Image Drift: Introducing Gaussian Split Detector

Maxime Fuccellaro
UMR 5800, LaBRI
University of Bordeaux
Bordeaux, France
maxime.fuccellaro@u-bordeaux.fr

Laurent Simon
UMR 5800, LaBRI
University of Bordeaux
Bordeaux, France
laurent.simon@u-bordeaux.fr

Akka Zemmari
UMR 5800, LaBRI
University of Bordeaux
Bordeaux, France
akka.zemmari@u-bordeaux.fr

***Abstract*—Recent research yielded a wide array of drift detectors. However, in order to achieve remarkable performance, the true class labels must be available during the drift detection phase. This paper targets at detecting drift when the ground truth is unknown during the detection phase. To that end, we introduce Gaussian Split Detector (GSD) a novel drift detector that works in batch mode. GSD is designed to work when the data follow a normal distribution and makes use of Gaussian mixture models to monitor changes in the decision boundary. The algorithm is designed to handle multi-dimension data streams thus suited for computer vision tasks and to work without the ground truth labels during the inference phase making it pertinent for real world use. In an extensive experimental study on real and synthetic datasets, we evaluate our detector against the state of the art. We show that our detector outperforms the state of the art in detecting real drift and in ignoring virtual drift which is key to avoid false alarms.**

***Index Terms*—Computer Vision, Concept Drift, Drift Detection**

I. Introduction

Few data distributions remain stationary over long periods of time. Machine learning models are constructed based on the assumption that the data encountered during the training stage and the future inference data share the same distribution. When models are trained and then deployed for inference, a change in the underlying data distribution can cause a drop in performance [16]. This shift in distribution is known as Concept Drift (CD) and detecting it will be the focus of this paper. The democratization of predictive modeling has made concept drift an active research topic as it cripples in production systems. Concept drift must not be confused with anomaly detection where few samples are out of distribution. Drift is not domain specific and impacts every field including text and video [24]. When dealing with computer vision tasks, drift can be caused by a wide array of factor including the change of equipment, unseen weather conditions or deterioration of the camera.

Real drift is a change in distribution that modifies the decision boundary. Real drift impacts a model's ability to predict instances classes. When real drift occurs, a drop of a model's predictive performances is systematically observed. *Virtual drift* is a change of distribution that does not affect the decision boundary. Since virtual drift does not impact the model's performance, we don't believe its occurrence warrants the re-training of the model. We believe a good drift detector should detect real drifts while ignoring virtual drifts.

Assume X is a set of variables used to predict the target class vector y. Three primary causes of concept drift are identified [27]: changes in the class distribution $\mathbb{P}(y \mid X)$, the feature space $\mathbb{P}(X \mid y)$ or class priors $\mathbb{P}(y)$. Drift of type $\mathbb{P}(y \mid X)$ happens when the decision boundaries change. Drift type $\mathbb{P}(X \mid y)$ happens when values reach unseen domains. Different drift speeds have been identified [21]. If the distribution change is sudden, it is referred to as *abrupt drift*, whereas if the distribution shift is slow over time, it is called *gradual drift*. When a distribution oscillates between two or more concepts, it is called *recurrent drift*.

In this paper, we introduce a novel drift detection method, Gaussian Split Detector (GSD). GSD computes decision boundaries of the most informative features on the training set. During the inference phase, the Expectation Maximization [9] (EM) algorithm estimates the parameters of the Gaussian mixture distribution on the selected features. The parameters are used to compute the new decision boundaries. Drift detection is based on the difference of the training boundaries and the inference boundaries. As GSD works in any number of dimensions, it does not need labels during inference and outperforms the state of the art in thorough experiments.

In Section 2, we present related work and position our paper. The GSD algorithm is described and evaluated in Section 3. Section 4 concludes this paper.

II. Related work

Concept drift has become the focus of recent work in the last few years [21]. Recent research yielded a wide array of models handling drift intrinsically [10]. As well as drift detectors that work jointly with a predictive model. If the ground truth is available shortly after inference, some detectors manage to achieve almost perfect drift detection. Two hypothesis are usually made in supervised drift detection. The first is that recent data shares the same distribution as upcoming data. The second is that a change in a model's error rate is a strong indicator of the presence of real drift.

With the assumption that recent and upcoming data share the same distribution, one approach is to continuously update a pool of models. In [10] a batch of new data is evaluated by a pool of models, and the contributions of each individual model are weighted based on its recent performance. With

979-8-3315-4184-2/24/$31.00 ©2024 IEEE

each new batch of data, a model is trained on it and added to the pool. Consistently poorly performing models are removed. This ensures quick adaptation to recurring drifts while keeping a strong level of performance. Techniques that operate on a dynamic pool of models have been extensively studied in [6], [17].

Detection methods based on the second hypothesis have received significant attention since they can reliably identify genuine drift while ignoring virtual drift. In [1] and [20] a model's error rate is monitored and a decline in performance is interpreted as drift. Various statistical tests [22] are employed to monitor the error rate and signal the occurrence of drift. These methods can reliably identify real drift while disregarding virtual drift as it does not impact a model's performances.

Updating a pool of models and monitoring the error rate are both effective techniques for handling drift. Both approaches require immediate access to the true labels after the inference stage. This is a very strong hypothesis for real-world situations where class labels may be never known [15]. Unsupervised methods have been the focus of research to deal with ground truth unavailability.

To detect drift based on the feature space instead of the true labels, window-based techniques have been studied. In [2] the authors introduce ADWIN that maintains a reference window consisting of past instances. The window size dynamically adjusts based on whether drift is detected. Specifically, when no change is detected, the window expands, while in the presence of drift, the window rapidly contracts. The detection mechanism of ADWIN involves repeatedly separating the window into two smaller windows based on the observation's age. Drift is detected when the averages of the values of these sub-windows exceed a threshold. Other window-based detectors have been proposed such as in [23].

Rather than relying on one-dimensional sliding windows, some methods detect drift on all the feature space. In [11], the authors introduce QT-EWMA where concept drift is detected using an exponential moving average to monitor the distribution of a QuantTree histogram built on the training data. The authors in [14] use minimum enclosing balls to identify abrupt or gradual changes. A ball is defined as a centroid and a minimum radius that encompasses all the samples in the ball. If a significant number of values are identified as outliers, a drift is detected and the centroid is updated to fit the latest concept.

Some authors use the hypothesis that if past and present data can be sorted, there is a drift. In [12], the authors train a model to distinguish between past and recent data. To prevent the model from identifying this distinction based on obvious timestamp-like aggregates, these were removed before training the model. The researchers evaluated the model's performance using the AUC metric. In [3], the authors use time as a means to detect drift by incorporating the timestamp attribute in the observations and training a model to predict the target variable for both past and recent data. If the timestamp attribute is considered as an informative feature, then the target variable is dependent on time, which indicates presence of a drift.

In [8], the authors propose a different unsupervised approach to detect drift. They train a teacher model on past labeled data, and then train a second model, known as the student model, to mimic the behavior of the teacher model. During the inference phase, the authors monitor the error rate of the student model and use the method presented in [2] to detect a drift if the error rate exceeds a certain threshold.

The authors in [24] investigate drift detection in the context of unsupervised image classification. The dimension is first reduced, followed by a two-sample test to identify drift. They evaluate a variety of dimension reduction and two-sample tests, including MMD [13] and KS, by applying various types of shifts to images. In [7], the authors propose an approach where the target class is incorporated in the dimension reduction mechanism, allowing the detector to ignore virtual drift.

The idea behind concept drift detection by statistical tests is that a distribution change will be a strong indicator of drift [14], [24]. A distribution change will enable the detection of drift, but will not discriminate between real drift and virtual drift. To the best of our knowledge, few algorithms are able to discriminate between real drift and virtual drift in a multivariate setting without access to true labels after detection such as Discriminative Drift Detector [12], QT-EWMA [11], Student-Teacher (ST) [8], Task Sensitive Drift Detector (TSDD) [7].

In this paper we introduce a novel algorithm to detect drift: Gaussian Split Detector (GSD). GSD is designed to handle binary classification problems. GSD successfully detects real drifts while triggering very few false positive by ignoring virtual drifts. The detector does not need true class labels to operate during the inference phase.

III. Gaussian Split Detector

In this Section we introduce GSD, an algorithm that monitors the decision boundaries shifts using a Gaussian mixture model in order to detect drift. We simplify the assumption made by the authors of [5] by making the hypothesis that the data distributions of any feature used for detection is a sum of two Gaussian distributions each corresponding to a class.

We start by building an ensemble of n single *Bayesian splits*. In a similar fashion to the random forest algorithm [4], each split is randomly given a subset of samples and features. During the training phase, for each feature ψ available to the split, the mean μ_c^ψ, variance σ_c^ψ and proportion p_c^ψ are calculated for each class c. The proportion is used to find the optimal decision boundary with regards to a possible class imbalance scenario, we have $p_0^\psi + p_1^\psi = 1$. Since we assume the variables are drawn from a Gaussian distribution we use these parameters to find the decision boundary. It is the intersection that minimizes the misclassification error rate of the two Gaussian curves. The feature selected for the split is the one that has the smallest misclassification area E^ψ. Figure 1 illustrates the process of computing the decision boundary.

During inference, the EM algorithm allows us to estimate the parameters of a Gaussian mixture distribution. The param-

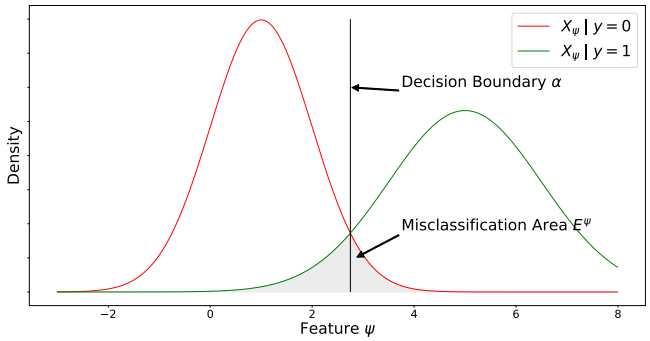

Fig. 1. In this figure, we plot the weighted density functions for the positive (in green) and negative (in red) samples. The decision boundary α is marked by the black line.

eters are then used to compute a new decision boundary $\hat{\alpha}$. The decision boundary difference is used to detect drift.

A. Mathematical formulation

1) Hypothesis: We assume that the predictive task is binary classification.

Furthermore, for each variable ψ, we make the hypothesis that:

$$\forall c \in \{0,1\} : X_\psi \mid y = c \sim \mathcal{N}(\mu_c^\psi, \sigma_c^\psi) \tag{1}$$

where $X_\psi \mid y = c$ denote the samples for feature ψ labelled as belonging to class c. GSD does not require the features *to be independent*.

2) Algorithm: Let $F_c^\psi(x)$ be the cumulative distribution function of the estimated Gaussian distribution for the variable ψ of the samples of class c. Let p_i^ψ be the proportion of samples of class i. Let α denotes the decision boundary that minimizes the misclassification area between the two classes. The misclassification area is calculated with equation 2.

$$\begin{aligned} E^\psi = \min\Bigg(& \min\left[p_0^\psi \times F_0^\psi(\alpha), p_1^\psi \times F_1^\psi(\alpha)\right] + \\ & \min\left[p_0^\psi \times (1 - F_0^\psi(\alpha)), p_1^\psi \times (1 - F_1^\psi(\alpha))\right] \Bigg) \end{aligned} \tag{2}$$

As in classic decision trees, the feature used in the split minimizes the misclassification probability across all features. For each feature ψ, the optimal boundary, α, is one of two intersections of the weighted Gaussian distributions. α can be computed analytically by solving a quadratic equation where the coefficients are:

$$a = \frac{1}{2(\overline{\sigma_0^\psi})^2} - \frac{1}{2(\overline{\sigma_1^\psi})^2}, b = \frac{\overline{\mu_1^\psi}}{(\overline{\sigma_1^\psi})^2} - \frac{\overline{\mu_0^\psi}}{(\overline{\sigma_0^\psi})^2}$$

$$c = \frac{(\overline{\mu_0^\psi})^2}{2(\overline{\sigma_0^\psi})^2} - \frac{(\overline{\mu_1^\psi})^2}{2(\overline{\sigma_1^\psi})^2} - \ln\left(\frac{\overline{p_0^\psi}\,\overline{\sigma_1^\psi}}{\overline{p_1^\psi}\,\overline{\sigma_0^\psi}}\right)$$

The α chosen minimizes the misclassification area. In some rare cases, the distribution functions do not intersect. This happens when the means are close and there is a steep class imbalance which causes a weighted distribution to encompass another. When encountered, the feature is dropped. We use here an ensemble of single splits for two reasons:

- In order to properly estimate the parameters of the Gaussian distribution we need a large enough sample size. This is not achievable when the data is partitioned by a large number of splits.
- It allows the model to work when the variables are not independent. In that case, a shift in the first split may drastically change the distribution of the final partitioning when we update its decision boundary.

B. Overview of the detection phase

Algorithm 1 GSD - Inference

Parameters:
- τ : Minimum ratio of drifting tree to flag drift
- β : Vector of thresholds to flag a drift

Inputs:
- $\mathcal{M} = \{\mathcal{M}_0, ..., \mathcal{M}_k\}$: Trained ensemble of $k+1$ single Gaussian splits
- $\alpha = \{\alpha_0, ..., \alpha_k\}$: Decision boundaries of the Gaussian splits
- $I \in \mathbf{R}^{\mathbf{m}\times\mathbf{d}}$: Test set with $\mathbf{d}$ features and $\mathbf{m}$ samples

Variables:
- $\gamma = 0$: Number of drifting splits

1: **for** all ψ used in $\mathcal{M}$ **do**
2: Run EM on $I[\psi]$ and get $\hat{\mu_i^\psi}, \hat{\sigma_i^\psi}, \hat{p_i^\psi}, i \in \{0,1\}$
3: Get the Gaussian decision boundary $\hat{\alpha}_\psi$
4: **if** $|\alpha_\psi - \hat{\alpha}_\psi| \geq \beta_\psi$ **then**
5: $\gamma = \gamma + 1$
6: **end if**
7: **end for**
8: **if** $\frac{\gamma}{k+1} \geq \tau$ **then**
9: A drift is triggered
10: **end if**

The drift detection phase is formally described in Algorithm 1. Let $\mathcal{M}$ be the ensemble of Gaussian splits computed during the training phase. In lines 1-2, for each feature in $\mathcal{M}$ we run the EM algorithm on the inference data. This gives the estimated parameters $(\hat{\mu_c^i}, \hat{\sigma_c^i}, \hat{p_c^i})$ for each subsequent feature i and class c of the two Gaussian distributions that constitute the overall distribution. In line 3, we find the new decision boundary $\hat{\alpha}_\psi$ by solving (2). It is the intersection of the two weighted Gaussian probability density functions. When the EM algorithm does not converge, the tree's output is not taken into account.

The β vector is used to determine if a change in the decision boundary in between the training data and inference data is high enough to signal a drift. To compute it, we split the labelled data into a train set containing 75% of the instances and a validation set. We then proceed to build a Gaussian split on all features of the training set. We then isolate the two components of the Gaussian mixture with the EM algorithm on the validation set as it is done in the detection phase. With

$\hat{\alpha}^i$ the estimated decision boundary on the validation data for feature i, the β vector is defined as $[|\alpha^0 - \hat{\alpha}^0|, ..., |\alpha^d - \hat{\alpha}^d|]$.

In lines 4 through 6, when the difference in the decision boundary exceeds the feature dependant β_ψ parameter, the γ count is incremented. In line 8, if the ratio of drifting split exceeds a user defined threshold τ, a drift is triggered. The τ value controls the sensibility of the detector. A low value will likely trigger false alarms while a high value might lead the detector to miss some real drifts.

C. Experimental results

In Figure 2 and Figure 3, we show how real drift affects the decision boundary and how the decision boundary can remain unchanged with virtual drift. In Figure 2, real drift is illustrated as the optimal split shifts with the distribution change. At time $t + \Delta t$, the decision boundary calculated at time t is no longer relevant. In Figure 3, virtual drift is shown as the decision boundary remains unchanged despite of the distribution change. We evaluate GSD against a panel of five state of the art detectors on both real and synthetic datasets. In order to test our method's ability to detect real drift and ignore virtual drift, perturbations are made on the datasets. The drift induction experimental procedure used in this section is standard when testing drift detectors [7], [25], [26].

1) Experimental protocol: The usual procedure to test algorithms suited to handle drift when true class labels are available after inference, is the test-then-train approach. A model predicts the class on a batch of samples, then, the true class is revealed and the model updates itself. The global prediction accuracy is then used to rank models.

This setup is not suited for models that do not rely on true labels availability. In most datasets used to benchmark drift handling methods, the presence of drift is only assumed or artificially introduced by sorting the observations on an attribute.

To know the exact drift occurrence and its impact, several distinct perturbations on images are applied. We add the image shift and Gaussian noise perturbations introduced by [24]. The image shift consists in applying rotations, (x-y) axis translation and zoom-in. Different standard deviations of Gaussian noise are added to corrupt the features. Several degrees of perturbations are added.

In order to assess if the perturbations yield real or virtual drift we fit a Random Forest Classifier on the unmodified train set before reporting its accuracy on the train set, unchanged validation set and the different drift sets. For the sake of robustness, we repeated the process ten times.

The drop of the model's accuracy between the different sets is used to classify drift as virtual or real. If the difference in accuracies between the validation set and the training set is lower than that of the validation set and the drift set, we consider the drift induced to be real, otherwise, it is considered as a virtual drift. We apply this protocol on all datasets. Our experiment is carried out with two dimensionality reduction techniques. In one setting we reduce the dimensionality to 32 features using Principal Component Analysis (PCA) which explains above 75% of the variance for all datasets. As GSD is designed to work when the data follows normal distributions. In a second setup, a Variational Autoencoder (VAE) is trained to project the data to a latent space of four dimensions that follow a normal distribution.

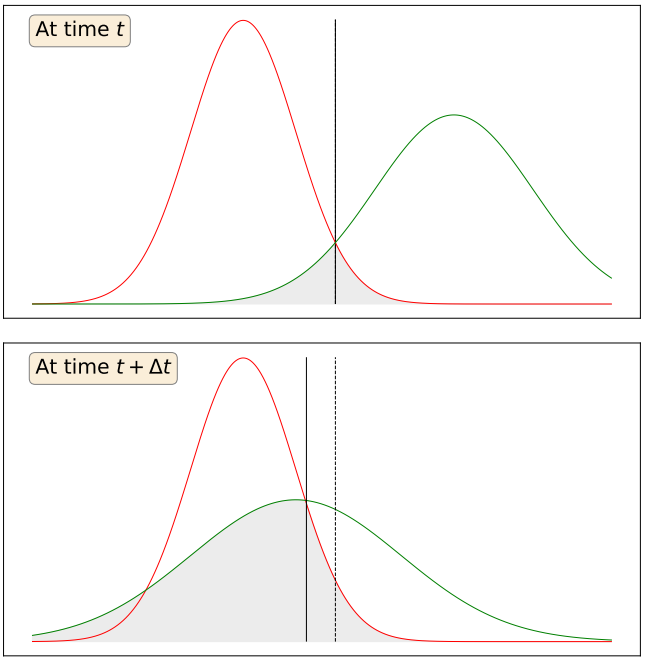

Fig. 2. Real drift

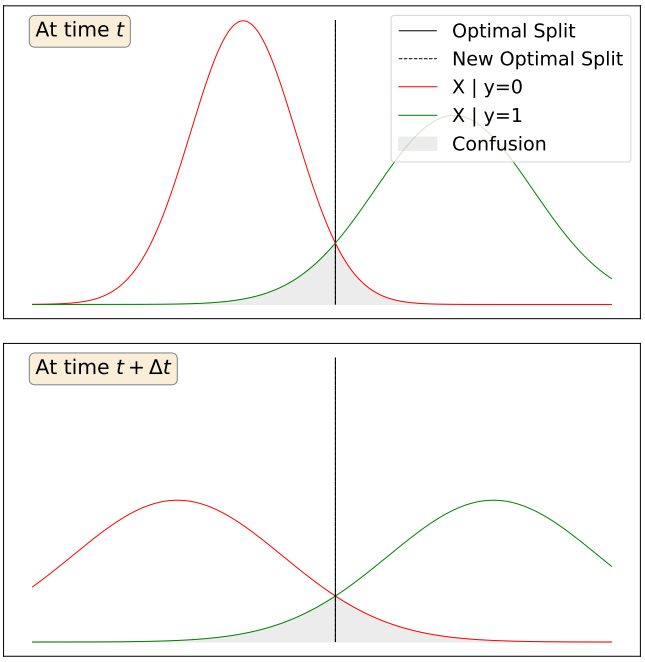

Fig. 3. Virtual drift

TABLE I
REAL AND VIRTUAL DRIFT CLASSIFICATION

	dataset	Cifar-10	Cifar-100	Fashion	Mnist
Type	Intensity				
Gaussian Noise	large	0.00	0.00	*1.00*	*0.67*
	medium	0.00	0.00	0.00	0.00
	small	0.00	0.00	0.00	0.00
Image Shift	large	*1.00*	0.33	*1.00*	0.00
	medium	0.33	0.17	*0.83*	*1.00*
	small	0.00	0.00	*0.67*	*0.67*

We evaluate the performance of GSD with $\tau = 0.25$ against other state of the art unsupervised drift detectors :

- Discriminative Drift Detector (D3) [12]
- Student-Teacher (ST) [8]
- Task Sensitive Drift Detector (TSDD) [7]

All detectors were used with their default parameter values.

Our experiment is carried out on the MNIST [19], CIFAR-10 [18], CIFAR-100 [18] and FASHION-MNIST [28] datasets.

In Table I, we show whether the combination of different types of perturbations and their respective strengths yields real drifts or virtual drifts. We consider a drift to be real if the perturbation generated a real drift at least in 50% of the cases. In italic are the settings that produce real drift. We note that when Gaussian Noise is added, it only produces real drift when the intensity is large. We also note that none of the perturbations yield real drift on the Cifar-100 dataset In order to correctly conduct our experiment, each experiment is performed over ten runs.

We group the experimental results in two tables (Table II and Table III). Table II reporting the performance of detectors when shown virtual drift, Table III reporting the results on real drift.

2) Results: Table II evaluates the detectors on their power to ignore virtual drift by reporting the true negative rate. A high true negative rate signifies that the detector accurately ignored virtual drift. When PCA is used, GSD consistently ignore the virtual drifts on the Fashion and MNIST datasets. GSD only comes last on the Cifar-10 and Cifar-100 datasets. D3 comes in second place with the lowest number of false positives on the Cifar-100 dataset and being the second best on the Fashion and MNIST datasets. TSDD takes the third place closely followed by the Student-Teacher model both achieving 47% on the Cifar-10 dataset. When a VAE is used, GSD achieves good results on the Mnist and Cifar-100 datasets. D3, ST and TSDD consistently ignore the virtual drifts on all the datasets.

Table III reports the detection results of models on real drift. A high true positive rate indicates a good detection by the model on real drift. When the dimensionality is reduced using PCA, GSD comes first with the best detection rate on the Fashion and MNIST datasets. GSD comes in second on the Cifar-100 dataset. The D3 model comes second on the Cifar-10 and Cifar-100 datasets. The ST model makes four correct detections. The Student-Teacher performs poorly with no consistent detections. Both TSDD and ST have trouble

TABLE II
VIRTUAL DRIFT DETECTION

Method	Dataset	Cifar-10	Cifar-100	Fashion	Mnist
PCA	GSD	0.33	0.33	**0.67**	**0.73**
	D3	0.45	**0.86**	0.48	0.64
	ST	**0.47**	0.83	0.28	0.55
	TSDD	**0.47**	0.83	0.39	0.64
VAE	GSD	0.60	0.89	0.40	**1**
	D3	**1**	**1**	**1**	**1**
	ST	**1**	**1**	**1**	**1**
	TSDD	**1**	**1**	**1**	**1**

TABLE III
REAL DRIFT DETECTION

Method	Dataset	Cifar-10	Fashion	Mnist
PCA	GSD	0.0	**0.80**	**0.40**
	D3	**0.10**	0.33	0.20
	ST	0.0	0.00	0.0
	TSDD	0.0	0.15	0.20
VAE	GSD	**0.10**	**0.80**	**0.10**
	D3	0.0	0.20	0.0
	ST	0.0	0.0	0.0
	TSDD	0.0	0.0	0.0

detecting drifts on the datasets that contain the most features. When a VAE is used, GSD comes first on all the datasets, although with a relatively low detection rate on the Cifar-10 and Mnist datasets. No drift detection is made by the ST and TSDD models, D3 manages to get a 20% true detection rate on the Fashion dataset.

In this benchmark, the drift detectors that ignore virtual drift, have trouble to detect real drift while the detectors that consistently detect real drift cannot ignore virtual drift. We demonstrated that GSD raises little to no false alarms, while achieving a good level of performance in detecting real drift offering a good compromise in between a high false positive rate and a low true positive rate. GSD is designed to work best when the input data follow Gaussian distributions. Our experimental protocol shows that it is the case but also suitable when it is not the case.

IV. CONCLUSION

The non discrimination of virtual and real drift is not suitable for industrial applications as it can generate a high number of false alarms, triggering costly label acquisition processes or long retraining times especially when dealing with computer vision tasks.

In this paper we introduced GSD, a novel drift detector that does not need labels to work. GSD works best when the data distribution is normal by design. However, our experiment shows that it is able to detect drift even when the normality hypothesis is not respected. We demonstrated its ability to ignore virtual drift while keeping a good level of detection in the presence of real drift. The focus of future work will be evaluating the pertinence of the model when the image variables are evaluated differently. Indeed, our method applies directly to the pixels and aims to detect drift if the pixel values change. Architectures such as Convolutional Neural Networks

(CNNs) can be used as feature extractors, with our detector connecting to the output of the layers just before the multilayer perceptron (MLP) classifiers.

References

[1] Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams. vol. 6, pp. 77–86 (2006)

[2] Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. pp. 443–448. SIAM (2007)

[3] Black, M., Hickey, R.: Learning classification rules for telecom customer call data under concept drift. Soft Computing **8**(2), 102–108 (2003)

[4] Breiman, L.: Random forests. Machine learning **45**, 5–32 (2001)

[5] Brzezinski, D., Minku, L.L., Pewinski, T., Stefanowski, J., Szumaczuk, A.: The impact of data difficulty factors on classification of imbalanced and concept drifting data streams. Knowledge and Information Systems **63**(6), 1429–1469 (2021)

[6] Brzezinski, D., Stefanowski, J.: Reacting to different types of concept drift: The accuracy updated ensemble algorithm. IEEE Transactions on Neural Networks and Learning Systems **25**(1), 81–94 (2013)

[7] Castellani, A., Schmitt, S., Hammer, B.: Task-sensitive concept drift detector with constraint embedding. In: 2021 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 01–08. IEEE (2021)

[8] Cerqueira, V., Gomes, H.M., Bifet, A.: Unsupervised concept drift detection using a student–teacher approach. In: International Conference on Discovery Science. pp. 190–204. Springer (2020)

[9] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society: series B (methodological) **39**(1), 1–22 (1977)

[10] Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10), 1517–1531 (2011)

[11] Frittoli, L., Carrera, D., Boracchi, G.: Change detection in multivariate datastreams controlling false alarms. In: Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21. pp. 421–436. Springer (2021)

[12] Gözüaçık, Ö., Büyükçakır, A., Bonab, H., Can, F.: Unsupervised concept drift detection with a discriminative classifier. In: Proceedings of the 28th ACM international conference on information and knowledge management. pp. 2365–2368 (2019)

[13] Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. The Journal of Machine Learning Research **13**(1), 723–773 (2012)

[14] Heusinger, M., Schleif, F.M.: Reactive concept drift detection using coresets over sliding windows. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1350–1355. IEEE (2020)

[15] Hinder, F., Vaquet, V., Brinkrolf, J., Hammer, B.: On the change of decision boundary and loss in learning with concept drift. In: International Symposium on Intelligent Data Analysis. pp. 182–194. Springer (2023)

[16] Kelly, M.G., Hand, D.J., Adams, N.M.: The impact of changing populations on classifier performance. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 367–371 (1999)

[17] Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. The Journal of Machine Learning Research **8**, 2755–2790 (2007)

[18] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[19] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[20] de Lima Cabral, D.R., de Barros, R.S.M.: Concept drift detection based on Fisher's exact test. Information Sciences **442**, 220–234 (2018)

[21] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363 (2018)

[22] Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Discovery science. vol. 4755, pp. 264–269. Springer (2007)

[23] Raab, C., Heusinger, M., Schleif, F.M.: Reactive soft prototype computing for concept drift streams. Neurocomputing **416**, 340–351 (2020)

[24] Rabanser, S., Günnemann, S., Lipton, Z.: Failing loudly: An empirical study of methods for detecting dataset shift. Advances in Neural Information Processing Systems **32** (2019)

[25] Sethi, T.S., Kantardzic, M.: On the reliable detection of concept drift from streaming unlabeled data. Expert Systems with Applications **82**, 77–99 (2017)

[26] Sethi, T.S., Kantardzic, M., Arabmakki, E.: Monitoring classification blindspots to detect drifts from unlabeled data. In: 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI). pp. 142–151. IEEE (2016)

[27] Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L., Petitjean, F.: Characterizing concept drift. Data Mining and Knowledge Discovery **30**(4), 964–994 (2016)

[28] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)