Adaptation Strategies for Automated Machine Learning on Evolving Data

Bilge Celik and Joaquin Vanschoren

Abstract—Automated Machine Learning (AutoML) systems have been shown to efficiently build good models for new datasets. However, it is often not clear how well they can adapt when the data evolves over time. The main goal of this study is to understand the effect of concept drift on the performance of AutoML methods, and which adaptation strategies can be employed to make them more robust to changes in the underlying data. To that end, we propose 6 concept drift adaptation strategies and evaluate their effectiveness on a variety of AutoML approaches for building machine learning pipelines, including Bayesian optimization, genetic programming, and random search with automated stacking. These are evaluated empirically on real-world and synthetic data streams with different types of concept drift. Based on this analysis, we propose ways to develop more sophisticated and robust AutoML techniques.

Index Terms—AutoML, data streams, concept drift, adaptation strategies

1 Introduction

The field of automated machine learning (AutoML) aims to automatically design and build machine learning systems, replacing manual trial-and-error with systematic, data-driven decision making [43]. This makes robust, state-of-the-art machine learning accessible to a much broader range of practitioners and domain scientists.

Although AutoML has been shown to be very effective and even rival human machine learning experts [13], [24], [29], [36], they are usually only evaluated on static datasets [43]. However, data is often not static, it evolves. As a result, models that have been found to work well initially may become sub-optimal as the data changes over time. Indeed, most AutoML techniques assume that earlier evaluations are forever representative of new data, and may therefore fail to adapt to changes in the underlying distribution of the data, also known as concept drift [5], [23], [31].

There has been very limited work on automating the design of machine learning pipelines in settings with concept drift. That would require that the AutoML process, the automated search for optimal pipelines, is adapted so that it can cope with concept drift and adjust the pipelines over time, whenever the data changes so drastically that a pipeline redesign or re-tuning is warranted. Simply re-running AutoML techniques from scratch on new data is often not feasible, since AutoML techniques are usually computationally expensive, and in most online learning settings good predictive performance must be achieved in a limited time window.

To truly understand the effects of concept drift and how to cope with it in AutoML systems, we systematically study how different AutoML techniques are affected by different types of concept drift. We discover that, while these AutoML methods often perform very similarly on static datasets [20], they each respond very differently to different types of concept drift. We propose six different adaptation strategies to cope with concept drift and implement these in open-source AutoML libraries. We find that these can indeed be effectively used, paving the way towards novel AutoML techniques that are robust against evolving data.

The online learning setting under consideration is one where data can be buffered in *batches* using a moving window. Hence, we assume limited memory and limited time for making new predictions, and evaluate the performance of different adaptation strategies paired with different AutoML methods accordingly. Online settings where only one new instance fits in memory and can only be viewed once (single pass) are beyond the scope of this paper.

The paper is organized as follows. First, we formalize the problem in Section 2. Section 3 covers related work as well as the most relevant AutoML systems and online learning techniques. Our adaptation strategies for AutoML methods are described in Section 4. Section 5 details our empirical setup to evaluate these strategies, and the results are analyzed in Section 6. Section 7 concludes.

2 Problem Definition

2.1 AutoML

AutoML methods aim to design an optimal series of operations that transform raw data into desired outputs, such as a machine learning pipeline or a neural architecture [27]. In this paper, we are concerned with optimizing machine learning pipelines, which can be formally defined as a *combined algorithm selection and hyperparameter optimization (CASH)* problem [36]. Given a training set X_{tr} with targets y_{tr}, and a set of algorithms $A = \{A^1, ..., A^p\}$, each with a space of hyperparameters $\{\Lambda^1, ..., \Lambda^p\}$, the CASH problem is defined as finding the algorithm A^* and hyperparameter settings λ that minimize a loss function L (e.g. misclassification loss) applied to a model trained on $\{X_{tr}, y_{tr}\}$, and

• *B. Celik and J. Vanschoren work at the Eindhoven University of Technology, Eindhoven, the Netherlands.*
E-mail: B.Celik.Aydin@tue.nl

Manuscript received ...

evaluated on a validation set $\{X_{val}, y_{val}\}$, e.g. with k-fold cross validation where $\{X^i, y^i\}$ is a subset of the data set:

$$A^*_\lambda \in \underset{\substack{\forall A^* \in A \\ \forall \lambda \in \Lambda}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} L(A^j_\lambda, \{X^i_{tr}, y^i_{tr}\}, \{X^i_{val}, y^i_{val}\}) \quad (1)$$

Instead of a single learning algorithm, A can be a full pipeline including data preprocessing, feature preprocessing, meta-learning and model post-processing, which typically creates a large search space of possible pipelines and configurations. This results in important trade-offs between computational efficiency (e.g. training and prediction time) and accuracy [43].

2.2 Online learning and concept drift

In online learning, models are trained and used for prediction without having all training data beforehand. The data stream should be considered to be infinitely long, has to be processed in order, and only a limited part of it can be in memory at any given time [16]. Oftentimes, the data points are processed in small batches, also called *windows*. The process that generates the data may change over time, leading to *concept drift*, a usually unpredictable shift over time in the underlying distribution of the data.

Consider a data stream $\{X, y\}$ generated from a joint probability density function $p(X, y)$, also called the *concept* that we aim to learn. Concept drift can be described as:

$$\exists X : p_{t0}(X, y) \neq p_{t1}(X, y) \quad (2)$$

where $p_{t0}(X, y)$ and $p_{t1}(X, y)$ represent joint probability functions at time t_0 and t_1, respectively [16]. Webb et al. [41] further categorize concept drift into 9 main classes based on several quantitative and qualitative characteristics, of which the *duration* and *magnitude* of the drift have the greatest impact on learner selection and adaptation. *Drift duration* is the amount of time in which an initial concept (a_0, at time t_0) drifts to a resulting concept (a_1, at time t_1):

$$Duration(a_0, a_1) = t_1 - t_0 \quad (3)$$

Abrupt (sudden) drift is a change in concept occurring within a small time window γ:

$$AbruptDrift \Rightarrow Duration(a_0, a_1) < \gamma \quad (4)$$

Drift magnitude is the distance between the initial and resulting concepts over the drift period $[t_0, t_1]$:

$$Magnitude(t_0, t_1) = D(a_0, a_1) \quad (5)$$

where D is a distribution distance function that quantifies the difference between concepts at two points in time. Webb et al. [41] argue that magnitude will have a great impact on the ability of a learner to adapt to the drift. A minor abrupt drift may require refining a model, whereas major abrupt drift may require abandoning the model completely.

In *gradual drift*, the drift magnitude over a time period is smaller than a maximum difference μ between the concepts:

$$GradualDrift \Rightarrow \forall_{t \in [t_0, t_1]} D(a_0, a_1) < \mu \quad (6)$$

It is crucial to understand the dynamics of concept drift and its effect on the search strategy used by the AutoML technique in order to design a successful adaptation strategy. Drift detection algorithms (e.g. DDM [17]) are a key part of these strategies. They determine the location of drifts to alarm the learner so that it can react in a timely manner.

3 Related Work

To the best of our knowledge, there has been very little work that aims to understand how AutoML techniques can be adapted to deal with concept drift, even though there is clear evidence that most current AutoML techniques are greatly affected by it. A recent AutoML challenge focused on concept drift and found that current AutoML approaches did not adapt to concept drift in limited time frames, and were outperformed by incremental learning techniques, especially gradient boosting, with clever preprocessing [12].

There exists interesting work on speeding up hyperparameter optimization by transfer learning from prior tasks [22], or continual learning [9] where the goal is to adapt (deep learning) models to new tasks without catastrophic forgetting. However, these do not consider concept drift in single tasks, and evolving data cannot always easily be split up into multiple tasks. In the online learning literature there exist many techniques for handling different types of concept drift [16], but these usually adapt a previously chosen model rather than redesigning entire machine learning pipelines, as is the goal here. Interesting early results in algorithm selection were obtained using meta-learning to recommend algorithms over different streams [15]. More recently, hyperparameter tuning techniques have been proposed which re-initiate hyperparameter tuning when drift is detected [8], [40]. However, these are tied to specific optimization techniques for single algorithms, while we aim to generally adapt AutoML techniques to redesign or re-optimize entire pipelines. There also exist strategies to adapt models previously learned by online learning algorithms to new data, usually through ensembling [2], but these do not re-optimize the algorithms or pipelines themselves.

There is some interesting work in optimization that could potentially be leveraged. For instance, there exists work that enables Bayesian models to detect changepoints in the optimization process [18], [19], which could potentially be used to adapt the Bayesian Optimization techniques used in certain AutoML methods, but to the best of our knowledge this has not been previously explored.

Most similar to our work is a recent study by Madrid et al. [28], in which a specific AutoML method (autosklearn) is extended with concept drift detection and two model adaptation methods. In this paper, however, we extend not one, but a range of very different AutoML techniques. We also generate datasets with very different types of concept drift to be able to understand how these AutoML approaches are each affected in their own way, as well as how quickly they recover from concept drift, if at all. In addition, we propose and evaluate five different adaptation techniques that radically differ in their resource requirements and how they train candidate models. All of this provides clear empirical evidence and guidance on how to develop novel AutoML techniques more suited to evolving data.

In the remainder of this section, we will discuss the AutoML and online learning techniques used in this study.

3.1 AutoML techniques and systems

Bayesian Optimization (BO) is one of the most successfully used AutoML approaches in the literature [7]. To efficiently explore the large space of possible pipeline configurations, it trains a probabilistic *surrogate model* on the previously evaluated configurations to predict which unseen configurations should be tried next. The trade-off in this process is between exploitation of currently promising configurations versus exploration of new regions. In *Sequential Model-Based Optimization (SMBO)* configurations are evaluated one by one, each time updating the surrogate model and using the updated model to find new configurations. Popular choices for the surrogate model are *Gaussian Processes*, shown to give better results on problems with fewer dimensions and numerical hyperparameters [34], whereas Random Forest-based approaches are more successful in high-dimensional hyperparameter spaces of a discrete nature [13]. *Tree-structured Parzen Estimators* (TPE) are more amenable to parallel evaluation of configurations [3].

Evolutionary computation offers a very different approach. For instance, pipelines can be represented as trees and genetic programming can be used to cross-over and/or mutate the best pipelines to evolve them further, growing in complexity as needed [29]. *Random search* also remains a popular technique. While less sample-efficient, it can be easily parallellized and/or combined with other strategies.

General strategies to improve AutoML include multi-fidelity optimization techniques, which first try many configurations on small samples of the data, only evaluating the best ones on more data. Ensembling techniques such as voting or stacking can combine many previously trained configurations, and correct for over- or underfitting. Finally, it is often beneficial to use *meta-learning* to build on information gained from previous experiments, for instance to *warm-start* the search with the most promising configurations, or to design a smaller search space. For a wide survey of AutoML techniques, see [27].

In this paper, we will study the impact of concept drift on each of these approaches (BO, evolution, and random search). Since we need to adapt these approaches with novel adaptation strategies to handle concept drift, we select open-source, state-of-the-art AutoML systems for each of these.

3.1.1 Autosklearn

Autosklearn [13] is an AutoML system using Bayesian optimization (SMBO). It produces pipelines consisting of a wide range of classifiers and pre-processing techniques from scikit-learn [32]. It supports warm-starting by initiating the search with pipelines that worked well on similar prior data sets, as well as a greedy ensemble selection technique to build ensembles out of the different pipelines tried.

3.1.2 H2O AutoML

H2O AutoML performs random search in combination with automated stacked ensembles [24]. The stacking technique can be Random Forests, Gradient Boosting Machines (GBM), Deep Neural Nets, or generalized linear models (GLM). The stacked ensembles can contain either the best of all base models, or the best model from each algorithm family.

3.1.3 GAMA

GAMA uses genetic programming to automatically generate machine learning pipelines [21]. It is similar to TPOT [29], but uses asynchronous rather than synchronous evolution, and includes automated ensembling as well.

3.2 Learning on evolving data

In order to evaluate AutoML techniques on evolving data, we build on best practices in online learning.

3.2.1 Forgetting mechanisms

In order to adapt better to the changing environment, irrelevant past data should be forgotten. A popular forgetting mechanisms is a sliding window where data is disregarded at a constant rate [16]. Alternatively, data can be down-weighted with a dynamic rate when concept drift is detected, or removed based on the class distribution. The most appropriate forgetting mechanism depends on the environment and drift characteristics. In this work we will explore strategies using both sliding windows and concept drift detection.

3.2.2 Online learning algorithms

Although many stream learning algorithms use a single learner (e.g. Hoeffding Trees [10]), ensemble learners (e.g. SEA [35], Blast [38]) are commonly used because of their higher accuracy and faster adaptation to concept drift. Oza's online version of bagging [31] performs a majority vote over the base models. Leveraging Bagging [5] adds more input and output randomization to the base models to increase ensemble diversity at the cost of computational efficiency. Blast [38] builds a heterogeneous set of base learners and selects the learner that performed best in the previous window to make predictions for samples in the current window. Finally, online boosting performs boosting by weighting the base learners based on their prior performance and adjusting these weights over time [30]. We will compare our AutoML strategies against several of these methods.

3.2.3 Evaluation

The most common procedures to evaluate online learning algorithms are holdout, prequential, and data-chunk evaluation. Since the data in a stream must be processed in their temporal order, cross validation is not applicable [16]. Holdouts can be applied by requiring that only the earlier data is used for training and the later data for testing. In *prequential evaluation* (or *interleaved test-then-train*), newly arriving instances are first used to calculate the performance (test) and then for updating the model (train) in the next iteration. Accuracy changes incrementally as new instances and their labels arrive. Although this type of evaluation has been shown to yield lower accuracy scores than holdout on average [37], it is very useful to evaluate models in case of concept drift. A popular middle-way approach that combines holdout and prequential evaluation is *data chunk evaluation*, which uses data chunks of size s instead of individual instances to apply the test-then-train paradigm [4].

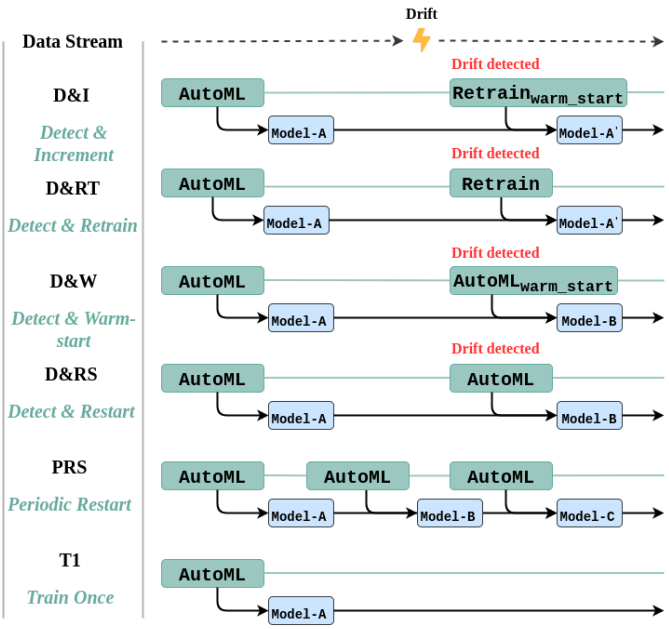

Fig. 1: Adaptation strategies

This approach doesn't punish algorithms for early mistakes and training time can be measured more consistently. We will therefore use data chunk evaluation in this paper.

4 Adaptation Strategies

Since AutoML methods usually optimize an objective function against a static training set (see Eq. 1), several modifications are required to apply them on evolving data. We call these modifications *adaptation strategies* because they can be generally applied to adapt AutoML methods. Madrid et al. [28] introduced two such strategies, but then specifically for auto-sklearn. In *global model replacement*, models are retrained on all prior data and recombined into ensembles based on meta-features when drift is detected. In *model management* only the weights of the base models in autosklearn's final ensemble are updated based on their performance.

4.1 Strategy definitions

In this paper, we propose and evaluate six different adaptation strategies, visualized in Fig. 1. In each of the following strategies, the AutoML algorithm is run at least once at the beginning with the initial batch of data. For the forgetting mechanism we use a fixed length sliding window, where the last s batches are used for training the learner and the earlier data is forgotten. The hyperparameter s can be chosen depending on the application: lower values imply lower memory requirements and faster adaptation to concept drift, while higher values imply larger training sets and less frequent re-optimization of the pipelines.

D&I Detect & Increment This strategy includes a drift detector that observes changes in pipeline performance and uses this to detect concept drift. AutoML is run on the first batch of data to build a first model (`Model-A`). We limit the AutoML search space to pipelines with incremental learning algorithms, i.e. learners that can continue to train on new data, such as random forests and gradient boosting. This restricts the AutoML methods, but it speeds up the retraining and we verified empirically that this still produces top-performing pipelines. `Model-A` is updated only if drift is detected, with the pipelines *trained incrementally* with the latest s (sliding window) batches. The configuration of the pipelines remains the same. This strategy assumes that the initial pipeline configuration will remain useful and only the models need to be updated in case their performance dwindles because of concept drift. It tests whether keeping a learner memory of past data is beneficial. This can give a performance boost if the learner can also adapt to the new data.

D&RT Detect & Retrain This strategy is similar, but runs AutoML without restricting the search space and retrains the pipelines *from scratch* on the latest s batches when drift is detected. The original pipeline configurations and ensemble weights are kept. This strategy also assumes that the initially found pipeline configuration will remain useful and only the models need to be retrained in case of concept drift. It eliminates the need for rerunning the (expensive) AutoML techniques with every drift, but may be less useful if the drift is so large that a pipeline redesign is needed.

D&WS Detect & Warm-start After drift is detected, the AutoML technique is re-run to find a better pipeline configuration. Instead of starting from scratch, however, the AutoML is rerun with a 'warm start', i.e. starting the search from the best earlier evaluated pipeline configurations. This strategy assumes that the initial pipeline configurations are no longer optimal after concept drift and should be re-tuned. Using a warm start mechanism can lead to faster convergence to good configurations, hence working better under small time budgets. However, in case of sudden drift the previously best pipelines may be misleading the search.

D&RS Detect & Restart After drift is detected, the AutoML technique is re-run from scratch, with a fixed time budget, to find a better pipeline configuration. This is a generalization of *global model replacement* [28] to other AutoML systems and an extension of the hyperparameter optimization in [8], [40] to full pipelines. This strategy also assumes that the pipelines need to be re-tuned after drift. Rerunning the AutoML from scratch is more expensive, but could result in significant performance improvements in case of significant drift.

PRS Periodic Restart Similar to 'Detect & Restart', except that instead of using a drift detector, AutoML is restarted with each new batch, or some other regular interval. This strategy tests whether a drift detector is useful, and whether it is worth to retrain at certain intervals in spite of the significant computational cost.

T1 Train once This is a baseline strategy added for comparison purposes. AutoML is run once at the beginning and the resulting model is used to test each upcoming batch. This tests the innate capabilities of AutoML methods to cope with concept drift.

TABLE 1: Implementation details for the AutoML method adaptations.

Strategy	H2O Adaptation	Autosklearn Adaptation	GAMA Adaptation
Detect & Increment	- Restrict the base models to GBM and RF, tune the hyperparameters with *RandomGridSearch*, and train a *StackedEnsemble* on the best pipelines using the initial batch. - When drift is detected, use the *checkpoint* option to train the base models incrementally on new data with the same hyperparameters. - Retrain the stacked ensemble with the updated models on the new data, with the same hyperparameters.	- Restrict the search space to GBM and RF, set option *warm-start = True*, and run AutoML on the initial batch. - When drift is detected, use the *refit* function to retrain the pipelines retrieved from the current ensemble on the latest data.	- Restrict the search space to GBM and RF, set option *warm-start = True*, and run AutoML on the initial batch. - When drift is detected, retrain the pipelines in the ensemble on the latest data by retraining the model retrieved from the previous AutoML run with *automl.model.fit*. Use the same hyperparameters.
Detect & Retrain	- Run AutoML on the initial batch. - When drift is detected, retrain the best model (*leader*). If this is a stacked ensemble, get the base models and retrain them on the new batches. - Retrain the stacked ensemble with the updated models on the new data.	- Run AutoML on the initial batch. - When drift is detected, use the *refit* function to retrain the pipelines in the ensemble model from the original AutoML run on the latest data.	- Run AutoML on the initial batch. - When drift is detected, use the *refit* function to retrain the pipelines in the ensemble model from the original AutoML run on the latest data with *automl.model.fit*
Detect & Warm-start	The best pipelines need to be retrieved from the previous *RandomGridSearch* and re-evaluated in the next, before retraining the *StackedEnsemble* on the best pipelines. Sadly, controlling the random grid search in this way is not currently supported in H2O. To approximate, we run a new random search and add the best new pipelines to a voting ensemble that also includes the previous best pipelines.	- To share the models between batches, run AutoML with options *parallel usage with manual process spawning* and *Ensemble_size = 0*. Then, build an ensemble on the trained models. - Warm-start the Bayesian optimization with the best previous pipelines. Keep 1/3 of the budget for building the ensemble (*fit_ensemble*). Use the default *Ensemble_size* and *ensemble_nbest*.	Use the *warm start* feature of the classifier fit function. This causes the evolutionary search algorithm to start from the best previous pipelines.
Detect & Restart	Use the drift detector after each tested batch. Retrain AutoML from scratch with a fixed time budget on the new data when drift is detected.		
Periodic Restart	Re-run the AutoML after every batch with a fixed time budget.		
Train once	No change. Only run the AutoML methods on the first batch.		

4.2 Integration in AutoML Systems

To evaluate the utility of these adaptation techniques, we have implemented them in Auto-sklearn, H2O and GAMA, representing AutoML approaches based on Bayesian optimization, random search, and evolution, respectively.

Drift detection. The first four adaptation strategies require a drift detector. We apply the Early Drift Detection Method (EDDM) [1] which is an improvement over the widely used DDM method [17] that better detects gradual drift. EDDM assumes that concept drift occurs when the learner's misclassifications are spaced closer together over time. After every misclassified sample i, it computes the average distance between misclassifications p_i and the standard deviation s_i in the current sequence (or batch). It also records value $(p_i + 2s_i)$, the 95% point of the distribution, and its peak value $(p_{max} + 2s_{max})$. Drift occurs when misclassifications are spaced more closely than usual:

$$(p_i + 2s_i)/(p_{max} + 2s_{max}) < \alpha \tag{7}$$

The threshold α is suggested to be set to 0.95.

Adapting AutoML techniques The remaining non-trivial adaptations required for each system are summarized in Table 1. First of all, none of the AutoML methods come with a built-in test-then-train procedure. We therefore perform data chunk evaluation by dividing the data into n batches and feeding them one by one in arriving order to the method for prediction/testing first, and training afterwards. None of the methods cover online learning algorithms. For *Detect & Increment* we restricted the search space to gradient boosting (GBM) and random forest (RF) models, which are supported by all methods and allow incremental learning.

5 Experimental Setup

In this section we describe the data streams used to evaluate the adaptation mechanisms and the setup of the AutoML systems. To ensure reproducibility, all data streams are publicly available at OpenML [39] together with results of experiments with different algorithms.[1] We also provide a github repository with our code and empirical results, including many plots that could not be fitted in this paper.[2]

5.1 Data streams

We selected 4 well-known classification data streams with real-world concept drift, and generated 15 artificial ones with different drift characteristics. The latter are generated using the MOA framework [6], and include gradual, sudden and mixed (gradual & sudden) drift. Within each drift type, the drift magnitude is changed by changing the underlying distribution functions, and in some cases a certain amount of artificial noise is added.

- AIRLINES is a time series on flight delays [23] with drift at daily and weekly intervals [42]. It has 539 383 instances and 9 numeric and categorical features.
- ELECTRICITY is a time series on electricity pricing [25] that has been shown to contain different kinds of drift [42]. It has 45 312 instances, 7 features, and 2 classes.
- IMDB is a text data stream with data from the Internet Movie Database [33]. An often-used task in drift research is to predict whether a movie belongs to the "drama" genre. It has 120 919 movie plots described by 1000 binary features.
- VEHICLE includes sensor data from a wireless sensor network for moving vehicle classification [11]. It has 98 528 instances of 50 acoustic and 50 seismic features.
- STREAMING ENSEMBLE ALGORITHM (SEA) is a data generator based on four classification functions [35]. We created data streams of 500 000 instances and 3 numerical features with concept drift. Abrupt drift is created by changing the underlying classification function at instance 250 000. The drift window, w, is the number of

1. Search www.openml.org for datasets tagged 'concept_drift'.
2. https://github.com/openml/continual-automl

instances through which the drift happens. The abrupt drift data streams were generated by switching between different functions with $w = 1$ and introducing 10% label noise. We also generated mixed drift streams by additionally adding two gradual drifts (before and after the abrupt drift) with $w = 100\,000$. The mixed drift data streams were generated with different magnitudes of sudden and gradual drift, without adding noise.

- ROTATING HYPERPLANE is a stream generator that generates d-dimensional classification problems in which the prediction is defined by a rotating hyperplane [26]. By changing the orientation and position of the hyperplane over time in a smooth manner, we can introduce smooth concept drift. We created a 5 gradual drift data streams with different drift magnitudes within a window w of $100\,000$. The data streams contain $500\,000$ instances with 10 features. 5% noise is added by randomly changing the class labels.

The data streams are divided into batches of equal size to simulate an online environment and given to the algorithms in arriving order, one batch at a time. We choose a batch size of 1000 to provide enough data for the AutoML algorithms and minimize the effect of accuracy fluctuations caused by small batches. For PRS, the batch size is larger ($\sim$20 000 instances) since it does not include a drift detector and requires retraining with every batch, which is much more expensive. We apply data chunk evaluation to evaluate the adapted AutoML systems. For training, a sliding window with a fixed length of 3 batches is chosen for each dataset.

5.2 Algorithms

We evaluate the adaptations of Autosklearn, GAMA and H2O. As baselines, we added Oza Bagging [4] and Blast [38]. Both are state-of-the-art online ensemble methods specifically designed for data streams with concept drift. We add gradient boosting as a baseline for incremental learning. We explain how each algorithm is configured. Hyperparameters that are not mentioned are used with their default setting.

- AUTO-SKLEARN The time budget for each AutoML run is the default 1 hour. The memory limit is increased to 12 GB to avoid memory errors.
- GAMA The time budget for each AutoML run is set to 1 hour. The option to build an ensemble out of the trained machine learning pipelines is activated.
- H2O The time budget for each AutoML run is set to 1 hour. All settings are kept at their defaults, including the default stacker, a Generalized Linear Model (GLM).
- OZA BAGGING from the scikit-multiflow library [31]. The window size is set to the batch size, 1000, and the pretrain size is also set to 1000. This way, training and prediction happens in the same way as for the AutoML methods, allowing a fairer comparison.
- BLAST from the MOA library [38]. The hyperparameters used are the defaults, and the window size is set to 1000, similar to the AutoML implementations.
- GBM Gradient boosting from the scikit-learn library [14]. The hyperparameters are set to their defaults and the window size is 1000. A drift detector is used to retrain the model when drift is detected.

Different drift detection algorithms were tested in multiple data streams prior to the selection. EDDM stands out as a robust and fast choice for different types of drifts, and hence used in all experiments.

6 RESULTS

We evaluate all algorithms on all data streams, and analyze the effects of drift characteristics, adaptation strategies and AutoML approaches.

6.1 Synthetic data streams

We first analyze the effect of two important drift characteristics: drift type and drift magnitude. Fig. 2-a to 2-c demonstrate the results of artificial data streams with gradual, abrupt and mixed drift, respectively. Each subgraph shows the results of one AutoML library, and each series shows the accuracy for a specific adaptation strategy after every batch. The markers on the lines show when drift was detected. PRS plots are smoother since the batch size was increased for efficiency. These figures show the results for the highest drift magnitude (for clearer analysis). Results for different levels of drift magnitude will be shown in Fig. 3.

6.1.1 Effect of Drift Type: High Abrupt Drift

As shown in Fig. 2-a, high abrupt drift severely affects all AutoML methods, and only some adaptation strategies help some of the AutoML methods to recover. GAMA is the quickest to recover after abrupt drift occurs when using warm-started AutoML (D&WS) or when retraining the previously best pipelines after the drift is detected (D&RT). Re-optimizing the pipelines from scratch after drift detection (D&RS) works for both GAMA and Autosklearn, but takes a bit longer to recover. H2O only recovers with warm-starting (D&WS). Periodically re-optimizing the pipelines from scratch without drift detection (PRS) also works for GAMA and AutoSklearn, but is of course more expensive. Blast is relatively unaffected by the abrupt drift and trails only slightly behind the recovered AutoML strategies. Oza Bagging is also unaffected by drift but performs significantly worse. GBM recovers after a delay but never regains its prior performance.

There are a few strategies that fail to recover at all. The default strategy of not updating the pipeline (T1) completely fails after sudden drift occurs. Incrementally updating the best pipeline (D&I) fails with Autosklearn and H2O, and never fully recovers with GAMA either. This indicates that after sudden drift, pipelines need to be either retrained or re-optimized. Interestingly, warm-starting (D&WS) doesn't help Autosklearn to recover, likely because the previously best pipelines are misleading its Bayesian Optimization approach. GAMA uses an evolutionary approach which does manage to evolve the previous pipelines into better ones. Surprisingly, for H2O also retraining (D&RT) and re-optimizing the pipelines (D&RS) fail to recover. This could be because the stacking procedure, a generalized linear model (GLM) by default, overfits on the predictions of the updated pipelines. This could potentially be solved with a more adaptive stacking method (e.g. GBM), larger batches, or maybe by feeding in the data points as well (cascade stacking). We will analyze this in more detail in Section 6.4.

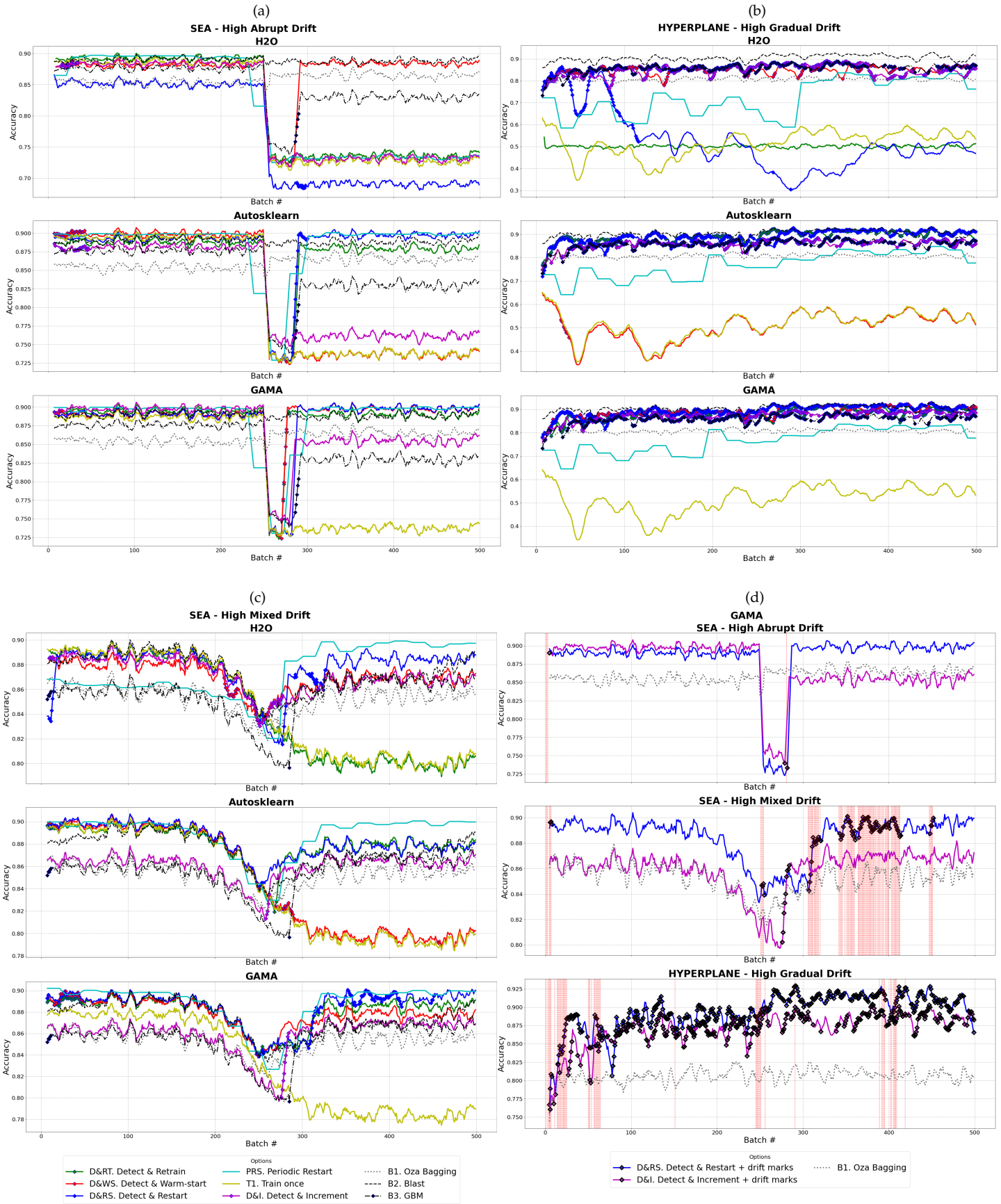

Fig. 2: Accuracy across batches for artificial data streams: (a) SEA - High abrupt drift; (b) HYPERPLANE - High gradual drift; (c) SEA - High mixed drift, and (d) All three artificial data streams with drift marks and pipeline change points (red)

6.1.2 Effect of Drift Type: High Gradual Drift

As shown in Fig. 2-b, D&RT and D&RS perform well under high gradual drift in both GAMA and Autosklearn, but not for H2O (again likely due to the GLM stacker overfitting). D&I works quite well for all AutoML techniques. D&WS works well for GAMA, but not for Autosklearn (again likely due to the Bayesian surrogate model being misled). Note that Blast is on par with the best options in GAMA and Autosklearn, and better than any option in H2O. OzaBagging is dominated by several AutoML adaptations. GBM performs better but still slightly worse than the best AutoML adaptations. The T1 baseline again performs poorly. PRS is also suboptimal, likely because the periodic restarts are not frequent enough and continually lag the high gradual drift.

6.1.3 Effect of Drift Type: High Mixed Drift

Fig. 2-c shows that in case of high mixed drift, restarting the AutoML on new data (D&RS) performs best overall. Compared to the abrupt drift data stream (Fig. 2-a, AutoML methods handle the sudden drift better and recover faster. This is likely because the gradual drift prior to and after the sudden drift alarms the drift detectors periodically and causes more frequent retraining. Oza Bagging and Blast handle the drift well, yet overall fail to match the performance of the best AutoML strategies (D&RS and D&RT), especially after the sudden drift point. GBM performs similarly to Oza Bagging, but with a better recovery after the sudden drift. As in Fig. 2-b, D&RT doesn't work well for H2O and D&WS doesn't work well for Autosklearn, likely for the same reasons. PRS recovers exceptionally well after the sudden drift, showing the advantages of re-running AutoML when significant drift occurs.

6.1.4 Pipeline analysis

In order to gain more insight into the underlying reasons behind these performance differences, pipelines at consecutive training points are compared. Fig. 2-d indicates when pipelines change (vertical red lines), and when drift is detected (black line marks) for each of the three artificial data streams and strategy D&RT. The accuracy plots of strategies D&RT and D&I are compared in addition to the Oza Bagging baseline. Retraining without re-optimizing the pipeline (D&I) can outperform the baseline but is generally worse than re-optimizing the pipeline (D&RT). This difference is more clear after the abrupt drift points. Therefore, while an initial AutoML optimization improves the performance compared to a baseline learner, re-optimizing the pipeline throughout the data stream can lead to different pipeline settings and clear advantages over static pipelines. The pipeline change points indicate that D&RT indeed finds new pipelines regularly, especially under gradual or mixed drift.

6.1.5 Effect of drift magnitude

Fig. 3 shows the results of GAMA library for increasing levels of *drift magnitude* on the SEA abrupt drift data stream. As the drift magnitude increases, the performance drop also increases, but the overall recovery period *decreases*. This is most likely due to the early triggering of the drift detector in high magnitude cases. The best adaptation strategy changes with the drift magnitude, as previously suggested in [28].

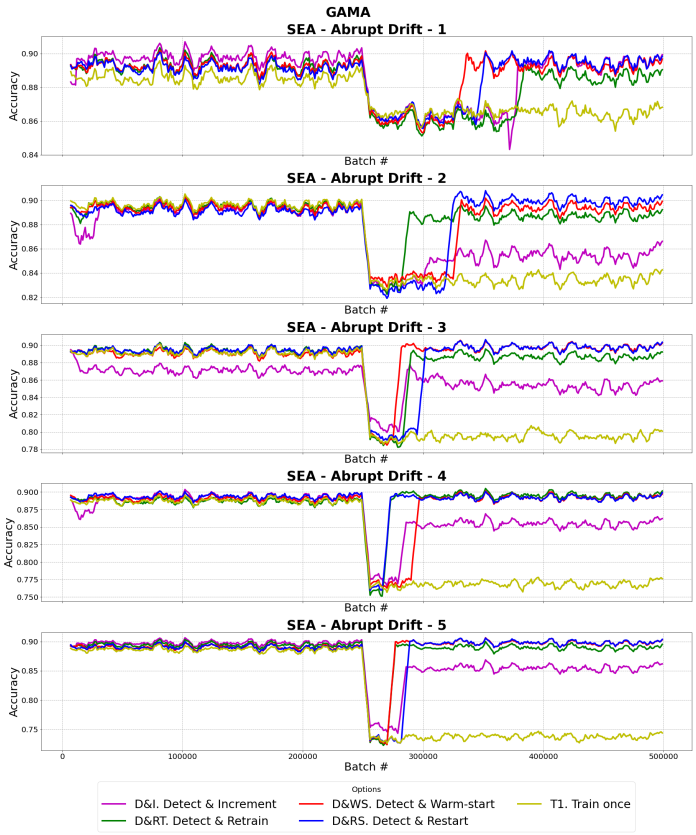

Fig. 3: Effect of increasing abrupt drift magnitudes (1 is lowest magnitude) on different strategies, for GAMA.

Retraining the models (D&RT), however, generally recovers faster than restarting the AutoML (D&RS). Warm-starting (D&WS) sometimes recovers faster, but not always. The latter may depend on exactly *how* the data drifts. If the concept underlying the data is still somewhat similar after the drift, i.e., it can be modelled with similar pipeline configurations, then warm-starting will speed up recovery, but if not, warm-starting could also be unhelpful.

6.2 Real Data Streams

The results on the real-world data streams are shown in Fig. 4-a to Fig. 4-d. The Airlines data (Fig. 4-a) has particularly frequent drift. D&RT and D&I clearly perform worse here, indicating that sticking to the originally optimized pipeline is not ideal. PRS doesn't adapt quickly enough: restarting AutoML with every batch could be better but is also prohibitively expensive. The other adaptation strategies behave roughly the same, especially for GAMA where they overlap almost completely. D&WS or D&RS work generally well. H2O is again the exception: D&RS does not recover well from drift (as observed earlier) and the other strategies fluctuate significantly. Blast works particularly well here. Oza Bagging is often outperformed by the best AutoML method but also fluctuates less. On IMDB data (Fig. 4-c) we can make similar observations, although the drift is more gradual here, allowing PRS to find better pipelines. Blast is roughly on par with the AutoML strategies, while Oza Bagging and GBM are clearly worse.

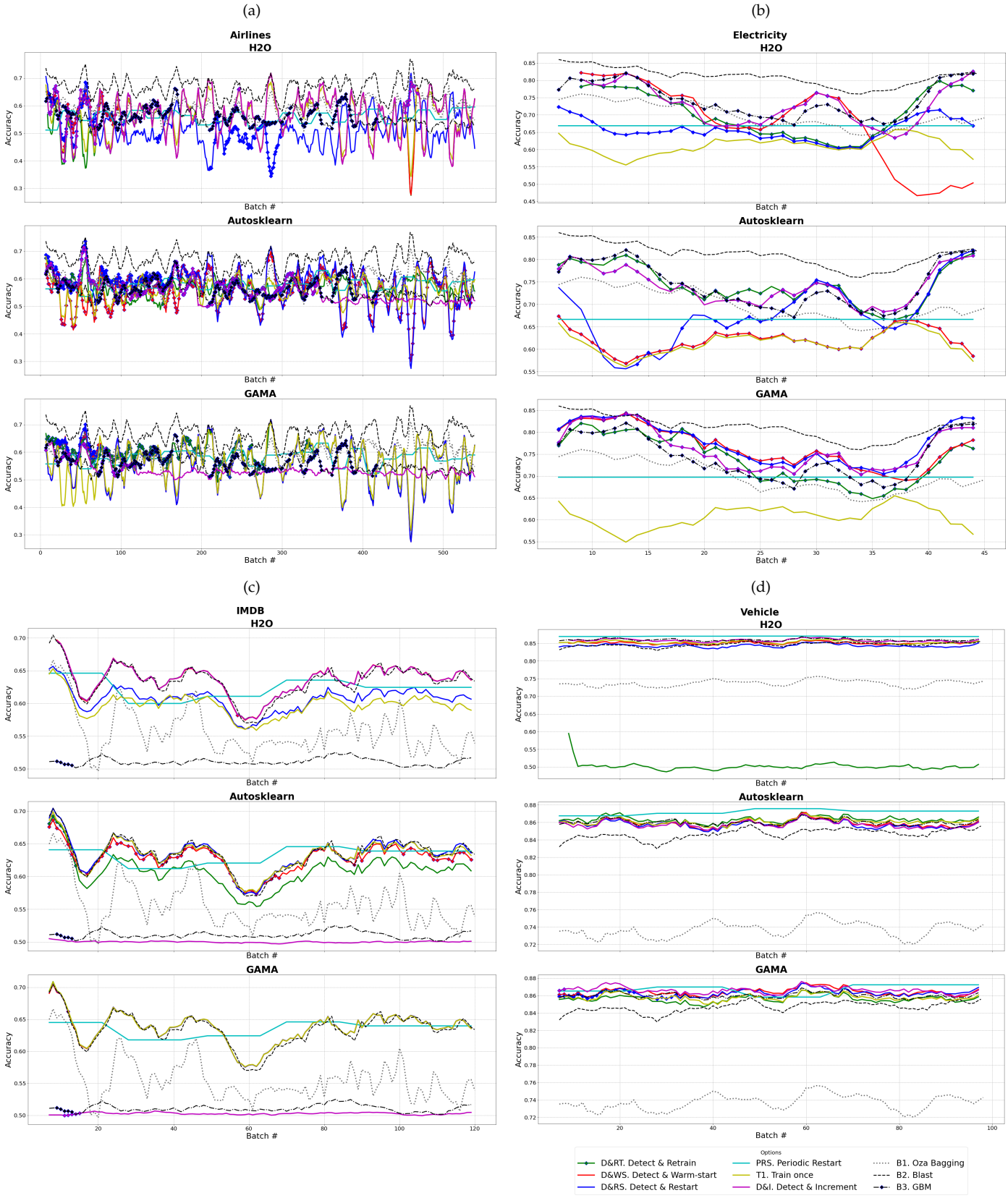

Fig. 4: Accuracy across batches for the real data streams: (a) Airlines; (b) Electricity; (c) IMDB; and, (d) Vehicle

Electricity (Fig. 4-b) is much smaller and has different kinds of drift, and hence we can observe more variation in the behavior of the different strategies and AutoML methods. Drift is detected with almost every batch. With GAMA, most strategies behave similarly as before. D&RT and D&I perform worse and D&WS or D&RS work generally well. AutoSKlearn and H2O have more difficulty with the smaller dataset, and re-optimizing the pipelines (D&WS or D&RS) often leads to worse results. PRS is a flat line because there aren't enough batches to warrant rerunning the AutoML techniques from scratch. Blast dominates the other methods here, GBM is mostly on par, Oza Bagging is much worse. The Electricity dataset is known to be heavily autocorrelated, which benefits Blast. On Vehicle, which is larger and had more gradual drift, both Blast and Oza Bagging are outperformed by the AutoML techniques and GBM. Note that few drifts are detected here, and hence the scores of the different strategies are very close. PRS works very well here, while H2O with D&RT performs badly, as we've seen earlier with high gradual drift.

6.3 Comparison of AutoML systems

In all, we found that the ideal adaptation strategy depends on the AutoML system and the characteristics of the data stream. On the real-world data streams (Fig 4), each AutoML system performs quite similarly when selecting the best strategy, but the best strategy differs between them. On large datasets, re-optimizing pipelines (e.g. D&WS or D&RS) works well for GAMA and Autosklearn. The cheaper retraining without re-optimization (D&RT) also works well on some datasets if regularly triggered to retrain. For H2O, D&RS is met with overfitting issues, but incremental learning (D&I) performs surprisingly well. On smaller datasets (e.g. Electricity), GAMA continues to re-optimize the pipelines well (D&WS and D&RS), while Autosklearn and H2O get better results by retraining the existing pipelines (D&RT and D&I). For data streams with abrupt concept drift, fast recovery is key. Restarting AutoML when abrupt drift is detected (D&RS) is a safe bet for GAMA and Autosklearn. GAMA can recover even faster when warm-starting the evolutionary search (D&WS), but this doesn't work with Autosklearn's Bayesian Optimization approach. This difference in recovery speed is bigger when the drift magnitude gets lower (Fig. 3). Similar behavior is observed for mixed drift (Fig. 2-c), whereas GAMA and Autosklearn perform quite similarly on gradual drift. H2O did not recover with D&RS in our experiments, which will be discussed in more detail next.

6.4 Interplay with drift detection

To understand the interplay between the adaptation strategy and the drift detector, Fig. 5-a shows the drift points (as vertical lines) for each adaptation strategy with GAMA on a mixed drift data stream. Re-optimizing the pipelines (D&RS) leads to more significant performance differences, which trigger the drift detector more often, in turn leading to more re-optimization. Warm-starting (D&WS) results in more subtle performance fluctuations and less frequent drift detector triggers. Other strategies mainly trigger the drift detector in the beginning and after abrupt drift.

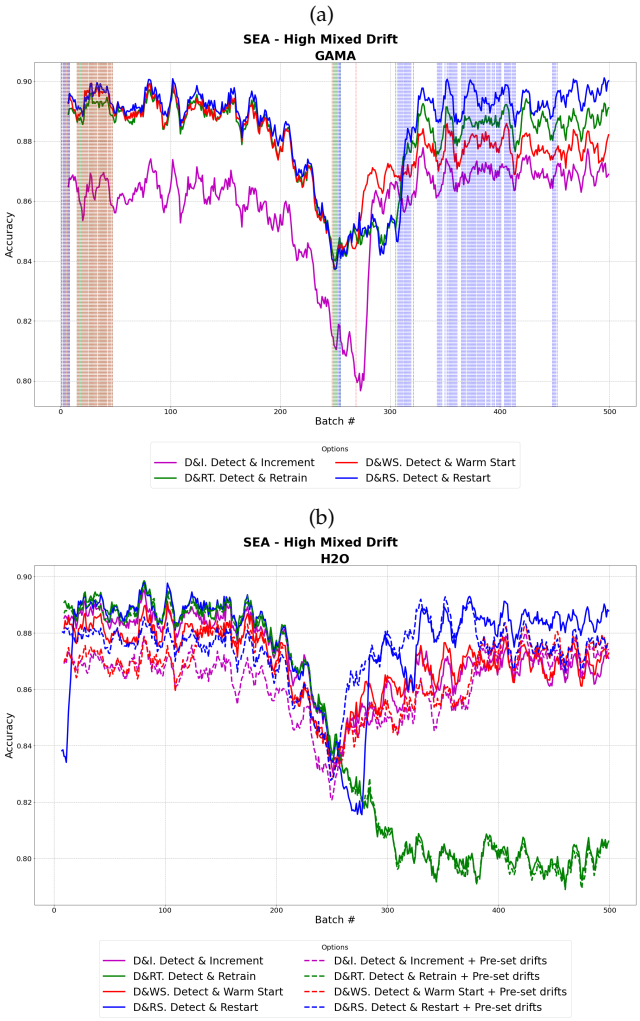

Fig. 5: Accuracy across batches on SEA - High mixed data stream for: (a) GAMA with detected drift points (vertical lines); and (b) H2O with detected and pre-set drifts

To evaluate the effect of the drift detector, we replaced it with five pre-set drift points: one after the abrupt drift and four after the mid-gradual drifts. Fig. 5-b compares the accuracy scores for H2O with the drift detector (full line) and the pre-sets (dashed). Although there are minor differences, e.g. faster recovery with D&RS, overall the relative performance of adaptation strategies does not change.

Finally, Fig. 6 marks the detected drift points (in red) for each library for the high abrupt drift data stream with D&RS. Drift is always detected after the sudden drift point at batch 250. Autosklearn and GAMA recover after the drift is detected and pipelines are re-optimized, but not H2O. Hence, a lack of drift detection is not the reason for H2O's performance after drift. Similar results were found for the abrupt and mixed drift data streams. This suggests that the linear model (GLM) used in H2O's stacking process is not adapting well to the changes in data distribution. Replacing this with a more adaptive stacker (e.g. GBM) may resolve this. H2O is also still a competitive option for gradual drift data streams with D&I chosen as the adaptation strategy.

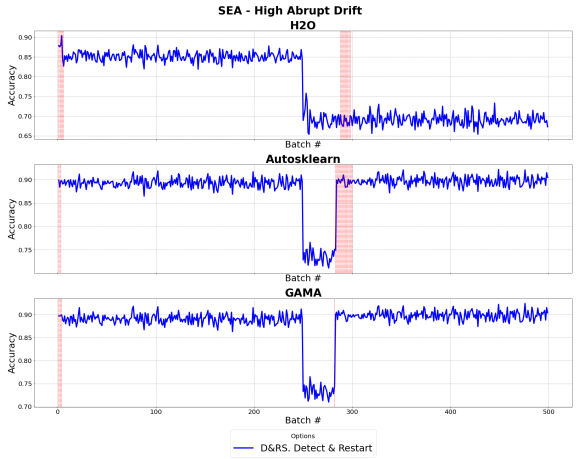

Fig. 6: Accuracy across batches with drift detection points for SEA - High abrupt data stream on D&RS strategy.

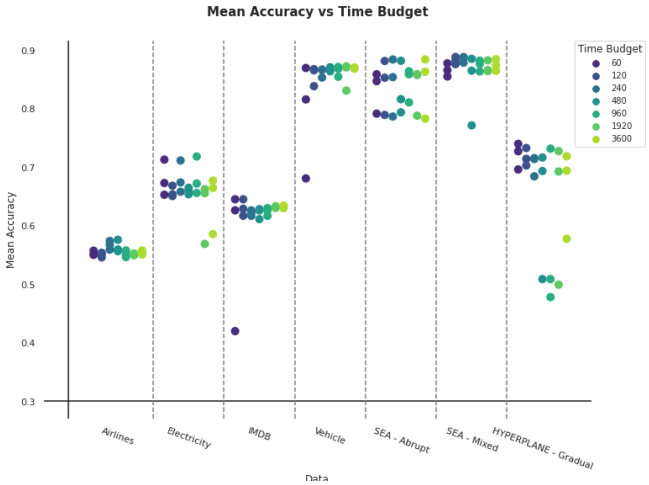

Fig. 7: Mean accuracy for all data streams and libraries with D&RS on time budgets between 60 and 3600 seconds.

6.5 Effect of the time budget

When we choose to rerun an AutoML technique (e.g. in D&RS and PRS), we can do so in parallel and keep using the previous pipelines until the new pipelines are optimized to new data. Still, we may want to limit the time budget given to the AutoML system so that our method reacts more quickly to concept drift. Fig. 7 shows the accuracy of all libraries with the D&RS strategy on each data stream, for time budgets varying from 60 sec to 3600 sec. Smaller budgets do affect performance, but overall most methods work well under stricter constraints. For data streams with a larger feature space (i.e. IMDB, Vehicle) the time budget has a more visible effect compared to artificial and lower-dimensional data streams. Blast, which trains multiple models in an ensemble, has similar accuracy and time performance. An advantage of the AutoML methods is that the time budget can be restricted without much loss in accuracy, which allows various online learning settings. An interesting research question would be to examine the relationship between the optimal time budget and the characteristics of the data stream.

7 Conclusions

The main goal of this study was to gain a deeper understanding on how current AutoML methods are affected by different types of concept drift, and how they could be adapted to become more robust. To this aim, we proposed six adaptation strategies that can be generally combined with AutoML techniques, integrated them with some of the most well-known AutoML approaches, and evaluated them on both artificial and real-world datasets with different types of concept drift.

We found that these strategies effectively allow AutoML techniques to recover from concept drift, and rival or outperform popular or online learning techniques such as Oza Bagging and BLAST. The comparisons between different AutoML techniques show that both Bayesian optimization and evolutionary approaches can be adapted to handle concept drift well, given an appropriate adaptation strategy and forgetting mechanism. Similar to previous studies, we find that different drift characteristics affect learning algorithms in different ways, and that different adaptation strategies may be needed to optimally deal with them.

On large datasets, re-optimizing pipelines after drift is detected works generally well. Evolutionary AutoML methods can so do even faster by evolving the previous best pipelines. Simply retraining pipelines on the most recent data after drift is detected, without re-optimizing the pipelines themselves, also works well if the concept drift is not too large and the pipelines are retrained frequently enough. Depending on the application, the additional computational time can be mitigated by decreasing the batch size, the optimization time budget, and warm-starting.

In all, this study contributes a set of promising adaptation strategies as well as an extensive empirical evaluation of these strategies, so that informed design choices can be made on how to adapt AutoML techniques to settings with evolving data. It also shows that there is ample room to improve existing AutoML systems, and even to design entirely new AutoML systems that naturally adapt to concept drift. We hope that this study will instigate further research into AutoML methods that adapt effortlessly to evolving data,

Acknowledgments

We would like to thank Erin Ledell, Matthias Feurer and Pieter Gijsbers for their advice on adapting their AutoML systems. This work is supported by the Dutch Science Foundation (NWO) grant DACCOMPLI (nr. 628.011.022).

References

[1] M. Baena-García, J. Campo-Ávila, R. Fidalgo-Merino, A. Bifet, R. Gavald, and R. Morales-Bueno, "Early drift detection method," in *Fourth International Workshop on Knowledge Discovery from Data Streams*, vol. 6, 2006, pp. 77–86.

[2] R. Bakirov, B. Gabrys, and D. Fay, "Generic adaptation strategies for automated machine learning," *ArXiv*, vol. 1812.10793, 2018.

[3] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 2546–2554.

[4] A. Bifet and R. Kirkby, "Data stream mining a practical approach," *J. Empirical Finance*, vol. 8, no. 3, p. 325–342, 2009.

[5] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases*, 2010, pp. 135–150.

[6] A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "MOA: A real-time analytics open source framework," in *Lecture Notes in Computer Science*, vol. 6913, 2011, pp. 617–620.

[7] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. abs/1012.2599, 2010.

[8] M. Carnein, H. Trautmann, A. Bifet, and B. Pfahringer, "Towards automated configuration of stream clustering algorithms," in *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019, pp. 137–143.

[9] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "Continual learning: A comparative study on how to defy forgetting in classification tasks," *arXiv preprint arXiv:1909.08383*, 2019.

[10] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, p. 71–80.

[11] M. Duarte and Y. H. Hu, "Vehicle classification in distributed sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, pp. 826–838, 2004.

[12] H. J. Escalante, W.-W. Tu, I. Guyon, D. L. Silver, E. Viegas, Y. Chen, W. Dai, and Q. Yang, "AutoML@NeurIPS 2018 challenge: Design & results," in *NeurIPS'18 Competition*. Springer, 2020, pp. 209–229.

[13] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, 2015, p. 2755–2763.

[14] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 10 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451

[15] J. Gama and P. Kosina, "Learning about the learning process," in *International Symposium on Intelligent Data Analysis*. Springer, 2011, pp. 162–172.

[16] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Compututer Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[17] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *In SBIA Brazilian Symposium on Artificial Intelligence*. Springer Verlag, 2004, pp. 286–295.

[18] R. Garnett, "Learning from data streams with concept drift," Ph.D. dissertation, University of Oxford, 2010.

[19] R. Garnett, M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts, "Sequential Bayesian prediction in the presence of changepoints & faults," *The Computer Journal*, vol. 53, no. 9, pp. 1430–1446, 2010.

[20] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An open source AutoML benchmark," *arXiv preprint arXiv:1907.00909*, 2019.

[21] P. Gijsbers and J. Vanschoren, "GAMA: Genetic automated machine learning assistant," *Journal of Open Source Software*, vol. 4, no. 33, p. 1132, 2019.

[22] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1487–1495.

[23] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017.

[24] H2O.ai, *h2o: Python Interface for H2O*, 2019, 3.24.0.1. [Online]. Available: https://github.com/h2oai/h2o-3

[25] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," The University of South Wales, Tech. Rep., 1999.

[26] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, p. 97–106.

[27] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automatic machine learning: methods, systems, challenges*. Springer, 2019.

[28] J. G. Madrid, H. J. Escalante, E. F. Morales, W. Tu, Y. Yu, L. Sun-Hosoya, I. Guyon, and M. Sebag, "Towards AutoML in the presence of drift: first results," *CoRR*, vol. abs/1907.10772, 2019.

[29] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, "Automating biomedical data science through tree-based pipeline optimization," in *Lecture Notes in Computer Science*, vol. 9597. Springer, 2016, pp. 123–137.

[30] N. C. Oza, "Online bagging and boosting," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2005, pp. 2340–2345 Vol. 3.

[31] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, p. 359–364.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *Proceedings of the 11th International Conference on Advances in Intelligent Data Analysis*. Springer-Verlag, 2012, p. 313–323.

[34] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2951–2959.

[35] W. Street and Y. Kim, "A streaming ensemble algorithm sea for large-scale classification," in *7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2001, pp. 377–382.

[36] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, p. 847–855.

[37] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Algorithm selection on data streams," in *Discovery Science*. Springer, 2014, pp. 325–336.

[38] J. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Having a blast : meta-learning and heterogeneous ensembles for data streams," in *15th IEEE International Conference on Data Mining*, 2016, pp. 1003–1008.

[39] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *ACM SIGKDD Explorations*, 2014.

[40] B. Veloso, J. Gama, and B. Malheiro, "Self hyper-parameter tuning for data streams," in *International Conference on Discovery Science*. Springer, 2018, pp. 241–255.

[41] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, p. 964–994, 2016.

[42] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Min. Knowl. Discov.*, vol. 32, no. 5, pp. 1179–1199, 2018.

[43] Q. Yao, M. Wang, H. J. Escalante, I. Guyon, Y.-Q. Hu, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, "Taking human out of learning applications: A survey on automated machine learning," *CoRR*, vol. abs/1810.13306, 2018.

Bilge Celik is a PhD researcher in the Department of Mathematics and Computer Science at Eindhoven University of Technology. She received her M.S. and B.A. degrees from Middle East Technical University. Her research interests include automated machine learning, data stream challenges and automation in adaptive learning.

Joaquin Vanschoren focuses his research on the progressive automation of machine learning. He founded and leads OpenML.org, an open science platform allowing scientists to share datasets and train many machine learning models from many software tools in a frictionless yet principled way. Subsequently, he uses this knowledge to create automatic machine learning (AutoML) techniques that learn from these experiments to help people build better models, faster, or automate the process entirely.