Elastic CatBoost Uncertainty (eCBU)

Elastic gradient boosting decision trees under limited labels by sequential epistemic uncertainty quantification.

Thesis report

by

Erik Sennema

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on Wednesday, September 13, 2023 at 13:30.

Thesis committee:	
Chair:	Dr. M.T.J. Spaan
Supervisors:	Dr. A. Lukina
	Dr. Y. Zhauniarovich
External examiner:	Dr. D.M.J. Tax
External supervisor:	Dr. E. Barbaro
Place:	Department of Computer Science, Faculty EEMCS, Delft
Project Duration:	3th of November, 2022 - 13th of September, 2023.
Student number:	4496582
Master track:	Artificial Intelligence Technology

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Department of Computer Science · Delft University of Technology

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction

Intrusion detection systems (IDSs) are critical components of modern cybersecurity that aim to identify and prevent unauthorised access to computer systems, networks, and other digital systems. A key challenge in developing effective IDSs is the ability to accurately identify anomalous behaviours and attacks, which can be difficult due to the high variability and complexity of the data. To address this challenge, machine learning (ML) techniques have been widely adopted in IDSs [1], with gradient boosting decision trees (GBDTs) being one of the promising techniques in this domain [2]. GBDTs are ensemble methods that combine weak learners (typically decision trees) into a strong learner by iteratively fitting the error of the previous learners. GBDTs have been shown to achieve high performance and robustness on tabular data [3]. However, these models face some difficulties when applied to dynamic real-world environments.

One significant challenge of intrusion detection in dynamic real-world environments, is the ever-evolving nature of threats. Attackers are constantly developing new tactics and techniques to evade detection, making it difficult for intrusion detection systems to keep up [4]. Another challenge in intrusion detection is the high rate of false positives, which can lead to alert fatigue and make it difficult for security teams to prioritise and respond to real threats. While false negatives can have a severe impact on the digital infrastructure one tries to protect, causing real-life consequences, such as personal data theft or fraudulent transactions. In response, IDS must be regularly updated and improved to stay ahead of evolving threats. Moreover, intrusion detection datasets frequently exhibit significant class imbalance due to challenges in obtaining adequate ground truth data on attack instances [4]. As a consequence, a minority class is present, which is of particular interest in the IDS. Although there has been research on imbalanced datasets for several decades, Das et al.[5] contend that it remains a complex challenge in the present era.

To address these challenges, techniques have been developed to adapt GBDTs to dynamic or non-stationary environments. The current state-of-the-art technique is the Adaptive Iterations [6] (AdIter) method. However, this technique assumes that the labels of data samples are available immediately after prediction. This assumption is unrealistic in the intrusion detection setting and many other domains, where obtaining labels is time-consuming and costly. A human expert, often referred to as the oracle in ML, has to determine the final true label of the sample. Hence, typical real-world ML tools have to operate under limited label availability.

In this thesis, we propose a novel elastic gradient boosting decision tree algorithm that addresses these limitations by using a novel uncertainty estimation method that gives an indication of the severity of the drift and copes with limited label availability. We evaluate our method on synthetic and real-world datasets and compare it with state-of-the-art methods. We show that our method achieves comparable accuracy and higher robustness than the existing methods in under limited label availability.

1.2 Motivation

GBDTs have demonstrated suitability for use in industrial applications due to their state-of-the-art performance in making point predictions on structured and tabular data [3]. These models exhibit

robustness to heterogeneous features [7] and have been shown to be competitive in terms of prediction speed and training time, often outperforming other methodologies [8, 9]. As a result, GBDTs present a valuable area of research for further investigation.

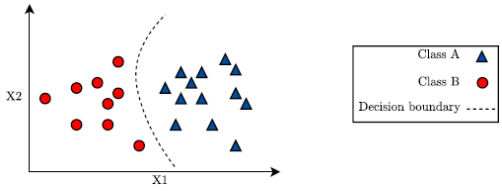
Machine learning models have evolved from static analytics tools to dynamic systems that can process incoming data on demand in real-world applications. However, this also poses a challenge of dealing with the variation of the prediction tasks due to the changing data over time. An open-world setting is considered to be a dynamic or non-stationary environment where an unseen sample can be supplied to the input of the model, an instance that is unlabeled and is not in the distribution of the training samples seen [10]. These non-stationary environments increase the risk of misclassification, particularly in critical safety domains such as automotive, healthcare, and cybersecurity. Deployed models in an open-world setting under real-world data can experience a changing distribution of the data over time, causing a decrease in accuracy [11].

In the context of intrusion detection, the adversarial objective of avoiding detection further exacerbates the non-stationary nature of the environment. The changes in the environment or distribution can be called the dataset shift [12, 13]. There are multiple types of these dataset shifts, including covariate, prior probability and *concept shifts*. The general definition for data shift, as Moreno-Torres et al. [13] define it, is *"Dataset shift appears when training and test joint distributions are different"*.

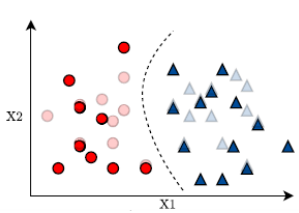
Dataset shift and *concept drift* have similar definitions in the literature, to distinguish these two, this work uses the distinction mentioned in [14]. Shift is considered to be related to batch learning where all data is in memory. While drift corresponds to the stream learning scenario, where data has a sequential order and is not necessarily fully stored in memory nor processed all at once.

In this work, we focus on concept drift as we are interested in the stream learning scenario. Stream learning is a more realistic scenario when considering real-world applications and machine learning used in business processes, such as intrusion detection systems, because with these real-world applications new data arrives every minute and needs to be processed quickly to enable fast and effective decision-making [15].

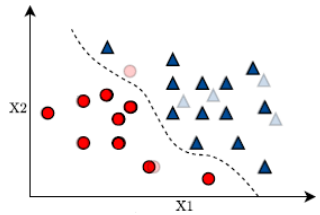
Concept drift is a phenomenon that occurs in non-stationary environments, where the change or drift leads to a decrease in the performance of the model [14]. An example of types of drifts can be seen in Figure 1.1c. Here we can see that some types of drifts do not result in a change of the optimal decision boundary and therefore no decrease in the performance is encountered Figure 1.1b, this is sometimes also called virtual drift. While other types do decrease performance Figure 1.1c, also called real concept drift.



(a) Normal data



(b) No change in optimal decision boundary



(c) A change in the optimal decision boundary

Figure 1.1: Types of drifts. Source: [14]

As concept drifts occur over time, generally a distinction is made between various types based on the speed of the change. Two often mentioned types in literature are abrupt (sudden) and incremental concept drift, depicted in Figures 1.2. Let us say we indicate the two concepts that might characterise the distribution as C1 and C2. With this, we can describe the types of speed of change in the following ways.

(a) Abrupt drift (b) Incremental drift

Figure 1.2: Different types of drifts that can affect the data stream. Source: [16]

- Abrupt drift is a sudden change in the underlying concept of the data, as illustrated in Figure 1.2a. Initially, the data follows the distribution of concept C1. At a certain point in time, the concept shifts abruptly to C2, and the data follows the new distribution thereafter.
- Incremental drift is a gradual change in the underlying concept of the data, as illustrated in Figure 1.2b. Initially, the data follows the distribution of concept C1. At a certain point in time, the concept begins to change slowly towards C2, and the data reflects this transition. During this period, the concept is a mixture of the distributions of C1 and C2.

Challenges Adaptation is crucial for preserving the predictive accuracy and reliability of models in dynamic environments. A trivial method to adapt models to concept drifts over time is to retrain the model periodically, which can be costly and inefficient for large models and datasets. Additionally, this approach is only robust if the retraining frequency is high enough, as the occurrence of concept drift is unknown. On the other side, this can result in unnecessary retraining. To limit unnecessary retraining, a reactive method is required, where in response to detecting a real concept drift the model should be adapted. The challenge here is to detect and respond to the drift promptly, where the timely detection of the drift depends on the speed of change and the appropriate response to the change on the optimal decision boundary. Another challenge indicated by [6] is adaption on different severities of drift, as different severities require a suitable adjustment of the model.

Adaptation of gradient boosting models To prevent unnecessary retaining alternative methods have been proposed that update the existing model instead of replacing it. Updating an existing model poses the challenge of balancing *stability* and *plasticity*. Stability refers to preserving relevant and recurrent knowledge, while plasticity refers to discarding outdated knowledge and incorporating new experience [14].

Most ensemble learning methods for non-stationary environments use passive adaptation techniques [17]. These techniques assume that concept drift can happen at any time and continuously learn from the input [18]. They balance stability and plasticity by dynamically adding and removing weak learners or assigning weights based on their performance. Gradient boosting algorithms are powerful techniques for static data analysis of both regression and classification problems, but they are harder to adapt in the same way as other ensemble techniques, due to their sequential architecture. In gradient boosting algorithms, the trees are dependent on each other, so adapting individual weak learners (trees) would affect the subsequent trees. This necessitates a gradient-boosting specific adaptation technique that can cope with different types and severities of concept drift. Overestimating the concept drift severity can result in discarding useful knowledge, while underestimating it can prevent the model from adapting quickly to new concepts, resulting in poor stability and plasticity, respectively.

Limited Labels Several methods exist for detecting concept drifts, but adaptation techniques for gradient boosting models are scarce [15] and limited by the assumption of immediate true label availability. This assumption is unrealistic [19] and impractical in many real-world scenarios, where obtaining true labels is difficult due to various factors, such as limited labelling resources, high labelling costs, or the inherent delay in label acquisition. This practical limitation of data streams is commonly referred to as *verification latency* [20]. Current adaptation techniques for GBDTs are therefore unsuitable for real-time streaming applications, where decisions need to be made in near real-time without waiting for all labels to become

available. Note that selection bias can induce concept drift itself as well. As Moreno-Torres et al. [13] explain, shifts can be caused by sample selection bias and by non-stationary environments. This implies that the effect of selecting samples to be labelled should be considered when assessing concept drift based on these samples.

In this work, we use the general definition of verification latency to define the *limited labels* in a specific and precise manner, as follows: Given a data stream, we consider this stream in batches of data arriving sequentially. In addition, we have a labelling budget that is expressed as the number of samples that can be labelled before the arrival of the next batch of samples. Then the labelling budget limits the number of labels available to adjust the model for any given batch. Therefore, the number of available labels is given by the number of samples of this batch that have a verification latency shorter than the arrival time of the next batch. Furthermore, we assume a constant budget over time.

The combination of challenges including, the timely detection and proper reaction based on the speed and severity of the drift, the stability-plasticity dilemma [21, 14], which states the balance between retaining and replacing knowledge in the model, and the limited availability of labels directly after the prediction time, indicate the progress that can be made. Hence, there is a need for adaptation techniques that can handle missing labels and enable gradient boosting models to cope with concept drifts in a timely and efficient manner.

1.3 Problem Statement

1.3.1 Informal problem formulation

To illustrate the challenge of adapting to concept drift under limited labels, we present the following scenario. We review the state-of-the-art passive adaptation method for gradient boosting, Adaptive Iterations [6]. This method employs the Elastic Gradient Boosting Decision Tree (eGBDT) [22] technique to determine the extent of the adaptation based on the change in data. It estimates the drift magnitude according to the loss of the available labels.

Assume we have a stream of data from which samples arrive sequentially and we collect them in a batch until a predefined size, after which we initiate the collection of the next (second) batch. Suppose an eGBDT model is trained on the first batch, resulting in a model with τ trees as illustrated in Figure 1.3a. Then the model predicts on the samples in the second batch as soon as the collection is completed. After this step, we aim to adjust the model based on the most recent data, namely the data in the second batch. The eGBDT does this by first reducing the size of the tree sequence in the model to a subsequence, which we call *pruning* the model. This pruned model is depicted in Figure 1.3b.

To decide how to prune the model, the method uses the loss of each subsequence. As all the labels of the second batch become available after prediction, the method can compute the loss on this batch for each subsequence of trees. Using Figure 1.3a as an example, all the subsequences start with the first weak learner h_1^a and may end with any other weak learner. The subsequence of h_k^a consists of all weak learners between and including h_1^a and h_k^a. The loss, also known as the residuals, of this subsequence is then obtained by the root mean squared error of the predictions on the batch of data compared to labels. The best performing subsequence with the minimal loss among all other subsequences determines the pruning point. By this method, the model is reduced to this best performing subsequence. In the example, the best subsequence ends at h_k^a and is therefore reduced to this subsequence shown in Figure 1.3b. By using this loss the size of the pruning is related to the magnitude of the drift in data.

After this pruning step, the model is either adjusted by appending new trees or fully retrained, depending on the number of remaining trees and a user-defined threshold. Appending new trees is called *learning continuation* and the result is shown in Figure 1.3c. Retraining the model is done based on only this new (second) batch of data, resulting in a completely new tree sequence depicted in Figure 1.3d.

This method, however, presupposes that the ground truth labels of each new batch are immediately accessible after prediction, which contradicts our setting of limited labels. With a restricted labelling budget, we can only approximate the drift magnitude with a subset of the ground truth labels for each batch. Suppose we can acquire only 15% of the ground truth labels for each batch before the arrival of the next batch. This may result in over-estimation or under-estimation of the drift magnitude.

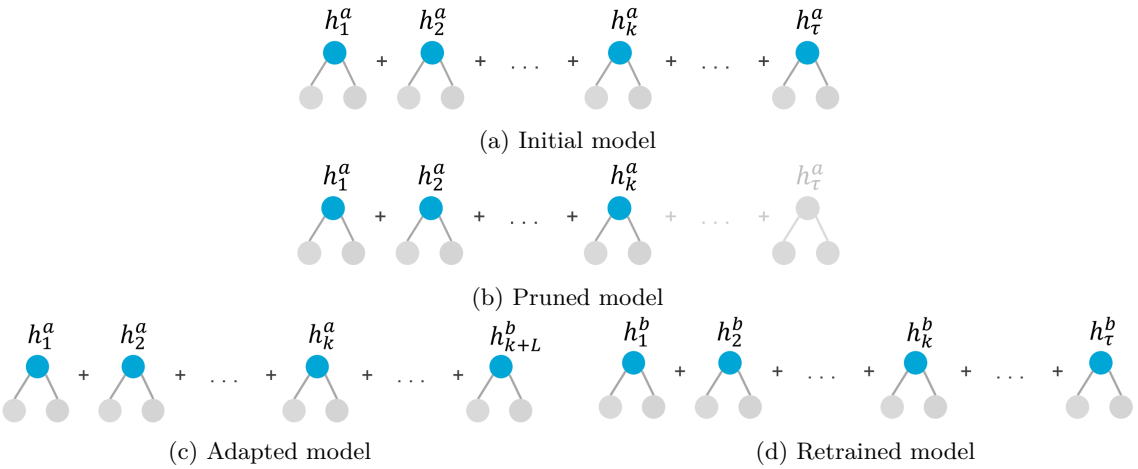


Figure 1.3: Adaptation cycle of the eGBDT model.

In the scenario of over-estimation of the drift, the number of pruned trees based on this subset exceeds the optimal number. In Figure 1.3b, the resulting tree sequence will be shorter than the optimal pruning at learner h_k^a. This implies that the model is actually discarding knowledge that is relevant, and this adversely affects the stability of the model. Moreover, if the resulting tree sequence is below the retraining threshold, the model will retrain, which is redundant. This redundant retraining can impair the performance of the model, as one assumes that the drift is sufficiently large that the previous knowledge is no longer relevant, while this may not be true, resulting in omitting useful information that may cause the model to underperform. An additional consequence of redundant retraining in real-world scenarios is the cost, the cost of labelling all the required samples and tuning the model.

Conversely, with the underestimation of the drift, the number of pruned trees based on this subset falls short of the optimal number. This underestimation of the drift severity may hinder the model's ability to adapt to concept changes. In this scenario, the model's plasticity decreases and it retains irrelevant or conflicting knowledge and may not retrain when actually necessary. Another drawback of under-estimation is the ever-increasing length of the tree sequence. As this increases the inference time.

To conclude, while the current state-of-the-art can adapt GBDT's to changing data over time to stabilise its performance and obviate periodic retraining, it may fail where only a limited amount of labels are available for each batch. Where the erroneous estimation of the required adaptation of the model may entail redundant retraining or inadequate adaptation of the model, potentially decreasing performance.

1.3.2 Formal problem formulation

Non-stationary environments Formally we define the problem of a non-stationary environment as follows, inspired by Bayram et al. [14]. Given an initial training dataset $D_{tr} = (x_i, y_i)_1^N$ for $N \geq 1$ where $x_i = \{x_i^1, ..., x_i^k\}$ is an arbitrary input vector of k features and $y_i \in \mathbb{R}$ is the target value. Where X_i in D_{tr} are independent and identically distributed random variables (i.i.d.). We define the initial stationary distribution $P_{tr}(x, y)$ as the joint distribution of the input and the target at time step $t = 0$, on our dataset D_{tr}.

For the data stream in this non-stationary environment, we have a potentially unbounded sequence of discrete data samples. Each sample is associated with a time stamp arriving in sequential order and the stream is defined as a sequence of $< s_i, s_{i+1}, \ldots, s_n, \ldots >$, where each element $s_j = (x_j, y_j)$ is a new sample. Note that at arrival y is unknown. This sequence of samples can be considered as the *test* set, D_{tst}. The concept drift in this environment considers the changes in statistical distributions of the data over time. Let us define $P_t(x, y)$ as the joint distribution of the input and the target at time step t, from which sequence element s_t is a sample and $P_{t+w}(x, y)$ as the joint distribution after the w time window has passed. Concept drift then occurs when $P_t(x, y) \neq P_{t+w}(x, y)$.

Adaptation To adjust the model to this changing data distribution $P_t(x, y) \rightarrow P_{t+w}(x, y)$, we consider an adaptation technique. As the state-of-the-art adaptation technique of the gradient boosting models, we review the elastic approach [22].

Let us say we train a gradient boosting decision tree on dataset $D_{tr} = \{s_i, \ldots, s_t\}$, with distribution $P_{tr}(x, y)$, resulting in a model of τ trees,

$$F_\tau(x) = \sum_{i=1}^{\tau} h_i(x) + \bar{y} \tag{1.1}$$

where $\bar{y}$ is the mean of the labels for D_{tr} and the initial prediction of the model. $h_i(x)$ represents the i tree in the sequence. We will explain the training of these models in more detail in Section 3.2, where Equation 3.1 describes the same function.

After training the model a batch of samples of the stream comes in, defined as $D_{new} = \{s_{t+1}, \ldots, s_j\}$, with $P_{t+1}(x, y)$. First, the model makes predictions on these samples, resulting in $\hat{y}$. Following the prediction, the elastic algorithm expects to receive the true labels y from an oracle, to adjust the model.

To determine how much the model needs to be adjusted to keep the performance on the same level. The algorithm aims to find the best performing (sub-)sequence of trees for the new data batch. The best performing (sub-)sequence with the minimal error is selected in the following manner,

$$I_{elastic} = \underset{\tau \in \mathbb{Z}^+_{\tau \leq T}}{\arg\min}\, MA(R_\tau) \tag{1.2}$$

where $I_{elastic}$ is the GBDT tree index, T is the maximum number of trees, MA is the mean absolute function and $R_\tau = \{r_{i\tau}\}_{i=1}^{D_{new}}$ are the 'pseudo-residuals' for all trees in the model for each sample in the new batch. Given by,

$$R_\tau = y - F_\tau(x) \tag{1.3}$$

where y is the true label.

The resulting model after pruning based on the new batch D_{new} is then $F_{I_{elastic}}(x)$. After which new trees are added or the model is retrained as explained in Section 1.3.1.

Limited labels When considering the case of *limited label availability*, we do not assume that all labels of a batch will be available at the adaptation time of the algorithm. We define this limited label availability in terms of verification latency and labelling budget (capacity).

Considering verification latency, the true labels are not available or only for a limited number of samples with an arbitrary delay. In this work, we consider a limited number of labels of samples of a batch to be available determined by a homogenous labelling budget over time. Given an arbitrary batch $D_b = \{s_i, \ldots, s_{i+w}\}$, with $|D_b| = N_b$ and a user-defined labelling budget β for time step i until $i + w$. Where we define t_i^x as the timestamp where x_i of sample s_i arrives and t_i^y as the timestamp of the arrival of the true label y_i by the oracle [23]. With

$$t_j^x < t_k^x \text{ for } j < k \in \mathcal{N}_{>0} \wedge t_i^x < t_i^y \text{ for } i \in \mathcal{N}_{>0}$$

and where the verification latency l_{s_i} of a sample s_i, is defined by $l_{s_t} = t_i^y - t_i^x$. Then for n samples of D_b we have,

$$\{l_{s_i} < t_{i+w+1}^x - t_i^x\}_i^n,\ n \leq \beta \leq N_b \tag{1.4}$$

Which simplifies to

$$\{t_i^y < t_{i+w+1}^x\}_i^n,\ n \leq \beta \leq N_b \tag{1.5}$$

Since we are only concerned with whether the delay is smaller or larger than $t_{i+w+1}^x - t_i^x$, the exact delay is irrelevant for the setting. Therefore, we use the term *limited labels*, as this captures the essence of the setting.

To determine $I_{elastic}$, the true labels of all the samples in D_{new} are required, indicating the difficulty of adaptation in the case of limited label availability. When drastically reducing the number of available true labels, the estimation of $I_{elastic}$ may not be accurate, resulting in over-estimation or under-estimation of the drift severity. This may lead to the underperformance of the model after adaptation and therefore this demonstrates the need for an algorithm that is more appropriate for a setting with limited label availability.

Setting and assumptions

We address the problem of gradient boosting adaptation under the following setting and assumptions: (1) the verification latency is finite and bounded, meaning that some labels will be available after a certain delay; (2) the oracle provides accurate labels that match the true class of the samples; (3) the concept drifts are either abrupt, incremental or gradual, and they occur over different time intervals; (4) no label noise, concept evolution or deletion are present in the data; (5) the initial training samples are independent and identically distributed (i.i.d.); (6) a stationary data set is available for the initial training of the model; and (7) we do not consider specific techniques for active learning in this research, as we assume random sample selection will be sufficient to evaluation our novel method.

1.4 Research questions

The problem formulation addresses the challenges of gradient boosting models in non-stationary environments. In real-world applications, the true labels are often unavailable directly after the time of prediction, which exacerbates the difficulties of adapting to concept drifts. Where concept drifts can affect the prediction performance of the model. The current state-of-the-art gradient boosting adaptation technique cannot handle limited labels. This leaves a gap in the research and, as such, the main purpose of this research can be described by the following main and sub-questions:

- *How to reliably adapt GBDTs under limited labels?*

A few relevant sub-research questions that can support this main research question are;

RQ1 How can the stability and plasticity of the model be balanced over time when adapting to concept drift under limited labels?

RQ2 How can the severity of the concept drift be accurately determined under limited labels?

RQ3 Which concept drift scenarios are the most challenging for adaptation under limited labels?

1.5 Contributions

The main contributions of this thesis are as follows:

- We propose a novel elastic gradient boosting decision tree algorithm, elastic CatBoost with Uncertainty (eCBU), that employs a novel sequential uncertainty estimation method to cope with concept drift under limited labels. This algorithm utilises a proxy of the error for the pruning by applying sequential uncertainty estimation. As this proxy relies on more data when label availability is limited, compared to the state-of-the-art methods, it enhances the stability of the pruning of the model. Making it less susceptible to the selection of the limited labels.
- We conduct extensive experiments on synthetic and real-world datasets and compare our method with state-of-the-art methods. We demonstrate that our method attains comparable accuracy and superior robustness to the existing methods under limited labels. This showcases that our novel method is applicable in real-world settings, including intrusion detection. For this intrusion detection scenario, we apply our method on a widely used public dataset and report the results on a proprietary dataset.

1.6 Outline

The report is organised as follows. Chapter 2 reviews the related work on tree-based algorithms, drift detection, model adaptation, and uncertainty quantification for data stream learning. Chapter 3 introduces the preliminaries, including the notation, terminology, and background knowledge. Chapter 4 presents the proposed method for elastic GBDT with uncertainty-based drift adaptation, and explains its rationale, intuition, and algorithmic details. Chapter 5 describes the evaluation framework, including experimental setup, evaluation metrics, baseline methods, and the datasets. Chapter 6 reports the empirical results of the experiments, highlights the key characteristics of the graphs and interprets the results in relation to the research questions, and hypotheses. Chapter 7 discusses the results and the method based on its strengths, weaknesses, implications, and limitations. Chapter 8 concludes the report and provides recommendations and suggestions for future research directions.

2

Related Work

2.1 Drift Detection

To reduce the problem of the decreasing performance of a model under non-stationary environments, we need to be able to detect these dataset drifts. In Open-world Machine leaning (OWML) [10] literature, this is often referred to as Out-of-distribution (OOD) detection [24]. In other works, Novelty, Anomaly and Outlier detection are argued to be similar to Out-of-distribution detection [25]. These techniques are often considered on single samples while drift detection is more often considered in the context of multiple samples at once.

Pimentel et al. [26] refer to the problem of novelty detection as "Novelty detection can be defined as the task of recognising that test data differ in some respect from the data that are available during training." In this review, multiple categories of novelty detection techniques are presented that aim to detect this scenario for various Machine Learning (ML) techniques. Some of the approaches declared in this work are probabilistic, distance-based, reconstruction-based, domain-based and information-theoretic novelty detection.

A related topic, which is often used as a synonym to novelty detection, is anomaly detection [26]. Multiple definitions of both can be found in the work by Xia et al. [27]. Different to novelty detection, anomalies are abnormal patterns in the data that are not a new type of class or input but are rather outliers [28]. Detected anomalies are therefore sometimes not incorporated into the model after detection. However, techniques used to detect novelties will often be similar to the ones used to detect anomalies. The most frequently mentioned application in literature for anomaly detection is intrusion detection [29].

To give an indication of how this detection is done we will elaborate on some of the most widely used detection techniques for concept drift. More drift detection methods can be found here [30]

2.1.1 Detection Techniques

Drift Detection Method (DDM) [31] aims to detect changes in the data distribution to identify the concept drift. The method tracks the error rate of the model as new data arrive. It raises a warning and saves the current model if the error rate increases beyond a threshold. When the error rate increases further the method switches to the saved model, using confidence intervals to determine the thresholds. With these error rates, it effectively detects the concept that may occur.

ADaptive WINdowing (ADWIN) [32], similar to DDM this technique uses the error rate as well to detect a difference in data distribution. This method uses windows over the data to detect differences. It assumes that the data within the window are stationary and divides the window into two sub-windows. The method performs a statistical test to compare the sub-windows and raises a detection warning if they differ significantly.

Uncertainty Drift Detection (UDD) [17] is an unsupervised method that uses uncertainty as a metric for detecting drift. This method is interesting because it correlates with our work. It uses the ADWIN algorithm but instead of the error rate, it uses uncertainty as the metric for the windows.

However, to be able to learn under the concept drift and keep the performance of the model at the same level after concept drift has been detected, we need to understand and adapt as well.

2.1.2 Understanding Concept drift

As explained by [33] , learning under concept drift consists of three parts, namely detection, understanding and adaptation. Under detection we understand detection when a concept drift has occurred, as explained in the concept drift section this can be done in many different ways. In the understanding part of the progress falls determining the severity and the regions of the concept drift. By gaining a better understanding of the concept drift the adaptation on the specific concept drift can be improved, resulting in better performance. At last, the adaptation involves the progress of adapting the model. Depending on the type and characteristics of the concept drift the model can be tuned or retrained. Both of these have their advantages and disadvantages. Additionally, for some models tuning is inherently difficult due to the architecture of the model.

2.2 Model adaptation

Model adaptation under concept drift is the task of updating machine learning models to cope with changes in the data distribution over time, which may affect the accuracy and performance of the models. This model adaptation is sometimes also referred to as, learning with concept drift [15, 33]. Model adaptation under concept drift can be achieved by various strategies, such as:

- Updating the model with new data, by retraining the model from scratch.
- Using ensemble methods, where the models are selected, weighted, or updated based on their relevance to the current data.
- Employing online learning algorithms, where the model is trained on streaming data and adapts to concept drift by using global optimization techniques, such as continuation or shrinkage.

The literature contains various active and passive adaptation techniques for concept drift [18], but the application of gradient boosting to cope with concept drift is less developed [15]. A few methods that use GBDTs and adapt to concept drift are discussed here.

Ensemble approaches Ensemble methods for handling concept drift can be classified into two types: active ensemble with drift detection and adaptation, and passive ensemble with forgetting strategy. The active type depends on a mechanism that detects the drift and triggers the model update accordingly. Passive handling methods however assume concept drift will occur at some moment and continuously add new data when it arrives.

Streaming Gradient Boosting algorithm (SGM) [34] adapts gradient boosting to the online setting, where the data arrives in batches and the weak learners are updated with new data. SGM also uses global optimization techniques known as continuation, which constrain and relax the difference between the learned and the behavior policies, to escape local optima and reduce the error in policy evaluation. However, while this method learns from new batches continuously it does not specifically adapt to concept drift that may occur.

OnlineBoosting [35] utilises the Adaptive Boosting (AdaBoost) [36] model in an online version capable of handling data stream learning. It adds ADWIN as a detector to deal with concept drift. However, these online learning techniques generally have lower performance than regular GBDTs. Making this method a less good option compared to our selected method.

2.3 Uncertainty Quantification

Uncertainty quantification (UQ) is the scientific process of quantifying, characterizing, and propagating uncertainties in both computational and real world systems. It aims to assess the likelihood of various outcomes under incomplete or imprecise knowledge of the system parameters or inputs. For instance, predicting the outcome of a coin flip requires accounting for the uncertainty in the coin's physical properties and initial conditions, which may influence the probability of obtaining heads or tails. UQ seeks to model and estimate these uncertainties and provide probabilistic predictions.

The objective of UQ with machine learning models is to infer predictive distributions that capture the uncertainty in the model parameters or outputs, instead of point estimates that ignore it. These uncertainties can be beneficial for decision-making, risk assessment, or model calibration. Learning predictive distributions with exact Bayesian predictors can be computationally prohibitive and inefficient for complex problems. Therefore, several approximation techniques have been developed to achieve similar goals with lower computational costs. Some additional approaches can be found here [37, 38] .

Gaussian Processes This technique [39, 40] is a non-parametric Bayesian method that defines a prior distribution over functions, rather than over parameters or outputs. The prior distribution is specified by a kernel function that encodes the similarity or correlation between any two points in the input space. Gaussian Processes can fit the observations exactly and provide posterior predictions with uncertainty estimates. They are adaptable and expressive, as various kernels can be chosen or inferred from data.

Monte Carlo Dropout This technique [41, 42] is an approximation technique that uses dropout as a Bayesian inference tool for neural networks. It randomly deactivates neurons in a neural network during both training and testing phases. Each dropout configuration represents a different sample from the approximate parametric posterior distribution over the network parameters or outputs. Monte Carlo Dropout allows stochastic predictions that can be regarded as samples from a probabilistic distribution.

Ensemble Distribution Distillation This technique [43, 44] is a compression technique that transfers the distribution of the predictions from an ensemble of neural networks, instead of only the mean prediction, into a single network. Ensemble Distribution Distillation allows a single network to preserve both the enhanced classification performance of ensemble distillation and the information about the variability of the ensemble, which is beneficial for uncertainty estimation.

UQ has many applications in various domains and tasks, such as computer vision, natural language processing, reinforcement learning, healthcare, robotics, finance and cybersecurity. Some examples of UQ applications are:

- Anomaly detection: UQ can help identify inputs that are likely to be misclassified by a classifier, such as ambiguous or noisy data. UQ can also help detect out-of-distribution inputs that are outside the scope or domain of the classifier, such as adversarial examples or novel classes.
- Active learning: UQ can help select informative inputs for labelling or querying in an active learning setting, where data is scarce or expensive to obtain. UQ can measure the uncertainty or informativeness of each input and prioritise the ones that are expected to reduce the uncertainty or increase the accuracy of the model.
- Exploration-exploitation trade-off: UQ can help balance exploration and exploitation in reinforcement learning, where an agent needs to learn from its interactions with an environment. UQ can quantify the uncertainty or novelty of each state-action pair and guide the agent to explore new or uncertain regions while exploiting known or rewarding regions.

2.3.1 Uncertainty Quantification for GBMs

For GBMs, uncertainty quantification is still an emerging field and only a few approaches have been suggested. Models that naturally provide uncertainty estimates over the outputs are Bayesian methods, but exact Bayesian approaches are often intractable. However, there are some interesting methods that can provide UQ in GBMs.

Natural Gradient Boosting (NGBoost) [45] uses a multiparameter boosting approach to model its conditional distribution, making it able to do probabilistic regression. In essence, each parameter of the output distribution has its own sequence of trees, the natural gradient optimises each set of sequential trees for these parameters together. This approach makes it possible to model any type of distribution, while in practice the time complexity of the prediction increases as the required number of parameters increases (typically two parameters are needed). However, this approach of modelling the probabilistic regression only captures the aleatoric uncertainty. Additionally, compared to other state-of-the-art GBMs[46] (CatBoost, etc.), NGBoost tends to underperform for point-predictions [47].

CatBoost with uncertainty For the sake of completeness, CatBoost with Uncertainty (CBU)[7], utilises an ensemble of GBMs to determine the epistemic uncertainty by the 'disagreement' between the multiple models. As the computation of this approach may be costly, they suggest using the ensemble nature of GBMs to estimate the uncertainty. The virtual ensembles of CBU use "truncated" sub-models of a single GBM model as elements of an ensemble to estimate the epistemic uncertainty. This framework can be applied to any GBM, however, it is limited in the quality of the uncertainty.

PGBM Probabilistic Gradient Boosting Machine (PGBM) [48] aims at determining the probabilistic regression that reflects the uncertainty in the model, by considering the leaf weights in the trees as random variables. PGBM is able to model different sets of posterior distributions while using only one GBM model. However, it does not examine the difference between aleatoric and epistemic uncertainty. Making it difficult to detect anomaly samples with confidence. While tree-based models are often picked for their interpretability, this approach reduces of ease of explainability by making the weights random variables.

Tree Flow More recently, a different approach has been introduced: Tree Flow [49]. It combines a GBDT with a conditional variant of normalizing flow to create the capability of modelling different distributions for the uncertainty than only a Gaussian. However, this approach can be challenging to tune and the shallow feature extractor based on an NN in combination with the two other elements makes it hard to explain the output in terms of the input.

IBUG Instance-Based Uncertainty estimation for Gradient-boosted regression trees (IBUG) [47], can be applied on any GBM. It uses a similarity measure between samples based on the number of the same occurrences samples have in leaves throughout the ensemble (sequence of trees). Nonetheless, this technique is slow at making predictions compared to other techniques (CatBoost, NGBoost), which can be a real problem in safety-critical environments. More importantly, it relies on the tuning of three hyperparameters in order to function properly, especially the 'k' parameter can take time to tune. Additionally, this approach only focuses on the predictive uncertainty and not specifically on epistemic uncertainty.

KGB The Kernel Gradient Boosting (KGB) [50] method proposes that by making certain assumptions, a GBM can be transformed into a kernel-based method that converges to a Gaussian Process' posterior mean. This enables the GBM to function as a posterior sampler, and by using Monte-Carlo estimation, one can estimate the variance and uncertainty of a prediction. Ustimenko et al. assume proper random and oblivious trees to establish the theoretical basis for KGB, and they demonstrate empirically that it outperforms SGB and SGLB. They mentioned that the SGB is bound because the limiting distribution concentrates on the minimum of RMSE obtained via the Gradient Flow-like dynamics [51]. Similarly, the SGLB cannot converge faster than the Euler-Maryama method. The KGB method is not limited by these factors and therefore has better performance. However, computing the kernel requires combining all possible tree structures, which is infeasible. To overcome this limitation, the authors use an ensemble of KGB models, similar to CBU, to estimate uncertainty.

cSGLB Cyclical Stochastic Gradient Langevin Boosting (cSGLB) [52] improves the virtual ensemble of CBU. It does this inspired on [53], this method uses a cyclical exploration and sampling phase to decrease the dependencies between the virtual members of the ensemble, depicted in Figure 2.1. This method shows comparable performance with an ensemble of around 5 SGLB models. By only using one model, the computational load is reduced drastically. However, to increase the independencies between the virtual

ensemble members longer sequences are needed. This suggests that the cSGLB is more prone to overfitting than regular SGLB and may therefore underperform in certain scenarios.

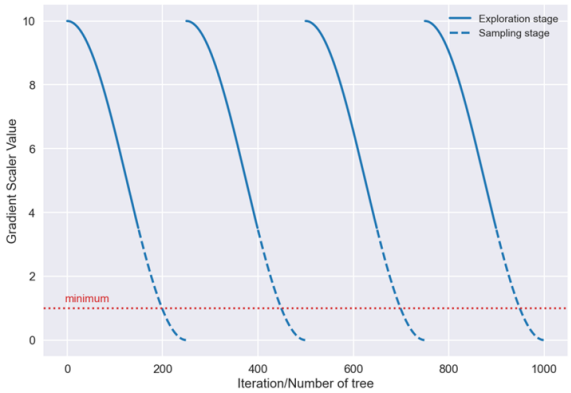


Figure 2.1: Illustration of cyclical schedule of the cSGBL method. Source: [52]

3

Preliminaries

In this chapter, we provide an overview of the technical background especially relevant to our novel method that aims to answer our research question. We first introduce gradient boosting, a machine learning technique that builds an ensemble of weak learners, usually decision trees, to form a strong learner that can achieve high accuracy for both regression and classification problems. We then discuss the challenges posed by non-stationary environments, where the data distribution or the underlying concept may change over time. This phenomenon, known as concept drift, can impair the performance of gradient-boosting models and necessitate frequent retraining or adaptation. After this, we revisit existing methods of model adaptation of gradient boosting models which are the building blocks of our approach. The methods aim at continuously finding the best sequence of weak learners on changing concepts over time. Finally, we show the chosen method of uncertainty quantification in gradient boosting models, which is utilised to provide our novel sequential uncertainty estimation. Uncertainty may arise from various sources, such as parameter estimation, input variability, model inadequacy, and numerical errors. The uncertainty quantification field aims to characterise and estimate the uncertainties and provide probabilistic information about the outcomes and the confidence of the predictions.

3.1 Tree-based algorithms

Tree-based algorithms use decision trees as their structural foundation. In a decision tree, each node is a decision on a certain feature in the dataset and each terminal node is associated with a class. A sample traverses through the tree based on the decision nodes and the feature values of the sample, reaching a terminal node that determines the class of the sample. For regression problems, these classes are scalar values.

To have a model that can predict an output for a new instance the model needs to be fit to the available data, this is done during training. While training the decision nodes for each consecutive split are determined by splitting criteria. The goal of a decision node is to increase the homogeneity of the split sets of samples. Often referred to as the information gain. To quantify the inhomogeneity or impurity, metrics such as entropy, Gini impurity or Twoing criterion are used [54]. The difference between the entropy of the dataset before the split and the weighted combination of the entropy of the datasets after the split reflects the information gain. These decision trees are used in the machine learning technique applied in this work, namely gradient boosting decision trees.

3.2 Gradient Boosting

Gradient Boosting Decision Trees (GBDT) [55] are a type of ensemble learning method that combines multiple weak prediction models, such as decision trees, to create a strong predictive model. GBDTs optimise the model by iteratively adjusting the weights, an additive manner, of each model based on the errors of the previous iterations, using the gradient descent algorithm to minimise a loss function.

Assume we have a dataset of i.i.d. samples $D_N = (x_i, y_i)_1^N$ for $N \geq 1$ from $D_N \sim \mathcal{D}$ where $\mathcal{D}$ is a training data distribution, $x_i = \{x_i^1, ..., x_i^k\}$ is an arbitrary input vector of k features and $y_i \in \mathbb{R}$ is the

output or target value. Gradient boosting aims to approximate the $F^*(x) : \mathbb{R}^k \to \mathbb{R}$ that maps input instances x to its output y, by minimizing the expected loss $\mathcal{L}(F|\mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\ell(F(x), y)]$. It does this by building a function by the following additive approach,

$$F_\tau(x) = F_{\tau-1}(x) + \rho_\tau h_\tau(x), \tag{3.1}$$

where ρ_τ is the weight of function $h_\tau(x)$. The function $h_\tau : \mathbb{R}^k \to \mathbb{R}$ is a base predictor, in this case, a shallow tree. The space of possible weaker learners is defined by $\mathcal{H} := \{h_s(x, \phi_s) : \mathbb{R}^k \to \mathbb{R}, s \in S\}$, where S is a finite index set and h_s linearly depends on its parameters ϕ_s. The parameters of a model F_τ are denoted by θ_τ. The decision tree recursively partitions the feature space into disjoint regions. Each final region or terminal R_j has an estimated response y value assigned for that region. The tree can be represented in the following way,

$$h(x) = \sum_{j=1}^{J} \phi_j \mathbb{1}(x \in R_j), \tag{3.2}$$

where J is the number of nodes, ϕ_j represents the estimated response values for terminal nodes and decision values for nonterminal nodes. The indicator function $\mathbb{1}(\cdot)$ has a value of 1 or zero depending on the evaluation of the arguments. These base learners are subsequently minimizing the loss for each greedy step by,

$$(\rho_\tau, h_\tau(x)) = \arg\min_{\rho, h \in \mathcal{H}} \sum_{i=1}^{N} \ell(F_{\tau-1}(x_i) + \rho h(x_i), y_i), \tag{3.3}$$

where h_0 is an initial guess and $\{h_\tau\}_1^T$ are successive increments. The consecutive h_τ base learners are minimised on the 'pseudo-residuals' derived from the preceding base learner. The 'pseudo-residuals' are a vital part of the GBDT adaptation algorithm and will be elaborated on in Section 3.4. The intermediate targets are determined by the following gradient-descent,

$$-g_\tau(x_i, y_i) = \left[\frac{\partial \ell(F(x_i), y_i)}{\partial F(x_i)}\right]_{F(x)=F_{\tau-1}(x)} \tag{3.4}$$

A common method to determine the next base learner is by using a least-squares approximation,

$$h_\tau(x) = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} [-g_\tau(x_i, y_i) - h(x_i)]^2 \tag{3.5}$$

Subsequently, a line search is done to optimise ρ_τ. To reduce the chance of overfitting most methods use a learning rate as a shrinkage method [46] $F_\tau(x) = F_{\tau-1}(x) - \epsilon\rho_\tau h_\tau(x)$, where $\epsilon = (0, 1]$. Moreover, most well-known libraries use additional regularisation methods.

This powerful approach has led to the development of several improvements of the algorithm over the past years, including Stochastic Gradient Boosting (SGB) [56], Categorical Boosting (CatBoost) [57] and Stochastic Gradient Langevin Boosting (SGLB) [58]. An elaborate analysis of CatBoost in multiple domains can be found here [59]

Note that in our approach we use Stochastic Gradient Boosting, where this method applies stochastic gradient descent to gradient boosting by using a random subset of the data to fit each weak learner, rather than the whole data. This reduces the correlation between the learners and improves the generalization performance of the ensemble. SGB also introduces randomness in the selection of features or split points for each tree, further increasing the diversity and robustness of the sequence of weak learners. As will be explained in Section 3.5.1, this stochastic plays a vital role in enabling the chosen uncertainty quantification method.

3.3 Concept Drift

3.3.1 Formal definition of Drift

In these non-stationary environments, this work talks about data streams, these streams are potentially unbounded sequences of discrete data samples. Each is associated with a time stamp arriving in sequential

order and is defined as a sequence of $< s_1, s_2, \ldots, s_n, \ldots >$, where each element $s_j = (x_j, y_j)$ is a new sample. This sequence of samples can be considered as the *test* set, D_{tst}. The concept drift in this environment considers the changes in statistical distributions of the data over time. Let us define $P_t(x, y)$ as the joint distribution of the input and the target at time step t, from which sequence element s_t is a sample and $P_{t+w}(x, y)$ as the joint distribution after the w time window has passed. Note that in Bayesian Decision Theory $P_t(x, y) = P_t(y|x) \times P_t(x) = P_t(x|y) \times P_t(y)$ [60]. Concept drift then occurs when

$$P_t(x, y) \neq P_{t+w}(x, y) \tag{3.6}$$

To differentiate between occasional anomalies and a drift, w is considered to be longer than one time-point [61]. Therefore we have $w \in \mathbb{Z}^+ \wedge w > 1$.

Other and more specific types of drift consider different probability distributions as well. This work differentiates between two types of drifts, ones that directly affect the prediction performance and ones that do not, depicted by Figure 1.1c and Figure 1.1b respectively. The first type that affects the prediction performance, is considered to be a change in the posterior probability distribution

$$P_t(y|x) \neq P_{t+w}(y|x) \tag{3.7}$$

While for the second type, scenarios where,

$$P_t(y|x) = P_{t+w}(y|x) \wedge P_t(x) \neq P_{t+w}(x), \tag{3.8}$$

hold, this does not directly influence the prediction performance. One type of drift that is left out of consideration is the prior-probability shift. Defined as,

$$P_t(y) \neq P_{t+w}(y) \tag{3.9}$$

This type of drift can have an effect on the prediction performance, however, these are scenarios where only the prior of the labels shows significant change. The flipping of the complete set of labels and concept-evolution and deletion are left out of consideration. Section 5.3.3 elaborates on why this choice has been made.

A more elaborate taxonomy of the various types of drift defined by the difference in changing probability distribution can be found in the work by Bayram et al. [14]

3.3.2 Change of the Concept

As these events of concept drift happen over time, We consider two types of the *speed of change*, in this work, namely, sudden, incremental concept drift [62]. The types shown in Figures 1.2 can formally be defined by Equation 3.10 and Equation 3.11 for sudden and incremental concept drift respectively. In these equations, the $P_0(x, y)$ and $P_1(x, y)$ distributions represent the concepts C1 and C2 of the figures displaying the concepts.

- Sudden concept drift

$$P_j(x, y) = \begin{cases} P_0(x, y), & \text{if } j < t \\ P_1(x, y), & \text{if } j \geq t \end{cases} \tag{3.10}$$

- Incremental concept drift

$$P_j(x, y) = \begin{cases} P_0(x, y), & \text{if } j < t_1 \\ (1 - \alpha)P_0(x, y) + \alpha_j P_1(x, y), & \text{if } t_1 \leq j < t_2 \\ P_1(x, y), & \text{if } t_2 \leq j \end{cases} \tag{3.11}$$

  where,

$$\alpha_j = \frac{j - t_1}{t_2 - t_1} \tag{3.12}$$

3.4 Model adaptation

The general goal of model adaptation is to learn the drift and ensure that the loss of a learning model is continuously optimised when concept drift occurs. The works by Sun et al. [63] and Wang et al. [22] describe the objective of model adaptation in the following manner. Given a batch of data samples $D = \{s_t, \ldots, s_i\}$, $s_t = (x_t, y_t)$, drawn from the same joint distribution $P_t(x, y)$ at time step t. We aim at minimizing the loss $\ell(F(x), y)$ by continuously finding the best model:

$$F^t = \arg\min_{F \in \mathcal{H}} \mathbb{E}_{(x,y) \in P_t(x,y)}[\ell(F(x), y)], \tag{3.13}$$

where $\mathcal{H}$ is the hypothesis set and $\mathbb{E}(\cdot)$ denotes the expected value. If we consider a sequence of $P_t(x, y)$ over time, where $P_t(x, y) \neq P_{t+w}(x, y)$ may occur in the sequence, the goal of dynamically adapting the model on this sequence is given by,

$$\min_{F^1, \ldots, F^t, \ldots} \sum_t \mathbb{E}_{(x,y) \in P_t(x,y)}[\ell(F^t(x), y)] \tag{3.14}$$

Equation 3.14 shows the selection of a model at each time step t that minimises the loss the most. To cope with different types of the speed of change described in Section 3.3.1 and depicted in Figure 1.2 model adaptation method continuously adapt based on incoming data.

Wang et al. [6] identify the problem of the rising loss affected by concept drift, where the learner is not adjusted to the concept drift $P_t(x, y) \neq P_{t+w}(x, y)$, by the increase of the loss after the change of concept. Let us say we have $\ell^t = \ell^t(F(x), y)$ for time t then the loss for $\ell^{t+w} = \ell^{t+w}(F(x), y)$ will increase, i.e.

$$\ell^t < \ell^{t+w}. \tag{3.15}$$

This problem of the rising loss is later used in a method to determine the local-minimum learner and with it, the severity of the drift, explained in Section 3.4.2.

The goal of adaptation is to reduce the loss to the same level as before the occurrence of drift. If the adjusted model is given by $F'(x)$ then the loss for time $t + w$ will be $\ell^{t+w'}(F'(x), y)$. Then the aim of the adaptation would be:

$$\ell^{t+w'}(F'(x), y) - \ell^t \leq 0 < \ell^{t+w} - \ell^t \tag{3.16}$$

3.4.1 Incremental GBDTs

To adapt the model to new patterns that occur in data streams due to concept drift, GBDTs need to adjust the sequence of trees. Incremental learning is a method of doing this update by adding trees at the end of the sequence, also known as *learning continuation*. Given the same scenario as above where we have batch $D_{init} = \{s_i, \ldots, s_t\}$, $D_{new} = \{s_{t+1}, \ldots, s_j\}$ and a model $F^t_\tau(x)$ trained on D_{init}. Then L new trees are added based on samples $\{(x_i, r_{i(\tau+L))}\}_{i=1}^{D_{new}}$, resulting in, similar to Equation 3.1,the following model,

$$F^{t+1}(x) = F_{\tau+L}(x) = F_{\tau+L-1}(x) + \rho_{\tau+L} h_{\tau+L}(x) \tag{3.17}$$

Note that all the 'pseudo-residuals' for sequence length τ are denoted by $R_\tau = \{r_{i\tau}\}_{i=1}^{D_{new}}$. To incrementally tune the model this procedure is executed repeatedly for each new batch in the data stream. The pseudocode for this method is shown in Algorithm 1. Note that generally, either or both regularization and stochasticity are added by methods mentioned in Section 3.2, therefore Equation 3.17 and Algorithm 1 are simplifications.

3.4.2 eGBDT

Elastic Gradient Boosting Decision Tree (eGBDT) [22] uses the incremental GBDT as an integral part of its algorithm. As a GBDT model makes predictions by adding up the outputs of a series of trees. The method can update the model by adding new trees based on new data or removing the last few trees to go back to a previous state of optimization. The eGBDT method updates the model and detects changes in the data distribution by finding the tree that minimises the error. The goal is the same as described by Equation 3.13. Finding the minimal (pseudo-)residuals to reduce the sequence of trees is described in Section 1.3.2. Where the $I_{elastic}$ is the resulting length of the reduced model, obtained by Equation 1.2.

Algorithm 1: Incremental GBDT

Init: D_{init}, D_{new}
Parameters:
Parameters GBDT, $GBDT_{parm}$
Num of incremental trees, L
Result: Model $F_{T+L}(x)$

build $F_T(x)$ on D_{init} with $GBDT_{parm}$
for $l \leftarrow 1$ **to** L **do**
  $R_{T+l-1} \leftarrow$ *compute pseudo-residuals by* $F_{T+l-1}(x)$ *on* D_{new}
  $h_{T+l} \leftarrow$ *fit* $h_{T+l}(x)$ *on* $\{(x_i, r_{i(T+l-1))}\}_{i=1}^{D_{new}}$
  $F_{T+l}(x) \leftarrow$ *update according to Eq. 3.17*
end
return $F_{T+L}(x)$

The search for the minimal (pseudo-)residuals, can be described by the gradient-drift learner and the local-minimum learner [6]. The gradient-drift learner is given in a similar notion as the rising loss problem, only here we consider the weak learners. Given a model $F_T(x)$ with weak learners $\{h_\tau\}_{\tau=1}^{T}$, where the expected loss for a h_τ is $\mathbb{E}[\ell_\tau] = \mathbb{E}[\ell(h_\tau(x), y)]$, then the gradient-drift learner is described as follows:

$$\mathbb{E}[\ell_\tau^t] \leq \mathbb{E}[\ell_{\tau-1}^t] \wedge \mathbb{E}[\ell_\tau^{t+1}] > \mathbb{E}[\ell_{\tau-1}^{t+1}] \tag{3.18}$$

The local-minimum learner precedes these drift learners and can be found where the expected loss is minimal:

$$h_\tau(x) = \arg\min_\tau \mathbb{E}[\ell(F_\tau(x), y)] \tag{3.19}$$

New trees are added by the incremental GBDT algorithm after pruning the model. Through the combination of pruning and the addition of trees, the model dynamically adapts to drifts that may occur. Algorithm 2 shows the procedure including the call to the Algorithm 1 as a subroutine.

Algorithm 2: Elastic GBDT

Init: Trained GBDT: $F_T(x)$, Data: D_{new}
Parameters: N/A
Result: Adjusted model $F_{T'}(x)$

predict with $F_T(x)$ on D_{new}
$(R_\tau)_{\tau=1}^{T} \leftarrow$ *calculate residuals for all trees on* D_{new} ▷ by Eq. 1.3
$\tau \leftarrow$ *find tree index with minimal mean absolute residual* ▷ by Eq. 1.2
if $\tau < T$ **then**
  retrain GBDT $F_T(x)$ on D_{new}
  return $F_T(x)$ *as* $F_{T'}(x)$
else
  $F_{T'}(x) \leftarrow F_\tau(x)$ ▷ remove redundant trees
  $F_{T''}(x) \leftarrow$ *run subroutine Alg. 1 without the first line on* $F_{T'}(x)$, D_{new}
  return $F_{T''}(x)$ *as* $F_{T'}(x)$
end

From the procedure in Algorithm 2 we can see that if $I_{elastic}$ is smaller than a predefined threshold the model will be retrained on the new batch. If this is the case, the authors say that there is a significant concept drift and retraining is needed. When the drift is less severe the model will be pruned and L number or trees are added by Algorithm 1. The threshold is set equal to the initial number of trees T, by the author of the method, but can be changed.

3.4.3 AdIter

The Adaptive Iterations (AdIter) [6] method is an adjustment on the eGBDT and tries to determine more accurately how many trees, also called iterations, need to be added in the case of tuning the model. The authors mention this problem as finding the appropriate number of trees under concept drift. It does this by having an ensemble of multiple GBMs and adjusting them with different amounts of trees. The models that predict the true labels are considered well-adjusted and, therefore, the number of added trees is considered appropriate. Each model in the ensemble is an elastic GBDT.

To allow the loss resulting from the concept drift to reduce to pre-drift levels, L trees will be added to help tune the model. The problem is how to determine an appropriate number for L. This problem is formulated as,

$$L = \arg\min_{L} \ell(F_{T+L}(x)), \tag{3.20}$$

where $F_{T+L}(x)$ is the model after tuning.

Algorithm 3: Adaptive Iterations GBDT

Init: Stream data: D
Parameters:
Set of Config, $\{eGBDT^{(m)}_{parm}\}^{M}_{m=1}$
Initial training batch size, $batch_{init}$
Sliding batch size, $batch_{slide}$
Initial number of iterations T
Result: Predictions $\{\hat{Y}_{batch_{slide}}\}$

for $eGBDT^{(m)}_{parm}$ *in* $\{eGBDT^{(m)}_{parm}\}^{M}_{m=1}$ **do**
  build eGBDT $F^{(m)}$ *on* $D_{batch_{slide}}$
end
while D *has next batch* $D_{batch_{slide}}$ **do**
  for $m = 1$ **to** M **do**
    $\hat{Y}_m \leftarrow F^{(m)}(D_{batch_{slide}})$ ▷ save prediction vector
    (run Alg. 2 as subroutine)
    $F^{(m)}_{\tau} \leftarrow$ *prune* $F^{(m)}(x)$ *on* $D_{batch_{slide}}$
    if $\tau < T$ **then**
      retrain $F^{(m)}(x)$ *on* $D_{batch_{slide}}$
    else
      tune $F^{(m)}_{\tau}(x)$ *on* $D_{batch_{slide}}$
    end
  end
  $\hat{Y}_{batch_{slide}} \leftarrow$ *majority vote on* $\{\hat{Y}_m\}^{M}_{m=1}$ ▷ by Eq. 3.21
end
return $\{\hat{Y}_{batch_{slide}}\}$ ▷ prediction results on all batches

AdIter aims at handling the problem of determining the best number of L for different degrees of the severity of the concept drift by an ensemble defined as $\{F^{(m)}(x)\}^{M}_{m=1}$ where each eGBDT in the ensemble has a different parameter for L, by $\{L_1, L_2, \ldots, L_m\}^{M}_{m=1}$. The resulting prediction of the ensemble is determined by a majority vote,

$$\hat{Y} = \begin{cases} 0, & \text{if } \sum_i \mathbb{1}(\hat{Y}_i = 0) \geq \sum_i \mathbb{1}(\hat{Y}_i = 1) \\ 1, & \text{else} \end{cases}, \tag{3.21}$$

where $\hat{Y}_i$ is denoted as the prediction by the ith model in the ensemble and $\mathbb{1}(\cdot)$ is the indicator function. Algorithm 3 gives the complete procedure of the AdIter algorithm.

3.5 Uncertainty quantification

In general, uncertainty in models arises when the model is under a shifted dataset, events of anomalies, noise or overlapping classes [41]. Its estimate can be used to indicate the models' confidence on a particular sample and so to detect an anomaly or novelty.

In Bayesian statistics, uncertainty can be modelled with the posterior distribution [40]. If we consider $\mathcal{H}$ to be the hypothesis space of probabilistic predictors, where h is a hypothesis in that space that maps an instance x to a target y, then we can formulate the following posterior distribution, $p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{p(\mathcal{D})}$. Here $\mathcal{D}$ is the data distribution, where $(x, y) \sim \mathcal{D}$. The $p(h)$ is the prior, which is the pre-known knowledge about the modelling. $p(\mathcal{D}|h)$ is the likelihood function of h, meaning how likely is the data if modelled by this hypothesis. However, in practice, we do not know the exact distribution of $\mathcal{D}$ and determining it by the integral over the space $\mathcal{H}$ is intractable. Therefore we seek alternative approaches to approximate the posterior distribution and thereby the uncertainty quantification. We elaborate in more detail on the modelling of uncertainty in Bayesian theory in Appendix A.1.1.

Uncertainty can be split into two parts, *aleatoric* and *epistemic* uncertainty. Aleatoric uncertainty is also known as data uncertainty and is not a property of the model, it's caused by noise or overlapping classes [64, 41]. Due to its origin, this part of the uncertainty is irreducible. Still, it is useful to know this part of the uncertainty especially if safety-critical decisions are based on the output of the model and the data is not perfectly controllable. Epistemic uncertainty is the uncertainty that arises due to the lack of knowledge in the model, it is also known as knowledge uncertainty. This part is reducible, making it possible to improve the model's performance and therefore making it interesting to quantify and determine its origin. However, this part of the uncertainty is significantly harder to determine well.

Specifics of epistemic uncertainty Hüllermeier et al. [40] differentiate between *model uncertainty* and *approximation* uncertainty as parts of epistemic uncertainty. Where *model* uncertainty is about the type of model used, so linear regression model, Gaussian process, GBDT, etc. The other part refers to the uncertainty on the approximation of the best function to model the prediction. In this work we will be looking at the *approximation* uncertainty part and neglect the *model* uncertainty part, as we assume that the model architecture namely, GBDTs, is capable and flexible enough.

3.5.1 Uncertainty Quantification by Ensembles

There are several techniques to determine the uncertainty in predictions made by machine learning models. Some relevant methods are described in Section 2.3. In this work, however, we consider the ensemble method to quantify the uncertainty, specifically a GBDT uncertainty quantification ensemble technique [7]. This approach uses multiple models with different seeds to represent a part of the hypothesis space. The multiple outcomes of the model together form the posterior.

Let us first define how the posterior distribution can be estimated by a Bayesian ensemble. Consider an ensemble of M number of probabilistic models $\{P(y|x;\theta^{(m)})\}_{m=1}^{M}$ sampled from the posterior $p(\theta|\mathcal{D})$, note that the θ is the representation of the hypothesis in the space of probabilistic predictors. In this case a Bayesian learner. The uncertainty can be estimated by the difference in predictions made by the models in the ensemble, as each model $P(y|x;\theta^{(m)})$ produces different predictions. This difference in prediction is a result from the stochasticity introduced by the SGB explained in Section 3.2. These models are trained on in-domain data and therefore yield a particular range of behaviours for this type of data, while it has an 'undefined' behaviour for out-of-distribution data. Due to this combination of stochasticity and 'undefined' behaviour the predictions on out-of-distribution data will be diverse and consistent on in-domain data for an ensemble of probabilistic models [43]. The intuition behind this statement is explained in the introduction of this section. As the exact Bayesian inference is often intractable, methods have been proposed to approximate the posterior.

To approximate the uncertainty by using the probabilistic gradient boosting models we use the following method. Note that in Section 3.2 it was explained that, the final GBDT model $F(x)$ given by Equation 3.1 is a sum of decision trees (weak learners) and the parameters of the full model are denoted by θ.

Let us take an ensemble of GBDT probabilistic predictors $\{P(y|x;\theta^{(m)})\}_{m=1}^{M}$. Malinin et al. [43] state that given the posterior distribution $p(\theta|\mathcal{D})$ the *predictive posterior* of the ensemble can be obtained by

taking the expectation with respect to the models in the ensemble:

$$P(y|x,\mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[P(y|x;\theta)] \approx \frac{1}{M}\sum_{m=1}^{M} P(y|x;\theta^{(m)}), \theta^{(m)} \sim p(\theta|\mathcal{D}) \tag{3.22}$$

According to the information theory [65], the Shannon entropy [66] is a measure of the uncertainty of a discrete probability distribution. It computes the expected amount of information or surprise contained by an event, given all possible outcomes. The entropy of the predictive posterior reflects the total uncertainty in the predictions, as shown by [64, 67]:

$$\mathcal{H}[P(y|x,\mathcal{D})] = \mathbb{E}_{P(y|x,\mathcal{D})}[-lnP(y|x,\mathcal{D})] \tag{3.23}$$

The total predictive uncertainty is obtained in the following manner:

$$\mathcal{U}^{p}(y|x;\theta) = \mathcal{H}[P(y|x,\mathcal{D})] \approx \mathcal{H}[\frac{1}{M}\sum_{m=1}^{M} P(y|x;\theta^{(m)})] \tag{3.24}$$

This uncertainty type contains both the aleatoric and the epistemic uncertainty. Where the aleatoric is derived by the following equation:

$$\mathcal{U}^{a}(y|x;\theta) = \mathbb{E}_{P(\theta,\mathcal{D})}[\mathcal{H}[P(y|x,\theta)]] \approx \frac{1}{M}\sum_{m=1}^{M} \mathcal{H}[P(y|x;\theta^{(m)})] \tag{3.25}$$

The estimate of the epistemic uncertainty can be obtained by subtracting the expected data uncertainty from the total uncertainty estimate, as formulated in Equation 3.26. This is known as the difference of the mutual information between the parameters θ and y.

$$\begin{aligned}\mathcal{U}^{e}(y|x;\theta) = \underbrace{\mathcal{I}[y,\theta|x,\mathcal{D}]}_{\text{Epistemic Uncertainty}} &= \underbrace{\mathcal{H}[P(y|x,\mathcal{D})]}_{\text{Total Uncertainty}} - \underbrace{\mathbb{E}_{P(\theta,\mathcal{D})}[\mathcal{H}[P(y|x,\theta)]]}_{\text{Expected Aleatoric Uncertainty}} \\ &\approx \mathcal{H}[\frac{1}{M}\sum_{m=1}^{M} P(y|x;\theta^{(m)})] - \frac{1}{M}\sum_{m=1}^{M} \mathcal{H}[P(y|x;\theta^{(m)})]\end{aligned} \tag{3.26}$$

This shows how we can approximate the different types of uncertainty by the ensemble technique. Note that for the sake of readability, a somewhat simplified notation $\mathcal{U}(x)$ for $\mathcal{U}^{p}(y|x;\theta)$ is occasionally used. It should be clear from the context on which learner θ the uncertainty is based.

4

Elastic gradient boosting decision trees under limited labels

In this section, we will present the main contributions of this thesis, which is a novel method that aims to adapt better under limited label availability.

4.1 Approach overview

To give an intuition of the novel algorithm we propose, which provides a proxy for the severity of the drift that may occur in data streams in a non-stationary environment, we build on the example of model adaptation explained in the informal problem formulation, Section 1.3.1.

As explained in that section, the current state-of-the-art adaptation method AdIter employs eGBDT as its technique to prune the model. The eGBDT technique aims to dynamically adapt a GBDT-based model to concept drift over time. We can divide the scenario of eGBDT described in the problem formulation into three logical consecutive steps. First, the model is trained on initial data, then the model is pruned based on a new batch, and finally, the model is adapted or retrained based on the same new batch.

With the eGBDT algorithm, this pruning step dynamically adapts to the severity of the concept drift. This part of the algorithm is responsible for the estimation of the severity of drift as well as for maintaining the stability and plasticity of the model under the changing environment. The amount of pruning is based on the loss over all the tree (sub-)sequences of the model, which depends on the batches of data that originate from a data stream. The objective of the pruning is to continuously minimise the loss for each new batch of data.

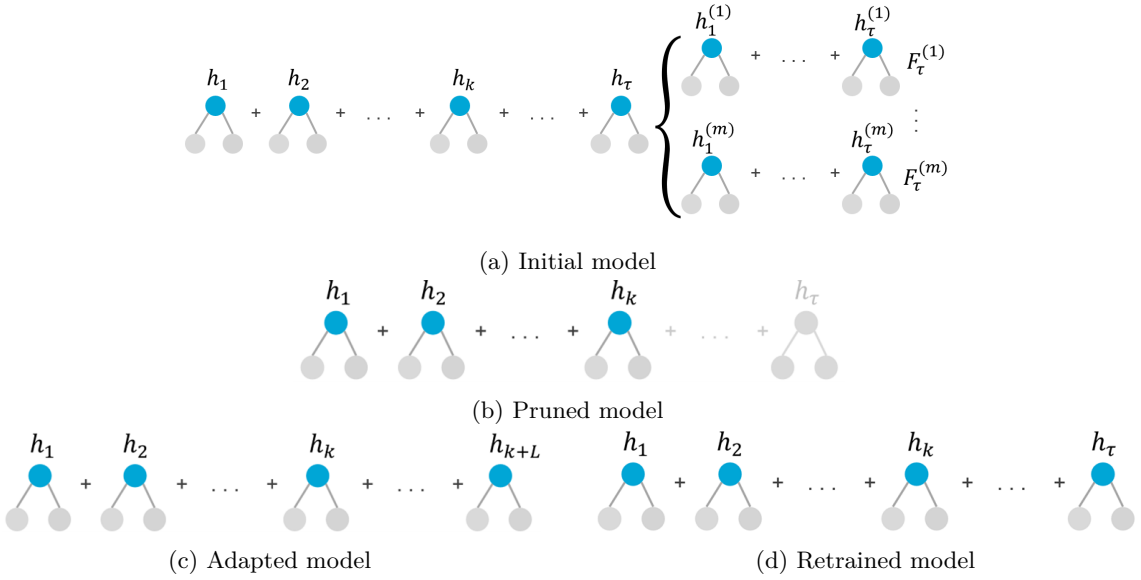
A limitation of this method, however, is the requirement for the true labels of all the samples in a batch. To determine the pruning point, the method requires that all the true labels arrive before the next batch to be able to adapt well. When the number of samples that are available before the next batch arrives is limited, however, this method is prone to over-estimation or under-estimation of the drift severity, resulting in a model that is not pruned correctly and therefore may have disadvantages such as excessive retraining or underperformance.

To make this pruning more accurate given only a limited number of true labels for each batch, we introduce our elastic CatBoost with Uncertainty (eCBU) approach. *This approach uses uncertainty quantification to estimate the pruning that is performed without the use of any labels.* This approach differs in the pruning step compared to the eGBDT approach and resembles the other steps compared to AdIter. As explained in Section 3.4.3, AdIter uses an ensemble of GBDTs to make the final prediction. Our method utilises an ensemble as well.

The same consecutive steps of eGBDT apply to eCBU as well. For the first step of training the initial model, eGBDT uses a single model, while our method with eCBU uses an ensemble of models, as depicted in Figure 4.1a. Each of the GBDT models in the ensemble has the same length.

For the second step, eGBDT evaluates the residuals for all (sub-)sequences and prunes the model to the (sub-)sequence with the minimum loss. With eCBU, we assess the uncertainty for all (sub-)sequences

by the ensemble and prune the model to the (sub-)sequence with minimum uncertainty. We measure the uncertainty of each (sub-)sequence by quantifying the disagreement among predictions by models in the ensemble that have the same length. To estimate this uncertainty, we need a probabilistic predictor, therefore we use a classifier instead of a regressor as used for eGBDT.



(a) Initial model

(b) Pruned model

(c) Adapted model

(d) Retrained model

Figure 4.1: Adaptation cycle of the eCBU model.

For instance, if we want to know the uncertainty for the (sub-)sequence until tree h_k, depicted in Figure 4.1a, with ensemble size M. We predict with each model $\{F_k^{(m)}\}_{m=1}^{M}$. We then measure the uncertainty at point k by the disagreement between the predictions. By doing this for every (sub-)sequence we can determine the minimal learner and thus the pruning point. If the ensemble has the lowest uncertainty at h_k, we prune the ensemble of models from this learner onwards, as illustrated in Figure 4.1b. The single sequence is a simplification of the full ensemble as shown in Figure 4.1a. By using this measure of *sequential uncertainty* we can quantify the best pruning point due to the change of uncertainty given by this new batch of unlabeled data.

For the third step, the model is tuned with a user-defined parameter L or retrained. Where L denotes the number of trees added by learning continuation. Similar to eGBDT and AdIter, if the pruning exceeds the threshold the model is retrained otherwise the learning continuation is applied, as shown in Figure 4.1d and Figure 4.1c, respectively.

4.2 Model Architecture

In this section, we will explain the technical details of our novel approach as well as the design choices and the multiple versions of the algorithm.

The general model architecture of our approach functions similarly to the AdIter method described in Section 3.4.3. The objective of our method is to improve the estimation of the severity of the concept drift under limited label availability and by doing this minimise unnecessary retraining of the model and maximise the predictive performance of the model. However, as explained our novel method is different in some aspects from the AdIter. While eCBU also utilises an ensemble of GBDTs, this method uses this ensemble for a different purpose. AdIter uses the ensemble to facilitate multiple different sizes of L to make the adjustment more flexible. Where eCBU uses the ensemble to estimate the uncertainty for each (sub-)sequence. How this uncertainty estimation relates to our objective of model adaptation and which techniques will be used for this are explained next.

4.2.1 Connecting Uncertainty and Concept Drift

So far, we have provided a comprehensive description of concept drift and uncertainty quantification, but we have not yet directly explored the link between them. Based on several works [68, 67], we have evidence to suggest that there is a correlation between them. Moreover, many other works indicate the relation between Out-of-Distribution (OOD) sample detection and uncertainty [41, 40, 69, 52]. This is a significant addition because samples from a different concept and hence, Equation 3.6, a different distribution, can be regarded as OOD-samples. The work of [67] proposes, by their experiments, that it is beneficial to decompose the uncertainty into an epistemic and an aleatoric component. They show that OOD-samples are characterised by having a high epistemic uncertainty and a low aleatoric uncertainty. A more direct connection is made by [68], who confirm that the model's confidence can be an unsupervised indicator of the presence of drift.

As we assume that somewhere during the arrival of new batches from a data stream the concept will change $P_t(x, y) \neq P_{t+w}(x, y)$, resulting in $P_t(y|x) \neq P_{t+w}(y|x)$ we want to adjust the *predictive posterior* to this new distribution $P_{t+w}(y|x)$. The predictive posterior for the ensemble obtained by Equation 3.22, by taking the entropy of this posterior we can generate an estimation of the uncertainty about the predictions. By minimising the entropy we can find a model that has the most proper predictive posterior given the new batch of data. To estimate this predictive posterior we need probabilistic predictors, in the next section we elaborate on the probabilistic predictors used.

4.2.2 Probabilistic predictor

The eGBDT and AdIter methods are both regression models where all the targets are either 0 or 1. The authors of the methods use this regression prediction and assign them to a particular class based on a threshold. For these methods pruning is done based on the residuals, the difference between the prediction and the label, for each tree in the sequence of trees of the regression model. To estimate the uncertainty for a regression GBDT however, we need to estimate both the mean and the standard-deviation of the posterior distribution [43]. NGBoost solves this for regression by using two sequences, together they model the distribution. However, under continuous learning, this model becomes unstable.

To avoid this instability, we propose eCBU as a classification model that can estimate uncertainty more reliably in continuous learning settings. Unlike the regression case, where we need to use NGBoost to estimate the uncertainty, we can directly obtain the uncertainty from the predicted probabilities of the classification model. For a general GBDT classification model, this probability is obtained by the fraction of training samples of the class in a leaf. Therefore, we do not require NGBoost for classification.

4.2.3 Sequential uncertainty

As mentioned in the related work, Section 2.1, there are methods that are able to detect concept drift in an unsupervised manner. In work [17], where the uncertainty is used to detect concept drift, the detection is triggered by the magnitude of the uncertainty. However, it is not straightforward how to infer the adaptation size of a GBDT model from the magnitude of the change of the uncertainty relative to a reference frame. Therefore, we need a metric that can measure the drift severity in an unsupervised way and directly correlates to the adaptation needed. To measure the severity, we use the same intuition as in eGBDT, where the best-performing (sub-)sequence of trees is given by $I_{elastic}$, similar to Equation 1.2. The resulting pruning size is a proxy for the drift severity, as a larger drift would require less retention of the original sequence of trees (plasticity) and a smaller drift would require more preservation of the prior knowledge in the model (stability).

To address the problem introduced in our problem formulation we introduce *sequential uncertainty*. This sequential uncertainty is the uncertainty for each (sub-)sequence with different lengths in increasing order of an ensemble of GBDT models.

As our method works with classification models we have probabilistic predictors at our disposal. Let us denote our ensemble of probabilistic predictors, GBDT classifiers, by $\{P(y|x; \theta^{(m)})\}_{m=1}^{M}$, where M is the size of the ensemble, similar to Section 3.5. For this complete ensemble of models, $\theta^{(M)}$ represents the parameters of the ensemble $F^{(M)}(x)$, where T are the number of weak learners in the model, denoted by $h_\tau^{(m)}$.

To determine the uncertainty for an arbitrary $\tau \in T^{(M)}$, for ensemble $F^{(M)}(x)$, we estimate the uncertainty in the same manner as Equation 3.24, the parameters for the (sub-)sequences are denoted by $\theta_\tau^{(M)}$. Only here τ can denote any $F_\tau^{(M)}(x)$. We define the approximation of the predictive uncertainty on the predictions by the ensemble for a given τ by:

$$\mathcal{U}^p(y|x;\theta_\tau^{(M)}) \approx \mathcal{H}[\frac{1}{M}\sum_{m=1}^{M} P(y|x;\theta_\tau^{(m)})] \tag{4.1}$$

The approximation of the expected aleatoric uncertainty for a given τ is obtained by:

$$\mathcal{U}^a(y|x;\theta_\tau^{(M)}) \approx \frac{1}{M}\sum_{m=1}^{M} \mathcal{H}[P(y|x;\theta_\tau^{(m)})] \tag{4.2}$$

Finally, the approximation of the epistemic uncertainty for a given τ is obtained by:

$$\mathcal{U}^e(y|x;\theta_\tau^{(M)}) \approx \underbrace{\mathcal{H}[\frac{1}{M}\sum_{m=1}^{M} P(y|x;\theta_\tau^{(m)})]}_{\text{Total Uncertainty}} - \underbrace{\frac{1}{M}\sum_{m=1}^{M} \mathcal{H}[P(y|x;\theta_\tau^{(m)})]}_{\text{Expected Aleatoric Uncertainty}} \tag{4.3}$$

4.3 Unsupervised pruning

In this section, we will elaborate on how we exploit the relationship between uncertainty and concept drift and introduce *rising uncertainty* and the *uncertainty drift learner*.

With this relation between the occurrence of concept drift and the magnitude of the uncertainty, seek to exploit this correlation with our novel method. By looking at the difference between the uncertainty for a given batch of data over the sequence of trees we aim to find the optimal learner. This is the same objective as stated by Equation 3.13, only our pruning approach is unsupervised, making it able to work with the same level of accuracy for any number of true labels at its disposal.

Rising uncertainty Now let us restate the problem of the rising loss, Equation 3.15, to rising uncertainty. Here we use $\mathcal{U}(x)$ as a simplification of $\mathcal{U}^p(y|x;\theta^{(M)})$ for a given ensemble of models $F^{(M)}(x)$, obtained by Equation 3.24. As explained in Section 3.5 the uncertainty is expected to grow for samples that fall out of the distribution. The rising loss is affected by concept drift $P_t(x,y) \neq P_{t+w}(x,y)$, where the model is not adjusted. Our proposed rising uncertainty is inspired by this effect, as the output of a learner is 'undefined' for the distribution $P_{t+w}(x,y)$ the disagreement between learners is likely to increase. The effect of the increase of estimation of the uncertainty when concept drift occurs and its correlation with the loss is empirically shown in [68] and used in various other works as explained in Section 4.2.1. As there is a high statistical coupling between the occurrence of concept drift and the uncertainty in the predictions made, we use this correlation to propose rising uncertainty.

Formally we state, given a ensemble GBDT classifier model $F^{(M)}(x)$, trained on $D \sim P_t(x,y)$ and $\mathcal{U}^t(x)$ for time t and if $P_t(y|x) \neq P_{t+w}(y|x)$ then the uncertainty $\mathcal{U}^{t+w}(x)$ for time $t+w$ will rise with high likelyhood, $\mathcal{U}^t(x) < \mathcal{U}^{t+w}(x)$.

The goal of adaptation is to reduce the uncertainty to the same level as before the occurrence of drift. If the adjusted model is given by $F'^{(M)}(x)$ and the uncertainty for time $t+w$ will be $\mathcal{U}'^{t+w}(x)$. Then the aim of the adaptation would be,

$$\mathcal{U}^{t+w'}(x) - \mathcal{U}^t(x) \leq 0 < \mathcal{U}^{t+w}(x) - \mathcal{U}^t(x) \tag{4.4}$$

Uncertainty-Drift Learner With this rising uncertainty and the high correlation between the uncertainty and the occurrence of concept drift, we seek to explore extending rising uncertainty over the sequence of trees in the ensemble. Similar to the gradient-drift learner described in Section 3.4.2 by Equation 3.18, we formulate an *uncertainty-drift learner*.

The gradient-drift learner is the tree in the GBDT sequence for which the loss is smaller than the preceding trees before the occurrence of concept drift and larger after the occurrence of concept drift. Without concept drift, the loss decreases until the end of the sequence as later trees minimise smaller and smaller losses, making them more specific for the current concept. Based on the loss, the gradient-drift learner (tree) directly succeeds the local-minimum learner Equation 3.19. With our uncertainty-drift learner, we expect that because of the correlation between the loss (error) and the uncertainty [70], an uncertainty-drift learner will also occur under concept drift. Where an uncertainty-drift learner is a learner of an ensemble $F^{(M)}(x)$ for which the uncertainty is lower than its predecessor, before the concept drift and higher after.

Again given an ensemble GBDT classifier models $F^{(M)}(x)$, trained on $D \sim P_t(x, y)$ we can obtain the uncertainty $\mathcal{U}_\tau^t(x)$ for an arbitrary τ, by Equation 4.1. We formulate an Uncertainty-Drift Learner to be the following:

$$\mathbb{E}[\mathcal{U}_\tau^t(x)] \leq \mathbb{E}[\mathcal{U}_{\tau-1}^t(x)] \wedge \mathbb{E}[\mathcal{U}_\tau^{t+1}(x)] > \mathbb{E}[\mathcal{U}_{\tau-1}^{t+1}(x)] \tag{4.5}$$

In Section Section 6.3 we show empirically to which extent this uncertainty-drift learner is present under concept drift.

Local-Minimum Learner Having determined the goal of the adaptation by the rising uncertainty. Our objective becomes finding the minimum learner that precedes the uncertainty-drift learner and reduces the uncertainty back to the original level. Where local-minimal learner is searched by the loss for eGBDT in Eq. 3.19, we use the uncertainty. The following equation gives us the learner (tree),

$$h_\tau^{(M)}(x) = \arg\min_\tau \mathcal{U}(y|x; \theta_\tau^{(M)}) \tag{4.6}$$

Having determined the local-minimum learner, we can prune the model to remove the uncertainty-drift learners succeeding this learner. By pruning the model with this method, we have determined a proxy for the severity of the drift with no labels.

Pruning by uncertainty types Our chosen method of uncertainty quantification is capable of estimating multiple types of uncertainty. Therefore, we can utilise these types to perform the unsupervised pruning. In our explanation of determining the local-minimum learner by uncertainty, we use predictive uncertainty. As this uncertainty encapsulates both the epistemic and aleatoric uncertainty, this is a good proxy for the loss and, therefore, a good metric to determine the pruning.

The aleatoric uncertainty only encapsulates the uncertainty that is related to the data and does not directly consider the knowledge of the probabilistic predictor. Therefore, we decided not to prune based on only the aleatoric uncertainty. As the epistemic uncertainty reflects the absence of knowledge in the model for a given sample, we do use this type to determine our local-minimal learner. Many works [40, 52, 69, 71] emphasise the usefulness of this type of uncertainty when aiming to detect out-of-distribution samples. As the samples originating from a different concept are out-of-distribution samples, we use this type of uncertainty next to predictive uncertainty to determine the local-minimum learner.

4.4 The algorithm

The algorithmic structure of eCBU is very similar to the algorithm of AdIter, Algorithm 3 and is shown in Algorithm 4. The main difference between the algorithms is their use of the ensemble of eGBDT models. Where the AdIter uses the ensemble for their adaptive learning continuation, we use it for our sequential uncertainty estimation.

In Algorithm 4 we first train the ensemble of eGBDTs on the initial training set available, as shown in lines 1 to 3. After this step the algorithm will aim to continuously adjust the model to keep performance on par, this objective is described by Equation 3.14. At line 4 we see that the algorithm runs while new batches come in for our data stream. Line 5 until 7 shows the prediction of the individual models on the current batch of data. These predictions are later used to make the final prediction shown at line 16. Then on line 8, the algorithm determines how much the models in the ensemble should be pruned by Equation 4.6, and prunes the models. Depending on the algorithm settings this can be either done based on the (total) predictive uncertainty or the epistemic uncertainty. This results in the desired length τ of the sequences in the ensemble. At last, from line 9 until 15 the models are then either fully retrained or tuned depending on the threshold, which is set to T. As our method is designed for a setting with limited label availability, the tuning at line 13 is done on the subset of the current batch for which labels are obtained by an oracle limited by the labelling budget. The retraining however needs to be done on the complete batch or more data, otherwise, the model will not be adapted well to the current concept and under perform.

Algorithm 4: Elastic CatBoost Uncertainty

Init: Stream data: D
Parameters:
Set of Config, $\{eGBDT^{(m)}_{parm}\}^{M}_{m=1}$
Initial training batch size, $batch_{init}$
Sliding batch size, $batch_{slide}$
Initial number of iterations T
Result: Predictions $\{\hat{Y}_{batch_{slide}}\}$

1 **for** $eGBDT^{(m)}_{parm}$ *in* $\{eGBDT^{(m)}_{parm}\}^{M}_{m=1}$ **do**
2 *build* $eGBDTF^{(m)}$ *on* $D_{batch_{slide}}$
3 **end**
4 **while** D *has next batch* $D_{batch_{slide}}$ **do**
5 **for** $m = 1$ **to** M **do**
6 $\hat{Y}_m \leftarrow F^{(m)}(D_{batch_{slide}})$ ▷ save prediction vector
7 **end**
8 $F^{(M)}_{\tau}(x) \leftarrow$ *prune* $F^{(M)}(x)$ *on* $D_{batch_{slide}}$ ▷ by Eq. 4.6
9 **for** $m = 1$ **to** M **do**
10 **if** $\tau < T$ **then**
11 *retrain* $eGBDT\ F^{(m)}(x)$ *on* $D_{batch_{slide}}$ ▷ collect all labels by an oracle
12 **else**
13 *tune* $F^{(m)}_{\tau}(x)$ *on* $D'_{batch_{slide}}$ ▷ subset labelled by an oracle, based on budget β
14 **end**
15 **end**
16 $\hat{Y}_{batch_{slide}} \leftarrow$ *majority vote on* $\{\hat{Y}_m\}^{M}_{m=1}$ ▷ by Eq. 3.21
17 **end**
18 **return** $\{\hat{Y}_{batch_{slide}}\}$ ▷ prediction results on all batches

4.4.1 Implementation details

Our implementation of all GBDT models is based on the CatBoost library that is known to achieve state-of-the-art results in a variety of tasks [57]. In our implementation we use the shrinking, learning continuation and uncertainty calculation capabilities, making the codebase for the algorithm light and robust.

4.5 RQ 1: Stability and Plasticity

In this section, we address the following sub-question: *How can the stability and plasticity of the model be balanced over time when adapting to concept drift under limited labels?*

To answer this question, we first revisit the stability-plasticity dilemma that we discussed in the problem formulation. This dilemma refers to the trade-off between preserving valuable knowledge (stability) and adapting to new concepts (plasticity) [21, 14]. This task becomes even more difficult when the true labels are limited or scarce, as the information available for adapting is reduced. Unlike the existing methods eGBDT and AdIter, which use all the true labels and thus have perfect information, our method relaxes this assumption and prunes in an unsupervised manner without requiring any true labels.

To ensure the plasticity of the model, we adopt the same approach as eGBDT and AdIter. As the data arrives in batches of samples, the model is updated every batch. Similar to Equation 3.14, the aim is to continuously find the best model given this stream of batches. For these methods, the only objective for this continuous adaptation is to minimise the loss. However, for our method this objective is twofold: the objective during pruning is to minimise the uncertainty, and during learning continuation, it is to minimise the loss on the available labels. This combination provides the plasticity of the model. Our method has the same level of freedom for the plasticity. The model can be reduced to a sequence length below the retraining threshold, resulting in the retraining of the model and enabling it to adapt to a completely new concept.

However, if the model is retrained on every batch, it would never retain any information and therefore be unstable. Similarly, if it would always apply learning continuation without pruning any trees, it would grow indefinitely, which is also not stable and reduces its plasticity.

By using unsupervised pruning, the model becomes more stable given limited labels, as the pruning is independent of the number of labels available. The advantage here is that the algorithm always uses the information of all samples in a batch, so the uncertainty does not change if the number of limited labels changes. However, to have a stable pruning, our uncertainty estimation also needs to be stable under model adaptation. We empirically demonstrate the robustness of the uncertainty estimation in Section 6.2, by using the robustness metric explained in Section 5.2.5.

4.6 RQ 2: Severity of the drift

In this section, we elaborate on how our novel method addresses the following sub-question:

How can the severity of the concept drift be accurately determined given limited labels?

To be able to reliably adapt the model under concept drift, it is important to be able to accurately determine the severity of the drift. The severity of the drift reflects how much different the new concept is compared to the former concept. If the drift is severe, the model needs to change a lot to be able to perform well on the new concept. If, on the other hand, the drift is not that severe, the model does not need to be changed that much. If the severity is estimated well for all severities of concept drifts, the model can adapt well to the changing environment while keeping unnecessary retraining limited. Therefore, it is important to be able to determine the severity of the drift well.

The current methods, eGBDT and AdIter, use the loss on the true labels available to determine the severity of the drift. As a sequence of trees of a GBDT model additively minimises the loss on a given dataset explained in Section 3.2, the trees at the end of the sequence will be more specific for the concept as later trees minimise a smaller and smaller loss. Therefore, if the algorithms prune more trees from the end of the sequence, the estimation of the drift is said to be more severe. By this analogy, these algorithms determine the severity of the drift by the number of trees pruned. However, as we know from the work of Moreno-Torres et al. [13], concept drifts can be caused by sample selection bias or by non-stationary environments. This implies that the effect of selecting samples to be labelled should be considered when assessing concept drift based on these samples.

As our setting is a more realistic real-world setting where only a limited number of true labels are available during the adaptation step, this selection bias will influence the detection of the drift. When the available labels are very limited, this selection bias will induce the appearance of concept drift based on

the loss metric used in the eGBDT and AdIter methods. To prevent the selection bias from influencing the determination of the severity on the drift, our method uses all samples available in an unsupervised manner. With this approach, the selection of samples does not influence the pruning step of the algorithm and, therefore, can not cause the appearance of concept drift while not present. However, to determine if the pruning by the uncertainty described by Equation 4.6 is accurate, we empirically show the effect of unsupervised pruning in Section 6.3.

5

Evaluation

This chapter presents the evaluation of the proposed method for solving the problem formulated in Section 1.3.1. The proposed method is compared with the original method and other baselines in the context of limited labels. The experiments are based on two types of datasets: synthetic and real-world.

In the synthetic experiments, we apply the methods to several synthetic datasets that simulate different types of drift scenarios. We know the timing and nature of the drifts in these datasets, which allows us to assess how well the methods can detect and adapt to them. To ease the comparison process with the original method, we report similar metrics and graphs. Additional metrics and graphs are used to test the stability and plasticity of the combination of the predictive and uncertainty estimation performance.

In the real-world experiments, we apply the methods to one dataset that was used in the original work and two datasets from the domain of intrusion detection. These datasets have different characteristics and challenges that test the robustness and generalizability of the methods. We use the same metrics and graphs as in the synthetic experiments to compare the methods.

The rest of this chapter is organised as follows: Section 5.1 describes the synthetic and real-world datasets used in the evaluation; Section 5.2 explains the metrics used to measure the performance of the methods; Section 5.3 presents the experimental design and settings.

5.1 Datasets

In this section, we describe the datasets that we use to evaluate our method. We use two types of datasets: synthetic and real-world. Synthetic datasets are generated by different methods that simulate concept drift or its absence. We know the exact nature and timing of the concept drift in these datasets. Real-world datasets are collected from various domains and applications. We do not know the exact concept drift in these datasets, if any. Hence, for these datasets, we focus on the temporal evolution of the model's predictive performance. All datasets involve binary classification tasks.

5.1.1 Synthetic data

The datasets used include the six synthetic datasets used by Wang et al. [6]. These datasets differ in the severity of the drift and the speed of the change of the drift. Further specifics on these datasets can be found in the paper by Wang et al. [6], in the explanations below and in an overview given in Table 5.1. The ratio column in the table specifies the ratio between the two classes.

These datasets are generated by a generator function that can be found in the documentation of scikit-multiflow[1]. However as the data from these generators have some random element, we will be using the datafiles from AdIter, to ease the comparison. These files can be found in their library[2]. The purpose of using these six synthetic datasets is to evaluate the performance of our proposed method on different kinds of concept drift and intensities. As well as to test if the method is applicable to datasets that are free of concept drift.

[1] https://scikit-multiflow.readthedocs.io/en/stable/index.html

[2] https://github.com/kunkun111/AdIter

SEAa This dataset is generated by the SEAGenerator [72] and contains 3 attributes and 2 classes. Each attribute is numeric between 0 and 10. There are 10,000 samples in this dataset which contains 3 abrupt concept drifts. Each batch contains 100 samples and the drift occurs at the 25th, 50th and 75th data batches.

RTG is generated by Random Tree Generator. A decision tree is built where the split nodes are chosen from randomly selected attributes, and the leaves are assigned to different classes. The concepts and classes are derived from a tree structure, which should favour decision tree learners. This dataset does not exhibit any concept drift.

RBF & RBFr The Radial Basis Function (RBF) generator is used to create the RBF and RBFr datasets with 10 features each. The number of centers are varied to create different concept drift situations in the data. RBF has 50 centers and all of them drift slowly with a margin of 0:0001. RBFr has 50 centers too, but only 10 of them drift faster with a margin of 0:01.

HYP is generated by a hyperplane generator [73] , which mimics incremental drift. Hyperplanes can model concepts that change over time, because we can adjust the direction and location of the hyperplane smoothly by changing the weights of the dimensions.

AGRa simulates the data stream with abrupt drift by using the AGRAWAL generator [74] . Similar to the SEAa dataset the concept drift occur on the 25th, 50th and 75th data batches, only more drastically. This dataset contains, 10,000 samples, 6 nominal and 3 continuous attributes, and 2 classes.

Synthetic	Samples	Features	Class	Ratio	Batch size	Drift type
SEAa	10,000	3	2	1:1	100	Abrupt
RTG	10,000	10	2	1:1	100	No
RBF	10,000	10	2	1:1	100	Incremental
RBFr	10,000	10	2	1:1	100	Incremental
HYP	10,000	10	2	1:1	100	Incremental
AGRa	10,000	9	2	1:1	100	Abrupt

Table 5.1: Seven Synthetic Datasets

5.1.2 Real world data

To show the function of the novel method on real-world data, we test the model on three real-world datasets. One of those, namely Electricity, is used by the AdIter paper as well. The other two datasets are related to our case study of intrusion detection. These datasets are real-world datasets, for which we do not know the nature nor the timing of the concept drifts that may occur. An overview of the details of these datasets can be found in Table 5.2. Similar to the table about the synthetic datasets the ratio column refers to the ratio between the two classes in the dataset.

Electricity This dataset is widely used and describes the Australian New South Wales Electricity Market. It has a binary target variable, which represents the direction of the electricity price change over time. The dataset contains 45,312 observations with various features, we initially set the batch size to 100.

NSL-KDD One of the few publicly available data sets for network-based anomaly detection systems is KDDCup99 [75], which contains a mixture of benign and attack traffic in a simulated environment. However, this dataset suffers from a high degree of redundancy among the records [76]. To overcome this limitation, we use the NSL-KDD[3] dataset by [77] instead. This dataset includes 67,343 samples of normal network traffic data and 58,630 attack data samples. These attacks include DOS, U2R, R2L and Probe. To simulate our stream-based scenario with this dataset we consider this training set in batches of size 100.

[3]https://www.unb.ca/cic/datasets/nsl.html

IDS dataset This dataset is a proprietary dataset of IDS alerts from a large organization collected over three months. The dataset comprises alerts from four different security detection systems: SIEM, UEBA, EDR, and IN-HOUSE. The original dataset has 113 alerts corresponding to real attacks, which are augmented using SMOTE. The final dataset contains 1,192 mainly synthetic attack samples and has a 2% attack rate. The stream is based on the last month of this dataset and contains 11 real attacks.

Real-world	Samples	Features	Class	Ratio	Batch size
Electricity	45,312	8	2	0.73:1	100
NSL-KDD [75]	125,973	41	2	0.46:1	100
IDS (private)	55,615	32	2	0.02:1	100

Table 5.2: Three real-world datasets

5.2 Metrics

This section describes the evaluation metrics for the experiment on both synthetic and real-world datasets. The metrics are consistent with those used by the original AdIter, namely Accuracy, F1-score, and Matthew correlation coefficient. We use similar metrics as the original work that introduced AdIter, such as Accuracy, F1-score and the Matthew correlation coefficient. Additionally, Balanced Accuracy is employed and a minor modification is made to adapt to the IDS dataset. Finally, the details of the robustness metric are given, which enables the assessment of both uncertainty quality and predictive performance simultaneously.

5.2.1 Classification

Let's first look and an ordinary confusion matrix for binary classification in Table 5.3. Each cell in the matrix is calculated by summing the predictions that fall in that category, by the following general formula for multi-class classification:

$$K_{ij} = \sum_{k} \mathbb{1}(y_k = i)\mathbb{1}(p_k = j), \tag{5.1}$$

where k is the number of samples on which a prediction are made. y_k is the actual label of the kth sample and p_k is the prediction. Table 5.3 shows how the confusion matrix looks for binary classification directly followed from Equation 5.1.

		Actual Class	
		Positive	Negative
Predicted Class	Positive	K_{11} True Positive (TP)	K_{21} False Positive (FP)
	Negative	K_{12} False Negative (FN)	K_{22} True Negative (TN)

Table 5.3: Confusion matrix for binary classification

Accuracy Accuracy is a common metric for evaluating the performance of a classifier. It is defined as the ratio of correctly classified instances to the total number of instances. Given by,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

5.2.2 F-beta score

The F_β is a popular metric for binary classification and was introduced in [78]. The score calculates the harmonic mean of precision and recall and is non-symmetric meaning one class is selected as the class of interest. The metric is defined by Equation 5.3, where β is set based on the required importance of the precision and recall. For the commonly used F_1 score this β is set to 1, giving 'equal' weight. For higher and lower values of β recall or precision will have more influence on the resulting metric, respectively.

$$F_\beta = (1+\beta^2)\frac{Precision * Recall}{\beta^2 * Precision + Recall}, \tag{5.3}$$

where if we take class 1 (positive class) to be our class of interest we have,

$$Precision = \frac{TP}{TP+FP}, \tag{5.4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5.5}$$

The precision in Equation 5.4 gives the fraction of positive predicted samples that actually belong to the positive class. While the recall in Equation 5.5 gives the fraction of the positive samples that are predicted correctly. Values of this metric range between 0 and 1, where better performance is indicated by larger values.

In Section 5.2.5 we use this same intuition to calculate a robustness metric for evaluating the uncertainty and predictive performance together.

5.2.3 Matthew correlation coefficient

As mentioned a frequently used metric is accuracy. Although this metric is generally applicable, it is inadequate for imbalanced data, where some classes have significantly fewer samples than others. This metric tends to overlook the model's performance on the minority classes in such scenarios. Therefore, researchers have suggested some balanced alternatives of these metrics, which account for the weight (frequency of samples) of each class.

The Matthew Correlation Coefficient (MCC) [79] is a metric used to evaluate the predictive performance of binary classification models. It takes into account all the entries of the confusion matrix to provide a measure of the quality of a model's predictions and is designed to give a fairer score on imbalanced datasets [80, 81].

The MCC ranges from -1 to 1, with 1 indicating perfect prediction, 0 indicating random prediction, and -1 indicating total disagreement between prediction and observation. It is calculated as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{5.6}$$

However, despite its advantages on imbalanced data, it is undefined if there are no positive samples. This scenario may happen in our experiments on the private IDS dataset. We will elaborate on this further in the next section.

5.2.4 Balanced Accuracy

Balanced Accuracy [82] is a metric design for imbalanced datasets. It is the arithmetic mean of the sensitivity and specificity. The sensitivity is the proportion of positive samples that are correctly predicted out of all positive predictions. The specificity is the same ratio only for the negative class. The sensitivity is defined by the same formula as the recall, Equation 5.5, and the specificity is defined as the recall if we take 0 to be our class of interest, defined by

$$Specificity = \frac{TN}{(TN+FP)} \tag{5.7}$$

The Balanced Accuracy is given by,

$$\text{Balanced Accuracy} = \frac{Sensitivity + Specificity}{2} \tag{5.8}$$

However, in our assessment of the private IDS dataset, the stream of samples is very unbalanced resulting in batches of data that do not contain any positive (attack) samples. If no positive samples are present, the sensitivity rate of the balanced accuracy becomes undefined. As we would like to visualise the performance of these experiments by a continuous line, we suggest an adjusted balanced accuracy to avoid the metric from being undefined.

Adjusted Balanced Accuracy To give continuous lines in our graphs on batches that contain only one class. We suggest using only the specificity if the positive class is missing. This metric shows the true negative rate of the model, showing how many of the samples were correctly classified in that batch. If in a later batch, both negative and positive samples occur we can use the balanced accuracy where both classes have equal weight. This adjusted balanced accuracy is used in our plots in Figure 6.7, and shows a continuous line without any interruptions.

5.2.5 Robustness metric

As we are interested in the predictive and uncertainty quantification performance under concept drift, we need a metric that can accurately review the combination of these outputs in this challenging setting.

The area under the retention curve (R-AUC), explained in detail in [64], can be used to evaluate both the robustness to distributional drift and the quality of the uncertainty estimates. The retention curve shows the mean error over the dataset as a function of the fraction of the dataset retained. Ideally, the uncertainty should be aligned with this curve. R-AUC can be improved by either reducing the overall error of the model or by producing uncertainty estimates that better correlate with error. However, a potential drawback of these curves is that they may be more sensitive to the predictive performance than to the uncertainty quality.

To address this issue, Malinin et al. [70] proposed uncertainty-based F1-retention curves. For binary classification, we define an error criterion based on the logloss. We say that the prediction is said to be acceptable if its logloss ε is below a certain error threshold T_e. This can be expressed by using,

$$\mathcal{A}_{T_e}(x) = \begin{cases} 1, & \text{if } \varepsilon(x) \leq T_e \\ 0, & \text{else} \end{cases} \tag{5.9}$$

Then, these acceptability labels are used to assess how well the model's uncertainty estimates $\mathcal{U}(x)$ can indicate the acceptability of a prediction. A threshold T_u on the uncertainty score is set, such that predictions with higher uncertainty are considered poor and predictions with lower uncertainty are considered acceptable. These acceptability labels based on uncertainty are given by the following equation,

$$\mathcal{A}_{T_u}(x) = \begin{cases} 1, & \text{if } \mathcal{U}(x) \leq T_u \\ 0, & \text{else} \end{cases} \tag{5.10}$$

In contrast to Malinin et al. [70], who apply their metric to models that predict on static datasets. We compute the metric over the batches of a stream, as we are interested in the stability of the model over time. To make use of their metric we combine the predictions and uncertainty estimation of all the samples in each batch into one set. This set is used to get the final robustness metric.

Consider a dataset of size N the 'true' acceptability labels for this set are given by $\{\mathcal{A}_{T_e(x_i)}\}_{i=1}^{N}$ and the uncertainty acceptability labels are given by $\{\mathcal{A}_{T_u(x_i)}\}_{i=1}^{N}$. By these two sets, we can determine the 'precision' and 'recall' to calculate the F1-score, similar to Section 5.2.2 for all the uncertainty thresholds,

$$P_i = \frac{\sum_{j=1}^{N} \mathcal{A}_{T_e}(x_j) \cdot \mathcal{A}_{T_{\mathcal{U}_i}}(x_j)}{N_b - i}, \tag{5.11}$$

$$R_i = \frac{\sum_{j=1}^{N} \mathcal{A}_{T_e}(x_j) \cdot \mathcal{A}_{T_{\mathcal{U}_i}}(x_j)}{\sum_{j=1}^{N} \mathcal{A}_{T_e(x_j)}}, \tag{5.12}$$

where the final F1-score can be obtained in a similar way as Eq. 5.3,

$$\mathrm{F1}_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{5.13}$$

Note that we determine each $\mathrm{F1}_i$ with the uncertainty threshold of $\mathcal{U}_i$ in decreasing order. The resulting curve shows all the scores $\{\mathrm{F1}_i\}_{i=1}^{N}$ opposed to the selection of the samples that are considered acceptable by the uncertainty threshold, $1 - \frac{i}{N}$.

To compare the curves and measure the variation over the batches, we use the area under the retention curve as our final metric, denoted by F1-AUC.[4].

[4]https://github.com/Shifts-Project/shifts

An example of the retention curves is depicted in Figure 5.1. This figure shows three retention lines. The green line shows the optimal retention line where the uncertainty estimation aligns perfectly with the error rate, in our case the logloss. The blue line shows the 'random' baseline, where the uncertainties are completely uncorrelated with the error. The orange line is a model showing the correlation between the errors and uncertainty predicted. The x-axis is the retention fraction, meaning the number of elements considered to be acceptable given the uncertainty. The y-axis gives the F1-score resulting from Equation 5.13 for each retention fraction. The area under the curve for the optimal and model curves are shown in the legend.

The intuition of these curves can be described in the following manner. The F1-score is based on the precision and recall given by the acceptable predictions. For high retention fractions, we accept predictions for all below a high uncertainty, which implies that more predictions will be accepted. However, as we can see from the 'optimal' curve, if the predictions are accepted that should not be accepted based on their 'true' acceptance, i.e. the error, the F1-score decreases. A better model would more closely resemble this optimal curve. Our plots in Section 6.2 follow the same structure.

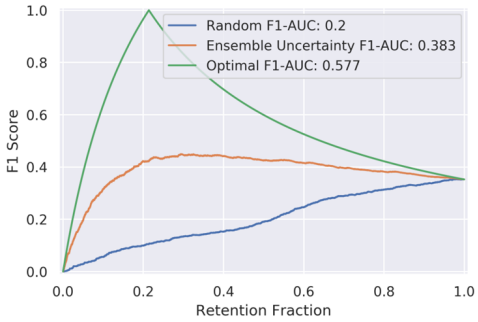


Figure 5.1: The retention curves of the optimal baseline (green), a model (orange) and random (blue). Source [70]

5.3 Experimental Design

5.3.1 Plasticity and Stability under adaptation

This section presents the methodology that aims to evaluate our first research sub-question: *How can the stability and plasticity of the model be balanced over time when adapting to concept drift under limited labels?*

Because our novel method prunes the model by using uncertainty estimation, the stability and plasticity of the model are dependent on the quality of this estimation. Poor uncertainty estimation will result in inaccuracy pruning and may cause the model to adapt poorly when concept drift arises. To assess the stability and plasticity of our novel method we need to be able to quantify the quality of our uncertainty estimation under continuous learning.

As explained in the metrics Section 5.2.5, a robustness metric can be used that aims to assess the joined predictive and uncertainty estimation performance. To assess the stability over time we use this metric where we compare the performance of our method with the optimal and random curves. The closer our curve is to the optimal curve the higher the correlation between the error and the uncertainty estimation. Because we aim to use our uncertainty estimation as a proxy of the loss this will give a good indication of the validity of our approach. Our method works on a stream we therefore combine all results into one set after all batches have been processed. The assessment is done on this combined set.

As we also would like to know if our method can provide stable uncertainty estimation under limited labels, we repeat the same experiment under a limited budget. The limited labelling budgets used are a 100%, 15%, 10% and 5% of the labels in each batch. We elaborate on the choice of these budgets in Section 5.3.2. The experiments are done on the SEAa dataset, as we know that this dataset contains concept drift and it is one of the most widely used synthetic datasets in concept drift literature [18]. Note

in our reporting of the results the value for T_e is set to 1 for determining the retention curve, this is the same as the default value the authors use for the metric. The uncertainty used in the metric is the total predictive of the eCBU method.

We hypothesise that our novel method can balance the stability and plasticity of the model over time by using the uncertainty to adapt to concept drift, as the limited budget would have minimal impact on the quality of the uncertainty estimation because the uncertainty is obtained based on unlabeled data.

5.3.2 Adaptation under limited labels

This section presents the methodology that aims to evaluate our second research sub-question: *How can the severity of the concept drift be accurately determined under limited labels?*

To evaluate the performance of our novel method in detecting the concept drift under limited labels, we first establish a baseline. This baseline compares the accuracy, the number of pruned trees and the number of retrains over the batches, of each method on each synthetic dataset. The goal of our method while under limited label availability, is to reduce the number of unnecessary retrains, as retraining involves selecting specific data or acquiring additional labels. This should be achieved while maintaining a comparable level of accuracy. After having established our baselines, we can evaluate the AdIter and our novel method with only limited labels.

Sample selection In an active learning approach, one aims to find the most useful sample(s) to use for training a model given a set of samples. Having selected this subset an oracle will label these instances and the model can be trained on these. In most real-world scenarios this oracle is a human determining the label for the given samples, as streams can consist of many samples and human label capacity will be limited, one wants to limit the number of samples that need to be labelled. When concept drift occurs, some of the incoming samples reflect the new concept more closely than others. In this case, active learning techniques aim at answering the question; How can we use an active learning approach to select a subset of these samples that can contribute the most to adapting to the drift while limiting the cost of labelling?

This question is beyond the scope of this work, but there are various existing methods in the literature that address it [83, 84, 16, 85, 86, 87]. The field of active learning can be divided into informativeness-based and representative-based selection. Both of these types of techniques have their advantages and disadvantages. To simulate the 'realistic' scenario of only limited label availability, without using a specific active learning method, and to minimise the influence of labelling selection influencing the concept drift detection in our experiments, we select samples in a random manner. The number of selected samples by this random selection method depends on the labelling budget.

Labeling budget To determine which labelling budgets are relevant to test, we refer to the work of Krawczyk et al. [88]. In this work, the authors found through an empirical analysis of multiple datasets under concept drift that, by reducing the number of labelled samples to 15% of the incoming data, the performance of models generally does not show a significant decrease when under these concept drifts. With these insights, we consider interesting labelling budgets to be 15% and below, as we assume by these findings the classifier will be able to reliably detect and adapt given more than 15% of the labels.

As the number of true labels available in a batch is limited by the user-defined labelling budget as defined by Equation 1.4. We consider multiple sizes of the labelling budget $\beta = \{15\%, 10\%, 5\%\}$. The question here is whether we can accurately determine the severity of the drift using no labels and in comparison to this limited budget.

Quantifying severity We quantify the severity of the concept drift by the number of trees that are pruned from the model. To assess the accuracy of the pruning, we compare it with the baseline methods that have access to all the labels. If the pruning exceeds a certain threshold, the model triggers a retraining process. Hence, overestimating or underestimating the severity of the concept drift can lead to excessive or insufficient retraining, respectively. We evaluate these two scenarios by contrasting the methods under limited labels with the baseline methods, which we assume to provide the optimal estimation of the 'true' severity of the concept drift, given their full label availability. Note that, regardless of the labelling budget, retraining the model always uses the labels of the entire batch. This is a necessary condition for the

adaptive learning algorithms to cope with severe concept drift, requiring fully changing the concept, which demands more labelled data for effective adaptation. However, selecting and labelling data for retraining is costly in real-world scenarios, therefore our objective is to minimise the frequency of retraining.

5.3.3 Challenging scenarios

This section presents the methodology that aims to evaluate our third research sub-question: *Which concept drift scenarios are the most challenging for adaptation under limited labels?*

The purpose of this research question is to explore the capability of our method on more challenging real-world datasets and with this assess the practicality of the approach in real-world scenarios. To evaluate this we use three real-world datasets, namely one Electricity dataset which is also used in the work that introduced AdIter and two intrusion detection datasets. We elaborate in more detail on the specifics of these datasets in Section 5.1.2.

We selected the Electricity dataset because it is one of the most commonly used datasets in the literature related to concept drift. This gives our work a broad base to be able to be compared with. For this reason of comparability, we report the performance on this dataset on multiple metrics explained in Section 5.2.

Next to this, we are interested in the intrusion detection setting as introduced in the introduction of the thesis. This setting is a challenging scenario as it is the objective of the hacker to make their attack look like benign data and continuously adjust their strategy to obtain this goal, effectively creating concept drift. Another challenge that adds to the complexity of intrusion detection is the rarity of attack events, creating an imbalance between attack data and benign data. The combination of these challenges makes this setting additionally interesting for our method in a real-world case.

With these challenging scenarios we seek to explore the impact limiting the label availability has on the performance of the AdIter method and if our method is able to cope with these. We evaluate this in a similar manner as our 'Adaptation under limited labels' evaluation. We take the AdIter method with access to 100% of the labels when adapting as the baseline for the pruning, retraining and performance. Each of the methods is tested on the same limited number of samples as explained in the previous section, 15%, 10% and 5%. For these cases, we are interested if the AdIter method initiates more retraining processes than the baseline indicating over-estimation of the drift. On the other side, we are interested in the performance of methods as this indicates how well the model can adapt to the real-world stream and indicates if the methods work while the data are imbalanced.

The experiments that will done on publicly available datasets are very similar to the experiments done in the work that introduced the AdIter method. The initial model is trained on the same number of samples in each batch that originate for the stream. For the private IDS dataset, however, we opt for a more realistic scenario that more closely resembles how it is done in practice. As this dataset contains 3-months of IDS alerts, we train the model on two months of data including the SMOTE data. To test the performance of the model we then evaluate it on the last month in the order the alerts have arrived without any SMOTE samples. The process of training the model with more than only a hundred samples when there is access to more data more closely resembles a real-world scenario.

We expect that our method will perform similarly despite the number of labels available. But as the number of labels is further limited, the adjustment of the model can be done less well, despite the same level of information at pruning. As we expect that our method will be influenced less compared to the AdIter method and therefore will not over-estimate or under-estimate as much as the AdIter method, we also expect that it will do a better job of reducing the number of false negatives, reducing alert fatigue.

5.3.4 Experimental setup

The methods are (re-)implemented and tested with the library of Catboost [57] version 1.2. Implemented in python 3.9. Experiments on the private dataset are run on a company computing cluster.

All experiments, excluding the private dataset, are run on a laptop with these specifications:

- CPU/GPU: AMD Ryzen 7 5700U with Radeon Graphics
- RAM: 16.0 GB

6

Results & Discussion

This chapter presents our results, starting with our baseline experiments to examine the slight difference between our baseline adaptation method and the original work. Then, we show results related to our first research question regarding the stability and plasticity of our method. Here, we show the robustness of the prediction and uncertainty quantification performance over time. After these results, we highlight the performance of the baselines on various labelling budgets as well as our novel method. Concluding this chapter with results on real-world datasets.

6.1 Baseline

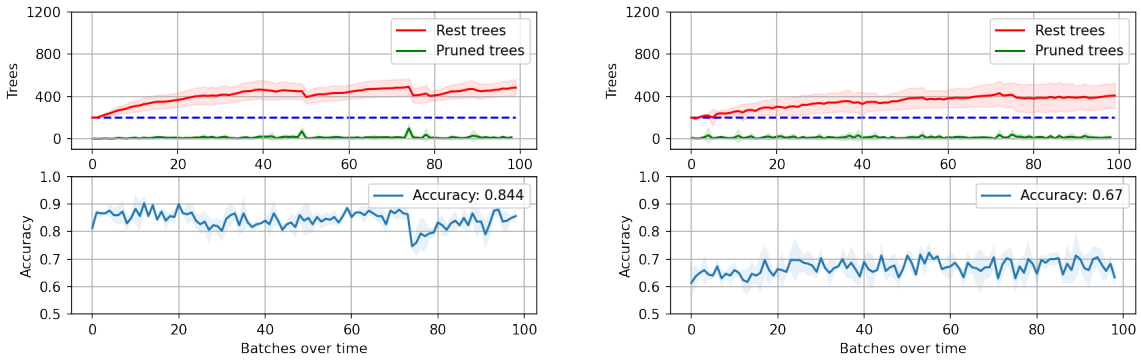
As we re-implement the AdIter algorithms with the CatBoost library to use as a base for our eCBU method, we provide the baseline result for this method. We evaluate our proposed method on all synthetic datasets from the work of Wang et al. [6] and compare it with their results. The adaption behaviour of our implementation of the AdIter method on all synthetic datasets is shown in Figure 6.1. The performance of both implementations can be found in Table 6.1.

To get the most similar result on both implementations. We use the same hyperparameters as used in the original work that was implemented with the sklearn library[1]. Namely, *(max depth = 4, subsample rate= 0.8, learning rate = 0.01)* which are specifically mentioned by in the work by Wang et al. [6]. Next to these, we use the default hyperparameters used with the sklearn library. The initial number of trees (iterations) is the same as the original work, *iterations = 200*.

However, due to the implementation dissimilarity, there are small differences. One of those differences is that CatBoost uses symmetric trees, and requires one to use this when using model shrinking. Another difference is the values used for L in our experiments. Instead of using the set of $\{L_1 = 25, L_2 = 50, L_3 = 75, L_4 = 100, L_5 = 125\}$ parameters, where each L_i defines the number of trees added when applying learning continuation, we use $\{L_1 = 5, L_2 = 10, L_3 = 15, L_4 = 20, L_5 = 25\}$. However, as pointed out by the authors of the AdIter method [6] the accuracy on the synthetic datasets is not significantly influenced by reducing the setting to a smaller number of iterations. Still, for the inference, the length of the sequence of trees influences the speed. To minimise the time the algorithm takes while keeping the accuracy on par we therefore pick these smaller sizes for L. Note that for the eCBU method we use *($L = 25$)* for all our experiments, by doing so our method has the same capability of plasticity as the AdIter method.

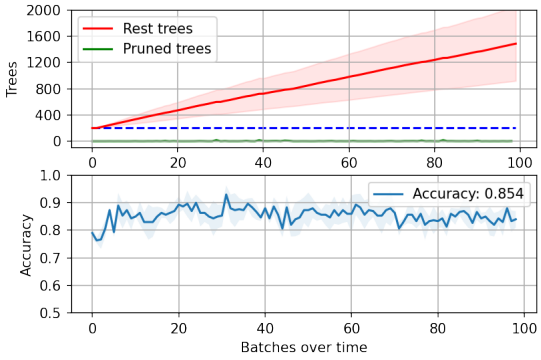
Each figure in Figure 6.1 consists of two subplots that illustrate the performance of the adaptation algorithm over the order batches from the data stream. The upper subplot displays the total number of trees (solid red line) after pruning, the initial number of trees and the retraining threshold (dashed blue line), and the number of pruned trees (solid green line) for each batch. The shaded areas around the solid lines represent the standard deviation of the results obtained from three repetitions of the experiments with different random seeds but identical generator settings. The lower subplot shows the prediction accuracy (solid blue line) for each batch, along with its standard deviation (shaded area).

[1] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

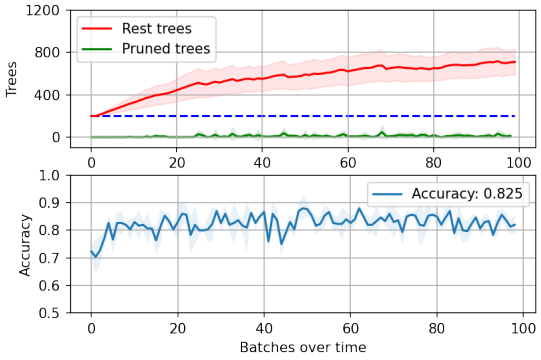


(a) SEAa

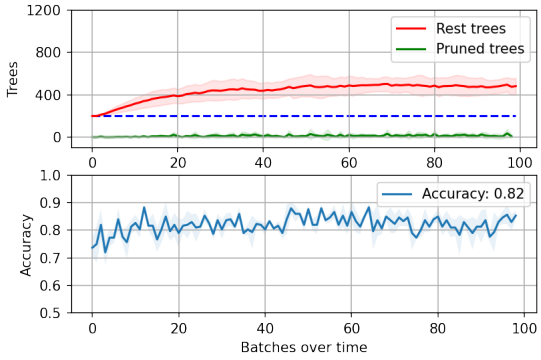
(b) RTG



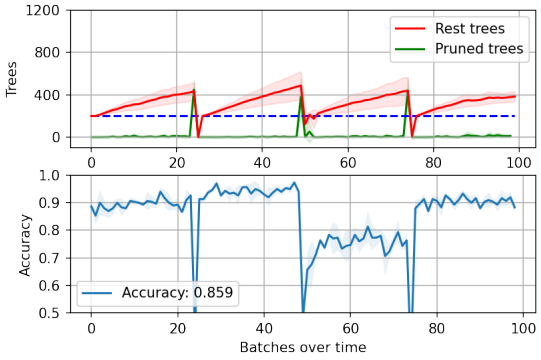
(c) RBF



(d) RBFr



(e) HYP



(f) AGRa

Figure 6.1: AdIter on six synthetic datasets, with all labels available. The dotted line indicates the retraining threshold.

Table 6.1 displays the accuracies for the AdIter method on six synthetic datasets. The first row are the accuracies given by Wang et al. [6]. The second row is our Catboost implementation of regression AdIter. Again the standard deviations for each of our experiments are obtained by three repetitions.

Methods	SEAa	RTG	RBF	RBFr	HYP	AGR
AdIter [6]	83.51 ± 0.86	65.96 ± 6.33	76.50 ± 1.47	82.03 ± 1.76	**86.93** ± 2.56	78.42 ± 1.13
AdIter (ours)	**84.43** ± 2.66	**67.01** ± 3.68	**85.43** ± 3.54	**82.46** ± 3.24	82.03 ± 3.00	**85.89** ± 2.30

Table 6.1: Accuracy of AdIter for both the original work and our CatBoost implementation, denoted in percentage (%). The best accuracy for each dataset is highlighted in bold.

.

From Figure 6.1 we can see that with our implementation retraining is only triggered for the AGR dataset. We see this as the number of rest trees gets below the blue dotted line, surpassing the retraining threshold. The points at which the retraining is triggered are the same at which the severe abrupt concept drifts occur, namely at the 25th, 50th and 75th batches. This is in line with what is presented in the work that introduced the AdIter method. If we look at the number of trees in the sequence indicated by the red line, all grow and stabilise, except for the behaviour on the RBF dataset. This is the same behaviour as the original work. The number of trees the algorithm grows to however is for all but RBF a bit higher, around 100 trees more. This behaviour can be explained by the slide differences between the libraries used.

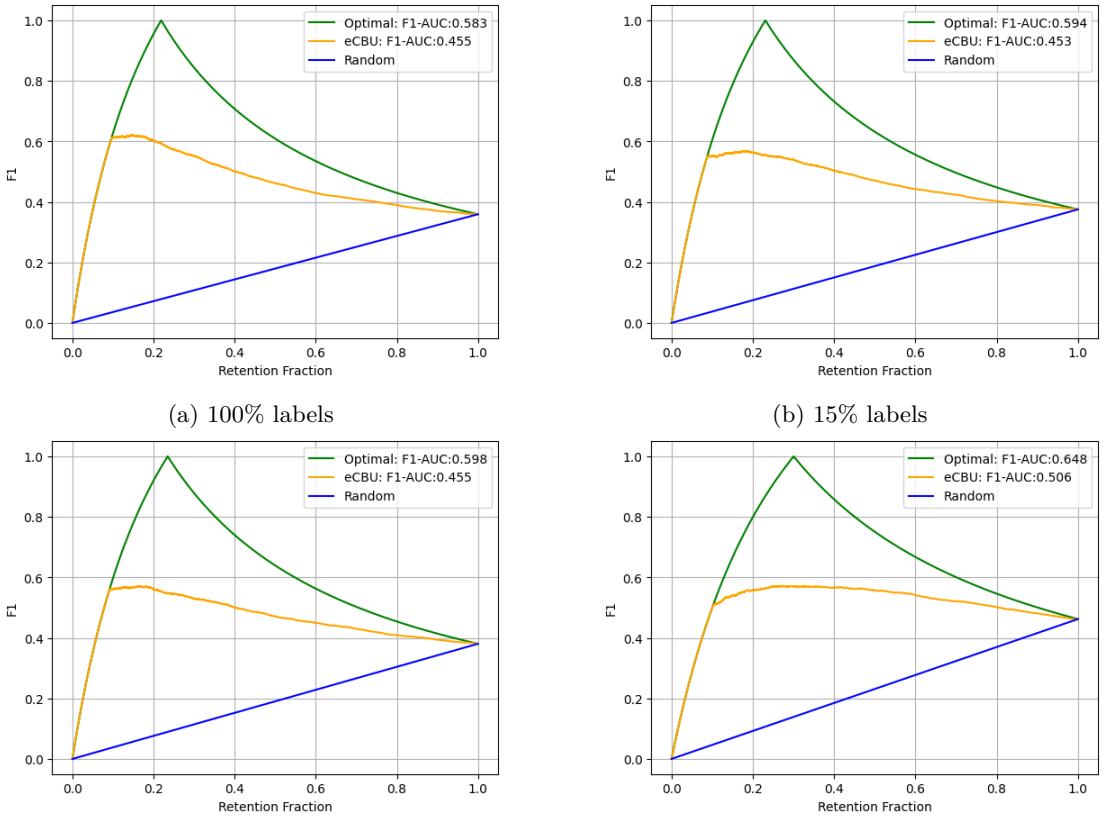
In terms of accuracy, which is shown in Table 6.1, our implementation performs better on most datasets except for the HYP dataset. This is most likely caused by the Catboost library as this generally outperforms the sklearn library, additionally, the number of trees in the sequence is slightly high with our implementation which can also cause an increase in performance.

Besides these slight differences, as we are mainly interested in the functioning of the algorithm under limited labels and given that for our implementation the adaptation behaviour is very similar, we can safely say that we can use this implementation as a baseline for our eCBU method. And because the AdIter is now implemented with the same library as the eCBU method can compare their performance well.

6.2 Plasticity and Stability under adaptation

As explained in our experimental design section we asses the stability and plasticity of our model, with the robustness metric 5.2.5. The curves of the multiple labelling budgets tested are shown in Figure 6.2. From these figures we can see that for all the labelling budgets tested the uncertainty estimation our method eCBU provides an uncertainty estimation that is positively correlated with the loss while under model adaptation and is better than random choice. From Figure 6.2a we see that the orange curve obtains a ROC-F1 score of around 0.45 and lies above the random curve. This shows that the uncertainty estimation is useful as a potential proxy for the loss. The same can be seen for the other labelling budgets.

Under the second research question, we seek to understand the stability and plasticity of our method that performs both prediction and uncertainty quantification. For the stability of our model, we use the retention curve to see the correlation between the F1-score and the retention fraction.



(a) 100% labels

(b) 15% labels

(c) 10% labels

(d) 5% labels

Figure 6.2: The retention curves with various label availability of the optimal baseline (green), eCBU (orange) and random (blue).

6.3 Adaptation under limited labels

In this section, we show the results concerning the second research sub-question *How can the severity of the concept drift be accurately determined under limited labels?*

In this section, we first show the problem that arises when applying the AdIter method under limited label availability, by the number of retraining done on our synthetic datasets. After this, we show the behaviour of the AdIter method and our eCBU method by similar curves introduced in Section 6.1. Finally, we give the accuracies obtained by the different methods and briefly discuss the results shown in this section.

Retraining Under the second research question, we aim to examine the effect of the limited labels on the existing method and whether our method provides an improvement. As a hypothesis in our problem formulation, we suggested that the AdIter method would be susceptible to under or over-estimation of the drift when lowering the number of labels available during adaptation. To test this hypothesis we evaluate the method under different labelling budgets and look at the behaviour and performance of the algorithm. When the algorithm overestimates the drift it would initiate unnecessary retraining of the model.

Based on our established selection techniques and labelling budget range. We have the following results, with three repetitions of each experiment, shown in Figure 6.3. This plot shows the percentage of batches that initiate the retraining process. We can see that for each dataset the more the labelling budget decreases the more the percentage of retraining batches increases. As the 100% blue bars indicate how often the retraining is initiated in our baseline case we see that for all synthetic datasets and all labelling budgets the retraining increases. If we have a closer look at dataset RTG, for this dataset we know that no concept drift occurs in the stream of data, however, the retraining process is initiated up to 70% of the time for a labelling budget of 5%. Based on this complete view of the different concept drift tested we can say that the AdIter method is pruning too much given the actual drift that occurs in the datasets and thus over-estimates the severity of the concept drift under labelling budgets that are 15% or smaller. With this, we can say that as we hypothesised this problem indeed occurs.

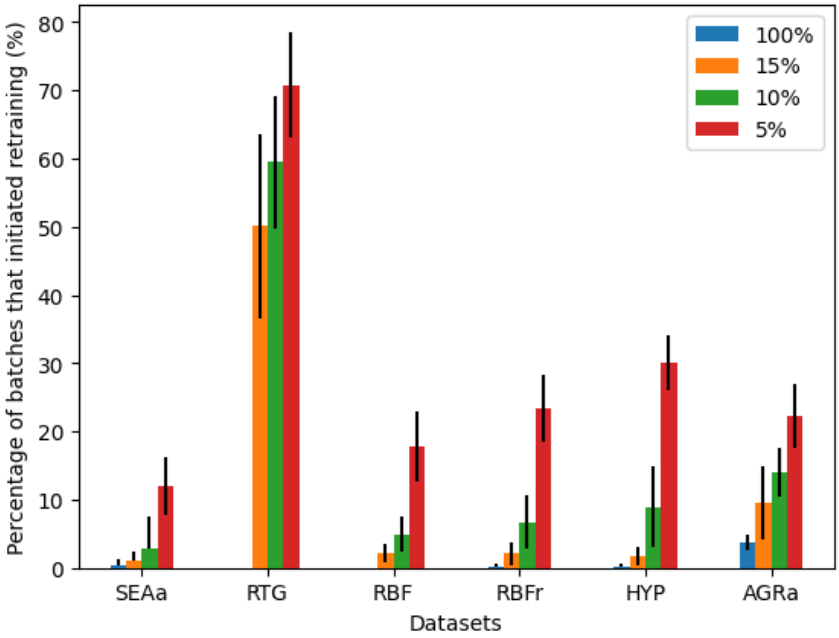
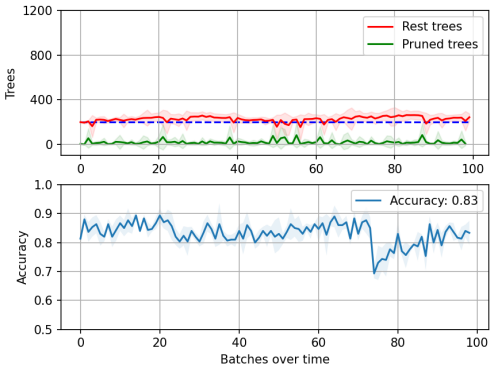


Figure 6.3: The percentage of batches triggering retraining, for the labelling budgets, 100%, 15%, 10% and 5%. For the six synthetic datasets.

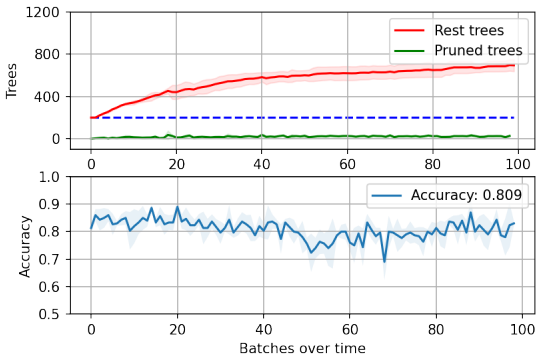
Pruning curves To show the behaviour of the pruning and continuous learning under limited labels of AdIter and our method we show the curves on three of the six datasets on a labelling budget of 5%. For datasets SEAa, RTG and HYP with abrupt, no and incremental concept drift, respectively. Similar to the graphs in Figure 6.1, the graphs in Figure 6.5 consist of the number of rest trees, the number of pruned trees, the retraining threshold and accuracy lines. With their shaded areas to indicate the variation between the three repetitions. The eCBU method shown uses sequential epistemic uncertainty as by the accuracy performance metric shown in Table 6.5 this type slightly outperforms the total predictive uncertainty type.

From these figures, we see that accurate pruning with the AdIter method is difficult. The method often prunes the sequence of trees such that the number of rest trees is below the retraining threshold. The extensive pruning causes the model to retrain frequently while this is not necessary if we compare this with the baseline. This is the same observation as made with the Figure 6.3.

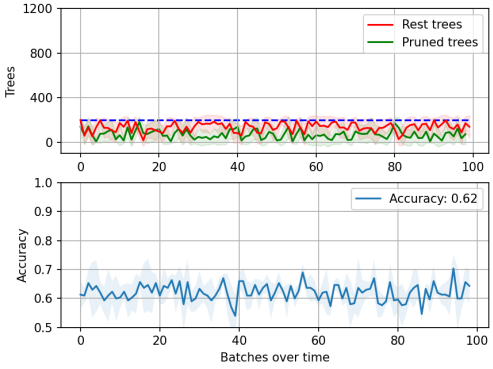
Whereas our method resembles the baselines of these curves given in Figure 6.1 more closely. From the



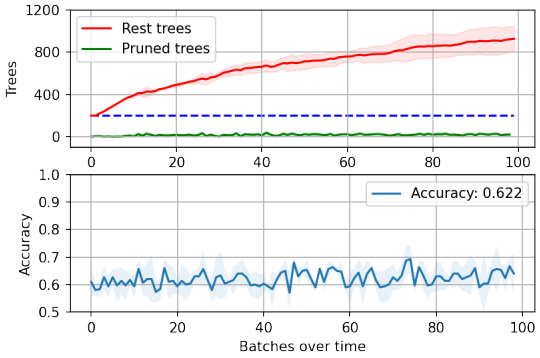
(a) SEAa - AdIter



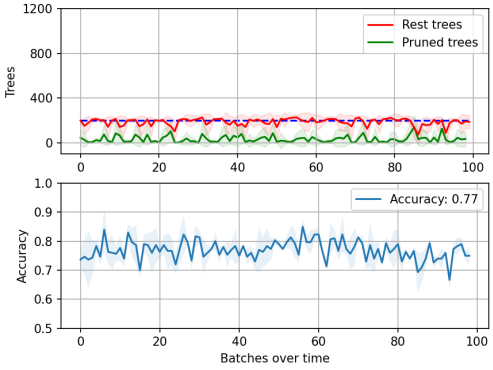
(b) SEAa - eCBU Epistemic



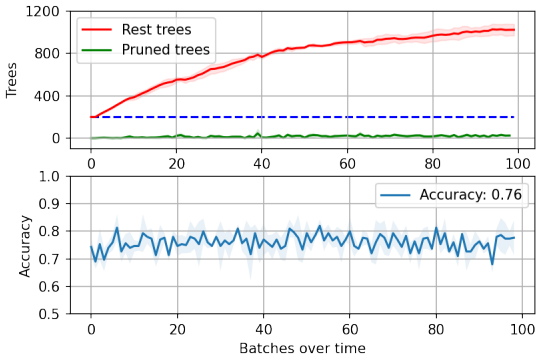
(c) RTG - AdIter



(d) RTG - eCBU Epistemic



(e) HYP - AdIter



(f) HYP - eCBU Epistemic

Figure 6.4: AdIter and eCBU on synthetic datasets SEA, RTG, HYP with a labelling budget of 5%

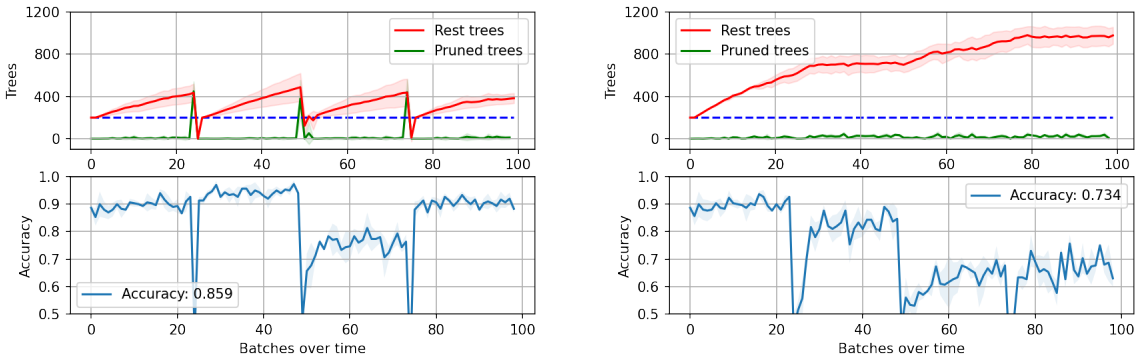
rest tree curves of the eCBU algorithm we see that despite that the method has only access to 5% of the labels, each of the curves follows a more similar profile as the baseline and does not retrain as much. We see that our method does not overestimate or underestimate the drift severity. We can infer this as the red lines, showing the number of trees, do not grow more rapidly than the baselines.

The accuracy curves in Figure 6.5 give an indication that the performance of the methods is similar. A more complete overview of all the accuracies on the various labelling budgets and methods, including both the eCBU by predictive and epistemic sequential uncertainty, can be found in Table 6.2.

If we compare the accuracy with the obtained by the AdIter method and the two types of the eCBU method we notice that they are comparable. With the exception of the AGR dataset which we will address later. Despite these similarities in accuracy, the AdIter requires a lot more labelling cost as it constantly retrains the full model, which uses all labels in a batch. Retraining is always done on the full batch, as we assume that when the algorithm initiates the retraining the concept has changed severely, One would like to collect all the data for this concept and therefore would invest the time to acquire these labels. These plots show the impracticality of the current method in a real-world scenario, where labelling and retraining are costly. Our method, however, does not overestimate or underestimate the drift severity. We inferred this by looking at the number of trees after pruning. The method achieves comparable accuracy while keeping retraining to a minimum and therefore putting less stress on the acquisition of labels.

One might notice that the accuracies in Table 6.2 are not very different. This is because the retraining is always done on the full batch, as we assume that for a severe concept drift that requires retraining the model, one would like to collect all the data for this concept drift and therefore invest the time to acquire these labels. However, it is interesting to see that for eCBU the accuracies do not decrease much while using only the given budget of labels and not retraining once.

As mentioned, the only noticeable difference in performance is on the AGR dataset which has severe abrupt drifts. If we look at Figure 6.5 we can see the pruning done over time for this dataset. The left figure displaying the results of the AdIter method shows clearly the point where concept drift occurs. For the eCBU method, the rest trees curve flattens at the moment of the first concept drift, indicating that more pruning is done after this point. However, the pruning does not exceed the retraining threshold. This modest pruning might be caused by a modest increase in uncertainty due to this concept drift. As the correlation between the error and the uncertainty is not perfect, the uncertainty estimation will naturally dampen extreme events.



(a) AGR - AdIter

(b) AGR - eCBU Epistemic

Figure 6.5: The AdIter and eCBU epistemic methods on the AGR severe abrupt drift dataset with 100% of the labels

By this evaluation, we can conclude that given these synthetic datasets, the sequential uncertainty estimation gives a better proxy for the drift severity than AdIter method under limited labels, excluding severe abrupt drift.

Methods	Budget	SEAa	RTG	RBF	RBFr	HYP	AGR
	100%	84.43 ± 2.66	67.01 ± 3.68	85.43 ± 3.54	82.46 ± 3.24	82.03 ± 3.00	85.89 ± 2.30
	15%	83.38 ± 2.72	63.39 ± 4.83	77.90 ± 3.51	77.90 ± 3.47	80.17 ± 3.20	84.56 ± 2.68
AdIter	10%	83.55 ± 2.94	63.16 ± 4.65	78.87 ± 4.20	78.02 ± 3.58	79.10 ± 3.48	84.39 ± 2.47
	5%	83.15 ± 2.87	62.09 ± 4.62	81.72 ± 3.80	77.46 ± 3.77	76.97 ± 3.47	84.20 ± 3.03
	100%	83.58 ± 2.87	65.96 ± 3.72	85.61 ± 4.36	82.50 ± 3.50	82.49 ± 2.88	60.61 ± 3.50
eCBU	15%	**82.12** ± 3.03	61.57 ± 4.35	74.12 ± 3.95	75.58 ± 3.69	**79.66** ± 3.33	59.43 ± 3.34
Total	10%	**80.93** ± 3.41	60.23 ± 4.66	69.81 ± 3.97	73.72 ± 3.84	77.93 ± 3.34	59.26 ± 3.45
	5%	78.55 ± 4.00	60.53 ± 4.70	66.97 ± 3.44	69.32 ± 3.98	66.30 ± 4.53	**62.17** ± 6.58
	100%	84.22 ± 2.67	68.06 ± 3.39	82.12 ± 3.32	82.33 ± 3.37	82.14 ± 2.67	73.44 ± 3.36
eCBU	15%	80.62 ± 2.84	**64.23 ± 4.71**	**76.67 ± 3.69**	**78.81 ± 3.83**	78.84 ± 3.17	**64.71 ± 3.56**
Epistemic	10%	80.31 ± 3.13	**62.24 ± 4.40**	**73.91 ± 3.88**	**77.65 ± 3.58**	**78.83 ± 3.02**	**61.22 ± 3.64**
	5%	**80.91 ± 3.55**	**62.15 ± 4.16**	**72.15 ± 3.36**	**75.87 ± 3.40**	**76.04 ± 3.74**	60.05 ± 3.53

Table 6.2: Accuracy, with random sample selection, for AdIter, eCBU total predictive uncertainty and eCBU epistemic uncertainty. For the eCBU method, we highlighted the best performance for each labelling budget and dataset.

6.4 Challenging scenarios

In this section, we show the results concerning the third research sub-question. *Which concept drift scenarios are the most challenging for adaptation under limited labels?*

This research question relates to the usefulness of the method on real-world data where concept drift might occur. As we do not know if or when concept drift occurs in these datasets we evaluate in comparison with the performance of the baseline. Our baseline is, just as with the synthetic datasets, the AdIter method with 100% of the labels available. First, we evaluate the AdIter method under limited labels and asses how much retraining is done. In Figure 6.6 we notice that with 100% labels available the algorithm only consistently retrains the model on the Electricity dataset. This indicates that for only this dataset there might be severe concept drift. For the others, it seems not to be necessary to retrain the model in order to keep its performance on the same level.

As we are interested in the performance of AdIter and our method under limited labels, we assess these methods again under 15%, 10% and 5% labelling budget. From the figure discussed, Figure 6.6, we can observe that for the NSL-KDD dataset, the retraining is not initiated much more when decreasing the available labels. The AdIter method stays fairly stable when limiting the number of labels for these datasets. However, when looking at the retraining percentage for the Electricity and private IDS dataset we notice an increase as we decrease the number of available labels. The percentage increases modestly for the Electricity dataset but increases to over 50% for the private IDS dataset when under 5% limited labels.

Electricity Table 6.3 shows the performance of the methods on the Electricity dataset by the MCC score. As the dataset is not perfectly balanced as with the synthetic datasets we use the MCC metric. From this table, we can observe that most scores are reasonably similar. However, the total predictive version of eCBU shows the best performance under the most limited budget.

Methods	100%	15%	10%	5%
AdIter	56.97	56.92	56.12	49.49
eCBU Total	50.88	55.13	52.59	54.30
eCBU Epistemic	56.25	56.55	54.46	47.60

Table 6.3: Performance on the Electricity dataset expressed in MCC (%).

NSL-KDD Table 6.4 shows the performance of the methods on the NSL-KDD dataset by the MCC score. Similar to the Electricity dataset we use this metric as the dataset is not balanced. From this table,

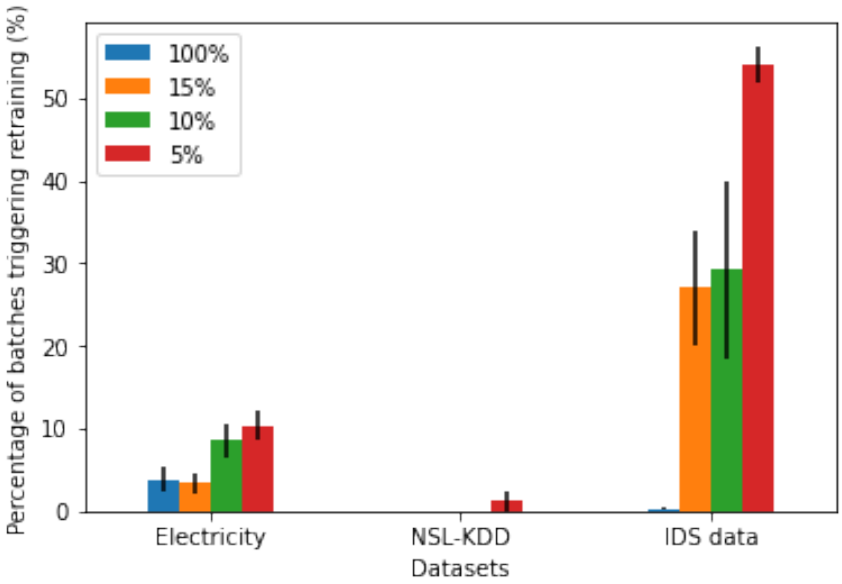


Figure 6.6: The percentage of batches by which retraining was initiated, for the labelling budgets, 100%, 15%, 10% and 5%. For the three real-world datasets.

we can observe that the eCBU method with epistemic uncertainty retains the most stable accuracy over all labelling budgets. On this dataset, it is interesting to see that for the AdIter method, the over-estimation which we have seen on all other datasets does not occur when limiting the number of labels.

Methods	100%	15%	10%	5%
AdIter	89.39	80.43	78.88	82.50
eCBU Total	90.76	84.74	78.68	77.04
eCBU Epistemic	88.38	84.78	82.51	81.93

Table 6.4: Performance on the NSL-KDD dataset expressed in MCC (%).

Proprietary IDS data The last dataset we test is the private company dataset on intrusion detection systems. As explained in the evaluation Section 5.3.3, the training process reflects a real scenario more closely. With this experiment, we use more data for the initial training. An additional difference from the previous two experiments is that we are using another metric. In this experiment, we use an adjusted version of the Balanced Accuracy, explained in Section 5.2.4. And report the number of attacks detected as well as the true negative rate.

From Table 6.5 we can see that the performance with our method is better under limited labels compared to the AdIter method while this method retrains frequently over time. To see more clearly what the behaviour of our method is compared to AdIter we provide the rest tree and pruning curve supported by the balanced accuracy in Figure 6.7. These figures show the graphs for the predictive uncertainty type of the eCBU method, figures on the epistemic version can be found in Appendix B.

From Figure 6.7, we can see that, the same as we observed by Figure 6.6, the AdIter prunes more as the number of available labels decreases. On the contrary, the eCBU method shows a more similar 'rest tree' curve of the batches compared with the baseline. We think that the rest three curves for the eCBU method shows these difference in growing and flatling because of the random sample selection. In our experiments on real-world datasets, we do not use any repetitions of the experiment with other seeds. The effect of some curves that flatline more while others grow more might be influenced by the particular samples selected.

When we look at the Balanced Accuracy lines shown in blue, in Figure 6.7, we notice that all have dips in between batches 40 and 60. These dips indicate the wrong classification of positive samples. As we are, next to limiting the number of false negatives, mainly interested in the number of attacks correctly

classified by the model over all the batches, we show the number of correctly classified positive samples in Table 6.6. From this table, we can see that the AdIter method identifies the most attack samples. This most likely is caused by the constant retraining of the model by AdIter, restoring knowledge in the model to a more balanced scenario. On the contrary, in the eCBU method, the number of trees is mostly growing and causing more overfitting on the majority class. This effect can also be seen in Table 6.7, which shows the true negative rate. This table shows that the eCBU, when using predictive uncertainty, correctly classifies up to 96% of the negative samples while performing the least on the attack samples.

Methods	100%	15%	10%	5%
AdIter	87.00	83.41	84.49	80.77
eCBU Total	87.40	87.94	95.30	93.29
eCBU Epistemic	89.34	85.23	85.62	82.45

Table 6.5: Performance on the proprietary IDS dataset expressed in adjusted Balanced Accuracy metric (%).

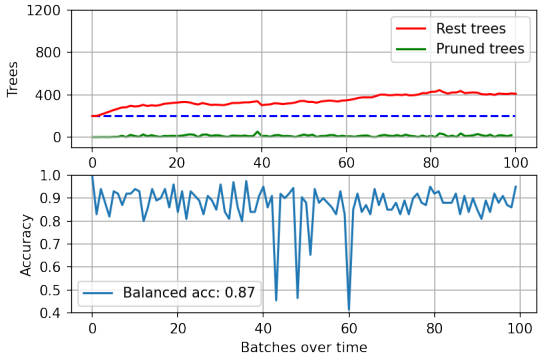
Methods	100%	15%	10%	5%
AdIter	7	8	10	10
eCBU Total	7	10	6	7
eCBU Epistemic	9	8	7	10

Table 6.6: Number of attack samples classified correctly out of 11 attack samples in the proprietary IDS dataset.

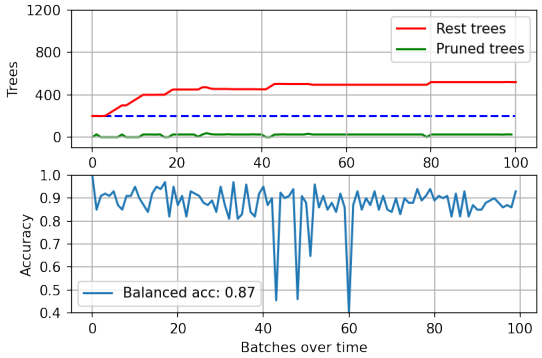
Methods	100%	15%	10%	5%
AdIter	88.44	83.77	83.93	80.72
eCBU Total	88.64	86.64	96.61	94.81
eCBU Epistemic	89.30	85.58	85.91	81.13

Table 6.7: The true negative rate (specificity) on the proprietary IDS dataset in percentages (%).

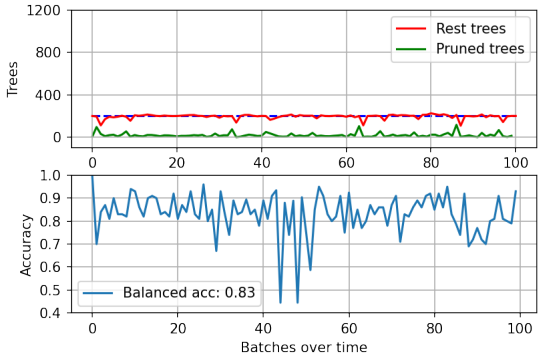
Overall, our introduced method does not generally underperform in comparison to the existing passive adaptation method AdIter on these three real-world datasets. On the Electricity and NSL-KDD datasets, the predictive performance is very comparable, while eCBU does not unnecessarily retrain the model. Moreover, our method is not prone to overestimation on the proprietary dataset as AdIter is under limited labels. However, in order to be less prone to overfitting on the majority class, we need to be more selective when selecting samples to be labelled



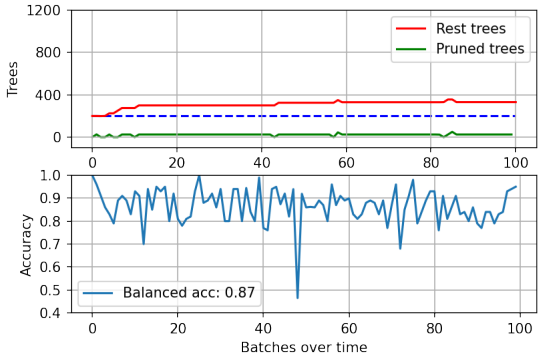
(a) AdIter - 100% labels



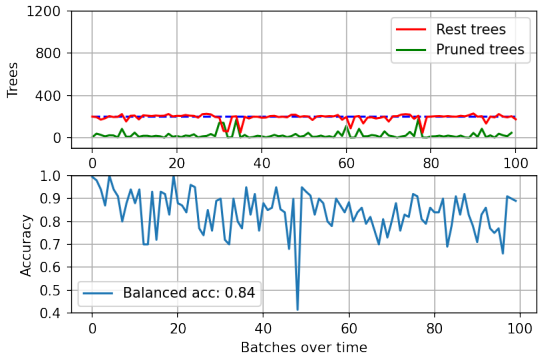
(b) eCBU - 100% labels



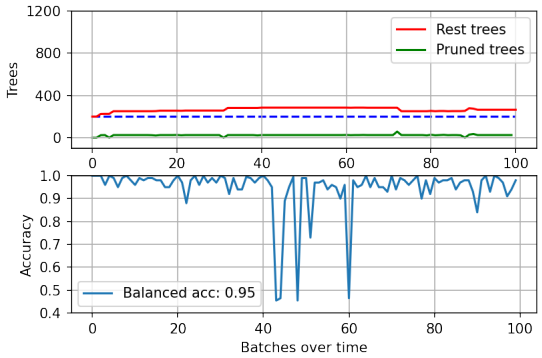
(c) AdIter - 15% labels



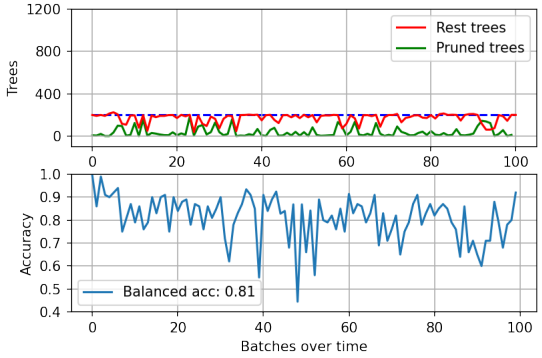
(d) eCBU - 15% labels



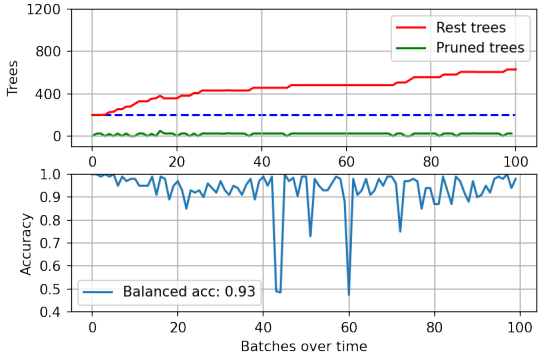
(e) AdIter - 10% labels



(f) eCBU - 10% labels



(g) AdIter - 5% labels



(h) eCBU - 5% labels

Figure 6.7: AdIter and eCBU on the IDS proprietary dataset with all labels available (a) and (b), 15% labelling budget (c) and (d), 10% labelling budget (e) and (f) and 5% labelling budget (g) and (h)

7

Closure

This chapter provides a summary of the main findings and contributions of the research, as well as a discussion of the limitations and suggestions for future work. The chapter begins by revisiting the research question and aims that guided the study, and highlighting how they were addressed by method and experiments. The chapter then acknowledges the limitations of the research design and method, and how they may have affected the validity and reliability of the results. Finally, the chapter presents some recommendations for future research that can build on the findings and address the gaps of this study. The chapter concludes with a brief summary of the main points.

7.1 Main insights

In this section, we summarise the main insights of our study and answer the research questions that motivated our work. Our main research question was: *How to reliably adapt GBDTs under limited labels?* To answer this question, we proposed a novel elastic gradient boosting decision tree algorithm that uses a sequential uncertainty estimation method to cope with concept drift and limited label availability in dynamic environments. We evaluated our algorithm on synthetic and real-world datasets and compared it to the Adaptive Iterations (AdIter) method. With our experiments, we aimed to answer the sub-questions resulting in an answer to the main question.

To answer the first sub-question, we conducted experiments on a synthetic dataset containing abrupt drift with different degrees of limited labels. We showed that our algorithm can provide accurate uncertainty estimations while continuously adapting the model to samples from a stream. This shows that despite these adaptations, the uncertainty estimation remained useful. This useful uncertainty estimation provided the method with the ability to balance the stability and plasticity of the model by using the sequential uncertainty estimation method when adjusting to concept drift under limited labels. This supports the adaptation of the model when needed.

To answer the second sub-question, we analysed the performance of the AdIter method on synthetic datasets, containing various types of concept drift. We found that the AdIter method tends to overestimate the concept drift on these synthetic datasets when given only a limited number of labels. We found that our algorithm has difficulty detecting big changes in the data, such as severe abrupt drifts. We observed that on the AGR dataset, which had severe abrupt drifts, our algorithm failed to detect the drift and was not able to adapt. This shows that our algorithm has limitations in handling extreme cases of concept drift. However, we also found that our algorithm can handle other types of concept drift, such as incremental or less severe abrupt, better than the AdIter method. And works much better when no concept drifts occur. However as indicated by the number of trees in the model when using eCBU it might also be more prone to underestimating drift, giving the model less chance to adapt well to new concepts. With this, we show that in most cases our algorithm can more accurately determine the severity of the concept drift under limited labels using the sequential uncertainty.

To answer the third sub-question, we identified the most challenging scenarios for adaptation under limited labels. To see whether our algorithm is applicable to real-world scenarios, we tested it on several real-world datasets, including intrusion detection. We observed that our algorithm is more stable and robust when given only a limited number of samples, and does not retrain when not necessary. We also

observed that our algorithm achieves comparable performance to the AdIter method on real-world datasets. With these insights, we answered our main research question by proposing a novel elastic gradient boosting decision tree algorithm that can reliably adapt to most concept drift types under limited labels.

7.2 Limitations & Future Work

In this section, we discuss the limitations of our study and the implications of our method for theory and practice. First, we start by pointing out the limitations of our study. After this, we discuss the limitations of our introduced method, which are either due to the architecture of the algorithm, used methods or observations based on experiments. Then we present some recommendations for future research and how our method can be improved.

7.2.1 Limitations of the study

In this study, we make a few assumptions most of which are pointed out in the introduction chapter in Section 1.3 under 'setting and assumptions'. Other assumptions and limitations of the study will be addressed here.

As our study revolves around concept drift adaptation with *limited labels*, the choice of labels has an influence on all the results. The selection of these labels influences both the detection and adjustment of the model. In this study, we chose to select our samples at random. In experiments that included repetitions, this effect is minimal. However, for other experiments, we need to keep in mind that this impacts the results, making them less generalisable. To mitigate this effect, repetitions of the same experiments should be done using different seeds for the random selection.

Another limitation of this sample selection method is that random selection does not provide us with an accurate reflection of reality. In most realistic scenarios, one would choose an active learning approach to select appropriate samples. The plasticity of the model while using eCBU can be increased when using proper active learning as the samples resemble more information about the concept drift, making the adjustment more rigorous in that direction while keeping it also stable through sequential uncertainty-based pruning. It might be that representative-based is even better in this case as it resembles more the whole batch and therefore the concept.

Another limitation of the work is the absence of a small sensitivity study conducted. To investigate if lowering the retraining threshold solves the problem of successive retraining of the model, we conducted a small analysis. Through this analysis, we concluded that when doing so, the problem is still present. By lowering the retraining threshold to near zero, we reduce the plasticity of the model such that it can not reliably adapt to concept drift anymore. However, a full and detailed enough experiment on this is absent due among other things to scarcity of time; therefore, this is a limitation of our study

7.2.2 Limitations of eCBU

Our proposed method has several limitations that we discuss in this section. The most significant limitation is the insufficient pruning of the tree sequence when facing severe abrupt concept drift. We show in Section 6.3 that the eCBU method does not prune enough trees to trigger retraining, which would improve the model's predictive performance.

Another limitation is the inability to handle concept drifts that involve concept evaluation or deletion, as these require the true label for detection. The AdIter method, which uses the loss on the labels, can detect these types of concept drift. Moreover, our method is restricted to classification GBDT models and cannot handle regression tasks.

The eCBU method employs the ensemble of GBDTs differently. For eCBU, the pruning point is determined collectively for all models in the ensemble. This reduces some of the flexibility that the AdIter method offers. In that method, each individual model has a different number of trees used for the adaptation, determined by parameter L. The pruning point is determined for each individual model, resulting in potentially different lengths of models. For our method, parameter L can still vary, but its effect is smaller.

Finally, another limitation is the accuracy of the uncertainty estimation. Literature has provided us with more accurate uncertainty estimation techniques for GBDTs that use the ensemble technique and can provide epistemic uncertainty. There are more accurate uncertainty estimation techniques for GBDTs that use the ensemble technique and can provide epistemic uncertainty as well. Two works KGB [50] and SGLB [58] provide more guarantees on the posterior estimation, enhancing the uncertainty estimation's reliability. However, these methods require a more complex GBDT model, making the learning continuation more complicated than trivial for SGB. This, along with Malinin et al. [7] stating that using SGLB does not significantly improve the uncertainty estimation to be useful in practice. However, this is not clear for KGB, showing a potential limitation of our method.

7.2.3 Future Work

Abrupt drift To overcome the inability of our method to cope with severe abrupt drifts, we could combine our method with UDD [17], which uses the uncertainty estimation to monitor the concept drift and trigger retraining of the model when a significant change is detected. This way, we could leverage the advantages of both methods: UDD would provide the sensitivity to abrupt drifts, while eCBU would provide the stability and plasticity to other drifts.

Model flexibility To reduce the problem of reduced flexibility due to collective pruning, we could extend our method to prune groups of ensembles within a larger ensemble. This would allow us to have different pruning points for different groups of models, increasing the flexibility and diversity of the ensemble. Moreover, this could also improve the accuracy of the uncertainty estimation, as it would reduce the correlation between the models.

Active learning To address the issue of realistic sample selection, we could adopt an active learning approach. By using either informative-based or representative-based [87] methods, samples could be more carefully selected and contribute more to the performance of the model. This could both improve the value of research as well as the practical application.

We believe that these directions for future work could enhance the performance and applicability of our method for handling concept drift under limited labels. We leave these extensions as future work.

7.3 Conclusion

In this thesis, we proposed a novel elastic gradient boosting decision tree algorithm that adapts various types of concept drift as well to dynamic real-world environments for intrusion detection under limited labels. Our algorithm uses a novel sequential uncertainty estimation method that indicates the severity of the drift without using any labels.

We evaluated our algorithm on synthetic and real-world datasets and compared it with a state-of-the-art method, the AdIter method. We demonstrated that our algorithm can balance the stability and plasticity of the model, more accurately determine the severity of most types of concept drifts, and handle various types of real-world scenarios under limited label availability.

However, our method has limitations as well. The most significant limitation is the inability to cope with severe abrupt drift. To mitigate this limitation, we suggested combining our method with a concept drift detection method in future work, combining their advantages.

Our work has several implications for both research and practice. For research, our work opens up new directions for exploring the estimation of severity in an unsupervised manner making adaptation techniques for gradient boosting decision trees in dynamic real-world environments more feasible. For practice, our work provides a practical step forward for intrusion detection systems or any other applications that involve a classification task in non-stationary stream-based environments with tabular data.

References

[1] Robin Sommer et al. "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". In: *2010 IEEE Symposium on Security and Privacy*. 2010, pp. 305–316. DOI: 10.1109/SP.2010.25.

[2] Nadezhda Bakhareva et al. "Attack Detection in Enterprise Networks by Machine Learning Methods". In: *2019 International Russian Automation Conference (RusAutoCon)*. 2019, pp. 1–6. DOI: 10.1109/RUSAUTOCON.2019.8867696.

[3] Yury Gorishniy et al. "Revisiting deep learning models for tabular data". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18932–18943.

[4] Ansam Khraisat et al. "Survey of intrusion detection systems: techniques, datasets and challenges". In: *Cybersecurity* 2.1 (2019), pp. 1–22.

[5] Swagatam Das et al. "On supervised class-imbalanced learning: An updated perspective and some key challenges". In: *IEEE Transactions on Artificial Intelligence* 3.6 (2022), pp. 973–993.

[6] Kun Wang et al. "Elastic gradient boosting decision tree with adaptive iterations for concept drift adaptation". In: *Neurocomputing* 491 (2022), pp. 288–304.

[7] Andrey Malinin et al. "Uncertainty in gradient boosting via ensembles". In: *arXiv preprint arXiv:2006.10562* (2020).

[8] Manuel Fernández-Delgado et al. "An extensive experimental survey of regression methods". In: *Neural Networks* 111 (2019), pp. 11–34.

[9] Ravid Shwartz-Ziv et al. "Tabular data: Deep learning is not all you need". In: *Information Fusion* 81 (2022), pp. 84–90.

[10] Jitendra Parmar et al. "Open-world machine learning: applications, challenges, and opportunities". In: *ACM Computing Surveys (CSUR)* (2021).

[11] Jobin Wilson et al. "Automatically optimized gradient boosting trees for classifying large volume high cardinality data streams under concept drift". In: *The NeurIPS'18 Competition*. Springer, 2020, pp. 317–335.

[12] Joaquin Quinonero-Candela et al. *Dataset shift in machine learning*. Mit Press, 2008.

[13] Jose G Moreno-Torres et al. "A unifying view on dataset shift in classification". In: *Pattern recognition* 45.1 (2012), pp. 521–530.

[14] Firas Bayram et al. "From concept drift to model degradation: An overview on performance-aware drift detectors". In: *Knowledge-Based Systems* (2022), p. 108632.

[15] Jie Lu et al. "Data-driven decision support under concept drift in streamed big data". In: *Complex & intelligent systems* 6.1 (2020), pp. 157–163.

[16] Davide Cacciarelli et al. "A survey on online active learning". In: *arXiv preprint arXiv:2302.08893* (2023).

[17] Lucas Baier et al. "Detecting concept drift with neural network model uncertainty". In: *arXiv preprint arXiv:2107.01873* (2021).

[18] Meng Han et al. "A survey of active and passive concept drift handling methods". In: *Computational Intelligence* 38.4 (2022), pp. 1492–1535.

[19] Bartosz Krawczyk et al. "Ensemble learning for data stream analysis: A survey". In: *Information Fusion* 37 (2017), pp. 132–156.

[20] Muhammad Umer et al. “Comparative analysis of extreme verification latency learning algorithms”. In: *arXiv preprint arXiv:2011.14917* (2020).

[21] Heitor M Gomes et al. “Adaptive random forests for evolving data stream classification”. In: *Machine Learning* 106 (2017), pp. 1469–1495.

[22] Kun Wang et al. “An elastic gradient boosting decision tree for concept drift learning”. In: *AI 2020: Advances in Artificial Intelligence: 33rd Australasian Joint Conference, AI 2020, Canberra, ACT, Australia, November 29–30, 2020, Proceedings 33*. Springer. 2020, pp. 420–432.

[23] Tuan Pham et al. “Stream-based active learning for sliding windows under the influence of verification latency”. In: *Machine Learning* (2022), pp. 1–26.

[24] Jingkang Yang et al. “Generalized out-of-distribution detection: A survey”. In: *arXiv preprint arXiv:2110.11334* (2021).

[25] Marc Masana et al. “Metric learning for novelty and anomaly detection”. In: *arXiv preprint arXiv:1808.05492* (2018).

[26] Marco AF Pimentel et al. “A review of novelty detection”. In: *Signal processing* 99 (2014), pp. 215–249.

[27] Xuan Xia et al. “GAN-based anomaly detection: A review”. In: *Neurocomputing* (2022).

[28] Varun Chandola et al. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[29] Ali Bou Nassif et al. “Machine learning for anomaly detection: A systematic review”. In: *IEEE Access* 9 (2021), pp. 78658–78700.

[30] Rosana Noronha Gemaque et al. “An overview of unsupervised drift detection methods”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10.6 (2020), e1381.

[31] Joao Gama et al. “Learning with drift detection”. In: *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*. Springer. 2004, pp. 286–295.

[32] Albert Bifet et al. “Learning from time-changing data with adaptive windowing”. In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM. 2007, pp. 443–448.

[33] Jie Lu et al. “Learning under concept drift: A review”. In: *IEEE transactions on knowledge and data engineering* 31.12 (2018), pp. 2346–2363.

[34] Hanzhang Hu et al. “Gradient boosting on stochastic data streams”. In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 595–603.

[35] Boyu Wang et al. “Online bagging and boosting for imbalanced data streams”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.12 (2016), pp. 3353–3366.

[36] Yoav Freund et al. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[37] Balaji Lakshminarayanan et al. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in neural information processing systems* 30 (2017).

[38] Valerio Freschi et al. “Evaluation of a sampling approach for computationally efficient uncertainty quantification in regression learning models”. In: *Neural Computing and Applications* 34.20 (2022), pp. 18113–18123.

[39] Matthias Seeger. “Gaussian processes for machine learning”. In: *International journal of neural systems* 14.02 (2004), pp. 69–106.

[40] Eyke Hüllermeier et al. “Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods”. In: *Machine Learning* 110.3 (Mar. 2021), pp. 457–506. DOI: 10.1007/s10994-021-05946-3. URL: https://doi.org/10.1007/s10994-021-05946-3.

[41] Moloud Abdar et al. “A review of uncertainty quantification in deep learning: Techniques, applications and challenges”. In: *Information Fusion* 76 (2021), pp. 243–297.

[42] Yarin Gal et al. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[43] Andrey Malinin et al. “Ensemble distribution distillation”. In: *arXiv preprint arXiv:1905.00076* (2019).

[44] Jakob Lindqvist et al. “A General Framework for Ensemble Distribution Distillation”. In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2020, pp. 1–6. DOI: 10.1109/MLSP49062.2020.9231703.

[45] Tony Duan et al. “Ngboost: Natural gradient boosting for probabilistic prediction”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2690–2700.

[46] Candice Bentéjac et al. “A comparative analysis of gradient boosting algorithms”. In: *Artificial Intelligence Review* 54.3 (2021), pp. 1937–1967.

[47] Jonathan Brophy et al. “Instance-Based Uncertainty Estimation for Gradient-Boosted Regression Trees”. In: *arXiv preprint arXiv:2205.11412* (2022).

[48] Olivier Sprangers et al. “Probabilistic gradient boosting machines for large-scale probabilistic regression”. In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 1510–1520.

[49] Patryk Wielopolski et al. “TreeFlow: Going beyond Tree-based Gaussian Probabilistic Regression”. In: *arXiv preprint arXiv:2206.04140* (2022).

[50] Aleksei Ustimenko et al. “GRADIENT BOOSTING PERFORMS GAUSSIAN PROCESS INFERENCE”. In: ().

[51] Clement Dombry et al. “A large sample theory for infinitesimal gradient boosting”. In: *arXiv preprint arXiv:2210.00736* (2022).

[52] Tian Tan et al. “Efficient and effective uncertainty quantification in gradient boosting via cyclical gradient MCMC”. In: *AAAI 2023 Workshop on Artificial Intelligence Safety*. 2023. URL: https://www.amazon.science/publications/efficient-and-effective-uncertainty-quantification-in-gradient-boosting-via-cyclical-gradient-mcmc.

[53] Ruqi Zhang et al. “Cyclical stochastic gradient MCMC for Bayesian deep learning”. In: *arXiv preprint arXiv:1902.03932* (2019).

[54] Leo Breiman et al. *Classification and regression trees*. Routledge, 2017.

[55] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[56] Jerome H Friedman. “Stochastic gradient boosting”. In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[57] Liudmila Prokhorenkova et al. “CatBoost: unbiased boosting with categorical features”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

[58] Aleksei Ustimenko et al. “SGLB: Stochastic gradient langevin boosting”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10487–10496.

[59] JT Hancock et al. *CatBoost for big data: an interdisciplinary review. J Big Data 7 (1): 94*. 2020.

[60] Richard O Duda et al. *Pattern classification and scene analysis*. Vol. 3. Wiley New York, 1973.

[61] Yiliao Song et al. “A segment-based drift adaptation method for data streams”. In: *IEEE transactions on neural networks and learning systems* 33.9 (2021), pp. 4876–4889.

[62] Łukasz Korycki et al. “Concept drift detection from multi-class imbalanced data streams”. In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE. 2021, pp. 1068–1079.

[63] Yu Sun et al. “Concept drift adaptation by exploiting historical knowledge”. In: *IEEE transactions on neural networks and learning systems* 29.10 (2018), pp. 4822–4832.

[64] Andrey Malinin. “Uncertainty estimation in deep learning with application to spoken language assessment”. Ph.D. Thesis. University of Cambridge, 2019.

[65] Thomas M Cover et al. “Elements of information theory second edition solutions to problems”. In: *Internet Access* (2006), pp. 19–20.

[66] Claude E Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[67] Niclas Ståhl et al. “Evaluation of uncertainty quantification in deep learning”. In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 18th International Conference, IPMU 2020, Lisbon, Portugal, June 15–19, 2020, Proceedings, Part I 18*. Springer. 2020, pp. 556–568.

[68] Jesus L Lobo et al. “On the Connection between Concept Drift and Uncertainty in Industrial Artificial Intelligence”. In: *arXiv preprint arXiv:2303.07940* (2023).

[69] Vu-Linh Nguyen et al. “Epistemic uncertainty sampling”. In: *Discovery Science: 22nd International Conference, DS 2019, Split, Croatia, October 28–30, 2019, Proceedings 22*. Springer. 2019, pp. 72–86.

[70] Andrey Malinin et al. “Shifts: A dataset of real distributional shift across multiple large-scale tasks”. In: *arXiv preprint arXiv:2107.07455* (2021).

[71] Vu-Linh Nguyen et al. “How to measure uncertainty in uncertainty sampling for active learning”. In: *Machine Learning* 111.1 (2022), pp. 89–122.

[72] W Nick Street et al. “A streaming ensemble algorithm (SEA) for large-scale classification”. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 2001, pp. 377–382.

[73] Geoff Hulten et al. “Mining time-changing data streams”. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 2001, pp. 97–106.

[74] Rakesh Agrawal et al. “Database mining: A performance perspective”. In: *IEEE transactions on knowledge and data engineering* 5.6 (1993), pp. 914–925.

[75] Dheeru Dua et al. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[76] Iman Sharafaldin et al. “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” In: *ICISSp* 1 (2018), pp. 108–116.

[77] Mahbod Tavallaee et al. “A detailed analysis of the KDD CUP 99 data set”. In: *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee. 2009, pp. 1–6.

[78] Amit Singhal et al. “Modern information retrieval: A brief overview”. In: *IEEE Data Eng. Bull.* 24.4 (2001), pp. 35–43.

[79] B.W. Matthews. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”. In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), pp. 442–451. DOI: `https://doi.org/10.1016/0005-2795(75)90109-9`. URL: `https://www.sciencedirect.com/science/article/pii/0005279575901099`.

[80] Sabri Boughorbel et al. “Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric”. In: *PloS one* 12.6 (2017), e0177678.

[81] Davide Chicco et al. “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”. In: *BMC genomics* 21.1 (2020), pp. 1–13.

[82] Kay Henning Brodersen et al. “The balanced accuracy and its posterior distribution”. In: *2010 20th international conference on pattern recognition*. IEEE. 2010, pp. 3121–3124.

[83] Xueying Zhan et al. “A Comparative Survey: Benchmarking for Pool-based Active Learning.” In: *IJCAI*. 2021, pp. 4679–4686.

[84] Alaa Tharwat et al. “A Survey on Active Learning: State-of-the-Art, Practical Challenges and Research Directions”. In: *Mathematics* 11.4 (2023), p. 820.

[85] Yifan Fu et al. “A survey on instance selection for active learning”. In: *Knowledge and information systems* 35 (2013), pp. 249–283.

[86] Sujoy Paul et al. “Learning with limited supervision: Static and dynamic tasks”. In: *Advanced Methods and Deep Learning in Computer Vision*. Elsevier, 2022, pp. 119–157.

[87] Sheng-Jun Huang et al. “Active learning by querying informative and representative examples”. In: *Advances in neural information processing systems* 23 (2010).

[88] Bartosz Krawczyk et al. “Combining active learning with concept drift detection for data stream mining”. In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE. 2018, pp. 2239–2244.

A

Additional details

A.1 Uncertainty Quantification

A.1.1 Bayesian Uncertainty

In Bayesian statistics, uncertainty can be modelled with the posterior distribution [40]. If we consider $\mathcal{H}$ to be the hypothesis space of probabilistic predictors, where h is a hypothesis in that space that maps an instance x to a target y. Then we can formulate the following posterior distribution,

$$p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{p(\mathcal{D})} \tag{A.1}$$

here $\mathcal{D}$ is data distribution, where $(x, y) \sim \mathcal{D}$. The $p(h)$ is the prior, which is the pre-known knowledge about the modelling. $p(\mathcal{D}|h)$ is the likelihood function of h, meaning how likely is the data if modelled by this hypothesis. The unconditional probability of the data, $p(\mathcal{D})$, can be modelled by,

$$p(\mathcal{D}) = \int p(\mathcal{D}|h)p(h)dh \tag{A.2}$$

However, this integral is over all possible values of h which is intractable in general. As a result, the posterior is considered to be proportional to the numerator of equation A.1.

$$p(h|\mathcal{D}) \propto p(\mathcal{D}|h)p(h) \tag{A.3}$$

This posterior distribution gives the likelihood of the model on the data and therefore reflects the knowledge in the model and epistemic uncertainty. The predictive posterior distribution is the former posterior under the mapping of the prediction $h \to p(y|x, h)$. So,

$$p(y|x) = \int_{\mathcal{H}} p(y|x, h)dP(h|\mathcal{D}) \tag{A.4}$$

This posterior takes all possible models and averages them out to get the predictive posterior, in practice, this is often difficult and computationally costly. Therefore generally one hypothesis is used.

$$h^{map} = \arg\max_{h \in \mathcal{H}} p(h|\mathcal{D}) \tag{A.5}$$

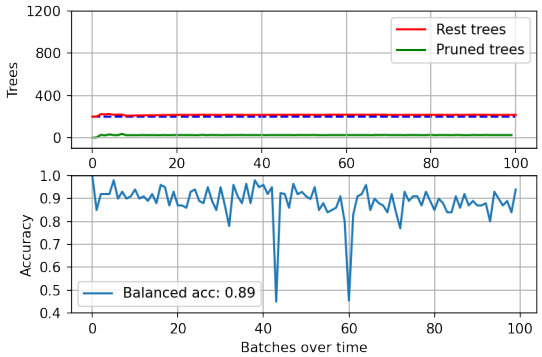
The highest posterior probability is adopted, as one wants to maximize the probability of the hypothesis given the data. This way the hypothesis can model the data best.

Unfortunately, due to the averaging effect of Eq. A.4, the aleatoric and epistemic uncertainty is no longer distinguishable. The same holds for Eq. A.5, due to only using one hypothesis. This shows the intractable nature of exact Bayesian uncertainty, however, there are methods to approximate this predictive posterior.

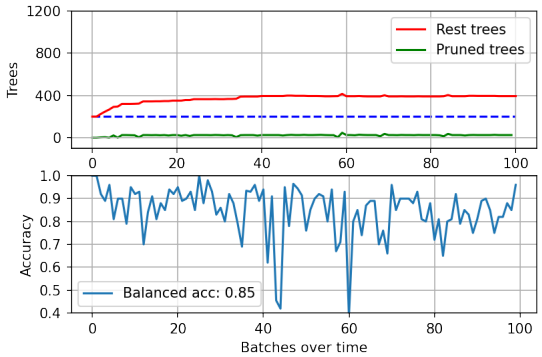
B

Results

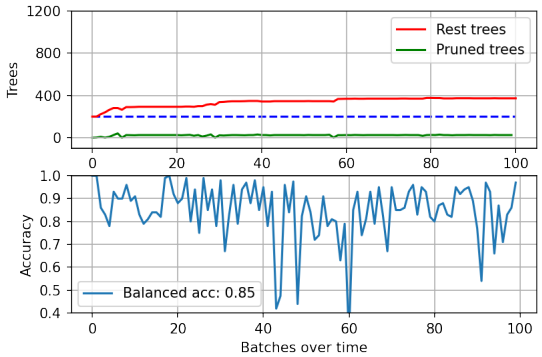
B.0.1 Challenging concept drifts



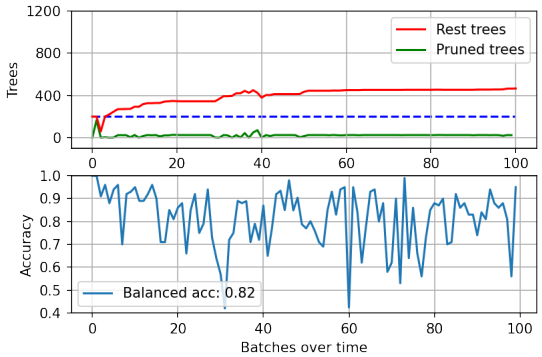
(a) eCBU Epistemic - 100% labels



(b) eCBU Epistemic - 15% labels



(c) eCBU Epistemic - 10% labels



(d) eCBU Epistemic - 5% labels

Figure B.1: eCBU Epistemic on the IDS proprietary dataset with all labels available.

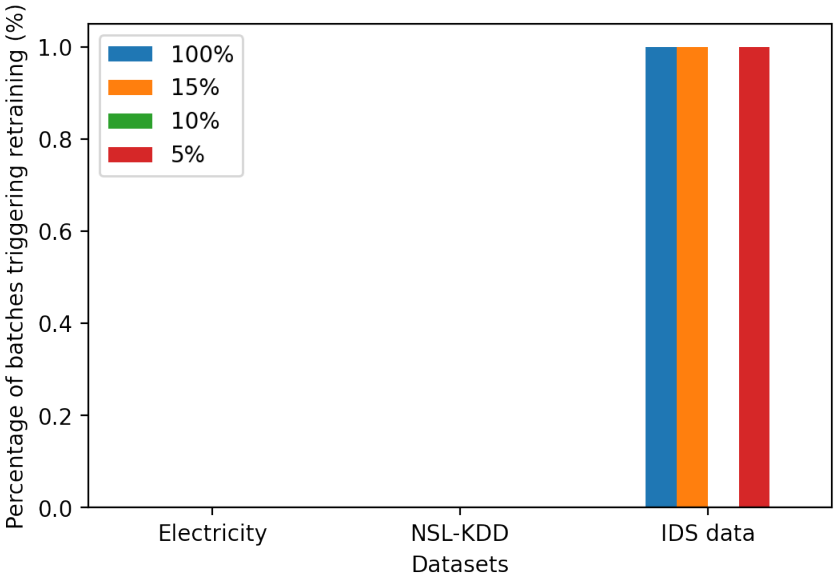


Figure B.2: bar plot ecbu total uncertainty

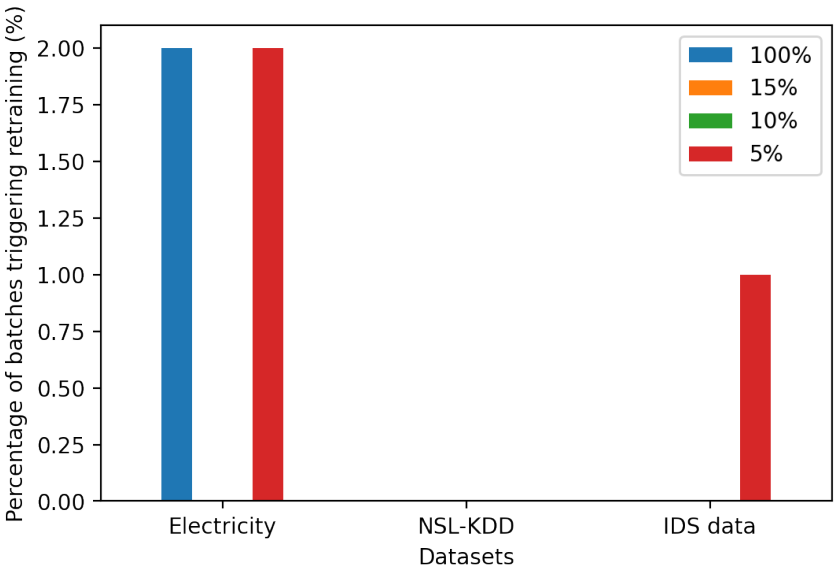


Figure B.3: bar plot ecbu knowledge uncertainty