Understanding and evaluating data drift in computer vision from edge computing perspective

Janusz Jakub Wilczek (Author)
IT University of Copenhagen
jawi@itu.dk

Supervisors:
Pinar Tözün IT University of Copenhagen, *pito@itu.dk*
Robert Bayer IT University of Copenhagen, *roba@itu.dk*

A Thesis presented for the Degree of
Bachelor of Science in Data Science

IT UNIVERSITY OF COPENHAGEN

IT University of Copenhagen
Course Code: BIBAPRO1PE

May, 15th 2024

Abstract

Concept drift (CD) detection in edge computing (EC) environments remains under-explored, with limited investigation into suitable techniques and their efficacy .Challenges such as inconsistent terminology complicate analysis, warranting a more stringent categorization approach. This study addresses this gap by examining CD detection methods in computer vision (CV) from EC perspective. However, due to incomplete experiments, further exploration is needed. Preliminary experiments highlight the critical role of batch size in detection accuracy. Overall, this study underscores the necessity for comprehensive exploration and evaluation of CD detection methods within EC and CV.

1 Introduction

The Internet of Things (IoT) presents a unique environment where devices are constrained in terms of computational power and memory. Despite these limitations, these devices can generate vast amounts of data [23]. However, cloud computing often fails to meet applications' needs due to factors such as "low throughput, high latency, bandwidth bottlenecks, data privacy, centralized vulnerabilities, and additional costs" [14]. A potential solution lies in performing calculations directly on the edge devices, known as edge computing (EC), despite their inherent limitations [14].

Various steps of the ML life cycle have been explored from an EC perspective, including training, deployment, and pre-processing [17][24][12]. However, ensuring the long-term success of ML models necessitates monitoring their performance. Young [32] has demonstrated the declining performance of various model architectures over time, where a decline has been observed for each. This phenomenon is known as model degradation.

ML engineers employ various approaches to address degradation, one of which is Continuous ML training and evaluation [20], which entails regular updates to the models alongside testing. However, in the context of IoT ML, additional workload and data requirements are a larger challenge than in a more standard setting. A possible way to reduce those expenses would be to investigate the reasons behind model degradation and find a solution there.

Given ML's limited ability to abstract rules [16] [22], ML models are overly reliant on the training data. Thus, their performance is sensitive to the differences between training and production data. Therefore, any series of methods that can detect those differences, which could be linked to model degradation, are desirable. This would achieve two goals: reducing data transfer (data does not need to be transferred to obtain labels) and optimizing retraining time frames (they could be executed only when data drift is detected).

There have been two prominent names to describe this phenomenon: Data Drift (DD) and Concept Drift (CD). Problematically, there is a lack of consensus regarding what those terms describe. Furthermore, a mathematical description is elusive. This leads to a natural problem, especially in computer vision (CV), where highly dimensional data leads to larger computational needs, pushing EC to the limit. Therefore, in this paper, I aim to answer"

- RQ: What is performance of unsupervised detection methods in detecting drift in CV tasks, in an EC environment?

2 Background

2.1 What are data and concept drifts?

Depending on the study, the meaning of data and concept drift is explained to various levels of detail. Some provide mathematical expressions, while others offer a couple of examples illustrating

	Concept Drift definitions
1	$P_t(X, y) \neq P_{t+w}(X, y)$
2	$P_t(y\|X) \neq P_{t+w}(y\|X)$
3	$P_t(y\|X) = P_{t+w}(y\|X)$ and $P_t(X) \neq P_{t+w}(X)$
4	$P_t(y\|X) \neq P_{t+w}(y\|X)$ and $P_t(X) = P_{t+w}(X)$
5	$P_t(X) \neq P_{t+w}(X)$
6	$P_t(y) \neq P_{t+w}(y)$

(1)

Table 1: Concept Drift mathematical definitions. t, w respectively signify a specific time point and the time window within which the test is conducted.

what constitutes drift, and still others stick with a brief verbal definition. This variability leads to ambiguity. For instance, [21] survey on concept drift defines it as:

> Concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways

Their definition originates from Widmer and Kubat's paper [31] and stands as one of the earliest formulations of the problem. However, since then, numerous other definitions and interpretations have emerged. This proliferation poses several challenges. Firstly, it complicates the translation of research insights into new scenarios, as the same term can encapsulate divergent ideas. Secondly, the abundance of terms and detection techniques unnecessarily complicates the evaluation process, leading to potential misapplication of techniques originally devised for different contexts.

Bayram [5] has recognized this problem and organized terminology into six mathematical expressions (Table 1). What can be noticed is that the concept drift term has been used to essentially describe any change between training and production data. This bears impact on detection techniques, as one cannot claim change in posterior probability when a detection method looks at joint distributions.

It is note-worthy that another name commonly used to describe the same concepts as captured in Table (1) is data drift (DD). This terminology is particularly prevalent in ML medical imaging studies [28], [25]. Some papers' terminology goes as far as defining DD as the group name for all the changes between training and production data, wherein concept drift is one of many subcategories [18].

Nonetheless, a large part of the most prominent surveys in the field use CD as the name for all specific drift cases [11], [21], [5], [4], [3], [19], [15]. Thus, from now on, I will exclusively use CD.

2.2 Types of concept drifts

As it stems from mathematical definitions, CD is not only described by the magnitude of a change but also its place in time. There appears to be unanimous consensus in the field regarding categorization regarding the velocity of change and cycles (see Table 1). Although similarly to the definitions, different terms have been used to describe the same notions. The types are a) sudden,

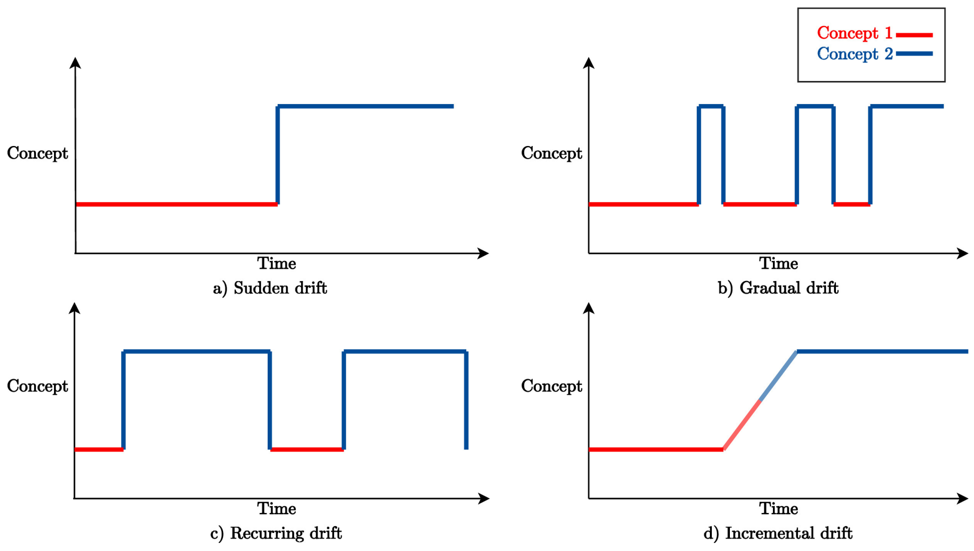

Figure 1: Concept drift types. Figure source: [5]

b) gradual, c) recurring, d) incremental [5], visualized in Figure 1.

2.3 What detection methods are available?

There have been several distinct approaches to organize all concept drift detectors (CDD). [5] organized performance-aware CDD according to the strategy used, e.g., ensemble learning. The issue with those methods from an EC perspective is that they require truth labels for the evaluation step, whose availability cannot always be assumed.

[11] created a conceptual overview of unsupervised CDDs, which do not require truth labels. In their taxonomy, models are first divided according to when and how detection is performed. The first distinction is made between online-based and batch-based methods, where the former checks each arriving data instance as they come, whereas the latter awaits until a batch is full.

Later, batch-based approaches can be divided based on whether they operate on a whole batch or a batch's subpart. In online-based methods, differentiation is made on how the reference window, i.e., the last known time before CD, is handled. Some CDDs have it fixed, whereas some slide it over time.

2.4 What are examples of concept drift in computer vision?

Notably, in the CD field there has been a predominant interest in synthetic datasets. A benefit of this approach is a high customizability and control of the drift properties, so they thoroughly follow cases as described in table (1). Furthermore, even with a higher level of scrutiny with employmnt of mathematical expressions, it is challanging to claim a drift occurence in a real-world dataset, not to mention of what type and when [6].

Many of common dataset generators are designed for classifications [4], such as Sine [2] and LED [10]. On regression side, available datasets are limited [5], and they can be artificial too [6].

Regarding CV datasets, no generational approaches have been developed, so real world data must be used. As established before, this situation becomes difficult. One cannot easily claim a drift in a real world dataset, and intuitively it is more challenging to accurately capture it. A systematic review study on CD adaptation in video surveillance [13] collected 56 datasets. Importantly, none of

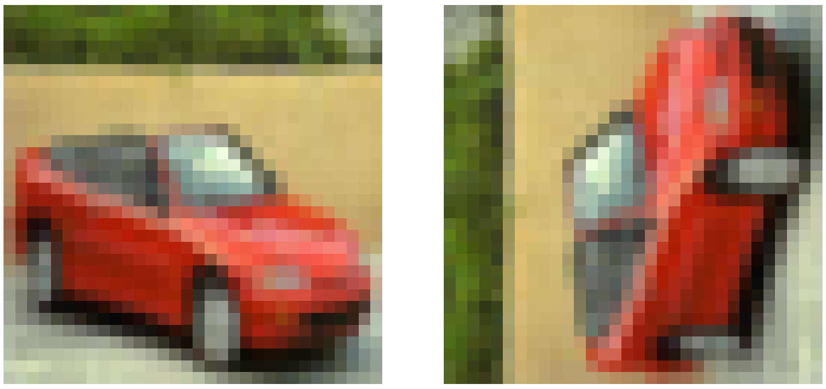
(a) CIFAR 10 original car image (b) Rotated car image by 90 degrees counter-clockwise

Figure 2: 2 Figures side by side

them had annotations for CD. Therefore, it is worth exploring how researches have handled it CD investigations in the domain.

One of the examples is a compound set of CheXpert and PadChest datasets [28], where data was combined and then divided according to the date of gathering. Those datasets are compound, i.e. contained both images and metadata. However, the study mostly focused on experimenting with metadata. Only one experiment had added lateral images, on which model was not trained on.

Another example is CXR dataset [18] in where COVID-19 emergence from 2020 onwards is present, alongside a decrease with non COVID related cases. Suprem [29] simulated CD in MNIST and CIFAR10 datasets. They 2 withhold classes for the training, which are later reintroduced in a production setting with a varying percentage of total classes they represent, between 0 to 50 percent.

2.5 Visualising CD

CD in CV presents a unique situation. Unlike regular tabular data, such as weather condition datasets with metrics like speed and temperature, for which humans are not capable of making factual assumptions about distributional changes that might have occurred just by looking at it. This holds especially true with datasets with a high number of features.

However, CV, by its nature, is different. People use their vision to discern whether images could have come from the same source. Figure 2 shows the same car images from the CIFAR10 dataset: one original (a) and one rotated by 90 degrees (b). We can tell that the image is the same, just rotated. We would say that it still shows the same car. However, if we consider how the RGB values representing it shift in the data space, it can give us an intuition as to what drift can look like. ML classifiers mislabel rotated images [7], unless they have been incorporated in the training, for example, by data augmentation [27].

Importantly, rotational drift can have an impact on what a correct label should be. Figure 3 contains a handwritten number 5 from the MNIST dataset, similarly as before in its unaltered state (a), and rotated (b). However, this time we would not necessarily intuitively claim that the rotated image still represents the digit 5. The interpretation depends on the design of the ML application [27]. This might be a blind spot in unsupervised CDD methods.

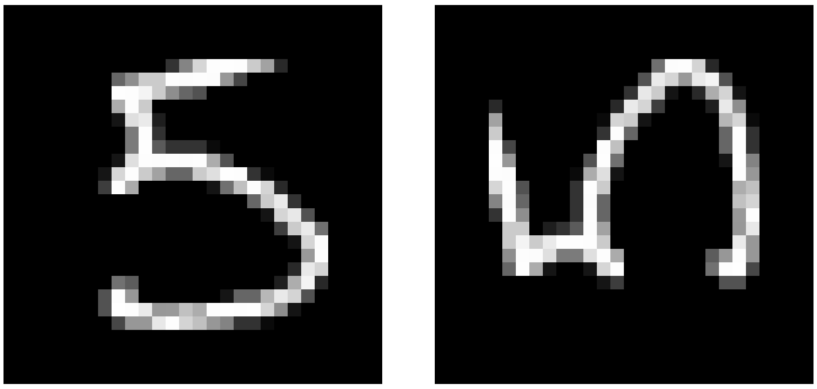
(a) MNIST no. 5 original image (b) Rotated no. 5 image by 90 degrees counter-clockwise

Figure 3: 2 Figures side by side

3 Experimental setting

So far, research has focused on applying CD in various fields, evaluating CDD techniques, and conducting systematic reviews of the knowledge. Yet, no attention has been paid to evaluating those approaches from an EC perspective, where the workload impact of CDD approaches influences the applicability of a given method.

Therefore, in this paper, I investigate to what extent unsupervised CDD methods are able to detect drift while measuring their CPU and memory utilization.

To select methods for the experiments, I used techniques outlined by [11]. Out of that list, I highlighted whether I was able to find a publicly available implementation. This narrowed the list from the initial 17 instances to only 3. One technique was discarded as it was implemented in Java, whereas the other 2 were in Python. Since Python is an interpreted language whereas Java is compiled, it has an impact on CPU and memory utilization. Therefore, only Python implementations were used, namely the drift detection method based on active learning DDAL [9] and the Incremental Kolmogorov–Smirnov-based Drift Detector (IKS-bdd) [26] [1].

3.1 CDD: IKS-bdd and DDAL

DDAL is a full batch-based approach, which calls for complete CD handling, from detection to re-training. However, I will focus on its ability to detect CD. This is achieved through the use of posterior probabilities from a model's output, compared to the user-defined hyperparameter called the uncertainty threshold. Each singular prediction is evaluated to determine whether it falls within confidence margins. This parameter governs the creation of virtual margins, set up as projections of hyperplanes. For instance, if the uncertainty threshold is set to 0.9, then for an object for which the maximum posterior value was lower than 0.9, it will be marked as significant. Once a batch is traversed, a density is calculated, which is the proportion of significant instances to the batch size. The minimum and maximum density values are stored and updated as the detection module evaluates incoming data. If the difference between the maximum and minimum densities exceeds another hyperparameter, named the drift threshold, drift is detected.

[1]Since authors did not assign a name to the proposed technique, it was done by Gemaque [11] in their survey.

IKS-bdd is an online detector with a fixed reference window. It uses the Kolmogorov–Smirnov test to detect drift, which is applied to each feature space individually, avoiding performing multivariate tests and reducing computational complexity. It keeps two windows for each feature: a fixed reference window and a sliding one, which is being updated at each iteration. The IKS test is performed between these two windows at each iteration. If a test rejects the null hypothesis that data comes from the same distribution, then drift is signaled.

3.2 Data

In the experiments, adapted versions of the MNIST and CIFAR10 datasets are used, as shown in the image of [29]. These datasets are a staple of CV, with images of low resolution, 28x28 pixels for MNIST and 32x32x3 pixels for CIFAR10. The reason being that neither DDAL nor IKS-bdd have been tested with image data. If they do not succeed in detecting drift on low-resolution images, then their ability to do so with higher resolution data is doubtful.

I apply dimensionality reduction to the CIFRAR10, converting it into a single channel, gray-scale pixels. I use PyTorch's function, which takes weighted sum from each channel.

3.3 Models for DDAL

DDAL works with posterior probabilities, so a trained classifier is needed. I decided on using CNN for two reasons: 1) it has not been explored in either the DDAL proposing study or any of the surveys. 2) There were preliminary problems which are discussed later. The PyTorch library is used for the creation of classifiers.

The architecture used is presented in Table 6 and Table 5 for CIFAR10 [8] and MNIST respectively. Both models have been trained with the Adam optimizer [1], a learning rate of 0.0009, a batch size of 32, for up to 30 epochs with early stopping.

3.4 Methodology

Given the use of 2 different datasets, 2 different CDD techniques, 2 different CD types, with 2 different scenarios. CIFAR10 and MNIST are used, DDAL and IKS are CDD techniques of choice, abrupt and gradual CD types are simulated, and withhold and rotation drift scenarios are used.

The experiments can be divided into three sections: Fine-tuning DDAL hyper-parameters, evaluating accuracy of the CDD techniques, and measuring computational needs of those algorithms.

Due to the lack of my intuition as to what are reasonable parameters, I run a series of runs across the whole hyperparameter space, i.e. from 0 to 1, with steps of 0.05, with additional finer steps within presumably optimal range, deduced from the initial random selection of hyper-parameter settings. Several batch sizes are tested too, withing range of 32 to 320, with each step being a multiple of 32. I run tests on each of the CDD cases, for both datasets, with an additional tests without the drift. Given EC setting, false positives can be detrimental, without providing increasing sensitivity.

Having established what are reasonable parameters for the DDAL, a fair accuracy testing can be performed. Importantly, given 2 distinct CD scenarios, a separate evaluation approach is needed. In the abrupt setting, the drift should be detected as soon as it happens, and only once, as other detection would be false-positives. As to the gradual case, I design the drift that it begins to take place roughly at the middle batch of testing data, and steadily a new concept is introduced up to when it fully replaces the training one. Here I wish to see it highlight drift multiple times, since in the real application one cannot know to what degree the drift will take place.

λ	θ	Batch Size	Drift Started At	Drift Detected At
0.975	0.850	160	31	31
0.900	0.900	96	52	52

Table 2: Best hyper parameters for abrupt withhold class MNIST

#	Dataset	Drift Case	Drift Started At	Drift Detected
0	cifar	abrupt_w-0	31.0	[3, 16, 28, 44, 61]
1	cifar	gradual_w-0	35.0	[3]
2	cifar	rotate_abrupt	31.0	[3, 16, 29, 39, 49, 59]
3	cifar	rotate_gradual	35.0	[3, 15, 27, 41, 51, 59]
4	mnist	abrupt_w-0	156.0	[140]
5	mnist	clean_test	NaN	[149]
6	mnist	gradual_w-0	160.0	[]
7	mnist	rotate_abrupt	156.0	[118]
8	mnist	rotate_gradual	160.0	[]

Table 3: DDAL: Drift Detection Results Across Different Datasets and Experiment Setups W-0 is a withold of Class 0 scenario, rotate is a case with rotated images, and clean had no drift at all

Regarding computational needs, finally the experiments are run on an Raspberry Pi device, during of which CPU and memory data is being collected every second.

4 Results

Fine-tuning results I analyse in the following manner, first I look at the runs without the drift. For the MNIST, it achieves no detections, which should be the case, for any tested batch size, for lambida's and theta's both being between 0.825 and 0.975. Notably, best runs did not detect any drift. For the CIFAR10 the results are less positive, In have found no pair of hyper-parameters that had no detections for the drift-less runs. The fewest number of false negatives was 4. Importantly, the best runs occur for similar values of theta and lambida, but only for the largest batch sizes of 288 and 320.

Secondly, I evaluate the specificity and sensitivity, with the values within those ranges, but with a drift introduced. There were two cases with a perfect detection for the MNIST abrupt withhold case, with the following parameters (Table: 2)

CIFAR10 runs with a drift performed highly regularly, yet disappointingly. Regardless of a whether a run had gradual or an abrupt drift, they all detected a drift at similar batch numbers, regardless of the hyper-parameters. I have not had many runs with small values of theta and lambida, but I would not expect them to perform much better given the experience with MNSIT. No run detects a drift at batch where it takes place for the abrupt, and I do not put faith in the true positive detections in the gradual scenarios. The only suggestion I can draw is that the best runs take place with a larger batch sizes.

Given all of the above, I settled on using the same set of hyper-parameters for both datasets and cases: $\lambda = 0.975$, $\theta = 0.850$, and a batch size of 160.

DDAL performance (Table: 3) is varied and quite different from what I have expected given the fine tuning. However, it arguably trumps IKS's performance (Table: 4).

Regarding computational needs, I experience that DDAL runs faster and drains less memory. However, I believe it is largely to unoptimised testing scenario, where many files of varied size,

#	Dataset	Drift Case	Drift Started At	Drift Detected
0	cifar	abrupt_w-0	156.0	[]
1	cifar	gradual_w-0	160.0	[]
2	cifar	rotate_abrupt	156.0	[]
3	cifar	rotate_gradual	160.0	[]
4	mnist	abrupt_w-0	156.0	[]
5	mnist	clean_test	NaN	[]
6	mnist	gradual_w-0	160.0	[]
7	mnist	rotate_abrupt	5000.0	[]
8	mnist	rotate_gradual	160.0	[]

Table 4: IKS: Drift Detection Results Across Different Datasets and Experiment Setups W-0 is a withold of Class 0 scenario, rotate is a case with rotated images, and clean had no drift at all

are saved, and IKS's setup makes it save more information. Quick and trivial optimisations, as introduction of batching resulted in significant improvements. Thus, I will come back to it and present more fair results during the defense.

5 Discussion

My immediate reflection on the accuracy results is that a more through fine-tuning could. It appears that there exist no set of hyper-parameters that satisfies all the experimental scenarios, leading to a vastly divergent results among setups. Furthermore, a more detailed analysis of the results would be helpful. We can see, e.g. that DDAL (table: 3) no. 5 detected drift in a clean setting, i.e. without any drift, although it had achieved to avoid so in some case in the fine-tuning. It might argue in favour of the DDAL's instability in detection, where in each run we could observe different results.

Nonetheless, it is infinitely better than IKS. It failed to detect anything at any time-point. I have followed what has been described by [26], double checked it, yet it still failed to show any signs of viability in a CV task.

The past decade has witnessed a significant surge in scientific interest in machine learning (ML), with the absolute volume of published papers doubling to 483,000 in 2021 from just under 200,000 in 2010 (Our World in Data). Despite that, there are many challenges with CD.

Preliminary experiments with a decision tree have shown that it is a crucial step. The model was overfitted to the training space, which resulted in frequent high posterior probabilities (¿ 0.9) for each test sample. Similar situations might have taken place, as observed baseline accuracies have been much lower than the ones I observed during development. Initially, I was able to achieve accuracy scores of over 80%, with a variance due to different hyperparameter choices and shuffling, on both datasets. However, in the end, I was reading partial accuracy scores of anything between 13% to 60%. It is an indication that either experimental data got corrupted, or artifacts had some issues. The accuracy score I mention here is a simple equal-weighted average since the datasets' class distributions were balanced.

There was a very limited amount of CDD techniques investigated. There are dozens of approaches, and using only 2 is not representative of them all. Furthermore, the lack of labels can be handled in semi-supervised ways, such as weak learning [30].

Evaluating CDD requires more thoroughness than is considered in this paper. Firstly, there is a lack of a fair comparison. Some of the CDD in DDAL is carried out in C language, as large parts of PyTorch are written in it, whereas IKS-bdd is pure Python. Secondly, if a different machine was used for the experiments, it might have changed the results.

A substantial part of the CD field is drift adaptation, which has been entirely omitted in this paper. Some approaches, especially ensemble-based [19], provide interesting and promising solutions to both detection and adaptation. The use of ensemble also indicates that investigation from a model selection perspective can bring benefits in an EC setting.

The complexity and diversity of CD pose many problems when trying to analyze in EC. Few studies alongside conflicting nomenclature and terminology make this process more difficult than it would have been with consistent naming conventions.

Given the brought-up research, it is hard to evaluate whether the addition of a more stringent approach to categorizing CDD regarding mathematical formulations of what drift they deal with. However, it would mitigate the aforementioned aspect of inconsistent namings.

The choice of batch size was mostly arbitrary. [2] has shown that its choice has a significant impact on DDAL's detection accuracy. Possibly, it is a trade-off between over-sensitivity and under-sensitivity, as batch size changes.

6 Conclusion

Rising interest in ML over years has brought many insights, including in regard to CD and CDD. Researchers have put effort into a creation of a systematical terminology and reviews, so to provide better overview and cope with term profeliation. Nonetheless, there are terminological issues, limited scope in CV and EC, and unreliable intuition what can be considered a drift. CDD techniques show promise for the low-resolution CV tasks, but are far from perfect. DDAL is dependent on the ML model it uses, a way in which it is trained, whereas IKS fails to the fullest degree. Regarding EC viability, it needs a further investigation.

References

[1] *Adam — PyTorch 2.4 documentation*. URL: https://pytorch.org/docs/stable/generated/torch.optim.Adam.html (visited on 08/15/2024).

[2] Manuel Baena-García et al. "Early Drift Detection Method". In: (Jan. 1, 2006).

[3] Roberto Souto Maior Barros and Silas Garrido T. Carvalho Santos. "A large-scale comparison of concept drift detectors". In: *Information Sciences* 451-452 (July 1, 2018), pp. 348–370. ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.04.014. URL: https://www.sciencedirect.com/science/article/pii/S0020025518302743 (visited on 05/13/2024).

[4] Roberto Souto Maior de Barros and Silas Garrido T. de Carvalho Santos. "An overview and comprehensive comparison of ensembles for concept drift". In: *Information Fusion* 52 (Dec. 1, 2019), pp. 213–244. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.03.006. URL: https://www.sciencedirect.com/science/article/pii/S1566253518308066 (visited on 05/13/2024).

[5] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. *From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors*. Mar. 21, 2022. arXiv: 2203.11070[cs]. URL: http://arxiv.org/abs/2203.11070 (visited on 05/10/2024).

[6] Rodolfo Cavalcante, Leandro Minku, and Adriano Oliveira. "FEDD: Feature Extraction for Explicit Concept Drift Detection in time series". In: July 1, 2016, pp. 740–747. DOI: 10.1109/IJCNN.2016.7727274.

[7] Gong Cheng, Peicheng Zhou, and Junwei Han. "Learning Rotation-Invariant Convolutional Neural Networks for Object Detection in VHR Optical Remote Sensing Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.12 (Dec. 2016), pp. 7405–7415. ISSN: 0196-2892, 1558-0644. DOI: 10.1109/TGRS.2016.2601622. URL: http://ieeexplore.ieee.org/document/7560644/ (visited on 05/15/2024).

[8] *CIFAR-10: Image Classification CNN (89%)*. URL: https://kaggle.com/code/sachinpatil1280/cifar-10-image-classification-cnn-89 (visited on 08/15/2024).

[9] Albert França Josuá Costa, Régis Ant\&nio Saraiva Albuquerque, and Eulanda Miranda dos Santos. "A Drift Detection Method Based on Active Learning". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018 International Joint Conference on Neural Networks (IJCNN). ISSN: 2161-4407. July 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489364. URL: https://ieeexplore.ieee.org/document/8489364 (visited on 05/15/2024).

[10] Isvani Frías-Blanco et al. "Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds". In: *IEEE Transactions on Knowledge and Data Engineering* 27.3 (Mar. 2015). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 810–823. ISSN: 1558-2191. DOI: 10.1109/TKDE.2014.2345382. URL: https://ieeexplore.ieee.org/abstract/document/6871418 (visited on 05/13/2024).

[11] Rosana Noronha Gemaque et al. "An overview of unsupervised drift detection methods". In: *WIREs Data Mining and Knowledge Discovery* 10.6 (2020), e1381. ISSN: 1942-4795. DOI: 10.1002/widm.1381. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1381 (visited on 02/21/2024).

[12] Ananda Mohon Ghosh and Katarina Grolinger. "Edge-Cloud Computing for Internet of Things Data Analytics: Embedding Intelligence in the Edge With Deep Learning". In: *IEEE Transactions on Industrial Informatics* 17.3 (Mar. 2021). Conference Name: IEEE Transactions on Industrial Informatics, pp. 2191–2200. ISSN: 1941-0050. DOI: 10.1109/TII.2020.3008711. URL: https://ieeexplore.ieee.org/abstract/document/9139356 (visited on 05/13/2024).

[13] Vinicius P. M. Goncalves et al. "Concept drift adaptation in video surveillance: a systematic review". In: *Multimedia Tools and Applications* 83.4 (Jan. 1, 2024), pp. 9997–10037. ISSN: 1573-7721. DOI: 10.1007/s11042-023-15855-3. URL: https://doi.org/10.1007/s11042-023-15855-3 (visited on 05/13/2024).

[14] Haochen Hua et al. "Edge Computing with Artificial Intelligence: A Machine Learning Perspective". In: *ACM Computing Surveys* 55.9 (2023), 184:1–184:35. ISSN: 0360-0300. DOI: 10.1145/3555802. URL: https://dl.acm.org/doi/10.1145/3555802 (visited on 05/12/2024).

[15] Adriana Sayuri Iwashita and João Paulo Papa. "An Overview on Concept Drift Learning". In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 1532–1547. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2886026. URL: https://ieeexplore.ieee.org/document/8571222 (visited on 05/13/2024).

[16] W. Jeffrey Johnston and Stefano Fusi. "Abstract representations emerge naturally in neural networks trained to perform multiple tasks". In: *Nature Communications* 14.1 (Feb. 23, 2023). Publisher: Nature Publishing Group, p. 1040. ISSN: 2041-1723. DOI: 10.1038/s41467-023-36583-0. URL: https://www.nature.com/articles/s41467-023-36583-0 (visited on 05/10/2024).

[17] Aymen Rayane Khouas et al. *Training Machine Learning models at the Edge: A Survey*. Mar. 13, 2024. DOI: 10.48550/arXiv.2403.02619. arXiv: 2403.02619[cs]. URL: http://arxiv.org/abs/2403.02619 (visited on 05/12/2024).

[18] Ali Kore et al. "Empirical data drift detection experiments on real-world medical imaging data". In: *Nature Communications* 15.1 (Feb. 29, 2024). Publisher: Nature Publishing Group, p. 1887. ISSN: 2041-1723. DOI: 10.1038/s41467-024-46142-w. URL: https://www.nature.com/articles/s41467-024-46142-w (visited on 05/13/2024).

[19] Bartosz Krawczyk et al. "Ensemble learning for data stream analysis: A survey". In: *Information Fusion* 37 (Sept. 1, 2017), pp. 132–156. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2017.02.004. URL: https://www.sciencedirect.com/science/article/pii/S1566253516302329 (visited on 05/13/2024).

[20] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. "Machine Learning Operations (MLOps): Overview, Definition, and Architecture". In: *IEEE Access* 11 (2023). Conference Name: IEEE Access, pp. 31866–31879. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3262138. URL: https://ieeexplore.ieee.org/abstract/document/10081336 (visited on 05/10/2024).

[21] Jie Lu et al. "Learning under Concept Drift: A Review". In: *IEEE Transactions on Knowledge and Data Engineering* (2018), pp. 1–1. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2018.2876857. URL: https://ieeexplore.ieee.org/document/8496795/ (visited on 05/13/2024).

[22] Gary Marcus. *Deep Learning: A Critical Appraisal*. Jan. 2, 2018. DOI: 10.48550/arXiv.1801.00631. arXiv: 1801.00631[cs,stat]. URL: http://arxiv.org/abs/1801.00631 (visited on 05/10/2024).

[23] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. "Edge Machine Learning for AI-Enabled IoT Devices: A Review". In: *Sensors* 20.9 (Jan. 2020). Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, p. 2533. ISSN: 1424-8220. DOI: 10.3390/s20092533. URL: https://www.mdpi.com/1424-8220/20/9/2533 (visited on 05/12/2024).

[24] M. G. Sarwar Murshed et al. "Machine Learning at the Network Edge: A Survey". In: *ACM Computing Surveys* 54.8 (2021), 170:1–170:37. ISSN: 0360-0300. DOI: 10.1145/3469029. URL: https://doi.org/10.1145/3469029 (visited on 05/12/2024).

[25] Luis Oala et al. *Data Models for Dataset Drift Controls in Machine Learning With Optical Images*. May 7, 2023. DOI: 10.48550/arXiv.2211.02578. arXiv: 2211.02578[cs]. URL: http://arxiv.org/abs/2211.02578 (visited on 05/13/2024).

[26] Denis Moreira dos Reis et al. "Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1545–1554. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939836. URL: https://doi.org/10.1145/2939672.2939836 (visited on 05/14/2024).

[27] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (July 6, 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (visited on 05/15/2024).

[28] Arjun Soin et al. *CheXstray: Real-time Multi-Modal Data Concordance for Drift Detection in Medical Imaging AI*. version: 2. Mar. 17, 2022. arXiv: 2202.02833[cs,eess]. URL: http://arxiv.org/abs/2202.02833 (visited on 05/13/2024).

[29] Abhijit Suprem et al. *ODIN: Automated Drift Detection and Recovery in Video Analytics*. Sept. 9, 2020. DOI: 10.48550/arXiv.2009.05440. arXiv: 2009.05440[cs,eess]. URL: http://arxiv.org/abs/2009.05440 (visited on 05/13/2024).

[30] Liyuan Wang et al. *ORDisCo: Effective and Efficient Usage of Incremental Unlabeled Data for Semi-supervised Continual Learning*. Apr. 8, 2021. arXiv: 2101.00407[cs,stat]. URL: http://arxiv.org/abs/2101.00407 (visited on 01/28/2024).

[31] Gerhard Widmer and Miroslav Kubat. "Learning in the presence of concept drift and hidden contexts". In: *Machine Learning* 23.1 (Apr. 1, 1996), pp. 69–101. ISSN: 1573-0565. DOI: 10.1007/BF00116900. URL: https://doi.org/10.1007/BF00116900 (visited on 05/07/2024).

[32] Zachary Young and Robert Steele. "Empirical evaluation of performance degradation of machine learning-based predictive models – A case study in healthcare information systems". In: *International Journal of Information Management Data Insights* 2.1 (Apr. 1, 2022), p. 100070. ISSN: 2667-0968. DOI: 10.1016/j.jjimei.2022.100070. URL: https://www.sciencedirect.com/science/article/pii/S2667096822000143 (visited on 05/10/2024).

Layer	**Description**
Convolution (Input)	1 channel, Output: 10 channels, Kernel Size: 5x5, Activation: ReLU, Pooling: Max Pooling (2x2)
Convolution	Input: 10 channels, Output: 20 channels, Kernel Size: 5x5, Activation: ReLU, Pooling: Max Pooling (2x2) with Dropout (before activation)
Flatten	–
Fully Connected	Input: 320 features, Output: 50 features, Activation: ReLU
Dropout	Training only
Fully Connected	Input: 50 features, Output: 10 features, Activation: Softmax

Table 5: MNIST CNN Architecture

Layer	**Description**
Convolution (Input)	3 channels, Output: 32 channels, Kernel Size: 3x3, Padding: 1, Activation: ReLU
Pooling	Max Pooling (2x2, stride 2)
Convolution	Input: 32 channels, Output: 64 channels, Kernel Size: 3x3, Padding: 1, Activation: ReLU
Pooling	Max Pooling (2x2, stride 2)
Convolution	Input: 64 channels, Output: 128 channels, Kernel Size: 3x3, Padding: 1, Activation: ReLU
Pooling	Max Pooling (2x2, stride 2)
Flatten	–
Dropout	Probability: 0.25
Fully Connected	Input: 128 features, Output: 128 features, Activation: ReLU
Dropout	Probability: 0.5
Fully Connected	Input: 128 features, Output: 10 features, Activation: Softmax

Table 6: CIFAR10 CNN Architecture

7 Appendix

7.0.1 CNN architectures

Tables 6 and 5 contain information on models structures.

7.1 Source code

Main project code is avaiable at: GitHub. DDAL implementation: GitHub. IKS-bdd implementation: GitHub.