Drift Detection for Black Box Deep Learning Models

Drift Detection for Black Box Deep Learning Models

Luca Piano
Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, 10129, Italy

Fabio Garcea
Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, 10129, Italy

Andrea Cavallone
Data Science Center of Excellence, Reale Mutua Assicurazioni, Turin, 10122, Italy

Ignacio Aparicio Vazquez
Data Science Center of Excellence, Reale Seguros, Madrid, 28002, Spain

Lia Morra
Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, 10129, Italy

Fabrizio Lamberti
Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, 10129, Italy

***Abstract*—Dataset drift is a common challenge in machine learning, especially for models trained on unstructured data, such as images. In this paper, we propose a new approach for the detection of data drift in black box models, which is based on Hellinger distance and feature extraction methods. The proposed approach is aimed at detecting data drift without knowing the architecture of the model to monitor, the dataset on which it was trained, or both. The paper analyzes three different use cases to evaluate the effectiveness of the proposed approach, encompassing a variety of tasks including document segmentation, classification, and handwriting recognition. The use cases considered for the drift are adversarial assaults, domain shifts, and dataset biases. The experimental results show the efficacy of our drift detection approach in identifying changes in distribution under various training settings.**

ARTIFICIAL INTELLIGENCE (AI) and especially machine and deep learning (ML/DL), has become increasingly pervasive in our society as the number of AI-based products and services skyrockets. Since training requires expertise and computing infrastructure, many companies choose to outsource the design and implementation of DL models or rely on commercial AutoML APIs; the resulting black box models can be less transparent and controlled than models developed in-house. Despite the impressive results achieved by DL techniques, there are growing concerns about the potential risks associated with their use, with an increasing need to regulate their development and deployment. Compliance with emerging AI regulations will require a post-market monitoring plan for medium- and high-risk AI systems to ensure that they operate correctly and consistently throughout their life cycle [1].

A significant issue that arises with ML/DL systems is dataset drift [2], i.e. a change in the distribution of the data over time that can lead to a degradation of model performance (model decay). Without suitable metrics, retraining is usually

Published by the IEEE Computer Society

triggered either by empirical evidences, without proper oversight by data scientists, or on a fixed time schedule. Both of these solutions may lead to premature or late retraining, with the former resulting in a waste of time and resources, and the latter to performance drops. If the AI solution is deployed in a business-driven environment, this may lead to important consequences for both the company and the customers.

Drift detection tackles this issues, by providing consistent evidences for optimal retraining. Most drift detection techniques are based on monitoring the error rate, thus assuming the availability of labels at inference time (supervised setting). This approach may be costly to implement if, e.g., labels are manually annotated; in some applications, labels could become available weeks or even months after the model prediction, thus incurring in delayed drift detection. For these reasons, there is an increasing interest into unsupervised drift detection, which shifts the focus to monitoring the statistical distribution of the input data. In the case of unstructured data, such as images, data samples must be transformed into a compact embedding space [3], [4], [5]. State-of-the-art techniques achieve this goal by either requiring access to the model and its internal features [4], [6], or by requiring access to the training set to train an auxiliary model [7], [8].

In this paper, we relax these constraints and simulate a model-agnostic scenario in which the architecture of the ML model is not known and its inner features are not accessible (black box setting); additionally, the user may not have access to the training dataset (or an equivalent dataset in terms of size and distribution). An embedding or feature vector must be extracted without relying on the internal features computed by the model, and any auxiliary model should be either pretrained or finetuned on a small reference set collected at the beginning of the operating phase. This reference sample would, in any case, be needed to establish the reference distribution, if the model is not trained in-house.

This paper extends our previous methodology primer on concept drift detection [4] by experimentally comparing different ways to achieve this goal. Empirical results on three different use cases, including document segmentation, image classification, and handwriting recognition, show that unsupervised drift detection is feasible even within this very challenging setting. The suggested strategy is expected to be beneficial to AI practitioners as well as IT professionals who need to integrate black box models into production environments.

INTRODUCING DATASET DRIFT

Dataset drift is a change in the distribution of source and/or target data that can degrade the performance of ML/DL models trained on historical data [2]. Dataset drift can limit the generalization capabilities of the models when deployed in the field, and detecting it is crucial to trigger a model update. For a more detailed introduction to the topic, the reader is referred to previous work [2], [4], [9].

Techniques to detect dataset drift can be broadly categorized into supervised [10], [11] and unsupervised methods [2], [5], [12]. The latter detect drift by checking whether the distribution of the input data differs from a reference one, computed on the the training set, representative of typical operating conditions or collected at the time of model deployment [5], [12]. Several works have tackled how to efficiently and effectively compare high-dimensional multi-variate data distributions. To this aim, distance measures [12], [4], [5], [6] were shown to be more practical than multi-variate statistical testing [3], as they are faster to compute and scale better to higher dimensional feature spaces.

For unstructured data such as images, however, it is necessary to first reduce the dimensionality of the input data before drift detection techniques can be applied [3], [4]. In this respect, existing approaches to drift detection often assume that the architecture of the model, the data used to train it, or the model itself are known. The most common and straightforward approach is to exploit the inner features computed by the model [4], [6], [3]. Other techniques exploit model-specific information, such as explanations computed by XAI techniques, to detect data changes [7]. Several recent studies required training an additional auxiliary model. For instance, the embedding space could be computed by an auto-encoder [3], [13] or by a teacher-student model [8]. The auxiliary model could be used to compute the embedding space directly; alterna-

tively, its predictions could be used to determine whether a given instance is consistent with the training set distribution by, e.g., measuring some form of reconstruction errors [13]. However, these approaches were always investigated assuming that a suitable training set is available; in some cases, they even require modifying the model or the training procedure [14], [15].

DEFINITION OF THE USE CASES

We investigated three different use cases: two standard benchmarks from the literature [3], and one dataset that was previously proposed for drift detection in business scenarios [4]. Each use case is characterized by a source task on which the black box model was trained (classification, semantic segmentation, recognition), a source dataset used for training, a data drift model used to simulate datasets with known drift, and a set of suitable techniques used to compute the embedding space for drift detection and correlate it with performance degradation. Combined, these use cases include a variety of situations in which adversarial attacks, domain changes, and dataset bias may pose challenges for ML/DL models. Datasets are available for download at https://gitlab.com/grains2/concept-drift-benchmarks.

Use case 1: CIFAR10 versus Adversarial

The first use case models adversarial attack as the source of drift, using CIFAR10 as base dataset and FGSM as the attack model [16]. The latter uses backpropagation to produce adversarial samples that are misclassified by the model, although they are indistinguishable from the original data by a human observer, as shown in Figure 1(a). To generate the Out of Distribution (OOD) adversarial samples, we set the intensity parameter (ϵ) to 0.05.

Use case 2: MNIST vs Colored

For the second use case, In Distribution (ID) images were sampled from MNIST and OOD images from Colored MNIST, in which the samples, originally in grayscale, have the background and the number colored (Figure 1(b)). The colored samples were generated by choosing two colors that satisfied the requirement of the WCAG WAI AA guideline, which defines an acceptable contrast for reading online. This use case allows

(a) Example of adversarial examples generated by FGSM[16].

(b) MNIST vs Colored MNIST.

(c) Example of documents.

Figure 1: Examples from each of the selected use cases.

to easily create datasets affected by systematic biases. Specifically, in this use case the network was trained on grayscale images and tested on colored ones. It is also possible to generate datasets affected by systematic biases by, e.g., associating specific colors to individual digits.

Use case 3: Document segmentation

The focus of the third use case is document segmentation. The DL model, described in previous work [4], was trained on synthetically generated data. New invoice templates with previously unseen attributes were produced to evaluate the ability of the model to manage deviations from the training data. Specifically, background color and background pictures were introduced as new features. A dataset of 30,000 invoices was created, with 10,000 documents for each scenario (Figure 1(c)): without new features, feature A (background color), and feature B (background image). A DL model was trained to identify invoice components (logo, header, table, footer, and background). Briefly, the DL model is an encoder-decoder which takes as input the images along with a binary mask generated by running optical character recognition (OCR), which represents presence or absence of text. The baseline

model and dataset are described in greater detail in [4].

Drift simulation

For all the use cases, we simulated 11 time windows where the percentage of OOD samples increased linearly from 0% to 100% with 10% steps. We randomly sampled $N = 1,000$ samples, without repetitions, from both the ID and the OOD sets in each window, where the window size N can be adjusted based on the expected workload of the model and the drift detector sensitivity.

Methodology

Drift detection for unstructured data usually follows a two-step approach: first, the raw input is compressed by applying a dimensionality reduction technique, then the distribution at the current time is compared against a reference distribution [4], [5], [12]. Once a measure of diversity is obtained, drift can be detected using a fixed or adaptive threshold [17]. In this paper, we assume that the Hellinger distance is used to compare the current and reference time windows. The Hellinger distance is a widely used metric for comparing distributions and is known to perform well in drift detection [4], [12]. Existing libraries, such as TorchDrift [17], are based on multivariate statistical models and do not scale well with the size of the embedding. This choice is also supported by recent work on textual data, using BERT as feature extractor, by Greco et al [6].

We focus here on the problem of projecting raw data onto an appropriate embedding space that is lower-dimensional and semantically rich. Model-aware techniques (which assume that the model, trained on the source task and being monitored, is known and can be inspected at inference time), and model-agnostic techniques (which assume that the model being monitored is unknown and that only its input and output will be accessible at inference time) can be used to extract these embeddings. Model-agnostic techniques may need to choose and/or train a surrogate model to project the input data into a small embedding space, and this surrogate model may differ from the model being trained, both in terms of structure and training data.

In this paper, we experimentally compare different model-agnostic techniques under the following experimental settings:

- Both the original model architecture and the training dataset are unknown;
- The original model architecture is known, whereas the dataset is unknown;
- The dataset is known, whereas the original model architecture is unknown.

Moreover, we consider two families of architectures to extract model-agnostic embeddings:

- **Pretrained ResNet**: The excellent performance of ResNet in image classification tasks and its capacity to serve as a pretrained model for transfer learning make it a commonly used neural network architecture. ResNets can effectively extract features from high-dimensional input data as a feature extractor, resulting in a semantically rich embedding space that can be applied to downstream tasks.
- **Autoencoders**: In this form of unsupervised learning, the autoencoder learns the representations (embeddings) of data samples by compressing the input into a low-dimensional latent space representation (*encoder*) and reconstructing the original image as output (*decoder*) [18]. This is achieved by minimizing a reconstruction loss $L_{\mathrm{r}}(x, \hat{x})$ between the original input and the reconstructed output. After unsupervised training, the encoder can be used as an auxiliary model to extract data representations in place of the black box source model.

RESULTS

We investigate different techniques to compute model-agnostic features, and compare them against a state-of-the-art baseline in which the features are those computed internally by the model.

In the first use case (CIFAR10 vs. Adversarial), we assumed that **the training dataset is known, but the model architecture is unknown**. A ResNet50 model was used as the black box model, and a ResNet18 model as a feature extractor for the drift. Both the networks were trained on CIFAR10[1], and the source model

[1]Weights are available at https://github.com/huyvnphan/PyTorch_CIFAR10

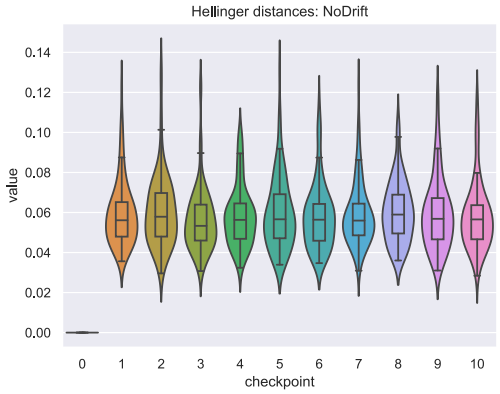
(ResNet50) was used to create the adversarial samples. Interestingly, as shown in Figure 2, a different model pretrained on the same dataset detects changes in the distribution due to adversarial examples. In fact, adversarial examples were shown in the literature to be dataset-dependent and, thus, transferable across different architectures, despite being generated to disrupt a specific target model [19]. We further calculated the model performance (measured by the AUC per class and the accuracy) for each time window, and detected a clear link between the introduction of adversarial examples and model performance, which dropped by more than 0.17. Thus, it may be possible to identify a drop in performance due to adversarial sample using the drift revealed by a different, and smaller, architecture.

In the second use case (MNIST vs. Colored) we investigated a scenario where **the training set is unknown or unavailable, but the black box architecture is known**. Therefore, we employed a ResNet18 pretrained on CIFAR10 as the feature extractor and a ResNet18 pretrained on MNIST as the model to monitor. Again, as shown in Figure 3, the Hellinger distance increased. We verified that the the target model performance dropped (by more than 0.21) when the drift was present.

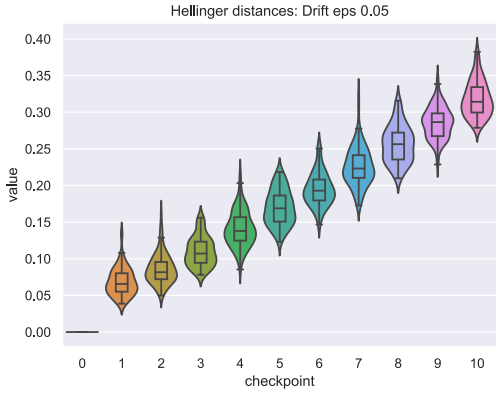
For the third use case (document segmentation), we considered **a setting in which neither the architecture nor the training dataset is known**. In this case, we considered as feature extractor a ResNet18 backbone pretrained on ImageNet. The encoder of a Variational Autoencoder (VAE) model pretrained on LAION-5B [20] was tested on all datasets[2].

We measured the ability of the proposed drift detectors to discriminate time windows in which drift is present from those sampled from the training distribution by using the Hellinger distance between the current and reference time windows as a "driftiness" score, as done in previous work[4]. The resulting corresponding Area under the ROC curves and selected ROC curves are reported in Table 1 and Figure 4, respectively.

In simpler benchmarks (CIFAR10 and MNIST), almost all the feature extractors are capable of detecting drift with good performance (AUC $>$ 0.8 with at least 20% OOD samples) and progressively match the baseline as the percentage of OOD samples increases to 30% or greater. At low rates of OOD samples (10%), pretraining on a very different dataset (e.g., pretraining on CIFAR10 to detect drift in the MNIST scenario) reduces the ability to detect drift (e.g., from AUC $=$ 1.0 to AUC $=$ 0.590). When extracting the features from the black box



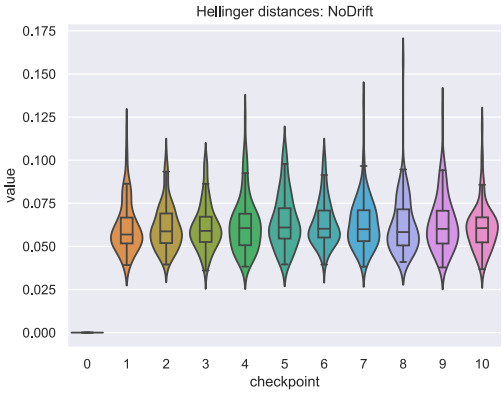
(a)



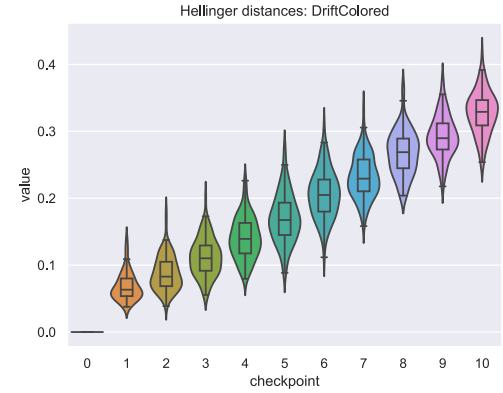
(b)

Figure 2: Distribution (violin plots) of the Hellinger distance on CIFAR10 vs. Adversarial,-black box model ResNet50, feature extractor ResNet18. No drift (a), Linear gradual concept drift (b). The Hellinger distance measures the distance between the current time window and a reference time window without drift ($p = 0$). $Z = 100$ time windows are sampled for each drift scenario and for each fraction p of OOD samples to generate the distribution.

[2]The model is available at https://huggingface.co/CompVis/stable-diffusion-v1-4



(a)



(b)

Figure 3: Distribution (violin plots) of the Hellinger distance on MNIST vs. Colored - Black box model: ResNet18 MNIST - Feature extractor: ResNet18 Cifar10. No drift (a), Linear gradual concept drift (b). The Hellinger distance measures the distance between the current time window and a reference time window without drift ($p = 0$). $Z = 100$ time windows are sampled for each drift scenario and for each fraction p of OOD samples to generate the distribution.

model itself, the Hellinger distance of different time windows sampled from the training dataset is on average smaller than when using features trained on different datasets, hence the lower separability between drift and no-drift windows.

In the document segmentation task, two types of drift were injected. We found that the ResNet18 pretrained on ImageNet was capable of detecting both type of drifts, but with substantial differences when compared to the embeddings extracted from the original encoder: the ImageNet model was more sensitive to drifts caused by the introduction of the background image (AUC $= 0.752$ vs. 0.503 at OOD sample rate of 10%), and slightly less sensitive to drifts due to changes in the background color (AUC $= 0.822$ vs. 0.975 at OOD sample rate of 10%). Indeed, in previous work it was shown that the performance of the segmentation model was marginally affected by the presence of background images [4]; we postulate that this feature is effectively attenuated by the segmentation model and, by extension, by the drift detection built on top of this model.

Finally, despite being trained in an unsupervised manner, the results obtained with the VAE are in line with those obtained using the other feature encoders. Interestingly, the VAE model detects adversarial examples with superior accuracy than the original model, even though the training dataset was different (AUC $= 0.603$ vs. 0.648 and AUV $= 0.887$ vs. 0.836 at 10% and 20% OOD sample rate, respectively). In the document segmentation task, the behavior of the chosen VAE is similar to that of the original document segmentation, which is also based on an encoder-decoder model, at very low OOD sample rates (AUC $= 0.503$ vs. 0.500), but then rapidly increases (AUC $= 0.536$ vs. 0.817).

To conclude, all the causes of drift can be accurately identified as long as more than 20% of samples are taken from the new distribution. The optimal threshold for the Hellinger distance could be selected based on the ROC curves in Figure 4, depending on the desired level of sensitivity.

CONCLUSIONS

Black box models are commonly used in ML/DL due to their high accuracy and ability to handle complex problems. However, monitoring their performance can be challenging, as the training data and model architecture may be unknown, and the internal model may not be accessible. This lack of transparency can prevent the detection of model degradation in cases of changes in the data distribution. In general, based on the outcomes of the current and previous studies [4], the inner features of the model ensure better correlation with the observed model degradation. Nonetheless, a variety of encoders can be successfully used to detect drift whenever

			Percentage of OOD samples					
Dataset	**Model**	**Pre-trained**	**10%**	**20%**	**30%**	**40%**	**50%**	**60% - 100%**
CIFAR10	ResNet50†	Cifar10†	0.603	0.836	0.936	0.983	0.999	**1.000**
	ResNet18	Cifar10	**0.654**	0.856	0.979	**0.999**	**1.000**	**1.000**
	VAE	LAION5B	0.648	**0.887**	**0.989**	**0.999**	**1.000**	**1.000**
MNIST	ResNet18†	MNIST†	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	ResNet18	Cifar10	0.590	0.839	0.954	0.990	0.998	**1.000**
	VAE	LAION5B	0.988	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Document segmentation (Color Drift)	Custom [4]†	Documents [4] †	**0.975**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	ResNet18	ImageNet	0.822	0.997	**1.000**	**1.000**	**1.000**	**1.000**
	VAE	LAION5B	0.878	0.995	**1.000**	**1.000**	**1.000**	**1.000**
Document segmentation (Image Drift)	Custom [4]†	Documents [4]†	0.503	0.536	0.606	0.649	0.763	0.832 $\rightarrow$ 0.959
	ResNet18	ImageNet	**0.752**	**0.983**	**1.000**	**1.000**	**1.000**	**1.000**
	VAE	LAION5B	0.500	0.817	0.992	**1.000**	**1.000**	**1.000**

Table 1: Drift detection performance (AUC) as a function of the fraction of OOD samples. For each scenario we report the target dataset/drift model, the architecture of the feature extraction and the dataset on which the feature extractor was trained. The best performance for each use case is highlighted in bold. Rows indicated by † correspond to the standard model-aware baseline in which the drift detector operates on the internal model features.

the inner features computed by the model cannot be inspected. In particular, general-purpose models, pretrained on large scale datasets such as ImageNet or LAION-5B, were shown to be effective and generalize across various types of drift, including adversarial attacks. Such solutions are simple to implement and incur in minimal computational costs. Better alignment between the black box model and the drift detector could be achieved, in principle, by training an auxiliary model under the teacher-student framework [8], [21]. Further studies are needed to understand if this approach is feasible under the constrained setting we considered, in which the student model should be trained on a few hundred samples. Additionally, when the black box model is provided by a third party, legitimate concerns related to model reverse engineering could arise.

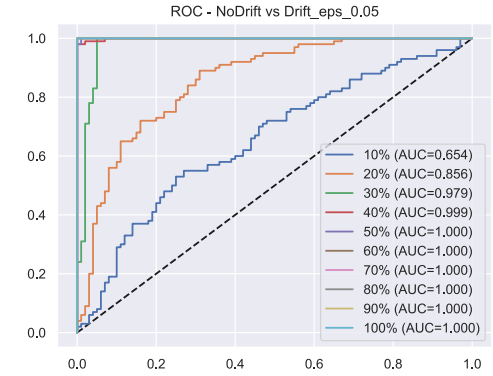
Nevertheless, the rich general-purpose representations computed by the feature extractor employed by the drift detector may be sensitive to changes in the input distribution that do not directly impact the performance of the black box model, thus potentially raising false alarms. The opposite is also possible, albeit less prominent in our experiments. Differences may be induced both by the different training set or by the architecture and learning paradigm (supervised or unsupervised). Further studies are needed to investigate these aspects from a theoretical, as well as empirical, standpoint. The investigation could be extended to other types of inputs (e.g., text, video), to features extracted from self-supervised or language-supervised models, as well as to other scenario and dataset drifts (e.g., datasets collected within autonomous driving exhibit distinctive drift patterns). Finally, it would be interesting to explore the proposed feature extractors in combination with other distance measures [6] to better understand the interplay between the two components.

ACKNOWLEDGMENTS

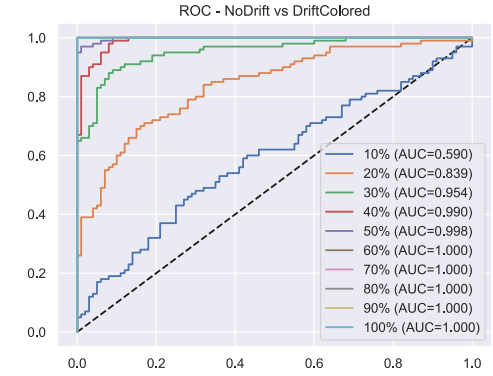
This work was supported by a grant from Reale Mutua Assicurazioni.

REFERENCES

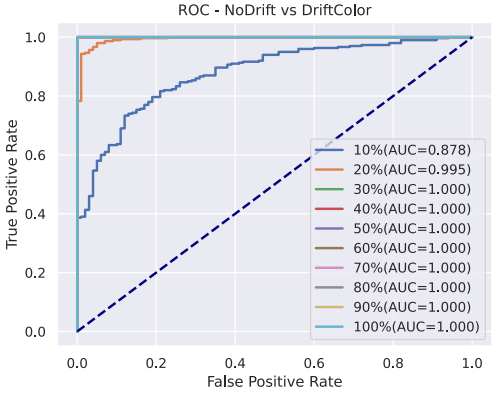
1. L. Floridi, M. Holweg, M. Taddeo, J. Amaya Silva, J. Mökander, and Y. Wen, "capAI – A procedure for conducting conformity assessment of ai systems in line with the EU Artificial Intelligence Act," *Available at SSRN 4064091*, 2022.
2. J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.
3. S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, pp. 1396–1408, 2019.
4. L. Piano, F. Garcea, V. Gatteschi, F. Lamberti, and L. Morra, "Detecting drift in deep learning: A methodology primer," *IT Professional*, vol. 24, no. 5, pp. 53–60, 2022.
5. T. Cerquitelli, N. Nikolakis, L. Morra, A. Bellagarda, M. Orlando, R. Salokangas, O. Saarela, J. Hietala,



(a)



(b)



(c)

Figure 4: ROC curves for each type of drift as a function of the percentage p of OOD samples for (a) CIFAR10/ResNet18, (b) MNIST/ResNet18, and (c) Document segmentation/VAE.

P. Kaarmila, and E. Macii, "Data-driven predictive maintenance: A methodology primer," in *Predictive Maintenance in Smart Factories*, 2021, pp. 39–73.

6. S. Greco and T. Cerquitelli, "Drift lens: Real-time unsupervised concept drift detection by evaluating per-label embedding distributions," in *Proc. International Conference on Data Mining Workshops*, 2021, pp. 341–349.
7. J. Demšar and Z. Bosnić, "Detecting concept drift in data streams using model explanation," *Expert Systems with Applications*, vol. 92, pp. 546–559, 2018.
8. V. Cerqueira, H. M. Gomes, A. Bifet, and L. Torgo, "Studd: A student–teacher method for unsupervised concept drift detection," *Machine Learning*, pp. 1–28, 2022.
9. J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer, *Dataset shift in machine learning*. Mit Press, 2009.
10. J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Brazilian Symposium on Artificial Intelligence*, 2004, pp. 286–295.
11. S. Disabato and M. Roveri, "Learning convolutional neural networks in presence of concept drift," in *Proc. International Joint Conference on Neural Networks*, 2019, pp. 1–8.
12. G. Ditzler and R. Polikar, "Hellinger distance based drift detection for nonstationary environments," in *Proc. IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, 2011, pp. 41–48.
13. T. Che, X. Liu, S. Li, Y. Ge, R. Zhang, C. Xiong, and Y. Bengio, "Deep verifier networks: Verification of deep discriminative models with deep generative models," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 7002–7010.
14. P. Wang, N. Jin, D. Davies, and W. L. Woo, "Model-centric transfer learning framework for concept drift detection," *Knowledge-Based Systems*, p. 110705, 2023.
15. C. Geng, S.-j. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *IEEE Transactions on Pattern Analysis and Mchine Intelligence*, vol. 43, pp. 3614–3631, 2021.
16. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
17. T. Viehmann, L. Antiga, D. Cortinovis, and L. Lozza, "Torchdrift: Drift detection for pytorch," 2019. [Online]. Available: https://torchdrift.org/
18. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
19. A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they

are features," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
20. C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *arXiv preprint arXiv:2210.08402*, 2022.
21. C. Chen, G. Yao, C. Wang, S. Goudos, and S. Wan, "Enhancing the robustness of object detection via 6G vehicular edge computing," *Digital Communications and Networks*, vol. 8, no. 6, pp. 923–931, 2022.

Luca Piano is PhD candidate at Politecnico di Torino. Contact him at luca.piano@polito.it.

Fabio Garcea is PhD candidate at Politecnico di Torino. Contact him at fabio.garcea@polito.it.

Andrea Cavallone has a MS degree in Complex Systems Physics and a PhD in Complex Systems for Life Sciences from University of Turin. He worked in the pharmaceutical industry and consultancy before joining Reale Mutua Assicurazioni. His areas of expertise are mathematical modeling and data science.

Ignacio Aparicio has a MSc degree in Data Science from the Universidad Politécnica de Madrid and a MSc degree in Data Science from the Royal Institute of Technology (KTH) in Stockholm. He has worked in the automotive and banking industries before joining Reale Seguros. His main area of expertise is data science.

Lia Morra is Assistant Professor at Politecnico di Torino. Her research interests include machine learning, image interpretation, and human-computer interaction. Contact her at lia.morra@polito.it.

Fabrizio Lamberti is Full Professor at Politecnico di Torino. His research interests include computational intelligence, human-computer interaction, computer graphics, and visualization. Contact him at fabrizio.lamberti@polito.it.