASETS

Detection method	Delay	Acc. degradn.	# False alarms
CDDBS$_{ks_s}$	**2.37**(±1.16)	**2.43**(±1.10)	3.49(±1.00)
CDDBS$_{ks_c}$	2.75(±1.12)	2.58(±1.08)	**3.42**(±0.92)
CDBD	4.84(±1.63)	4.60(±1.75)	4.97(±1.83)
HDDDM	3.42(±2.03)	3.41(±1.92)	4.74(±1.46)
DDM	3.65(±1.46)	4.07(±1.27)	4.06(±1.45)
EDDM	5.85(±1.72)	5.85(±1.72)	3.81(±1.35)
ADWIN	5.12(±0.98)	5.06(±1.03)	3.50(±1.03)

Table IV lists the average ranks along with the SDs for each detection method on all datasets. We ranked these methods where the method obtaining the best score as 1 and that obtaining the worst score as 7 for each trial on each dataset. Detailed results are listed in the middle tables of Table VI at the end of this paper. The ranks of CDDBS$_{ks_c}$ or CDDBS$_{ks_s}$ were the best or the second-best for all metrics in this scenario. This indicates that CDDBS enables us to detect drift sensitively with less delay, less accuracy degradation, and fewer false detections.

C. Scenario 3: with high-dimensional real image in CIFAR-10

We finally evaluated CDDBS with high-dimensional real images with synthetic drift.

1) Simulation conditions: We used images in the CIFAR-10 dataset as input examples and added six types of synthetic drift to their test examples: (a) GaussianBlur (Blur), (b) Darken, (c) Rotate, (d) GaussianNoise (Noisy), (e) Rain, and (f) Snow. Examples of images with each type of drift are shown in Fig. 6. We gradually increased degrees of drift in the drift types (a), (b), (c), and (d), while we gradually increased the number of classes whose images are drifted in (e) and (f) [1]. These drift were not real concept drift but virtual drift [1] or covariate shift since the ground truth of images did not change with those drift. We also used MobileNetV2 as f_{ori} and trained it through the stochastic gradient descent (SGD) with a learning rate 1.0×10^{-2}.

2) Design of inspector models: In the same manner as the previous scenarios, we empirically set thresholds of sureness ρ_i, $i \in \mathcal{C}$ so that 75 to 80% of training examples in each class would be removed. Then, we train $\hat{f}_i$, $i \in \mathcal{C}$ in accordance

[1]We added drift in (e) Rain and (f) Snow with the augmentation code provided by https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library.

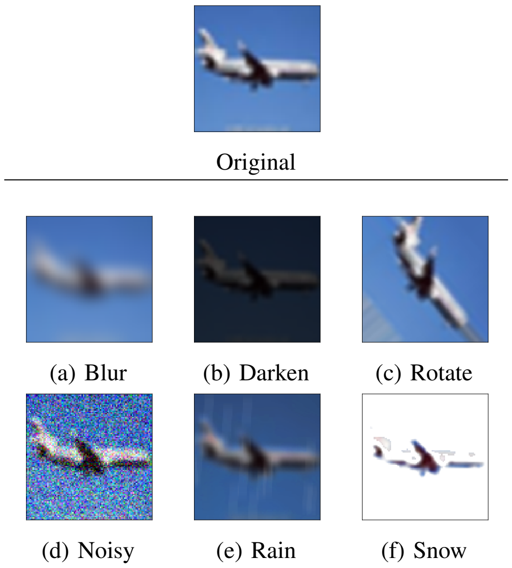


Fig. 6. Examples of original and drifted images in CIFAR-10 dataset. Degrees of drift are gradually increased in (a), (b), (c), and (d), while the number of drifting classes are gradually increased in (e) and (f).

with Algorithm 1, the network architectures of which are also MobileNetV2.

3) Evaluation results: We evaluated the effectiveness of CDDBS for each type of drift with the metrics used in the previous scenarios. We first randomly selected different 500 non-drifted images from test dataset 5 times. We then randomly selected different 500 images from test dataset with each type of drift 5 times. We repeated this evaluating trial 25 times for each type of drift. In this scenario, both Algorithm 2 and the KS test were used for drift-detection. The results by using Algorithm 2 are denoted as CDDBS$_{kl}$ and those by using the KS test are denoted by CDDBS$_{ks_s}$ and CDDBS$_{ks_c}$ as used in Scenario 2. We also evaluated and compared CDDBS with the other unsupervised detection methods: CDBD and a PCA-based method as used in [24] and [25], while we did not use HDDDM because of its computation cost.

Table V lists the average ranks along with the SDs for each detection method on all types of drift. We ranked the above-mentioned methods under the same manner as that used in Scenario 2. Their detailed results are listed in the bottom tables of Table VI at the end of this paper. CDDBS obtains the best for all metrics in this scenario. This also indicates that CDDBS enables us to detect drift sensitively even if the input examples are high-dimensional images.

VI. Conclusion

We proposed an unsupervised concept-drift-detection method with an ensemble of inspector models called "Concept-Drift Detection via Boundary Shrinking (CDDBS)". CDDBS detects concept drift sensitively in classification problems by using the ensemble of the inspector models, the decision boundaries of which are shrunk in a certain classification region. Furthermore, these inspector models identify where the drift occurs without using true labels of input examples since they are react sensitively to changes in their corresponding classes. We also introduced a specific design procedure to train the inspector models used in CDDBS and a drift-detection and identification algorithm using classification scores obtained from these inspector models. We evaluated the effectiveness of CDDBS through experimental verification with a simple numerical dataset, several public synthetic benchmark datasets and high-dimensional real images with synthetic drift. CDDBS outperformed the other drift-detection methods in sensitivity of drift detection without increasing the number of false alarms and identified where drift occurs without using true labels of input examples in operation.

TABLE V
Average rank of results on drifting images

Detection method	Delay	Acc. degradn.	# False alarms
CDDBS$_{kl}$	$\mathbf{2.44}(\pm0.85)$	$\mathbf{2.53}(\pm0.87)$	$3.23(\pm1.10)$
CDDBS$_{ks_s}$	$2.82(\pm0.76)$	$2.92(\pm0.80)$	$\mathbf{2.47}(\pm0.77)$
CDDBS$_{ks_c}$	$3.15(\pm0.80)$	$3.27(\pm0.83)$	$2.52(\pm0.89)$
CDBD	$3.33(\pm0.92)$	$3.44(\pm0.92)$	$3.33(\pm1.10)$
PCA-based	$3.26(\pm1.70)$	$2.84(\pm1.68)$	$3.46(\pm1.14)$

References

[1] 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, p. 44, 2014.

[2] J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. Dillon, J. Ren, and Z. Nado, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 13 969–13 980.

[3] T. S. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," *Expert Systems with Applications*, vol. 82, pp. 77 – 99, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417302439

[4] 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, 2019.

[5] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: an empirical study of methods for detecting dataset shift," in *Advances in Neural Information Processing Systems*, 2019, pp. 1394–1406.

[6] L. Deshpande and M. Rao, "Concept drift identification using classifier ensemble approach," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 19–25, 02 2018.

[7] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[8] Y. Freund, R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning (ICML)*, 1996, pp. 148–156.

[9] V. Losing, B. Hammer, H. Wersing, and A. Bifet, "Randomizing the self-adjusting memory for enhanced handling of concept drift," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[10] J. Montiel, R. Mitchell, E. Frank, B. Pfahringer, T. Abdessalem, and A. Bifet, "Adaptive xgboost for evolving data streams," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[11] H. M. Gomes, J. Montiel, S. M. Mastelini, B. Pfahringer, and A. Bifet, "On ensemble techniques for data stream regression," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[12] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *30th AAAI Conference on Artificial Intelligence*, 2016, pp. 1652–1658.

TABLE VI
DETAILED RESULTS OF EXPERIMENTAL EVALUATION IN SCENARIOS 1, 2, AND 3

Scenario 1

Drift type	Detection method	and rule	Delay	Acc. degradn. %	# False alarms
I	**CDDBS**	1/1	**1.93**	**−0.03**	3.08
	CDBD	1/1	3.51	−0.06	**2.77**
	CDDBS	2/3	**4.01**	**−0.03**	**0.23**
	CDBD	2/3	6.90	−0.22	0.64
	CDDBS	3/5	**4.95**	**−0.06**	**0.05**
	CDBD	3/5	9.06	−0.38	0.31
II	**CDDBS**	1/1	**1.66**	**−0.11**	**2.88**
	CDBD	1/1	2.05	−0.11	3.32
	CDDBS	2/3	**3.63**	**−0.21**	**0.23**
	CDBD	2/3	3.83	−0.25	0.98
	CDDBS	3/5	**4.76**	**−0.37**	**0.05**
	CDBD	3/5	5.17	−0.44	0.52

Drift type	Detection method	and rule	Delay	Acc. degradn. %	# False alarms
III	**CDDBS**	1/1	**2.02**	**−0.17**	3.18
	CDBD	1/1	2.27	−0.20	**2.84**
	CDDBS	2/3	4.97	**−0.51**	**0.19**
	CDBD	2/3	**4.83**	−0.56	0.75
	CDDBS	3/5	6.39	**−0.70**	**0.03**
	CDBD	3/5	**6.28**	−0.73	0.28
IV	**CDDBS**	1/1	**0.96**	**−0.24**	**2.71**
	CDBD	1/1	1.15	−0.25	2.86
	CDDBS	2/3	**2.33**	**−0.47**	**0.23**
	CDBD	2/3	2.39	−0.54	0.88
	CDDBS	3/5	**3.39**	**−0.76**	**0.09**
	CDBD	3/5	3.43	−0.78	0.52

Scenario 2

Datasets	Detection method	Delay	Acc. degradn. (%)	# False alarms
2CDT	**CDDBS**$_{ks_s}$	0.50	−18.20	**0.00**
	CDDBS$_{ks_c}$	**0.06**	**−10.90**	**0.00**
	CDBD	1.28	−21.54	7.40
	HDDDM	7.66	−36.68	0.88
	DDM	0.38	−17.20	3.58
	EDDM	0.46	−17.94	0.28
	ADWIN	0.58	−20.86	0.10
2CHT	**CDDBS**$_{ks_s}$	0.54	−10.23	**0.00**
	CDDBS$_{ks_c}$	**0.28**	**−7.13**	**0.00**
	CDBD	3.54	−36.37	**0.00**
	HDDDM	1.56	−18.69	0.84
	DDM	2.60	−30.33	1.20
	EDDM	0.60	−12.57	4.58
	ADWIN	1.58	−23.21	1.56
4CR	**CDDBS**$_{ks_s}$	**6.82**	**−0.02**	0.04
	CDDBS$_{ks_c}$	17.94	−1.22	0.02
	CDBD	9.68	−0.70	3.32
	HDDDM	7.74	−0.32	0.98
	DDM	12.10	−1.06	**0.00**
	EDDM	27.34	−7.66	**0.00**
	ADWIN	19.94	−2.08	**0.00**
4CRE-V1	**CDDBS**$_{ks_s}$	**0.00**	**−0.10**	**0.00**
	CDDBS$_{ks_c}$	**0.00**	**−0.10**	0.08
	CDBD	2.00	−46.85	**0.00**
	HDDDM	0.72	−7.11	0.94
	DDM	0.76	−6.45	**0.00**
	EDDM	2.00	−46.85	**0.00**
	ADWIN	1.00	−8.33	**0.00**
4CRE-V2	**CDDBS**$_{ks_s}$	**0.08**	**−1.39**	**0.02**
	CDDBS$_{ks_c}$	0.24	−1.77	**0.02**
	CDBD	1.48	−5.70	0.04
	HDDDM	0.36	−3.00	0.04
	DDM	1.46	−5.82	3.36
	EDDM	1.50	−5.75	0.52
	ADWIN	1.76	−6.93	0.34
UG_2C_2D	**CDDBS**$_{ks_s}$	1.60	**−0.31**	0.04
	CDDBS$_{ks_c}$	1.80	−0.41	**0.00**
	CDBD	3.30	−1.07	0.24
	HDDDM	**0.56**	−0.57	0.58
	DDM	1.28	−0.56	**0.00**
	EDDM	7.34	−5.76	**0.00**
	ADWIN	4.56	−1.94	**0.00**
MG_2C_2D	**CDDBS**$_{ks_s}$	1.48	−0.32	1.82
	CDDBS$_{ks_c}$	1.72	−0.32	1.40
	CDBD	2.56	−0.34	3.70
	HDDDM	5.50	−1.13	1.16
	DDM	**0.52**	**−0.29**	**0.00**
	EDDM	9.14	−1.88	**0.00**
	ADWIN	4.30	−0.72	**0.00**
UG_2C_3D	**CDDBS**$_{ks_s}$	**0.38**	**−0.02**	**0.00**
	CDDBS$_{ks_c}$	1.00	−0.05	**0.00**
	CDBD	3.58	−0.30	3.96
	HDDDM	5.14	−4.14	0.96
	DDM	2.52	−0.23	**0.00**
	EDDM	8.64	−2.94	**0.00**
	ADWIN	6.20	−1.10	**0.00**
UG_2C_5D	**CDDBS**$_{ks_s}$	**0.20**	**0.00**	0.04
	CDDBS$_{ks_c}$	1.08	−0.02	0.08
	CDBD	5.98	−0.94	0.06
	HDDDM	1.78	−0.11	0.34
	DDM	2.92	−0.38	**0.00**
	EDDM	8.46	−3.00	**0.00**
	ADWIN	5.50	−0.86	**0.00**

Scenario 3

Datasets	Detection method	Delay	Acc. degradn. (%)	# False alarms
Blur	**CDDBS**$_{kl}$	**0.76**	**−7.19**	0.76
	CDDBS$_{ks_s}$	1.00	−8.94	0.16
	CDDBS$_{ks_c}$	1.00	−8.94	**0.08**
	CDBD	1.12	−11.85	0.48
	PCA-based	1.28	−16.06	1.00
Darken	**CDDBS**$_{kl}$	**2.40**	**−4.27**	0.52
	CDDBS$_{ks_s}$	3.56	−8.19	**0.08**
	CDDBS$_{ks_c}$	3.56	−8.76	0.28
	CDBD	3.76	−9.46	0.48
	PCA-based	1.40	−1.20	0.48
Rotate	**CDDBS**$_{kl}$	**1.16**	**−6.81**	0.36
	CDDBS$_{ks_s}$	1.24	−7.81	0.24
	CDDBS$_{ks_c}$	1.72	−10.56	**0.20**
	CDBD	2.00	−12.59	0.52
	PCA-based	2.12	−9.13	0.84
Noisy	**CDDBS**$_{kl}$	**0.00**	**−14.25**	0.60
	CDDBS$_{ks_s}$	**0.00**	**−14.25**	0.16
	CDDBS$_{ks_c}$	**0.00**	**−14.25**	**0.12**
	CDBD	**0.00**	**−14.25**	0.80
	PCA-based	2.44	−20.72	0.72
Rain	**CDDBS**$_{kl}$	**1.08**	**−7.41**	0.36
	CDDBS$_{ks_s}$	1.72	−9.55	0.12
	CDDBS$_{ks_c}$	2.00	−11.23	**0.04**
	CDBD	2.04	−11.38	0.60
	PCA-based	1.84	−10.24	0.52
Snow	**CDDBS**$_{kl}$	**0.44**	**−6.84**	0.64
	CDDBS$_{ks_s}$	1.20	−10.40	0.12
	CDDBS$_{ks_c}$	1.12	−9.51	**0.04**
	CDBD	1.00	−8.35	0.28
	PCA-based	1.24	−9.64	0.56

[13] A. C. Gorgônio, A. M. de P. Canuto, K. M. O. Vale, and F. L. Gorgônio, "A semi-supervised based framework for data stream classification in non-stationary environments," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[14] L. Korycki and B. Krawczyk, "Unsupervised drift detector ensembles for data stream mining," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2019, pp. 317–325.

[15] P. Lindstrom, B. Mac Namee, and S. J. Delany, "Drift detection using uncertainty distribution divergence," in *2011 IEEE 11th International Conference on Data Mining Workshops*, Dec 2011, pp. 604–608.

[16] S. Yu, Z. Abraham, H. Wang, M. Shah, Y. Wei, and J. C. Príncipe, "Concept drift detection and adaptation with hierarchical hypothesis testing," *Journal of the Franklin Institute*, vol. 356, no. 5, pp. 3187 – 3215, 2019.

[17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*. org, 2017, pp. 1321–1330.

[18] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analysis and Data Mining*, vol. 2, pp. 311 – 327, 12 2009.

[19] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2015, pp. 873–881.

[20] G. Ditzler and R. Polikar, "Hellinger distance based drift detection for nonstationary environments," in *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*. IEEE, 2011, pp. 41–48.

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

[22] 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.

[23] 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.

[24] L. I. Kuncheva and W. J. Faithfull, "Pca feature extraction for change detection in multidimensional unlabeled data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 69–80, 2014.

[25] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 935–944.