Highlights

Online Ensemble Model Compression for Nonstationary Data Stream Learning

Rodrigo G. F. Soares, Leandro L. Minku

- An online, real time compression method for neural networks that delivers similar performance to ensemble techniques at the time cost of a single model.
- An online neural network with improved predictive performance via weight averaging through time.
- An online neural network capable of finding stable regions in weight space through the search of wide minima.

Online Ensemble Model Compression for Nonstationary Data Stream Learning

Rodrigo G. F. Soares[a], Leandro L. Minku[b]

[a]*Department of Statistics and Informatics, Federal Rural University of Pernambuco, Rua Dom Manoel de Medeiros, s/n, Recife, 52171-900, Pernambuco, Brazil*
[b]*School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom*

Abstract

Learning from data streams that emerge from nonstationary environments has many real-world applications and poses various challenges. A key characteristic of such a task is the varying nature of the underlying data distributions over time (concept drifts). However, the most common type of data stream learning approach are ensemble approaches, which involve the training of multiple base learners. This can severely increase their computational cost, especially when the learners have to recover from concept drift, rendering them inadequate for applications with tight time and space constraints. In this work, we propose Online Weight Averaging (OWA) – a robust and fast online model compression method for nonstationary data streams based on stochastic weight averaging. It is the first online model compression for nonstationary data streams, which is capable of compressing an evolving ensemble of neural networks into a single model continuously over time. It combines several snapshots of a neural network over time by averaging its weights in specific time steps to find promising regions in the loss landscape with the ability to forget weights from outdated time steps when a concept drift occurs. In this way, at any point in time, a single neural network is maintained to represent a whole ensemble, leveraging the power of ensembles while being appropriate for applications with tight speed requirements. Our experiments show that this key advantage of our proposed method also translates into other advantages such as (1) significant savings in computational cost compared to state-of-the-art data stream ensemble methods while (2) delivering similar predictive performance.

Keywords: Online model compression, Nonstationary environments, Data stream learning, ensemble learning, neural networks

1. Introduction

The amount of data with potential relevance for real-world applications of Machine Learning (ML) has been growing exponentially in the last few years. Such ever-increasing volumes of data pose significant scalability challenges to offline ML, which requires the whole training set to be stored in memory and iterated through several times. This is especially the case for applications with tight time and space constraints such TinyML applications, which require real-time monitoring and data analytics in compact devices, and are beneficial in multiple domains such as agriculture, healthcare, conservation and environment, etc [1].

Online approaches able to learn additional training examples without requiring to reprocess all past training points are an effective alternative for solving the issue of training and predicting with high-speed data for this kind of application. However, data streams formed by incoming data in real-world applications are often nonstationary, with their underlying data distributions changing over time. This phenomenon is referred to as concept drift [2], and typically causes ML models to start underperforming. To maintain their predictive performance, learning models must be updated or even reset, depending on the severity and speed of the concept drift.

Ensemble algorithms have been widely used in data stream learning scenarios due to their ability to dynamically select appropriate learners according to the current concept and to potential drifts [3, 4] while reducing learning variance. However, since ensemble learning relies on the training of multiple learners, these techniques may be prohibitive for certain data stream learning scenarios [5], especially when there are strict time and space requirements. In this context, techniques able to compress ensembles would be desirable.

The offline learning literature has a number of methods proposed to compress large models, so that smaller models can be adopted without significant loss in predictive performance [6], especially in the area of deep learning [7, 8]. However, existing approaches operate by taking an existing large model (pre-trained) and then compressing it. In data stream learning, a model compression approach able to operate in online mode is necessary, such that compression occurs continuously over time and the learning system can maintain its models in compressed format at all times while being able to cope with concept drift.

In the context of ensemble learning, Fast Geometric Ensembling (FGE) [9] chooses multiple nearby points in the weight space to create neural network ensembles with high predictive performance in the time required to train a single neural network. Izmailov et al. [10] suggest that the weights of the networks combined by FGE are on the borders of the most desired region of the weight space. They proposed Stochastic Weighting Averaging (SWA) and showed that it was more promising to average the points in weight space and use them as network weights, instead of training an ensemble with averaged network outputs (in model space). SWA uses an equally weighted average of the points traversed by Stochastic Gradient Descent (SGD) to form the weights of a single network with higher predictive performance using virtually no computational overhead. SWA produces wider solutions in weight space than SGD, which leads to better generalization. However, SWA is not suited for nonstationary data stream learning, where concepts may change rapidly, rendering the averaged weights obsolete.

This paper aims to develop the first online model compression learning approaches for nonstationary data streams. We propose a more efficient stochastic weight averaging approach that can be used as an online model compression approach for nonstationary data streams in which any type of concept drift may occur. Each example is learned as soon as it arrives. Overall, we answer the following Research Questions (RQs):

- RQ1: *How to compress an ensemble of neural networks into a single neural network in online nonstationary data stream learning?* We propose a novel approach called Online Weight Averaging (OWA). This approach uses weight averaging over time as an online technique to obtain a high-performing single network with a simple drift detection procedure to update the model when a concept drift occurs. OWA is able to maintain a compressed ensemble of neural networks at all times in the form of a single neural network, rather than relying on first growing an ensemble and then compressing it. We analyze the landscape of the weights during stable periods and during concept drifts to show that the optimization path of the proposed approach is able to recover from concept drifts.

- RQ2: *How does the proposed approach perform compared with existing ensembles of neural neural networks for nonstationary data streams, and the existing weight averaging method?* As the proposed approach is compressing an ensemble, it should ideally achieve predictive performance that is no worse than that of ensembles themselves. Moreover, as the existing weight averaging method SWA is not suited for nonstationary data streams, the proposed approach should ideally outperform SWA. We compare the proposed approach to existing ensembles of neural networks for nonstationary environments and SWA in experiments with 20 artificial and 10 real-world data streams. The results confirm that the proposed approach is able to obtain superior overall predictive performance compared to SWA, especially during concept drifts, while maintaining similar predictive performance to existing data stream learning techniques. Our experiments also show that OWA is robust to different types of concept drifts.

- RQ3: *How does the proposed approach perform in terms of computational time compared to existing ensembles of neural networks for nonstationary environments, and the existing weight averaging method?* The main motivation for the proposed compression approach is to reduce the computational time of ensembles. Therefore, the proposed approach should ideally obtain lower computational time than ensembles. Meantime, whilst the existing weight averaging method SWA is unprepared for concept drifts, it is able to compress ensembles of neural networks, reducing their computational cost. Therefore, the proposed approach should ideally have a computational cost that is no larger than that of SWA. Our experiments with 20 artificial and 10 real world data streams show that our proposed approach was able to obtain computational cost from 1.05 to 130 times lower than that of existing ensemble methods, and similar to that of SWA.

The remainder of this paper is organized as follows. Section 2 presents our data stream learning scenario. Section 3 presents the state-of-the-art of stream learning and model compression. Section 4 introduces our approach to data stream learning. Our experiment design is introduced in Section 5. Section 6 presents an analysis of the trajectory of the weights learned by the proposed approach as a proof of concept for its ability to produce wider solutions in the weight space in non-stationary environments. Together with the proposed approach itself, this answers RQ1. This section also presents the analysis of predictive performance to answer RQ2, and analysis of computational cost and time complexity to answer RQ3, and an analysis of sensitivity to hyperparameters. Section 7 presents our conclusions and future work.

2. Problem formulation

We consider that examples arrive in the form of a data stream, which is a sequence of examples $S = \big(s^{(1)}, \ldots, s^{(t)}, \ldots\big)$, where $s^{(t)} = (\mathbf{x}^{(t)}, y^{(t)})$, $\mathbf{x}^{(t)} \in \mathbb{R}^D$, D is the dimensionality of the input space, and $y^{(t)} \in \{y_1, \ldots, y_K\}$ denote the labels. Each value of t is referred to as a time step. We focus on strict online learning algorithms, which update predictive models whenever a new training example becomes available and discard it right after being learned [11]. This is different from chunk-based learning, where incoming examples arrive in or are stored into chunks/batches, and each chunk can be iterated through multiple times for learning [3]. Strict online learning can act as an enabler for learning in applications with high-speed data streams or with strict time requirements [3], where it is not possible to iterate through examples more than once.

In our learning scenario, we also assume that the model will have a warm-start, that is, there will be a limited amount of data available to tune the models. Thus, the early predictions would be more accurate. This is a different scenario from [12], which assumes a cold-start in the learning process. Warm-start is the typical scenario for applications that provide previous historical data to tune the model before the online learning process starts. For example, in an application for personalized recommendations in Internet-of-Things devices, an initial set of examples collected from different people could be used for warm-start before the model starts to learn a specific user's preferences.

Each labeled example $s^{(t)}$ from the data stream is drawn from an unknown distribution $\mathcal{P}^{(t)}$. A concept drift is a change in the joint data distribution, that is, $p^{(t+1)}(\mathbf{x}, y)$ might differ from $p^{(t)}(\mathbf{x}, y)$ within the length of a stream. When concept drifts occur in a data stream, we say that the data stream is operating in a nonstationary environment, which is the focus of this work. Concept drifts can be categorized according to the rate at which a new concept becomes the true target. In abrupt concept drifts, the change occurs instantly (in one time step). In gradual concept drifts, the new concept takes over slowly over time. Data stream learning algorithms need to adapt to such concept drifts to avoid poor predictive performance over time [11].

Assume that the parameters of a compressed model being learned are $\mathbf{w} = (w_1, \cdots, w_d)$, where d represents the size of the parameter vector. At each time step t, our problem consists of using $\mathbf{w}^{(t-1)}$ and s^t to learn a compressed model $\mathbf{w}^{(t)}$ capable of representing an ensemble with $m^t \cdot d$ parameters, $m^{(t)} >$

1, without having to use s^t to update each of these m^t models separately.

3. Related work

As our proposed model compression approach compresses an ensemble of neural networks into a single neural network, we discuss both related work on single neural networks for nonstationary environments, ensemble methods for nonstationary environments, and compression methods. Data stream learning algorithms for nonstationary environments are typically categorized into explicit or implicit. Explicit methods use explicit drift detection mechanisms to activate an adaptation procedure. Implicit methods are continuously updated over time without explicit drift detection. Explicit algorithms are often the most effective techniques for abrupt drifts, whereas implicit algorithms often deliver better generalization for gradual drifts [2]. We will make use of this terminology when discussing the approaches.

3.1. Single neural networks for nonstationary data streams

There are a few studies on neural networks for data stream learning that use evolving architectures to deal with concept drift without the use of pre-tuned hyperparameters. Autonomous Deep Learning (ADL) [13] is a explicit deep neural network for chunk-based continual learning that grows and prunes its architecture according to the network significance metric. The NADINE approach [14] tackles chunk-based continual learning by dynamically evolving the depth and width of a MultiLayer Perceptron (MLP) structure to quickly react to concept drifts. It is also an explicit approach that employs network significance to trigger network changes. MUSE-RNN is an explicit chunk-based recurrent neural network that grows and prunes hidden nodes on demand. Network significance also drives the changes in its architecture. These techniques focus on chunk-based learning, that is, they do not deal with strict online learning, in which only a single instance arrives at each time step.

The authors in [15] proposed a one-pass model to tackle online learning without any assumption about the data distribution. However, they assume a linear model to achieve certain theoretical learning properties. While a linear model could be seen as a neural network with no hidden layer, their approach is not applicable to neural networks with one or more hidden layers as its formulation is based on a linear classifier. The Multiclass Imbalanced and Concept Drift Network Traffic Classification Framework based on Online Active Learning (MICFOAL) [16] can tackle classification problems by making use of an uncertain label request strategy based on the variable least confidence threshold vector.

3.2. Ensemble learning for nonstationary data streams

Ensemble learning has been widely employed in data stream learning [3, 4]. Their ability to create strong learners for stable concepts while being able to create new models to learn new concepts and discard past obsolete models [17, 18] has made them attractive approaches to handle concept drift.

The Adaptive Random Forest (ARF) [19] is an online ensemble method for data stream learning, it is based on using a drift detection mechanism for each base tree to monitor warnings and drifts. It trains new trees in the background in case of a warning before replacing them when a drift is detected. In [20], ARF received a feature extraction step to reduce the dimensionality of the data before learning and improve generalization.

Online Bagging (OzaBag) is the online version of the Bagging ensemble [21]. Instead of sampling with replacement, it assigns each a weight to each example according to a Poisson probability distribution, so that the models can be updated with each example separately upon arrival. OzaBag can be enhanced with an explicit drift detection method such as Adaptive Windowing (ADWIN) [22], so that the ensemble can be reset when a concept drift occurs. Diversity for Dealing with Drift (DDD) [23], Diversity Pool (DP) [24] and Concept Drift Handling Based on Clustering in the Model Space (CDCMS) [25] are explicit online ensemble approaches that employ diversity to select and train base learners, improving robustness to different types of concept drift. Recurring Concept Drift (RCD) [26] is an explicit ensemble approach that employs a statistical test and memory management to identify recurring concepts. Kappa Updated Ensemble (KUE) [27] is an ensemble that combines chunk-based and online learning, it employs Kappa statistic for dynamic weighting and selection of base learners. It train base learners following a Poisson distribution and encourages diversity among them by using random feature subsets. The Streaming Random Patches classifier (SRPC) is an online ensemble algorithm that is able to tackle concept drifts by combining random subspaces and bagging, and by using an explicit drift detection mechanism [5]. SRPC can be interpreted as an adaptation of batch learning ensemble methods that combines random samples of examples and random subspaces of features.

Dynamic Weighted Majority (DWM) [17] is an implicit online approach that adds and removes base learners according to global and local performance on the stream. To perform such updates, DWM uses four mechanisms to handle concept drifts: trains its online base learners, weights base learners according to their performance, it removes them, also based on their performance, and it adds new experts based on the global predictive performance of the ensemble. Online Accuracy Update Ensemble (OAUE) [18] is an implicit ensemble approach that maintains a weighted majority vote ensemble with a pruning mechanism. The Cost-Sensitive Boosting (CSB2) online algorithm combines mechanisms from AdaBoost and AdaC2 [28]. When the ensemble correctly classifies a test example, CSB2 updates weights as in AdaBoost. Otherwise, it updates weights as in AdaC2 as an attempt to deal with concept drifts that may have occurred.

Except for ARF ans CSARF, all of the ensemble approaches above can be adopted with any type of online base learner. While these ensembles produce high generalization accuracy, they have the computational burden of training multiple base learners, which is aggravated by the amount of data that must be processed in typical data stream applications. Typically, both training and prediction time required by these methods grow with the number of base learners. This can make their computational cost prohibitive for high-speed data streams or applications with strict time requirements.

The Comprehensive Active Learning Method for Multiclass Imbalanced Streaming data with concept drift (CALMID) [29] is an ensemble that is based on an asymmetric margin threshold matrix, and a class imbalance evaluation method under an online active learning strategy. Authors could show that it is effective and efficient on general classification tasks.

3.3. Model Compression

Model compression techniques have been widely applied to offline learning, more particularly, to deep learning for reducing the computational burden of large neural networks [7, 8]. With deeper networks, model size, inference latency and energy cost have become critical issues. Therefore, model compression is becoming increasingly demanded and studied [30, 31, 6]. However, these approaches operate by taking an existing large model (pre-trained) and then compressing it.

A few online compression techniques were proposed to handle this train-and-compress issue [32, 33]. For example, distillation algorithms have been developed to perform knowledge online teaching in a one-phase training procedure. However, these techniques are designed for the sole purpose of reducing the computational cost of the compression [34]. They were designed for offline problems. In [35], a student-teacher online approach was proposed for dealing with time-series analysis. Such algorithm uses a compact model to approximate the function learned by more complex and highly performing model. However, this approach still has to maintain and to train the original larger and slower model, which limits the potential gains in computational time provided by compressing the model. In data stream learning, a model compression approach able to operate in online mode at the same time as the model being compressed is being trained is necessary. This is so that compression occurs continuously over time and the learning system can maintain its models in compressed format at all times to achieve computational efficiency while being able to cope with concept drift.

In the context of ensemble learning, Garipov et al. [9] and Izmailov et al. [10] showed that local optima produced by Stochastic Gradient Descent (SGD) can be connected by simple curves of near constant loss. FGE [9] is a neural network ensemble technique that uses a cyclical learning rate to combine models that are spatially close to each other and produce diverse predictions. These base models are used to train ensembles with no computational overhead when compared to single neural networks. SWA [10] is a technique that averages multiple points in weight space along the trajectory of SGD with cyclical or constant learning rates. The solutions provided by SGD are approximately sampled from the loss surface of the neural network, leading to stochastic weights that are, in turn, stochasticaly averaged by SWA. Such weight averaging can be seen as an ensemble compression technique and was able to improve the predictive performance of a number of network architectures over several benchmarks with no significant increase in time complexity. That is, SWA is able to achieve ensemble-like performance without the time overhead since it effectively employs only a single network. The reason for this performance is that averaging weights leads to solutions that are wider than the optima found by SGD. SGD generally converges to a point near the boundary of the wide flat region of optimal points. SWA, on the other hand, is able to find a point centered in this region, often with slightly worse train loss but with substantially better generalization [10].

Neither FGE nor SWA are suitable for data stream learning, especially in the presence of concept drifts. We show in Section 6.1 that, when a concept drift occurs, SWA is no longer able to find flat regions in weight space. The stochastic selection of points in weight space performed by SWA becomes obsolete and the algorithm is unable to recover from the changes in that space, rendering the model with a low generalization performance. A novel approach able reduce computational time while maintaining state-of-the-art predictive performance in nonstationary data streams is necessary.

4. Online Weight Averaging (OWA)

The proposed method consists of a single MultiLayer Perceptron (MLP) with weights $\mathbf{w}$ and an averaged vector of weights $\boldsymbol{\mu}$. The weights $\mathbf{w}$ correspond to a neural network that is being trained on the current concept, whereas the averaged vector of weights $\boldsymbol{\mu}$ represents the compressed version of an ensemble composed of MLPs previously trained on this concept. The latter weights $\boldsymbol{\mu}$ are used as the weights of a MLP for prediction purposes. The former weights $\mathbf{w}$ are updated continuously with new examples to reflect the current concept and are periodically incorporated into $\boldsymbol{\mu}$ through weight averaging. Such periodic incorporation enables $\boldsymbol{\mu}$ to become a stronger model through the incorporation of knowledge from additional models trained on the current concept. A simple drift detection method that does not cause computational overhead is used to handle concept drift. When this method triggers a drift detection, the averaged vector of weights $\boldsymbol{\mu}$ is reset to the value of the current $\mathbf{w}$, so that it immediately reflects only the most recent MLP model. OWA as proposed in this section and its analysis performed in Section 6.1 are targeted at answering RQ1.

Algorithm 1 depicts OWA, where the function f represents an MLP network. Learning is performed while there are examples arriving in the data stream (Line 6). Before θ examples have arrived in the data stream, there is a single MLP f with weights $\mathbf{w}^{(t)}$. OWA's MLP $\mathbf{w}$ is trained with SGD in this work (Line 12). Other online training algorithms can be used. When used for neural networks in data stream learning, SGD provides, at each time step t, a solution in the weight space that represents the current concept as in Equation 1, where $L(\mathbf{x}^t, y^t)$ is a loss function and η is the learning rate:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial L(\mathbf{x}^{(t)}, y^{(t)})}{\partial \mathbf{w}^{(t)}}. \tag{1}$$

Note that the application of this equation at each iteration of the while loop corresponds to the optimization process in stochastic gradient descent being applied over time, to learn each example from the data stream.

Different from existing work adopting SGD, the learning rate η used in OWA can and should be relatively large for the following reasons: (1) it enables more diverse models to be incorporated into the compressed ensemble represented by $\boldsymbol{\mu}$, making it stronger during periods of stable concepts; (2) it enables faster adaptation to concept drift, due to the stronger emphasis that it places on the current concept. Such benefits are an advantage of OWA over standard online neural networks, for which small learning rates would hinder adaptation to concept drifts while large learning rates would hinder predictive performance during stable concepts, preventing them from being successful in nonstationary data streams.

Once more than θ examples have been received, OWA starts averaging points in the weight space to produce an effective model for periods of stability (Lines 22-26), where each point $\mathbf{w}^{(t)}$ incorporated into the average corresponds to an MLP model that was being learned at a different point in time. In particular, at every C time steps, the current $\mathbf{w}^{(t)}$ is incorporated into the average of weights $\boldsymbol{\mu}$ as described in Equation 2:

$$\boldsymbol{\mu}^{(t+1)} = \frac{m^{(t)}\boldsymbol{\mu}^{(t)} + \mathbf{w}^{(t)}}{m^{(t)} + 1}. \tag{2}$$

where $m^{(t)} = i/C$ is the number of models learned in the current concept and i estimates the length of time covered by the current concept so far by counting the number of time steps since the last drift detection (or since the beginning of learning, when no concept drift has been detected yet). The length of the cycle C controls the rate at which the averaging of the network weights is performed, that is, it regulates the extent at which the MLP's optimization path is included in the current weight average. The averaging process to compose $\boldsymbol{\mu}$ starts only after the data stream produces a pre-defined number of examples θ, as OWA would not benefit from averaging undertrained weights.

When a concept drift occurs, the prequential accuracy naturally drops. This is detected by OWA with a light drift detection mechanism (Lines 14–21) that keeps track of the prequential accuracy $preq(\hat{y}, preq^{(t-1)})$, where $\hat{y}$ is the prediction for $\mathbf{x}^{(t)}$. If the performance drops consecutively for F time steps, this is likely to represent a concept drift. So, the average $\boldsymbol{\mu}$ is reset to the weights of the MLP $\mathbf{w}$ and the running averaging μ and $m^{(t)}$ restarts with the time step count $i = 1$ so that OWA can adapt to the new incoming concept (Lines 28–30). F is the maximum number of time steps where accuracy has been lower than the maximum accuracy, that is, it regulates the number of time steps the OWA is allowed to perform worse than its previous best accuracy before resetting the weight average. Drift detection is only performed from time step θ onwards, where θ is a predefined hyperparameter representing the initial training period.

5. Experiment Design

In this section, we present the experimental setup used to answer RQ1-3, to validate the ability of OWA to find local optima in broad regions of the train loss, and the usefulness of using a weight averaging procedure in data stream learning in terms of computational time and predictive performance.

5.1. Data streams

We employed artificial and real-world data streams. We used four different generators to obtain these artificial data streams. The Sine generator [36] produces streams that follow various forms of the sine function. The Agrawal generator [37] generates categorical and ordinal data to predict certain groupings of individuals. The SEA generator [38] uses numerical attributes to produce streams with different forms of hyperplanes as the correct decision boundaries. STAGGER [39] contains categorical attributes – size, color and shape – for the prediction of a simple binary function of a set of objects. Each generator contains a set of functions $\{r_i\}$, where each function represents a different concept. Different data streams can be produced by using different sequences of functions. The sequences used in this work are shown in Table 1, where each concept extends for 1000 time steps.

We produced different versions of these streams with different types of concept drift, including abrupt, gradual, recurrent, non-recurrent and different levels of severity. These streams were tailored to assess the adaptation mechanisms and efficiency of the classifiers. For each sequence shown in Table 1, we generated 2 artificial data streams – one with abrupt and one with gradual concept drifts. Abrupt drifts occur in one time step and gradual drifts take 100 time steps to complete. The streams Sine1, Sine2, Agrawal3, SEA1, SEA2, STAGGER1 and STAGGER2 possess recurrent concepts, as shown in the column containing the concept sequences. The different sequences of concepts also lead to drifts with different severities. The severities are shown in Table 2.

The severity of a drift is calculated as the percentage difference between the previous and the incoming concept as follows [25], based on one million

Algorithm 1 OWA algorithm

```
 1: function OWA($\theta, C, F$)
 2:     $t \leftarrow i \leftarrow 1$                          ▷ time step and length of current concept
 3:     $fails \leftarrow 0$                                  ▷ current number of fails
 4:     $bestacc \leftarrow -\infty$                          ▷ Best accuracy in the concept
 5:     $\boldsymbol{\mu} \leftarrow \mathbf{0}$
 6:     while $t <$ end of stream do
 7:         if $t <= \theta$ then
 8:             $\hat{y} \leftarrow f(\mathbf{x}^{(t)}, \mathbf{w})$
 9:         else
10:             $\hat{y} \leftarrow f(\mathbf{x}^{(t)}, \boldsymbol{\mu})$
11:         end if
12:         Update MLP weights using SGD (Eq. 1)
13:         if $t > \theta$ then
14:             $acc \leftarrow preq(\hat{y}, y^{(t)})$
15:             if $acc > bestacc$ then
16:                 $fails \leftarrow 0$
17:                 $bestacc \leftarrow acc$
18:             else
19:                 $fails \leftarrow fails + 1$
20:             end if
21:             if $fails < F$ then
22:                 if $\mod(i, C) = 0$ then
23:                     $m^{(t)} \leftarrow i/C$
24:                     Update OWA weights with Eq. 2.
25:                 end if
26:                 $i \leftarrow i + 1$                      ▷ Update the concept length
27:             else
28:                 Reset ensemble weights to $\mathbf{w}$
29:                 $\boldsymbol{\mu} \leftarrow \mathbf{w}$
30:                 $i \leftarrow 1$, $fails \leftarrow 0$, $acc \leftarrow -\infty$
31:             end if
32:         end if
33:         $t \leftarrow t + 1$
34:     end while
35: end function
```

Table 1: Summary of data streams.

Data Stream	Concept sequences	Number of instances	D	K; Class distribution
Artificial data streams				
Sine1	$r_3 \rightarrow r_4 \rightarrow r_3$	3000	4	2; uniform
Sine2	$r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_4 \rightarrow r_1$	5000	4	2; uniform
Agrawal1	$r_1 \rightarrow r_3 \rightarrow r_4 \rightarrow r_7 \rightarrow r_{10}$	20000	36	2; uniform
Agrawal2	$r_7 \rightarrow r_4 \rightarrow r_6 \rightarrow r_5 \rightarrow r_2 \rightarrow r_9$	24000	36	2; uniform
Agrawal3	$r_4 \rightarrow r_2 \rightarrow r_1 \rightarrow r_3 \rightarrow r_4$	20000	36	2; uniform
Agrawal4	$r_1 \rightarrow r_3 \rightarrow r_6 \rightarrow r_5 \rightarrow r_4$	20000	36	2; uniform
SEA1	$r_4 \rightarrow r_3 \rightarrow r_1 \rightarrow r_2 \rightarrow r_4$	5000	3	2; uniform
SEA2	$r_4 \rightarrow r_1 \rightarrow r_4 \rightarrow r_3 \rightarrow r_2$	5000	3	2; uniform
STAGGER1	$r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow r_2$	16000	7	2; uniform
STAGGER2	$r_2 \rightarrow r_3 \rightarrow r_1 \rightarrow r_2$	16000	7	2; uniform
Real-world data streams				
Electricity	–	27549	7	2; 58%:42%
NOAA	–	18159	8	2; 42%:58%
Chess	–	503	8	2; uniform
Keystroke	–	1600	10	4; uniform
Luxembourg	–	1901	31	2; uniform
Ozone	–	2534	72	2; 5%:95%
Power S. Day/Night	–	29928	2	2; uniform
Power S.	–	29928	24	24; uniform
Sensor	–	130073	5	2; uniform

randomly generated examples:

$$diff(r_a, r_b) = \frac{\sum_{i=1}^{n} |y^i_{r_a} - y^i_{r_b}|}{n}, \tag{3}$$

where $y^i_{f_a}$ and $y^i_{f_b}$ are the class labels determined by the a-th and b-th functions of a generator, respectively, and n is the total number of examples generated uniformly at random to calculate the severity. According to [25], a concept drift could be considered severe when the concepts before and after the drift have at least around 50% difference, and mild if the difference is around 25%.

We preprocessed all data streams by using one-hot encoding for categorical attributes and by transforming ordinal attributes into real-valued variables. All attributes were scaled into $[-1, 1]$ and all artificial streams consist of binary classifications.

To assess the performance of the algorithms in diverse real-world domains, we also used ten real-world data streams to evaluate classifiers, which are also summarized in Table 1. The Electricity Pricing Data (Electricity) [40] stream contains data from the electricity market. The NOAA stream [41] has decades of daily weather measurements, such as temperature, pressure,

Table 2: Severity of Drifts as the Percentage Difference Between the Old and New Concepts. Adjusted from: [25]

	Sine			SEA				STAGGER	
	r1	r2	r3	r1	r2	r3	r4	r1	r2
r2	100.0%	-	-	8.5%	-	-	-	59.3%	-
r3	26.8%	73.2%	-	7.4%	16.0%	-	-	77.8%	48.1%
r4	73.2%	26.8%	100.0%	13.1%	4.6%	20.6%	-	-	-
r5	-	-	-	23.9%	32.5%	16.5%	37.1%	-	-

	Agrawal								
	r1	r2	r3	r4	r5	r6	r7	r8	r9
r2	53.9%	-	-	-	-	-	-	-	-
r3	53.1%	50.8%	-	-	-	-	-	-	-
r4	53.9%	20.5%	50.8%	-	-	-	-	-	-
r5	53.4%	47.6%	50.7%	47.7%	-	-	-	-	-
r6	69.9%	28.9%	51.2%	35.5%	48.1%	-	-	-	-
r7	50.5%	53.3%	50.1%	53.5%	60.1%	57.2%	-	-	-
r8	33.5%	60.4%	46.5%	59.6%	59.6%	59.8%	49.8%	-	-
r9	50.4%	53.3%	50.2%	53.5%	59.9%	57.3%	6.0%	49.5%	-
r10	32.9%	61.3%	46.5%	61.3%	60.0%	59.9%	51.1%	1.8%	51.1%

All percentage differences were calculated using Eq. 3 based on one million random generated examples.

visibility, and wind speed. The task is to predict whether or not it will rain in a given day. The Chess data stream [42] consists of chess game records from *chess.com* from December 2007 to March 2010 with seven attributes. Each player has a rating, which changes over time. Potential sources of concept drift are changes in the player's skills over time, participation in various types of tournaments and competitions. The Keystroke data stream [43] is a subset of a larger dataset [44], which is formed of recordings of the typing rhythm from 51 users typing a certain password and the *enter* key 400 times over eight sessions spread along different days. In the Keystroke data stream, the typing rhythm is used to predict four different users. Ten attributes were extracted from the flight time for each pressed key (the time difference between the instants when a key is released, and the next key is pressed). This data stream incrementally evolves as the users practice. The Luxembourg data stream [42] is based on the European Social Survey from 2002 to 2007. Each example represents a subject and his/her attributes are based on the answers to a survey questionnaire and 1,901 examples collected over five years. This classification aims to predict whether his/her internet usage is high or low. Since internet usage changes over time, this data stream may present concept drifts. The Ozone data stream [45] consists of air readings collected from 1998 to 2004 at the Houston, Galveston and Brazoria areas. This learning task aims to predict the ozone level (ozone day and normal day) eight hours ahead of time. The Power Supply stream [46] contains three years of hourly measurements of electricity supply. The task consists of identifying at which hour of the day a given power supply has been recorded. The Power Supply Day/Night is its binary classification version, we grouped the 24 classes (hours) into two classes: day and night. The Sensor stream [46] contains data of temperature, humidity, light and voltage collected from sensors in a research laboratory. The task is to identify the sensor that produced a given measurement. We considered the two largest classes in this stream: sensors 29 and 31.

5.2. Experimental setup

We answer RQ2 and RQ3 by comparing OWA against six ensemble learning approaches that are also strict online learning approaches able to adopt neural networks as base learners:

- Ozabag [21] – This is the baseline for our proposed approach and has been chosen due to its popularity in online data stream learning for stationary environments. To have a successful predictive performance, OWA should perform at least as well as the Ozabag ensemble. Since this method consumes very little computational time beyond the base learners' training, it will also be a good benchmark for efficiency.

- Ozabag-Adwin [21, 22] – We also employ a drift detection mechanism (ADWIN) to consider the impact that an explicit warning of a drift causes to the performance of our baseline (Ozabag).
- DWM [17] – This is a well established ensemble method dynamically adapts its structure to a new incoming concept. DWM handles concept drift by creating and removing weighted base learners in response to changes in performance calculated at each time step. This method will allow the comparison of OWA to an ensemble that changes over time and can adapt to new concepts using only a few base learners at a given time step.
- CSB2 [28] – This ensemble approach is based on AdaBoost and AdaC2; and uses the ADWIN change detector. We aim to compare OWA to a AdaBoost-based technique for data stream learning in nonstationary environments.
- SRPC [5] – This ensemble approach is based on bagging and random subspaces. It has been chosen for being a state-of-the-art approach that can be used as a benchmark of predictive performance in data stream learning. We expect OWA to deliver at least a comparable predictive performance to SRPC while requiring less computational time.
- KUE [27] – This ensemble technique is based on Kappa statistic to update its base learners. We expect to perform similarly to KUE with less computational time.
- CALMID [29] – This ensemble algorithm uses a asymmetric margin threshold matrix to tackle imbalanced data. We expect OWA to perform similarly to CALMID on balanced data streams and worse on streams with imbalanced classes.
- MICFOAL [16] – This technique uses an uncertain label request strategy based on the variable least confidence threshold vector. We expect

MICFOAL to perform similarly to CALMID when compared to OWA.

- SWA [10] – We aim to assess the predictive performance of our approach when compared to SWA, which is an approach based on weight averaging that was not designed for data stream learning. It is not prepared to cope with concept drift, even though it is a strict online learning method when SGD is applied with a single epoch.

Izmailov et al. [10] studied SWA with two learning rate schedules: high and constant; and cyclical. The design of a cyclical learning rate schedule for data stream learning depends on the correct match (sync) of the learning rate and the incoming of a new concept. Such a design is challenging because a concept drift can occur at any given moment in time. That is, such a change can occur when the learning rate is low, which would hinder the learning of a new concept. Therefore, we employed a more aggressive constant learning rate schedule as it generally leads to faster convergence [10] and OWA needs to recover rapidly from concept drifts, as explained in Section 4.

To perform the hyperparameter tuning of each method, we extracted the initial 10% of each data stream to form a validation stream. We performed a random search with 200 iterations on the validation streams [47] following the probability distributions in Table 3. The predictive performances of the selected models were evaluated on the remainder of the streams with the prequential G-Mean [48] measured over 30 replicates. G-Mean is appropriate for both balanced and imbalanced datasets as it accounts for the model's accuracy on each class k. For time step t, it is defined as

$$gmean^{(t)} = \left(\prod_k sens_k^{(t)} \right)^{\frac{1}{K}}, \tag{4}$$

where

$$sens_k^{(t)} = \begin{cases} \frac{ps_k^{(t)}}{pn_k^{(t)}}, & \text{if } y^{(t)} = k, \\ sens_k^{(t-1)}, & \text{otherwise.} \end{cases}$$

$sens_k^{(t)}$ is the prequential sensitivity for class k, $ps_k^{(t)} = acc_k + fading * ps_k^{(t-1)}$, $pn_k^{(t)} = 1 + fading * pn_k^{(t-1)}$, $acc = 1$ if the model correctly classified instance $s^{(t)}$ as class k and 0 otherwise, and *fading* is the fading factor fixed at 0.999.

For a fair comparison able to assess the impact of the various ensemble approaches themselves, we employed the same MLP model as base learner to all compared techniques. Following Demsar [49]'s recommendations on comparisons of multiple classifiers over multiple datasets, we used the Friedman statistical test followed by the Nemenyi post-hoc test with significance level of 0.05. One Nemenyi test was conducted to compare the approaches across all artificial data streams, one was conducted to compare the approaches across all real world data streams and one was performed to compare all methods across all streams.

6. Analyses

6.1. Optimization paths and solution widths through concept drifts

According to [50, 51], the generalization produced by weights in a local optimum is influenced by the width of that solution, where a large width refers to solutions that have a wide neighborhood of other weights whose predictive performance is similar to that of the solution, whereas a small width refers to solutions whose neighbors have considerably poorer predictive performance than that of the solution. The reason for this relationship is that the surfaces of train loss and test error have an offset between them. Therefore, it is useful to find a local optimum in a broad region of the train loss so that the test error has a higher probability to be found in that same high-performing region. Such wide solutions stay approximately optimal in the presence of small perturbations. SWA is able to find such broad regions, that accommodate local optima for both train loss and test error, and produce good predictive performance for stationary classification scenarios [10].

However, such a characteristic does not hold for data stream learning. As SWA maintains its averaged weights since the start of the stream, it becomes obsolete when a concept drift occurs. That is, SWA is no longer able to produce informative broad local optima after a concept drift. We highlight this fact in Figure 1, where we use random projections of weights to visualize the prequential accuracy landscape [52] and the optimization path of SWA when a concept drift occurs. The Sine data stream is described in Section 5.1. Both OWA and SWA use the same MLP in the figures of this section. All hyperparameters, including the constant learning rate, were tuned according to Section 5.2. We depict a gradual drift that starts at time step 1000. Figure 1a shows that SWA was able to correctly find a wide optimum for the current concept, which ranges from time step 1 to 1000. A

Table 3: Table of hyperparameter ranges for random search.

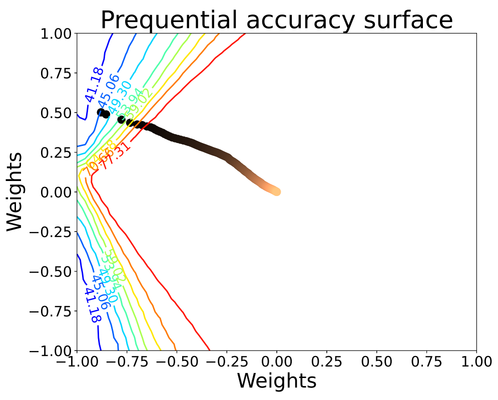
Hyperparameter	Probability distributions for randomized search
Ozabag	
ensemble size	uniform in $[1, 50]$
Ozabag-Adwin	
ensemble size	uniform in $[1, 50]$
DWM	
ensemble size	discrete uniform in $[1, 50]$
β	uniform in $[1^{-5}, 1^{0}]$
θ	uniform in $[1^{-5}, 1^{-1}]$
period	discrete uniform in $[1, 100]$
SRPC	
ensemble size	discrete uniform in $[1, 50]$
subspace size	discrete uniform in $[1, 20]$
training method	discrete uniform in {random subspaces, resampling, random patches}
λ	uniform in $[1^{-3}, 10]$
CSB2	
ensemble size	discrete uniform in $[1, 50]$
positive cost	uniform in $[0, 1]$
negative cost	uniform in $[0, 1]$
KUE	
ensemble size	discrete uniform in $[1, 50]$
CALMID	
ensemble size	discrete uniform in $[1, 50]$
weight shrink	discrete uniform in $[1, 1000]$
active threshold	uniform in $[0, 1]$
active budget	uniform in $[0, 1]$
random ratio	uniform in $[0, 1]$
window size	discrete uniform in $[1, 500]$
MICFOAL	
active threshold	uniform in $[0, 1]$
active budget	uniform in $[0, 1]$
random ratio	uniform in $[0, 1]$
low F1 score	uniform in $[0, 1]$
high F1 score	uniform in $[0, 1]$
SWA	
start	discrete uniform in $[1, 100]$
OWA	
cycle	discrete uniform in $[1, 500]$
maximum fails	discrete uniform in $[1, 100]$
start	discrete uniform in $[1, 100]$

gradual concept drift starts at time step 1001 and has a length of 100 time steps. In Figure 1b, the weight solution starts to deteriorate as the solutions are found near the borders of a wide optimum. This situation is aggravated 50 time steps later in Figure 1c, where the path shows solutions in sub-optimal regions. Figure 1d shows that, even after 100 time steps after the end of the concept drift, solutions are not produced in wide optimal regions. That is, SWA could not recover from the concept drift within 100 time steps. This fact might indicate that keeping the obsolete weight solutions in the current averaging negatively affects current solutions.

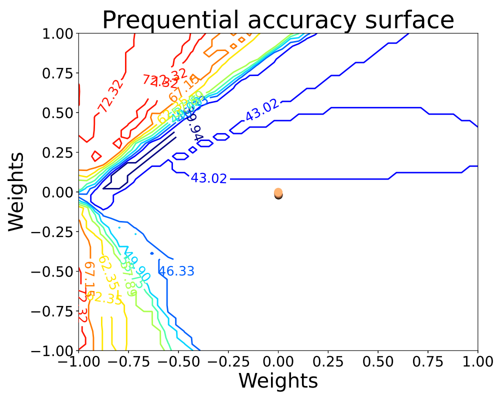
In contrast, the proposed method is able to recover and find wide optimum shortly after a concept drift occurs. This fact is shown in Figure 2, where we depict random projections of the prequential accuracy surface and the optimization path of OWA during a gradual concept drift that ranges from time steps 1001 to 1100. In Figure 2a, OWA could find a wide optimum as SWA, which leads to better generalization performance [10]. After 50 time steps within the drift, OWA no longer produces solutions in middle of wide regions (Figure 2b). However, 100 time steps later, in Figure 2c, the drift ends and OWA finds again a solution in wide optimum region due to its drift-handling mechanism. In Figure 2d, we show that, after 100 time steps after the drift end, OWA could completely recover from the drift and find a good weight solution in the middle of a wider optimum in the prequential accuracy surface when compared to SWA.

In Figure 3, we plot the projection of the optimization path of SWA before and 50, 100 and 200 time steps after an abrupt drift (it has a length of one time step). This is a more severe change, as a new concept arrives suddenly and the previous one becomes totally absent after a single time step. Before this concept drift, SWA could find a wide optimum in weight space. However, when a sudden change occurs (concepts are switched in a single step), the weight average becomes outdated, which start drastically and negatively affecting the quality of the current solution (Figure 3b). After 100 time steps (Figure 3c), SWA cannot adapt to the new concept. Even after 200 time steps (Figure 3d), SWA still could not produce weights in potentially interesting regions. This degradation of the optimization is aggravated when many concept drifts took place as SWA does not have mechanisms to forget the weights found for previous concepts. That is, SWA is incrementally and negatively affected by past concepts.

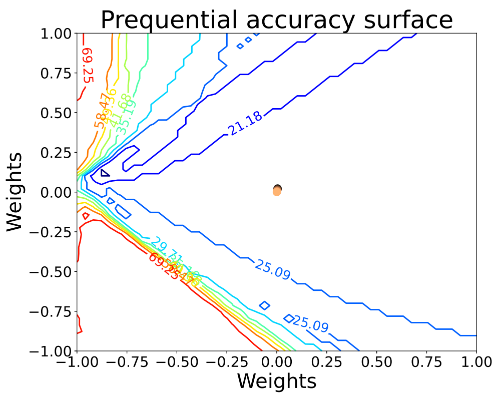
On the other hand, OWA is able to correctly learn the current concept via its cyclical optimization process and to rapidly adapt to sudden changes



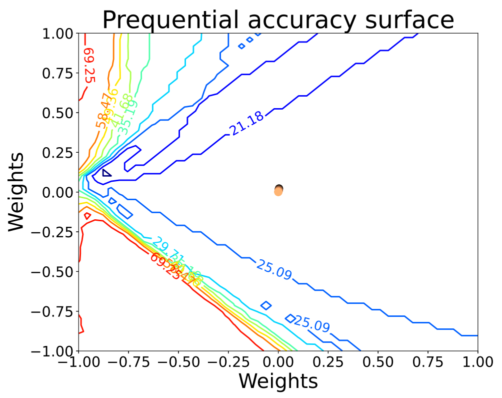
(a)



(b)

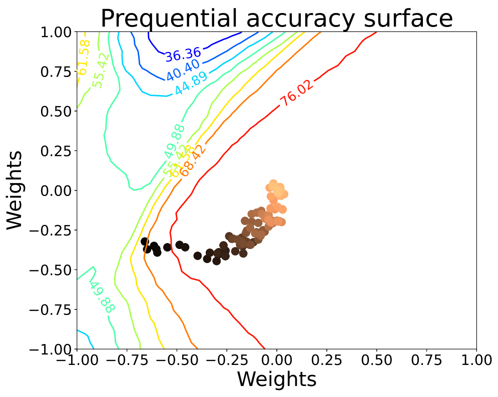


(c)

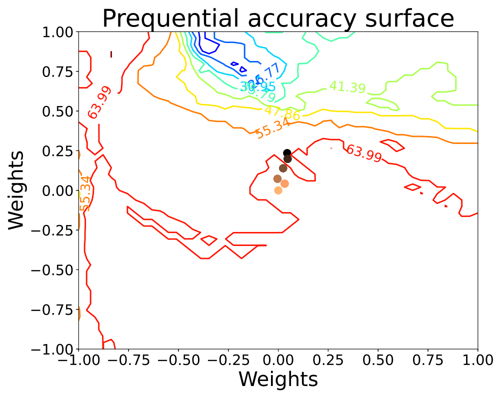


(d)

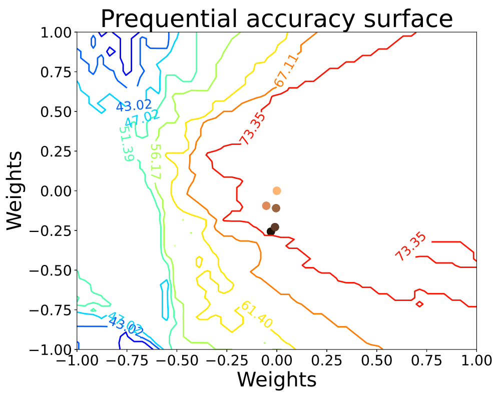
Figure 1: Prequential accuracy surfaces for SWA at 0, 50, 100 and 200 time steps after a gradual concept drift of length 100 starts on the Sine data stream. Circles denote points in weight space retrieved by the SWA optimization path, where darker/lighter circles represent earlier/later positions in the path.



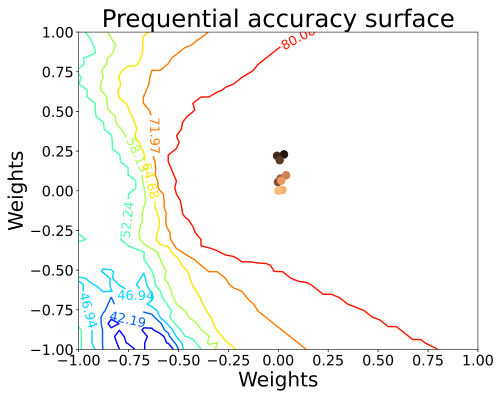
(a)



(b)



(c)



(d)

Figure 2: Prequential accuracy surfaces for OWA at 0, 50, 100 and 200 time steps after a gradual concept drift of length 100 starts on the Sine data stream. Circles denote points in weight space retrieved by the OWA optimization path.

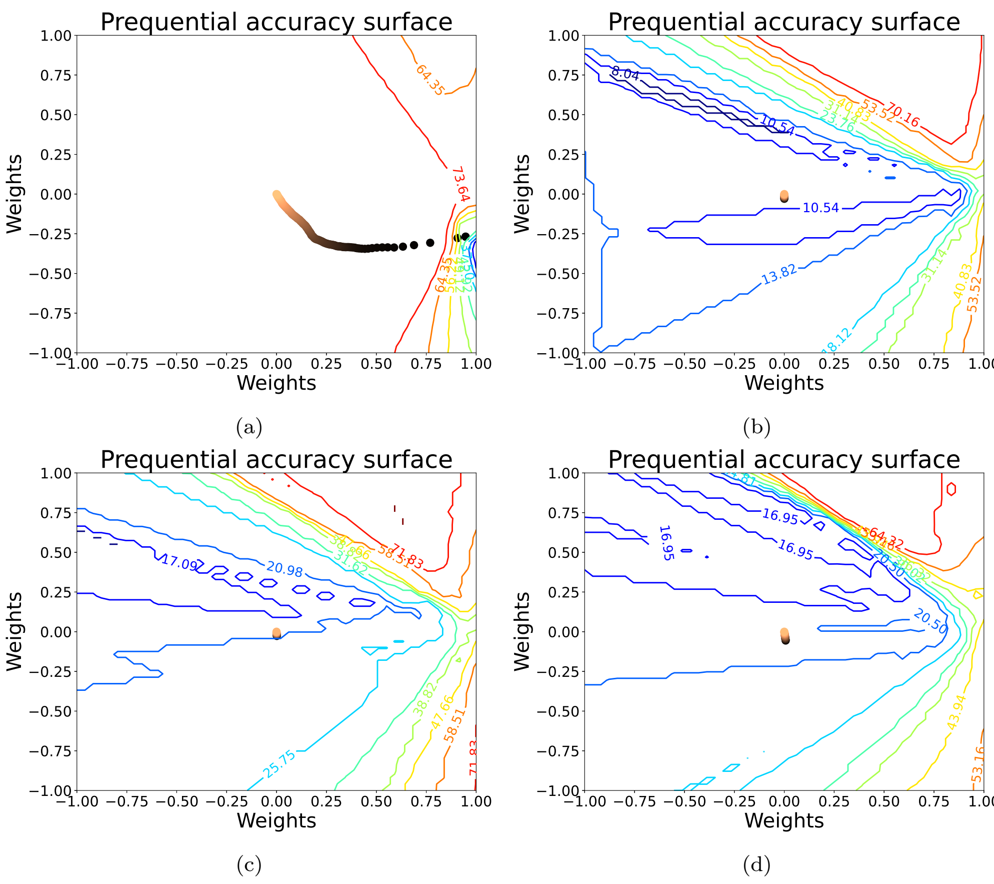
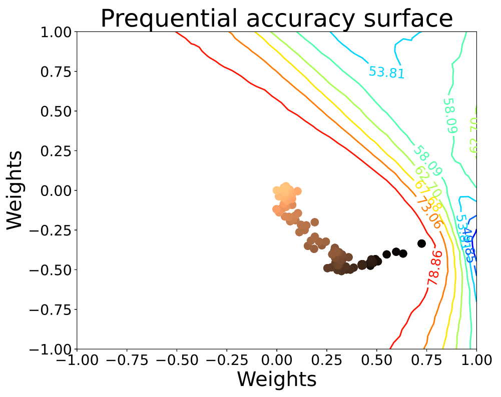
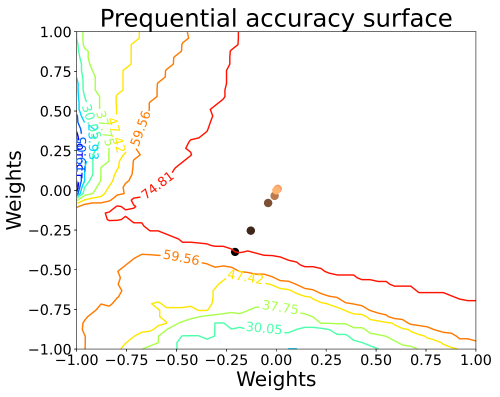


Figure 3: Prequential accuracy surfaces for SWA at 0, 50, 100 and 200 time steps after an abrupt concept drift occurs on the Sine data stream. Circles denote points in weight space from the SWA optimization path.

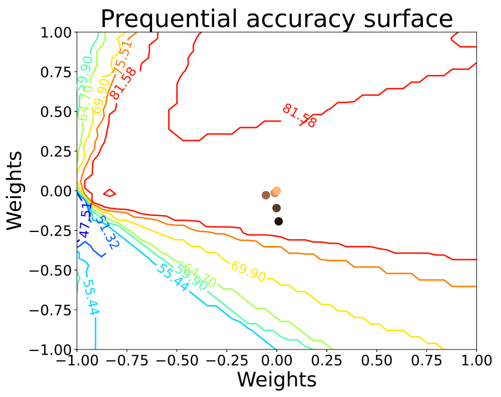
by forgetting obsolete points in its averaging, considering only the current concept in the training of its weights. Figure 4 depicts OWA's optimization path when a abrupt concept drift occurs. The proposed method could use the average of weight vectors from different points in time to produce good weight solutions for the current concept (Figure 4a). Only 50 time steps after a sudden change (Figure 4b), OWA could find good weights in a wide informative region for the new concept. And 200 time steps after the abrupt concept drift, the proposed method continued to find solutions in safer regions (further from the boundaries) of the wide optima, which leads to good generalization.



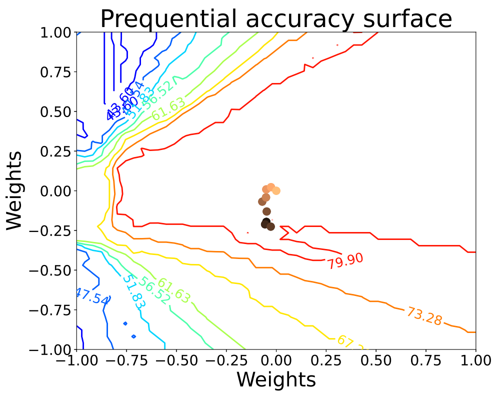
(a)



(b)



(c)



(d)

Figure 4: Prequential accuracy surfaces for OWA at time steps 0, 50, 100 and 200 after an abrupt concept drift occurs on the Sine data stream. Circles denote points in weight space from the OWA optimization path.

We answered RQ1 by proposing OWA – a single neural network for nonstationary data stream learning that compresses an ensemble based on weight averaging. OWA is able to find solutions in wide optima both before and after drifts. Solutions in wide optima are known to lead to good generalization [50, 51]. OWA will be further validated through a comparison of predictive performance and computational cost against existing approaches in Sections 6.2 and 6.3.

6.2. Predictive performance

This section validates OWA in terms of its predictive performance, answering RQ2.

Artificial Data Streams. We conducted Friedman test and Nemenyi post-hoc tests across all artificial data streams and types of concept drifts to compare the G-Mean obtained by the methods. Friedman test detected significant difference at the level of significance of 0.05 (p-value of $2.42 * 10^{-11}$). Figure 5 depicts the critical difference diagram for the Nemenyi post-hoc tests to identify which methods perform differently from each other. Based on these tests, none of the compared methods produced significantly superior predictive performance to OWA across artificial data streams. In particular, OWA was significantly better than SWA and no significant difference has been found compared to Ozabag, Ozabag-Adwin, DWM, CSB2, KUE, CALMID, MICFOAL and SRPC.

Figures 6 and 7 further depict the heatmaps of G-Mean of each compared method across several numbers of base learners (10, 20, 30, 40 and 50) on each data stream separately with gradual and abrupt concept drifts, respectively[1]. Regardless of the number of base learners, OWA delivered similar generalization to the best methods. This is a very positive result, given that OWA is based on a single MLP and a weight average, whereas all other methods except for SWA are ensemble techniques[2].

OWA outperformed SWA because SWA rapidly becomes obsolete when a new concept arrives, whilst OWA can quickly adapt and produce good generalization even after abrupt drifts. This can be observed based on Figures 8

[1]Missing values denote that the method failed to run on the stream, even though all public available implementation of these methods were tested.

[2]Due to space restrictions, plots with G-Mean across different numbers of base learners are available in the Supplementary Material.

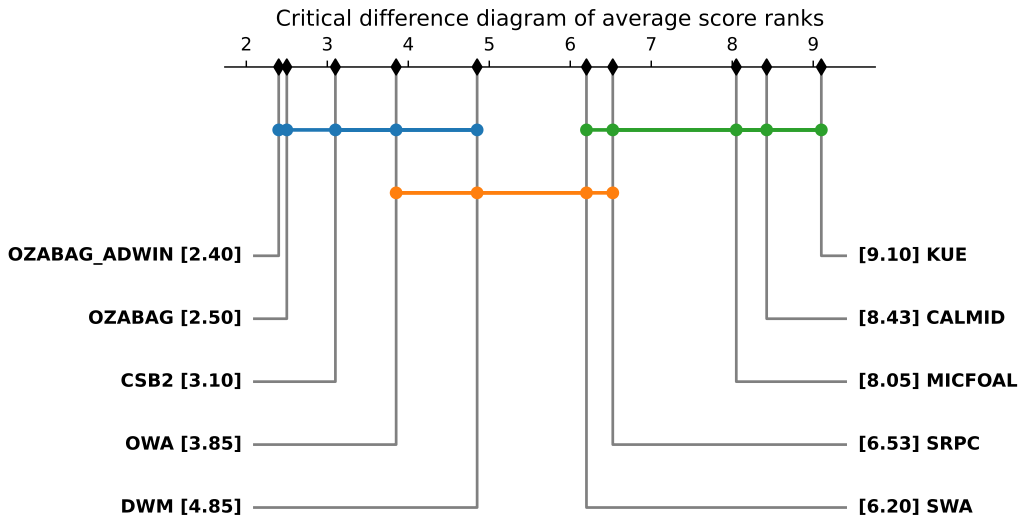
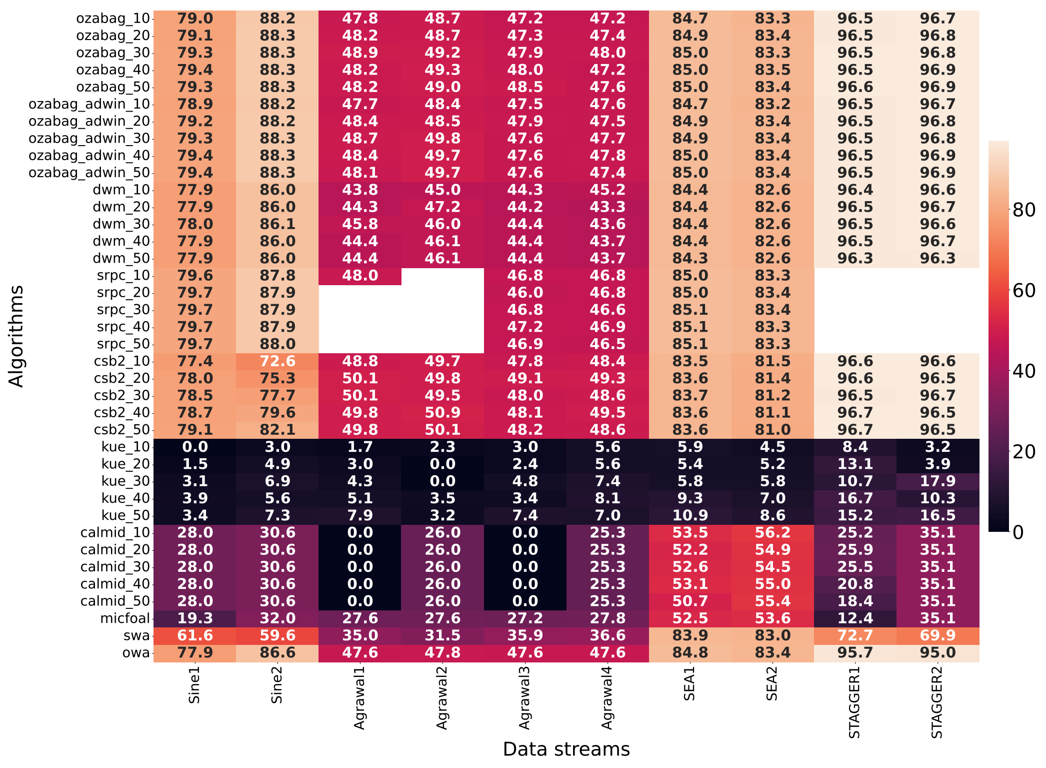


Figure 5: G-Mean Nemenyi post-hoc statistical test across artificial data streams. For ensemble methods, the number of base learners is 50. Values in brackets are mean ranks, where smaller ranks represent higher G-Means. Groups of algorithms that are not significantly different are connected with a bold solid line.

and 9, which show the G-Mean over time of the best run of each algorithm on artificial data streams with gradual and abrupt concept drifts, respectively. In particular, SWA's performance tends to considerably decrease over time in the presence of concept drift, whereas OWA's performance presented a more rapid recovery from concept drift, as its drift detection mechanism resets the averaging procedure.

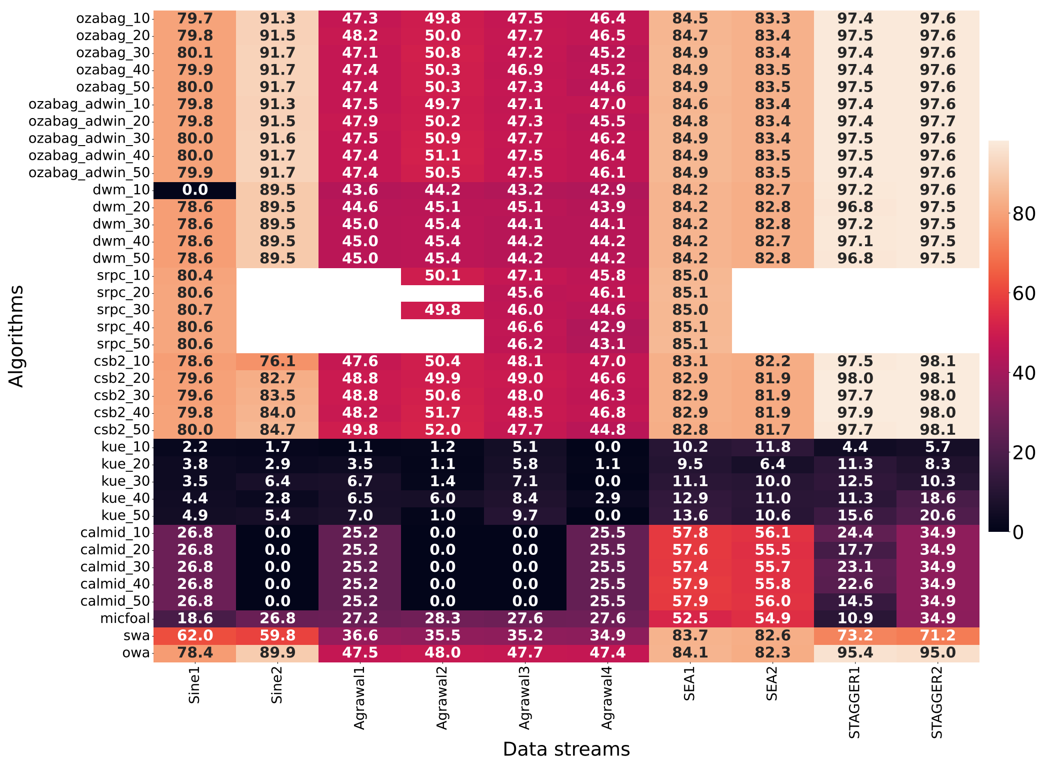
Figures 8 and 9 also show that OWA delivered G-Mean similar to the best ensemble methods over time. It is important to highlight that the compared ensemble methods have sophisticated learning mechanisms (for example, DWM creates and removes base learners as necessary and KUE updates its base learners according to an online statistic), whereas OWA achieves similar performance with a simple weight averaging through time that resets its weights to $\mathbf{w}$ when a concept drift is detected.

Real World Data Streams. According to Friedman statistical test with level of significance of 0.05 on the G-Mean score run across all real-world data streams, significant differences were found among the methods (p-value of $3 * 10^{-4}$). The results of the Nemenyi post-hoc statistical test to identify



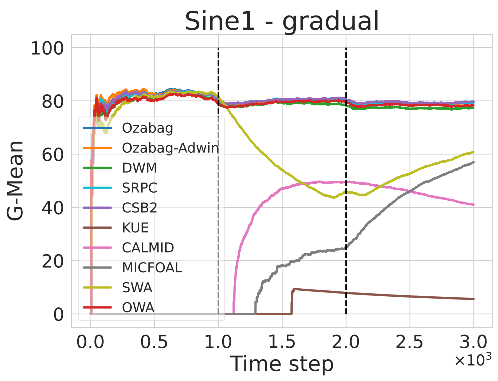
Algorithms	Sine1	Sine2	Agrawal1	Agrawal2	Agrawal3	Agrawal4	SEA1	SEA2	STAGGER1	STAGGER2
ozabag_10	79.0	88.2	47.8	48.7	47.2	47.2	84.7	83.3	96.5	96.7
ozabag_20	79.1	88.3	48.2	48.7	47.3	47.4	84.9	83.4	96.5	96.8
ozabag_30	79.3	88.3	48.9	49.2	47.9	48.0	85.0	83.3	96.5	96.8
ozabag_40	79.4	88.3	48.2	49.3	48.0	47.2	85.0	83.5	96.5	96.9
ozabag_50	79.3	88.3	48.2	49.0	48.5	47.6	85.0	83.4	96.6	96.9
ozabag_adwin_10	78.9	88.2	47.7	48.4	47.5	47.6	84.7	83.2	96.5	96.7
ozabag_adwin_20	79.2	88.2	48.4	48.5	47.9	47.5	84.9	83.4	96.5	96.8
ozabag_adwin_30	79.3	88.3	48.7	49.8	47.6	47.7	84.9	83.4	96.5	96.8
ozabag_adwin_40	79.4	88.3	48.4	49.7	47.6	47.8	85.0	83.4	96.5	96.9
ozabag_adwin_50	79.4	88.3	48.1	49.7	47.6	47.4	85.0	83.4	96.5	96.9
dwm_10	77.9	86.0	43.8	45.0	44.3	45.2	84.4	82.6	96.4	96.6
dwm_20	77.9	86.0	44.3	47.2	44.2	43.3	84.4	82.6	96.5	96.7
dwm_30	78.0	86.1	45.8	46.0	44.4	43.6	84.4	82.6	96.5	96.6
dwm_40	77.9	86.0	44.4	46.1	44.4	43.7	84.4	82.6	96.5	96.7
dwm_50	77.9	86.0	44.4	46.1	44.4	43.7	84.3	82.6	96.3	96.3
srpc_10	79.6	87.8	48.0		46.8	46.8	85.0	83.3		
srpc_20	79.7	87.9			46.0	46.8	85.0	83.4		
srpc_30	79.7	87.9			46.8	46.6	85.1	83.4		
srpc_40	79.7	87.9			47.2	46.9	85.1	83.3		
srpc_50	79.7	88.0			46.9	46.5	85.1	83.3		
csb2_10	77.4	72.6	48.8	49.7	47.8	48.4	83.5	81.5	96.6	96.6
csb2_20	78.0	75.3	50.1	49.8	49.1	49.3	83.6	81.4	96.6	96.5
csb2_30	78.5	77.7	50.1	49.5	48.0	48.6	83.7	81.2	96.5	96.7
csb2_40	78.7	79.6	49.8	50.9	48.1	49.5	83.6	81.1	96.7	96.5
csb2_50	79.1	82.1	49.8	50.1	48.2	48.6	83.6	81.0	96.7	96.5
kue_10	0.0	3.0	1.7	2.3	3.0	5.6	5.9	4.5	8.4	3.2
kue_20	1.5	4.9	3.0	0.0	2.4	5.6	5.4	5.2	13.1	3.9
kue_30	3.1	6.9	4.3	0.0	4.8	7.4	5.8	5.8	10.7	17.9
kue_40	3.9	5.6	5.1	3.5	3.4	8.1	9.3	7.0	16.7	10.3
kue_50	3.4	7.3	7.9	3.2	7.4	7.0	10.9	8.6	15.2	16.5
calmid_10	28.0	30.6	0.0	26.0	0.0	25.3	53.5	56.2	25.2	35.1
calmid_20	28.0	30.6	0.0	26.0	0.0	25.3	52.2	54.9	25.9	35.1
calmid_30	28.0	30.6	0.0	26.0	0.0	25.3	52.6	54.5	25.5	35.1
calmid_40	28.0	30.6	0.0	26.0	0.0	25.3	53.1	55.0	20.8	35.1
calmid_50	28.0	30.6	0.0	26.0	0.0	25.3	50.7	55.4	18.4	35.1
micfoal	19.3	32.0	27.6	27.6	27.2	27.8	52.5	53.6	12.4	35.1
swa	61.6	59.6	35.0	31.5	35.9	36.6	83.9	83.0	72.7	69.9
owa	77.9	86.6	47.6	47.8	47.6	47.6	84.8	83.4	95.7	95.0

Figure 6: Heatmap of the mean G-Mean of each method using different numbers of base learners (10, 20, 30, 40, 50) on artificial data streams with gradual concept drifts.

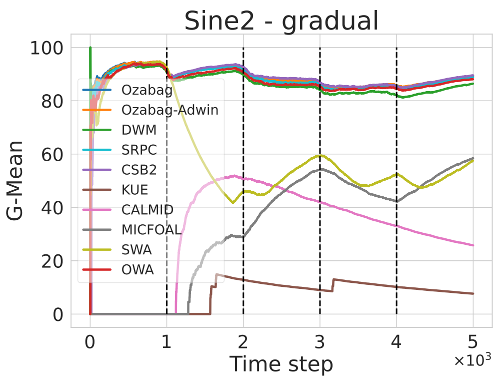


Algorithms	Sine1	Sine2	Agrawal1	Agrawal2	Agrawal3	Agrawal4	SEA1	SEA2	STAGGER1	STAGGER2
ozabag_10	79.7	91.3	47.3	49.8	47.5	46.4	84.5	83.3	97.4	97.6
ozabag_20	79.8	91.5	48.2	50.0	47.7	46.5	84.7	83.4	97.5	97.6
ozabag_30	80.1	91.7	47.1	50.8	47.2	45.2	84.9	83.4	97.4	97.6
ozabag_40	79.9	91.7	47.4	50.3	46.9	45.2	84.9	83.5	97.4	97.6
ozabag_50	80.0	91.7	47.4	50.3	47.3	44.6	84.9	83.5	97.5	97.6
ozabag_adwin_10	79.8	91.3	47.5	49.7	47.1	47.0	84.6	83.4	97.4	97.6
ozabag_adwin_20	79.8	91.5	47.9	50.2	47.3	45.5	84.8	83.4	97.4	97.7
ozabag_adwin_30	80.0	91.6	47.5	50.9	47.7	46.2	84.9	83.4	97.5	97.6
ozabag_adwin_40	80.0	91.7	47.4	51.1	47.5	46.4	84.9	83.5	97.5	97.6
ozabag_adwin_50	79.9	91.7	47.4	50.5	47.5	46.1	84.9	83.5	97.4	97.6
dwm_10	0.0	89.5	43.6	44.2	43.2	42.9	84.2	82.7	97.2	97.6
dwm_20	78.6	89.5	44.6	45.1	45.1	43.9	84.2	82.8	96.8	97.5
dwm_30	78.6	89.5	45.0	45.4	44.1	44.1	84.2	82.8	97.2	97.5
dwm_40	78.6	89.5	45.0	45.4	44.2	44.2	84.2	82.7	97.1	97.5
dwm_50	78.6	89.5	45.0	45.4	44.2	44.2	84.2	82.8	96.8	97.5
srpc_10	80.4			50.1	47.1	45.8	85.0			
srpc_20	80.6				45.6	46.1	85.1			
srpc_30	80.7			49.8	46.0	44.6	85.0			
srpc_40	80.6				46.6	42.9	85.1			
srpc_50	80.6				46.2	43.1	85.1			
csb2_10	78.6	76.1	47.6	50.4	48.1	47.0	83.1	82.2	97.5	98.1
csb2_20	79.6	82.7	48.8	49.9	49.0	46.6	82.9	81.9	98.0	98.1
csb2_30	79.6	83.5	48.8	50.6	48.0	46.3	82.9	81.9	97.7	98.0
csb2_40	79.8	84.0	48.2	51.7	48.5	46.8	82.9	81.9	97.9	98.0
csb2_50	80.0	84.7	49.8	52.0	47.7	44.8	82.8	81.7	97.7	98.1
kue_10	2.2	1.7	1.1	1.2	5.1	0.0	10.2	11.8	4.4	5.7
kue_20	3.8	2.9	3.5	1.1	5.8	1.1	9.5	6.4	11.3	8.3
kue_30	3.5	6.4	6.7	1.4	7.1	0.0	11.1	10.0	12.5	10.3
kue_40	4.4	2.8	6.5	6.0	8.4	2.9	12.9	11.0	11.3	18.6
kue_50	4.9	5.4	7.0	1.0	9.7	0.0	13.6	10.6	15.6	20.6
calmid_10	26.8	0.0	25.2	0.0	0.0	25.5	57.8	56.1	24.4	34.9
calmid_20	26.8	0.0	25.2	0.0	0.0	25.5	57.6	55.5	17.7	34.9
calmid_30	26.8	0.0	25.2	0.0	0.0	25.5	57.4	55.7	23.1	34.9
calmid_40	26.8	0.0	25.2	0.0	0.0	25.5	57.9	55.8	22.6	34.9
calmid_50	26.8	0.0	25.2	0.0	0.0	25.5	57.9	56.0	14.5	34.9
micfoal	18.6	26.8	27.2	28.3	27.6	27.6	52.5	54.9	10.9	34.9
swa	62.0	59.8	36.6	35.5	35.2	34.9	83.7	82.6	73.2	71.2
owa	78.4	89.9	47.5	48.0	47.7	47.4	84.1	82.3	95.4	95.0

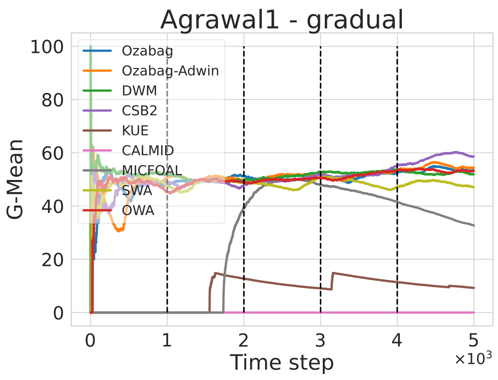
Figure 7: Heatmap of the mean G-Mean of each method using different numbers of base learners (10, 20, 30, 40, 50) on artificial data streams with abrupt concept drifts.


Sine1 - gradual
G-Mean
Time step
×10³
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

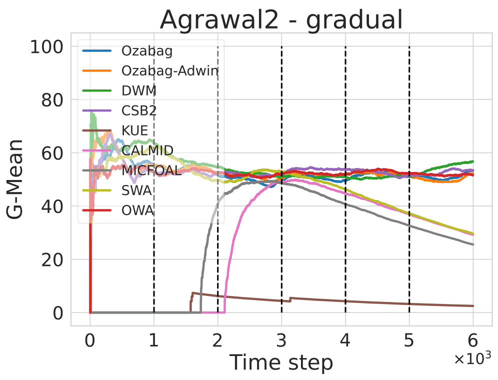
(a)


Sine2 - gradual
G-Mean
Time step
×10³
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

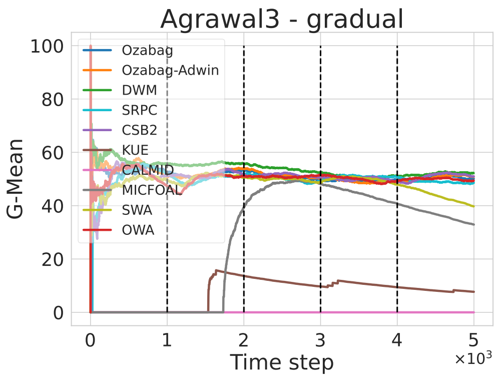
(b)


Agrawal1 - gradual
G-Mean
Time step
×10³
Ozabag
Ozabag-Adwin
DWM
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

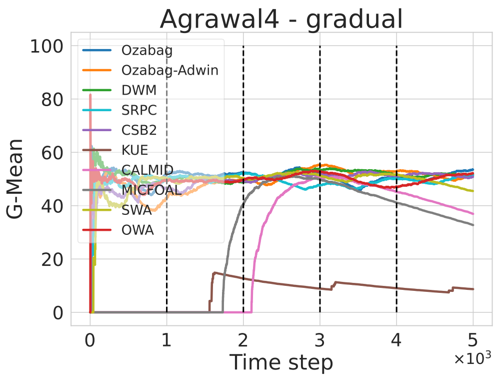
(c)


Agrawal2 - gradual
G-Mean
Time step
×10³
Ozabag
Ozabag-Adwin
DWM
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

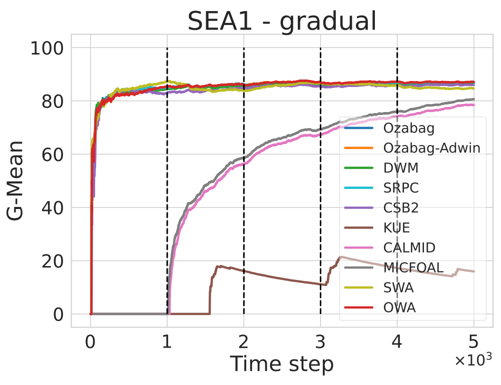
(d)


Agrawal3 - gradual
G-Mean
Time step
×10³
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

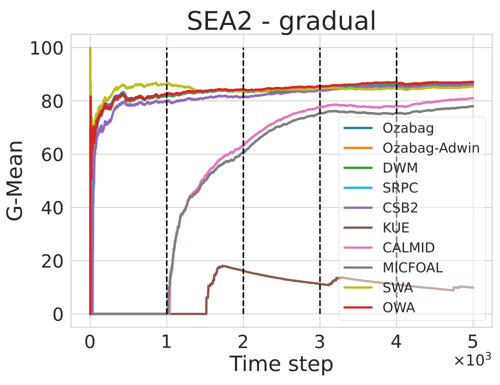
(e)


Agrawal4 - gradual
G-Mean
Time step
×10³
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

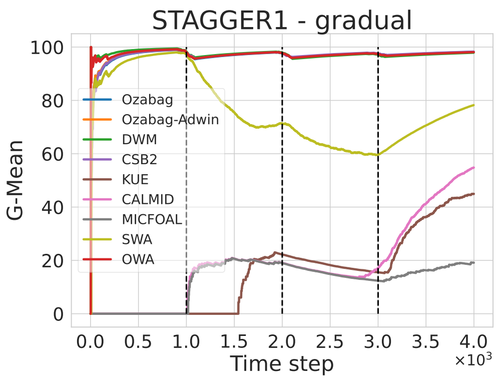
(f)



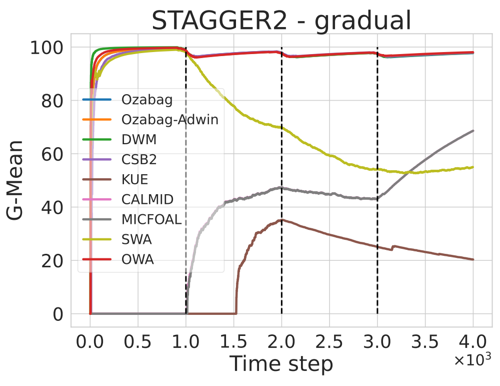
(g)



(h)

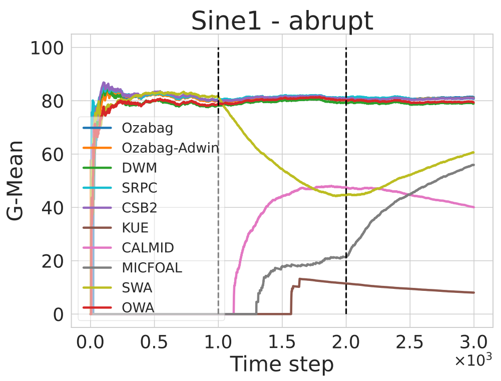


(i)

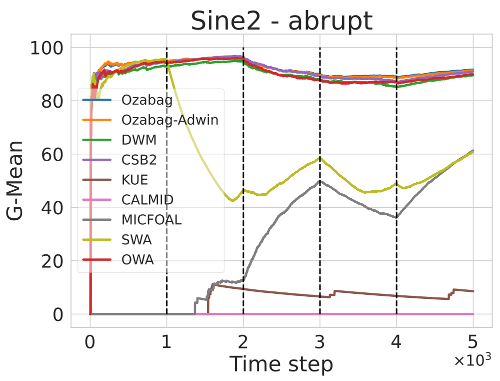


(j)

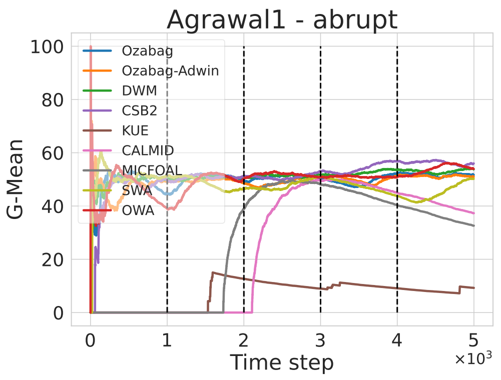
Figure 8: Plots of G-Mean produced by the best run of each of the methods for artificial data streams with gradual concept drifts. Dashed lines denote concept drifts.


Sine1 - abrupt
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

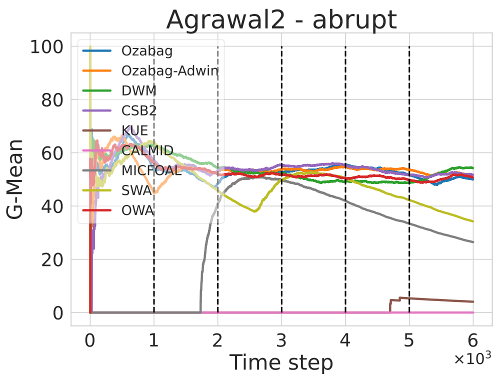
(a)


Sine2 - abrupt
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

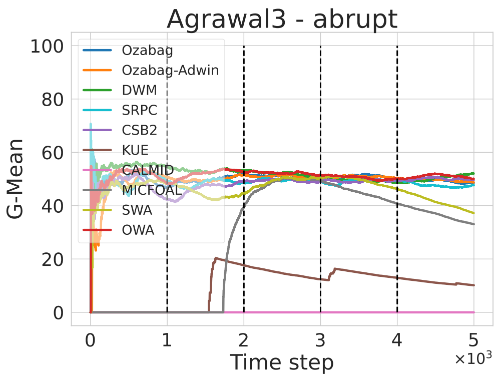
(b)


Agrawal1 - abrupt
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

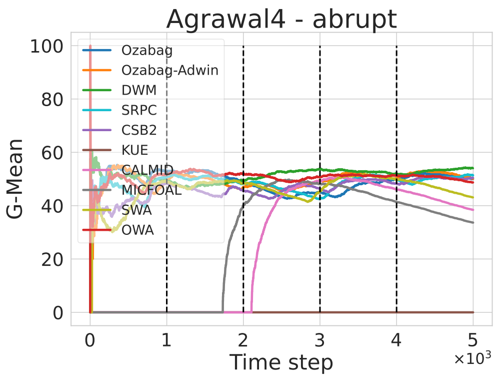
(c)


Agrawal2 - abrupt
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

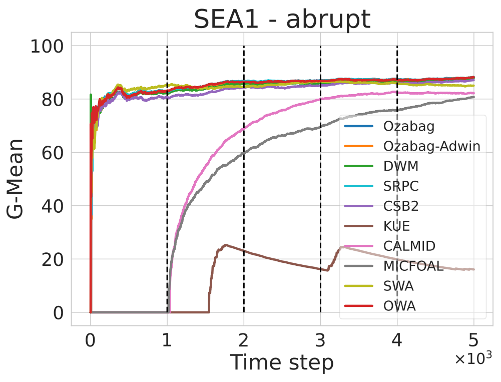
(d)


Agrawal3 - abrupt
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

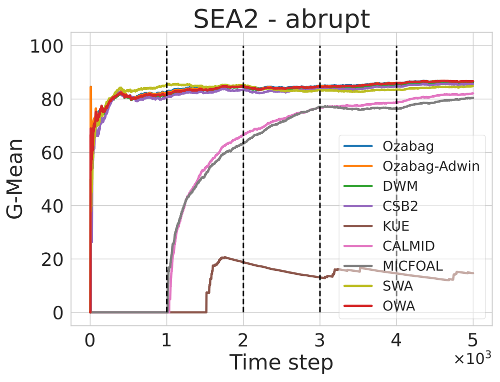
(e)


Agrawal4 - abrupt
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

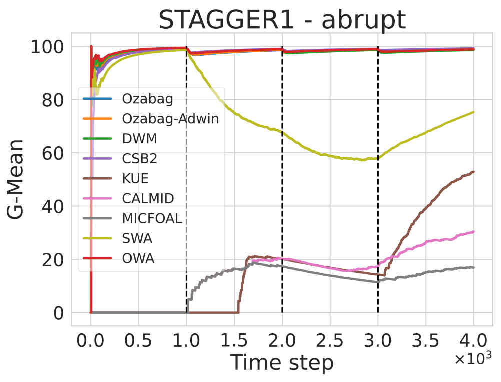
(f)



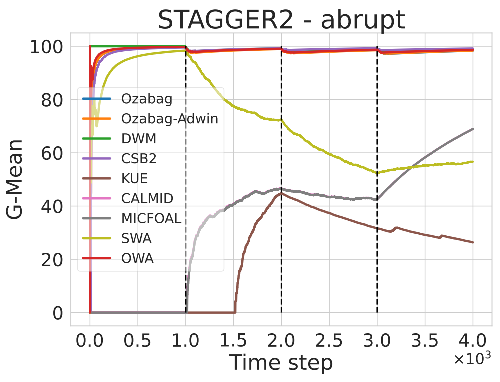
(g)



(h)



(i)



(j)

Figure 9: Plots of G-Mean produced by the best run of each of the methods for artificial data streams with abrupt concept drifts. Dashed lines denote concept drifts.

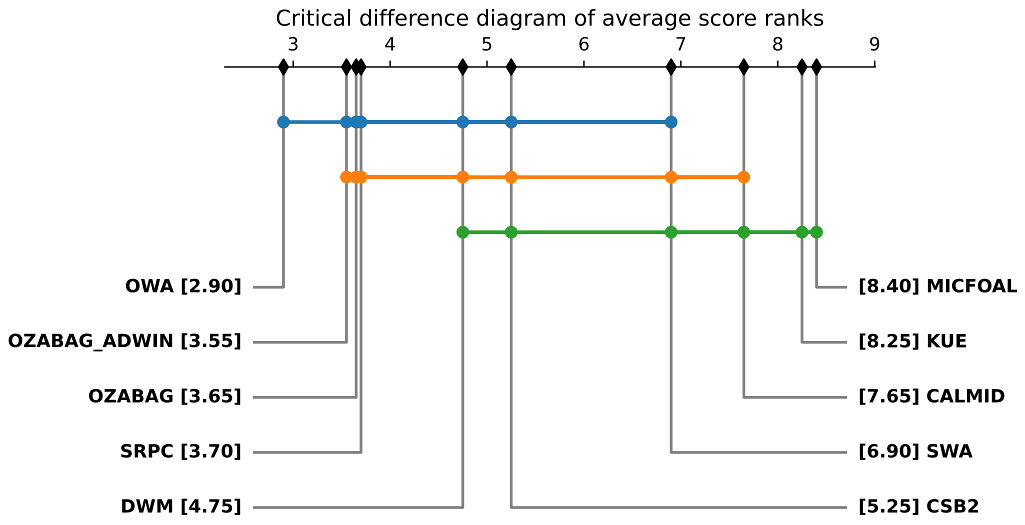


Figure 10: G-Mean Nemenyi post-hoc statistical test across real-world data streams. For ensemble methods, the number of base learners is 50. Values in brackets are mean ranks. Groups of algorithms that are not significantly different are connected with a bold solid line.

which methods perform differently are shown in Figure 10. None of the compared methods produced significantly superior predictive performance to OWA across real-world data streams. As with the artificial data streams, no significant difference has been found between the performance of the proposed approach and Ozabag, Ozabag-Adwin, DWM, CSB2 and SRPC. On the other hand, OWA was significantly superior to KUE, CALMID and MICFOAL. This illustrates OWA's ability to find more stable minima by averaging weight solutions on the borders of wide minima. This leads to an ensemble-like performance at the cost of a single method. OWA was significantly superior to its offline counterpart – SWA – as expected. This result indicates the usefulness of employing an online approach specifically designed for data stream learning with a concept drift detection mechanism. Such an averaging helps the neural network to obtain similar generalization to ensemble methods while training only a single model.

Figure 11 depicts the heatmap of the G-Mean score for each algorithm across several numbers of learners (10, 20, 30, 40 and 50) on each specific data stream. For the Chess stream, OWA was superior to KUE, CALMID, MICFOAL, SWA and DWM, and similar to the other techniques. Only

SWA and OWA delivered good predictive performances for Keystroke. In particular, the other techniques are making large errors in a specific class, which leads to poor G-Mean score. For Luxembourg, SRPC and OWA obtain similar results, however OWA produced higher variance due to a potentially large architecture for this stream. For Ozone, which is heavily imbalanced (5%:95%), and for NOAA (32%:68%), SRPC and DWM were the superior methods, while OWA was superior to KUE, CALMID and MICFOAL and was similar to the other algorithms. This might indicate that SRPC could handle imbalanced classes better than the other methods, which translates to a much higher G-Mean score. A smaller architecture for OWA could mitigate the higher variance. OWA also obtained similar performance to the top techniques for the Electricity and Power Supply Day/Night streams. This shows that the proposed approach is robust to large number of classes (40). For Power Supply, which has 24 classes, OWA delivered significantly superior performance, which denotes that it could find wide regions in weight space to compute better minima than standard neural networks. OWA was significantly better than SWA when all real-world streams were considered. As mentioned earlier, these streams are likely to present drifts, which hinders SWA's performance. In contrast, OWA is able to deliver good generalization by being able to find wide optima even after drifts in real-world data streams.

We plot in Figure 12 the G-Mean score over time of the best run of each algorithm in each stream. For Chess and Keystroke, all methods delivered similar performance (except for SWA) as shown in Figures 12a and 12b. For Luxembourg (Figure 12c), DWM produced superior performance, which might indicate that it could adapt to its small number of instances. For Ozone (Figure 12d), OWA was superior to the other algorithms for the longest section of the stream. On the Electricity data stream (Figure 12e), all algorithms, except MICFOAL and SWA, produced similar predictive performance. For NOAA (Figure 12f), SRPC was the top-scoring method throughout the stream. For both Power Supply and Power Supply Day/Night (Figures 12h and 12g), OWA delivered top G-Mean despite being a single learner compared with ensembles. On the Sensor stream (Figure 12i), DWM was the superior technique and the other techniques delivered similar performance (except KUE and MICFOAL).

We also performed a Friedman followed by a Nemenyi post-hoc test with all streams and all numbers of base learners (Figure 13). OWA is among the top performing ensembles and is significantly superior to CALMID, MICFOAL, KUE and especially SWA. This evidence supports our hypothesis

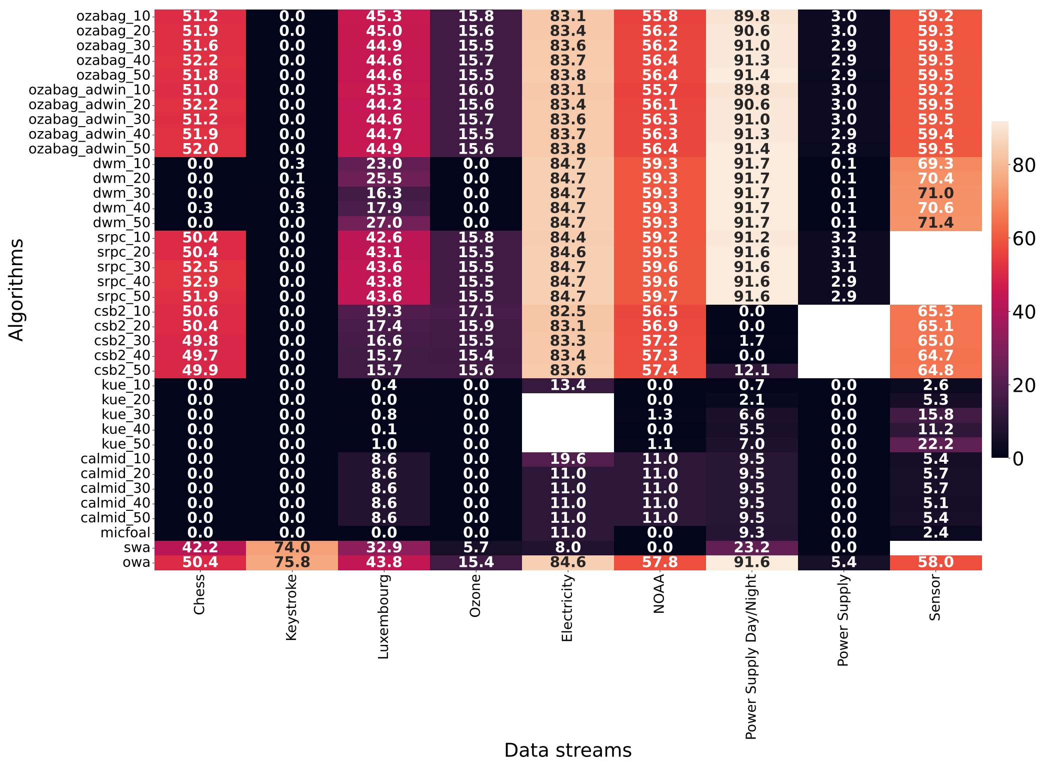
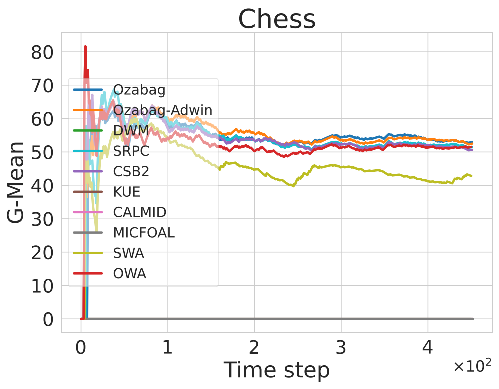
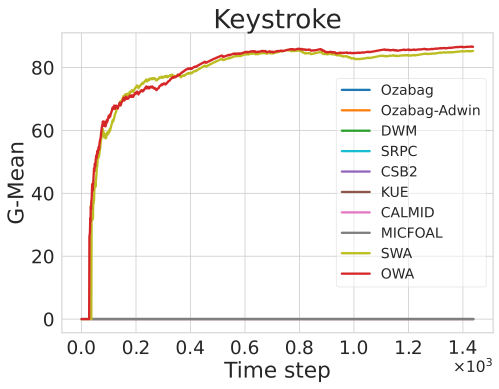


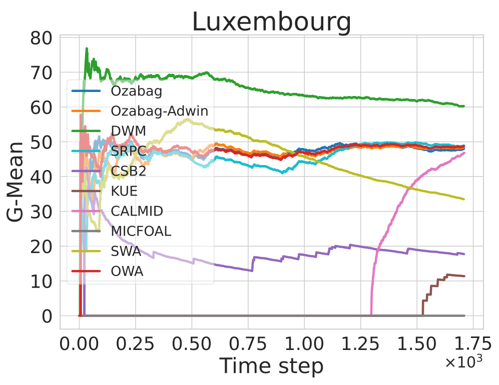
Figure 11: Heatmap of the mean G-Mean of each method using different numbers of base learners (10, 20, 30, 40, 50) from real-world data streams.


Chess
G-Mean
Time step
×10^2
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

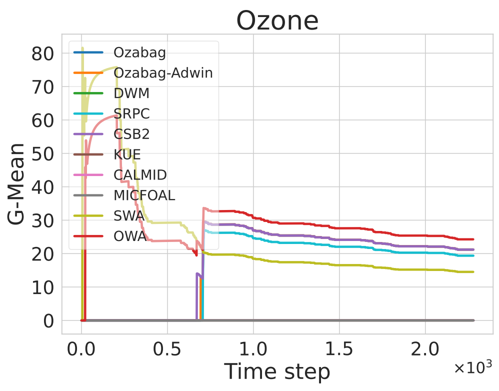
(a)


Keystroke
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

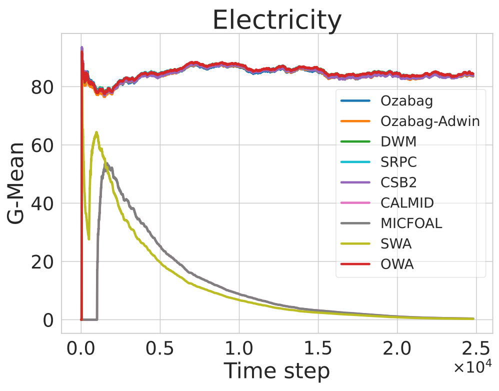
(b)


Luxembourg
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

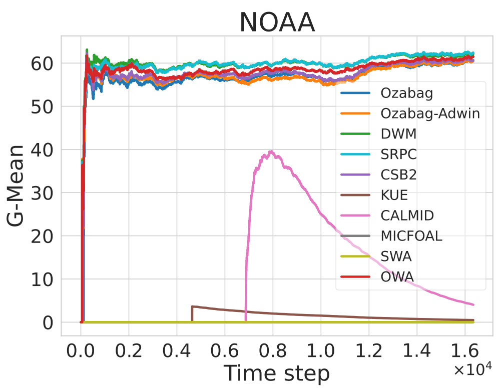
(c)


Ozone
G-Mean
Time step
×10^3
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
KUE
CALMID
MICFOAL
SWA
OWA

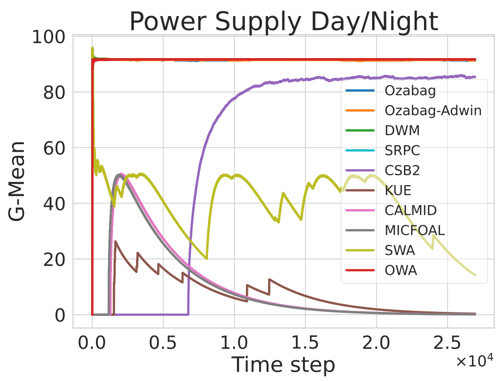
(d)


Electricity
G-Mean
Time step
×10^4
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
SWA
OWA

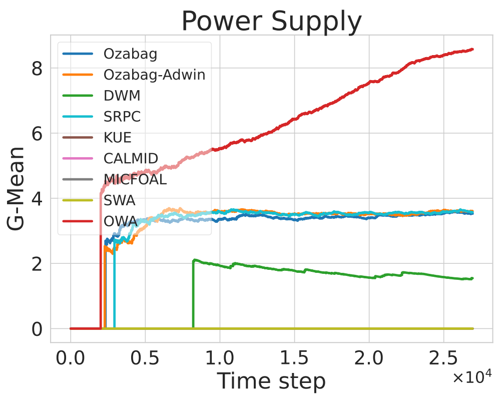
(e)



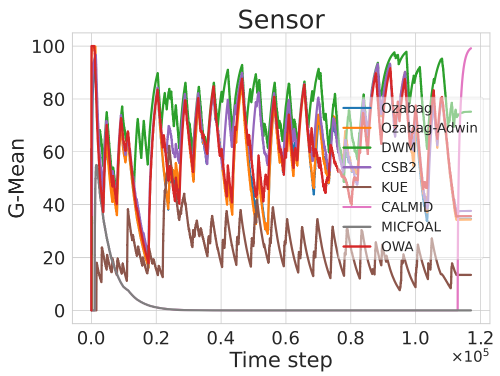
(f)



(g)



(h)



(i)

Figure 12: Plots of G-Mean produced by the best run of each of the methods for real-world data streams.

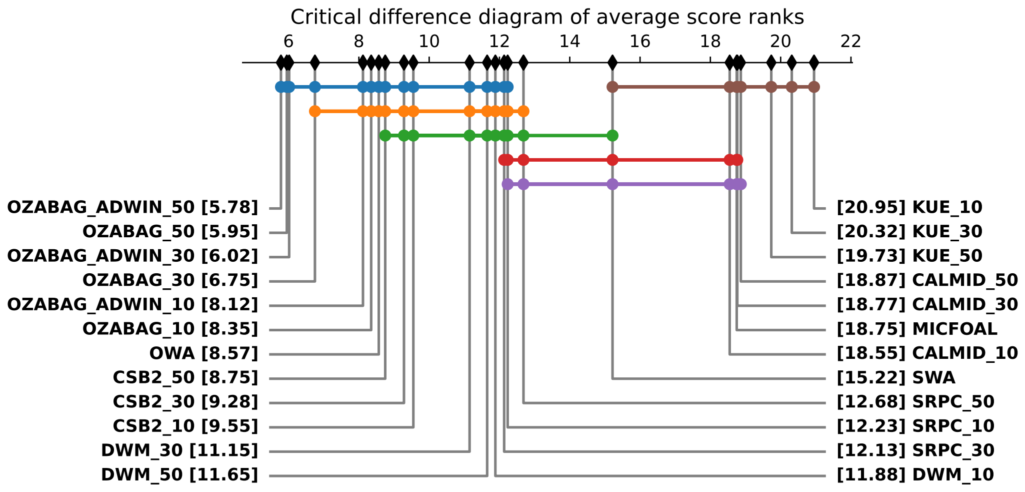


Figure 13: G-Mean Nemenyi post-hoc statistical test across all data streams and, for ensemble methods, the numbers of base learners are 10, 30, 50. Values in brackets are mean ranks. Groups of algorithms that are not significantly different are connected with a bold solid line.

that weight averaging is useful for nonstationary data stream learning, delivering ensemble-like generalization while being able to adapt to all types of drifts.[3]

We answered RQ2 by performing several experiments with artificial and real-world streams with statistical and visual examination. The results showed that OWA produced better generalization than SWA in all cases, and similar or better predictive performance across data streams than existing data stream learning techniques. This fact denotes that the cyclical learning and the ability to adapt to new concepts positively affect the predictive performance in weight averaging approaches on nonstationary environments.

6.3. Computational time

This section validates OWA in terms of its time complexity and computational time, answering RQ3.

[3]We provide a comprehensive and detailed analysis with boxplots of predictive performance in the Supplementary Material.

Assessing the time complexity of state-of-the-art data stream learning ensembles is challenging, because it mainly depends on the base learners and on the drift detection mechanisms employed. For example, let t_{base} be the computational time of a base learner and g_{ens} be the number of base learners, then OzaBag is in $\mathcal{O}(g_{\text{ens}} t_{\text{base}})$. Typically, such ensemble methods have drift detection mechanisms. For instance, Ozabag may be implemented with ADWIN, which is based on the Bernstein Bound to detect concept changes. ADWIN maintains a window (W) of examples at a given time and compares the mean difference of any two sub-windows of older and recent examples from W. It requires multiple passes in the current window. The time complexity of ADWIN is $\mathcal{O}(\log |W|)$. With ADWIN, Ozabag becomes, at each time step, $\mathcal{O}(g_{\text{ens}} t_{\text{base}} \log |W|)$. With the same base learner (MLP), the minimum difference in computational time between OWA and ensemble methods mainly depends on the number of base learners to be trained (we are not considering other learning mechanisms as they vary between ensemble techniques), as their computational complexity usually grows linearly with the number of base learners and the number of base learners can grow large depending on the application and the ensemble learning approach. As our experiments will show in this section, with the same MLP model as base learner, OWA is able to learn with the computational time orders of magnitude less than other methods, as its time complexity does not grow with the number of models compressed into the ensemble.

It is important to highlight that OWA only keeps a shallow MLP and the averaged weights $\boldsymbol{\mu}$, as it is based on a single model. OWA's time complexity, with a single hidden layer of size H, to predict and train on a single time step is $\mathcal{O}(H(D + K))$. As will be shown in this section, OWA is more time efficient than state-of-the-art methods, whilst preserving the benefits of ensemble learning as shown in Section 6.2.

We have performed the Friedman followed by the Bonferroni-Dunn statistical test [49] to establish whether the computational time of OWA is significantly different than that each other technique. Bonferroni-Dunn was used instead of Nemenyi because it is stronger when using a single control method, which is the case for the analysis done in this section, where OWA is the control method. Based on the Friedman test, there are significant differences among methods at the level of significance of 0.05 (p-value of $7,8 * 10^{-3}$). Based on the Bonferroni-Dunn tests, for all artificial and real world data streams, OWA significantly reduced the computational time compared to all ensemble methods according to this test with the largest p-value of $7,8 * 10^{-3}$. This result is further illustrated by Figures 14, 15, 16 which contain plots of the computational time (in seconds) for all methods. As expected, OWA had similar computational time to SWA since they are both single neural networks. However, OWA is tailored for data stream learning, producing significantly higher predictive performance as demonstrated in Section 6.2.

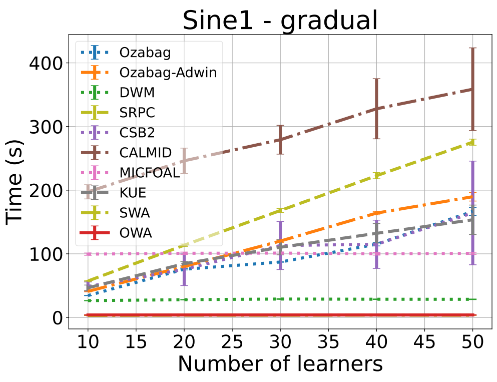
It is worth highlighting that, as expected, the computational time for most ensemble approaches grows linearly with the number of base learners. In contrast, SWA and OWA are significantly faster because they compress their ensembles into single models. Although OWA and SWA have similar computational time, OWA has the advantage of delivering better predictive performance.

We answered RQ3 by studying the computational time of the proposed approach. Our comprehensive experiments confirmed that OWA requires significantly smaller computational time than ensemble approaches, and similar computational time to the existing weight averaging approach SWA.

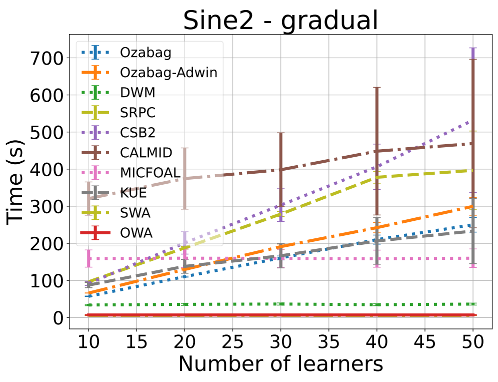
6.4. Sensitivity analysis

C is the main hyperparameter of OWA regarding its predictive performance according to our experiments. So, we further analyze it in detail here. This hyperparameter regulates the rate at which the weight average is updated in OWA and is related to the number of learners in ensembles. Since we are comparing the proposed technique to ensembles, we analyze the impact of the choice of C on OWA's performance by depicting its prequential G-Mean score on two artificial streams, namely, Sine and SEA for different values of C in Figure 17. In Figure 17a, it is possible that a more intense rate of averaging is necessary for the Sine2 stream ($C = 5$) with gradual concept drifts (Figure 17a). With abrupt concept drifts (Figure 17c), a more intense rate of averaging also obtained good performance (similar G-Mean to that obtained by $C = 10$ and $C = 50$). This illustrates the robustness of our methods to C with respect to the type of concept drift. The results on the SEA stream (Figures 17b and 17d) also indicate the approach's robustness to C. In particular, the choice of C had a negligible impact for both gradual and abrupt concept drifts on this stream. In general, we suggest C to be optimized in $[1, 500]$, with smaller values possibly being among the top performing ones.

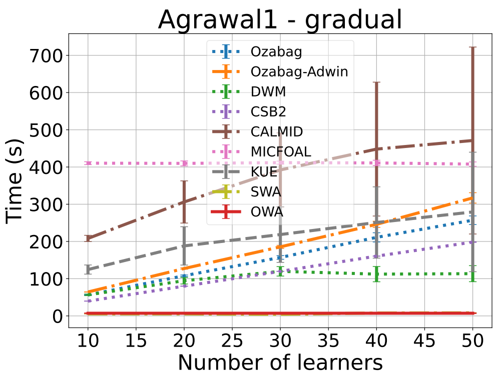
We have also measured OWA's time sensitivity to C. The impact of C on


Sine1 - gradual
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA
Time (s)
Number of learners

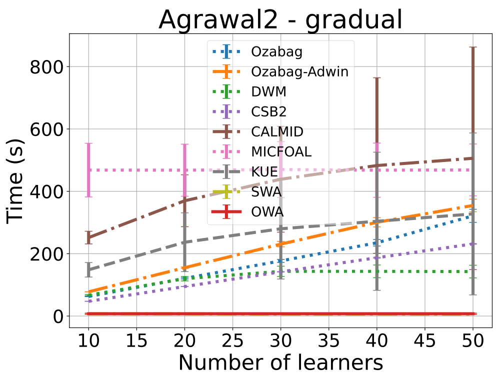
(a)


Sine2 - gradual
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA
Time (s)
Number of learners

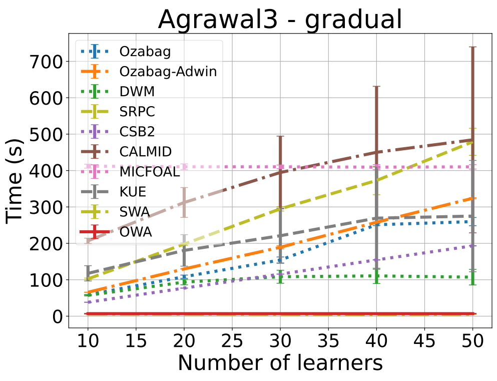
(b)


Agrawal1 - gradual
Ozabag
Ozabag-Adwin
DWM
CSB2
CALMID
MICFOAL
KUE
SWA
OWA
Time (s)
Number of learners

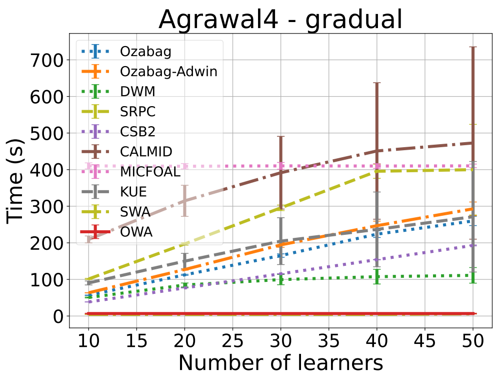
(c)


Agrawal2 - gradual
Ozabag
Ozabag-Adwin
DWM
CSB2
CALMID
MICFOAL
KUE
SWA
OWA
Time (s)
Number of learners

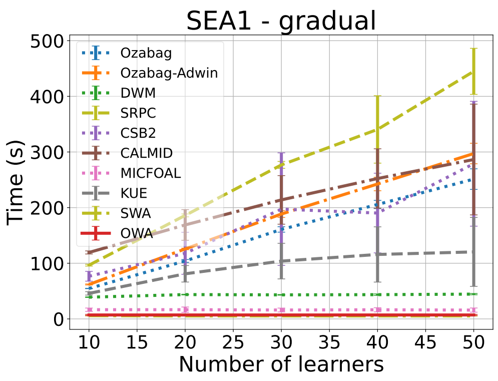
(d)


Agrawal3 - gradual
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA
Time (s)
Number of learners

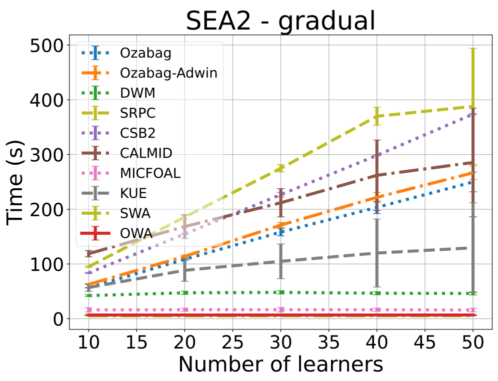
(e)


Agrawal4 - gradual
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA
Time (s)
Number of learners

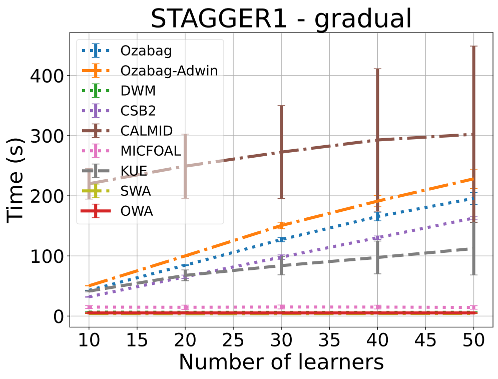
(f)



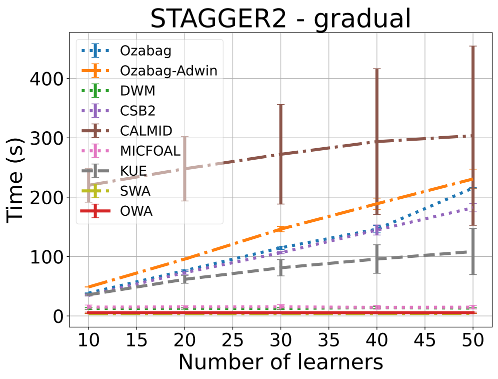
(g)



(h)

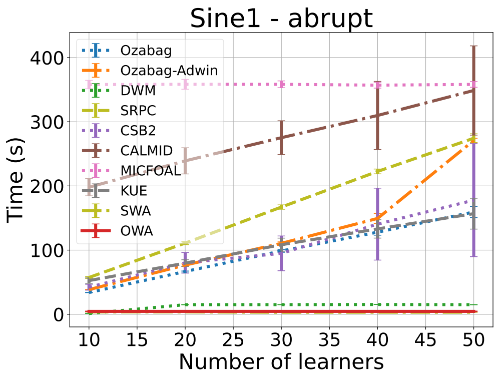


(i)

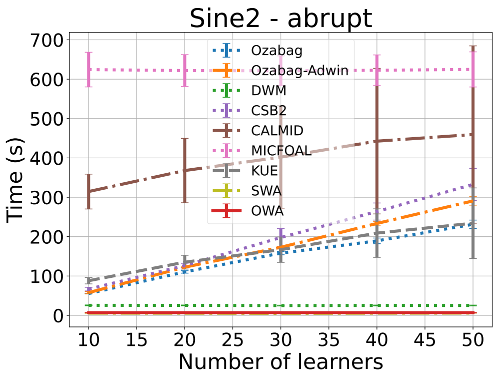


(j)

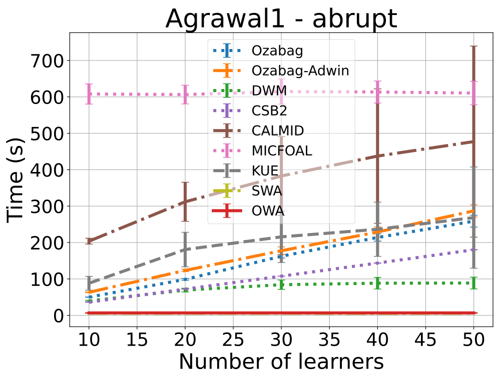
Figure 14: Plots of computational time required by the best run of each of the methods for artificial data streams with gradual concept drifts.


Sine1 - abrupt
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

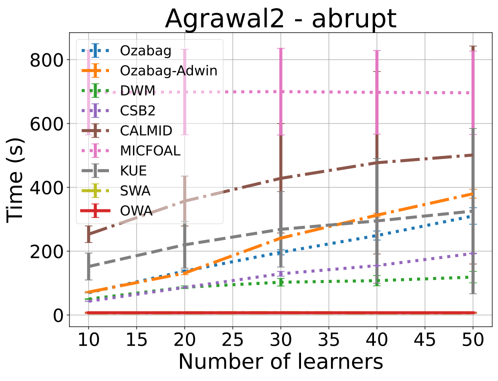
(a)


Sine2 - abrupt
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

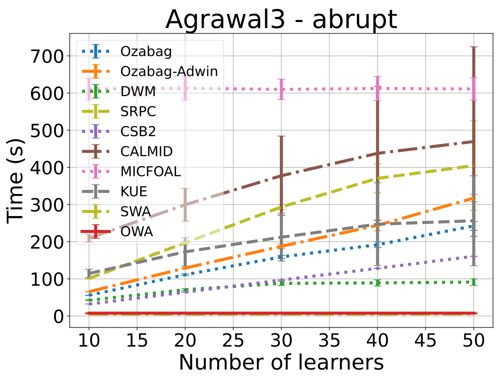
(b)


Agrawal1 - abrupt
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

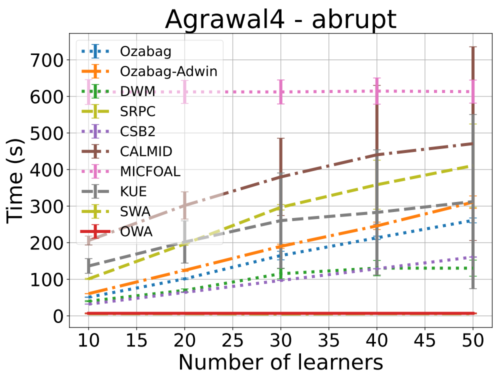
(c)


Agrawal2 - abrupt
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

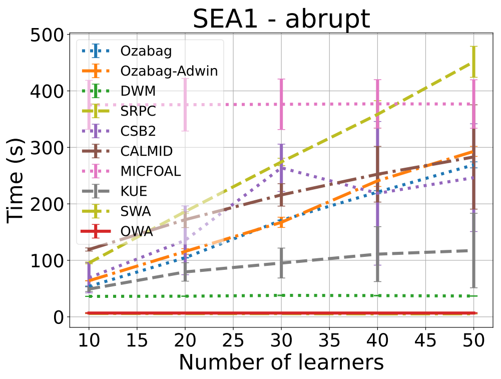
(d)


Agrawal3 - abrupt
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

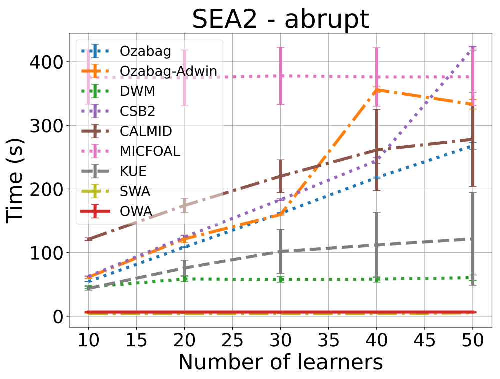
(e)


Agrawal4 - abrupt
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

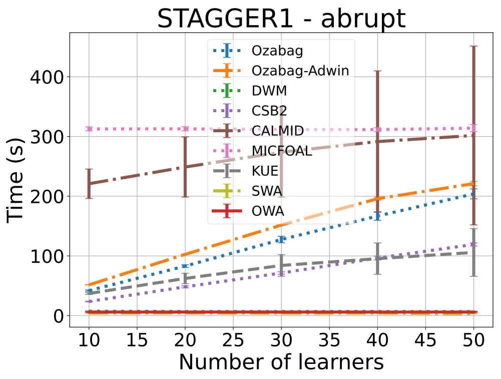
(f)



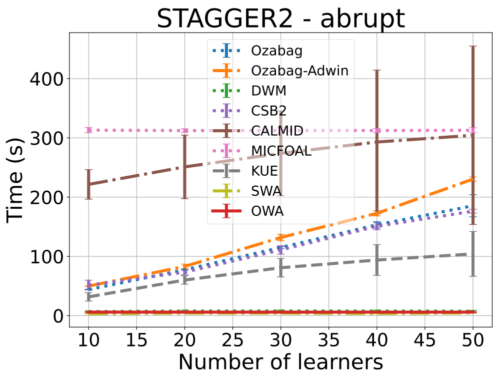
(g)



(h)

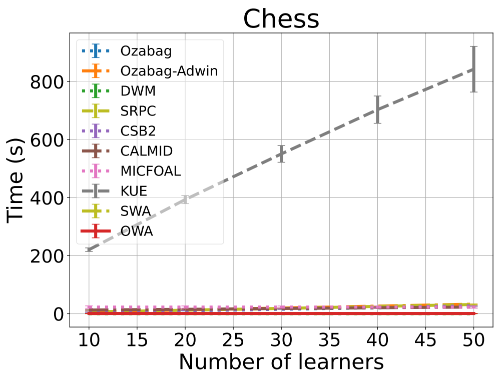


(i)

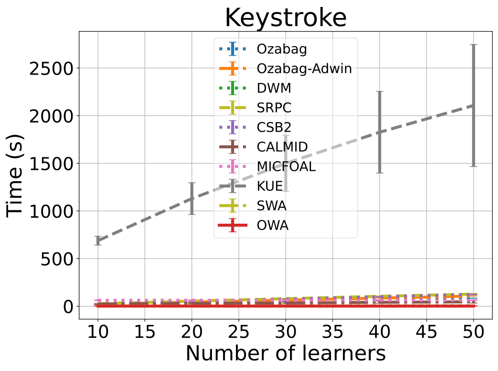


(j)

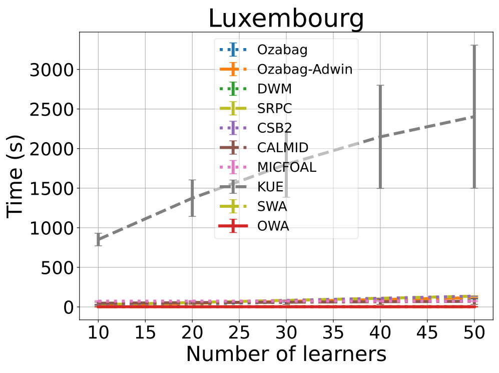
Figure 15: Plots of computational time required by the best run of each of the methods for artificial data streams with abrupt concept drifts.


Chess
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

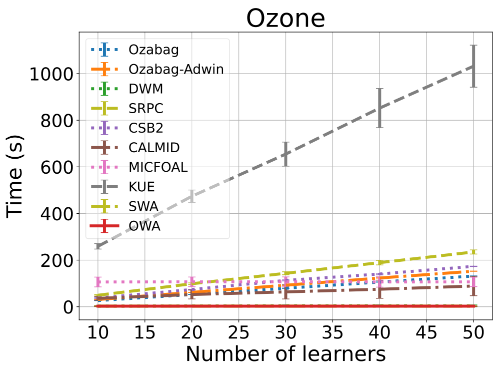
(a)


Keystroke
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

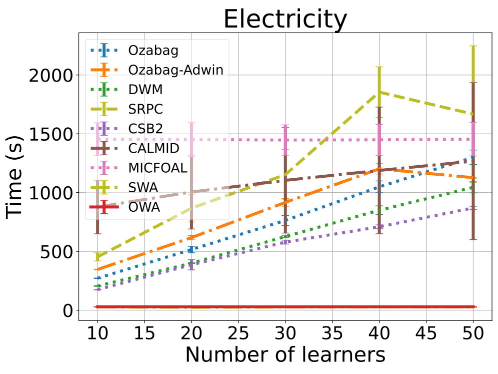
(b)


Luxembourg
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

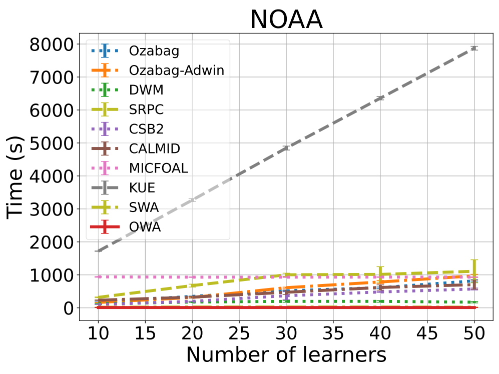
(c)


Ozone
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
KUE
SWA
OWA

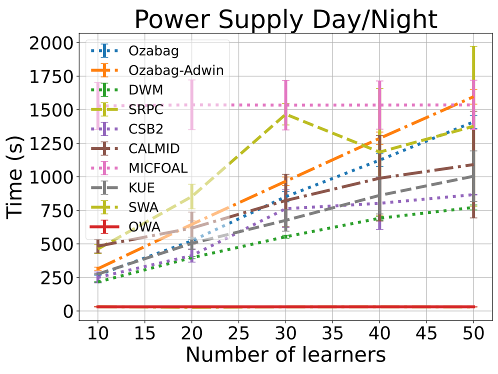
(d)


Electricity
Time (s)
Number of learners
Ozabag
Ozabag-Adwin
DWM
SRPC
CSB2
CALMID
MICFOAL
SWA
OWA

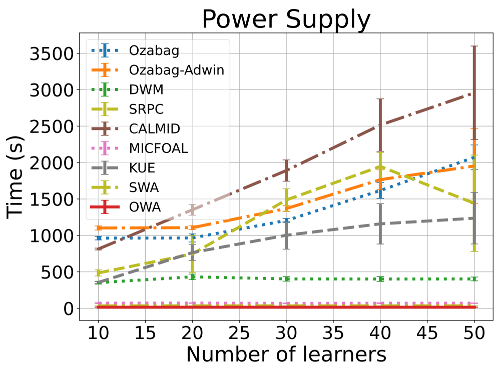
(e)



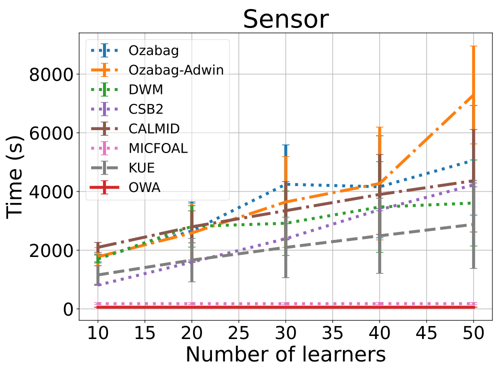
(f)



(g)

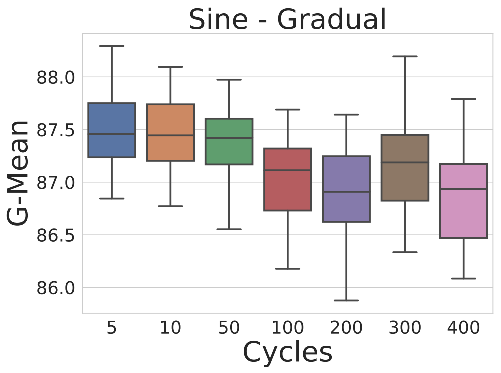


(h)

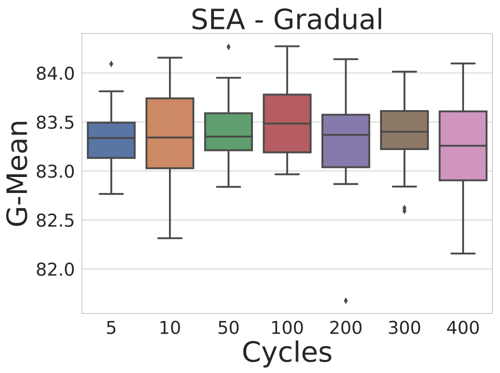


(i)

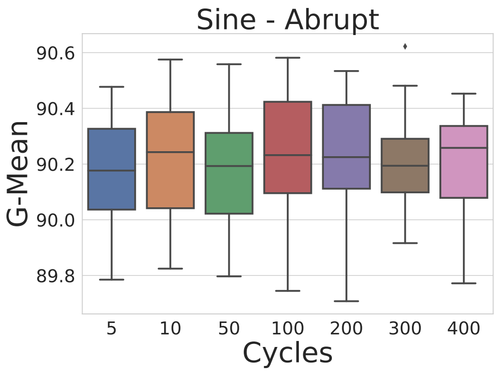
Figure 16: Plots of computational time required by the best run of each of the methods for real-world data streams.



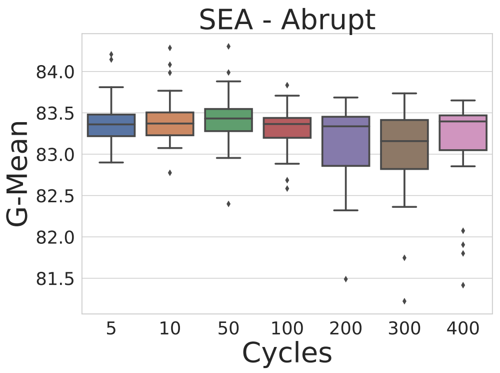
(a)



(b)



(c)



(d)

Figure 17: Box plots of G-Mean for several cycle lengths in OWA.

the computational time was marginal. For instance, for the Sine data stream with abrupt concept drifts, the choice of C in $[1, 500]$ caused around a 0.2s variation in computational time. Thus, in terms of efficiency, OWA was the statistically least time-consuming (together with SWA) and this result was robust to different choices of C.

According to our preliminary experiments, the choice of θ has little impact on the performance (typically less than 1% variation in G-Mean) as long as a relatively small number is chosen, because it should be less than the length of the first concept of the stream. F controls the extension of performance degradation that is allowed in OWA, leading to the model being reset if this threshold is exceeded. Both of these hyperparameters are problem-dependent and should be optimized, for example, via random search or bayesian optimization. According to our preliminary experiments, we suggest tuning these both of these hyperparameters in $[1, 100]$. We have also found that OWA is sensitive to the size of the MLP, which is also true for the ensemble techniques that employ several MLPs.

7. Conclusions

Typical data stream learning scenarios require high predictive performance and low computational time to handle the high volumes of incoming data or to operate in applications with strict time requirements. Often, in such scenarios, concept drifts pose additional challenges for learning algorithms, which have to adapt to potentially sudden and severe target changes. This is usually tackled by ensemble learning approaches, whose computational time cost is high. In this work, we proposed the first online model compression approach for nonstationary data streams. Our comprehensive experiments with artificial and real-world data streams from multiple domains and different types of concept drifts show that OWA was able to deliver similar or better predictive performance to existing data stream ensemble learning methods, while requiring significantly less computational time to be trained. Such ensemble-like performance is due to its optimization via averaging weights through time, which finds solutions in wide regions of weight space. The larger width of solutions leads to robust generalization. Its superior time efficiency is due to the use of a single model to represent an ensemble, instead of maintaining a collection of base learners. In future studies, we aim to further investigate the influence of adaptive learning rates and momentum to improve OWA's recovery time from concept drifts, and the impact of averaging parameters over time in other types of learning model, such as decision trees.

Acknowledgments

This work was funded by the National Council for Scientific and Technological Development – CNPq.

References

[1] Vijay Janapa Reddi. *Machine Learning Systems – Principles and Practices of Engineering Artificially Intelligent Systems*. Last updated in 2024.

[2] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[3] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, and M. Wozniak. Ensemble learning for data stream analysis: a survey. *Information Fusion*, 37:132–156, 2017.

[4] Heitor Murilo Gomes, Jean Paul Barddal, Fabricio Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys*, 50(2):23.1–23.36, 2018.

[5] Heitor Murilo Gomes, Jesse Read, and Albert Bifet. Streaming random patches for evolving data stream classification. In *2019 IEEE International Conference on Data Mining*, pages 240–249, November 2019.

[6] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A Survey of Model Compression and Acceleration for Deep Neural Networks. *arXiv:1710.09282*, June 2020.

[7] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. In *British Machine Vision Conference*, July 2015.

[8] Ke Tan and DeLiang Wang. Towards Model Compression for Deep Learning Based Speech Enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1785–1794, 2021.

[9] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8803–8812, 2018.

[10] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. *arXiv:1803.05407*, February 2019.

[11] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44.1–44.37, March 2014.

[12] Bruno Veloso, João Gama, Benedita Malheiro, and João Vinagre. Hyperparameter self-tuning for data streams. *Information Fusion*, 76:75–86, dec 2021.

[13] Andri Ashfahani and Mahardhika Pratama. *Autonomous Deep Learning: Continual Learning Approach for Dynamic Environments*, pages 666–674. Society for Industrial and Applied Mathematics, Philadelphia, PA, May 2019.

[14] Mahardhika Pratama, Choiru Za'in, Andri Ashfahani, Yew Soon Ong, and Weiping Ding. Automatic construction of multi-layer perceptron network from streaming examples. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1171–1180, November 2019.

[15] Peng Zhao, Xinqiang Wang, Siyu Xie, Lei Guo, and Zhi-Hua Zhou. Distribution-free one-pass learning. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):951–963, 2021.

[16] Weike Liu, Cheng Zhu, Zhaoyun Ding, Hang Zhang, and Qingbao Liu. Multiclass imbalanced and concept drift network traffic classification framework based on online active learning. *Engineering Applications of Artificial Intelligence*, 117:105607.1–105607.17, 2023.

[17] J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(91):2755–2790, 2007.

[18] Dariusz Brzezinski and Jerzy Stefanowski. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265:50–67, 2014.

[19] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

[20] Maroua Bahri, Heitor Murilo Gomes, Albert Bifet, and Silviu Maniu. CS-ARF: Compressed adaptive random forests for evolving data stream classification. In *2020 International Joint Conference on Neural Networks*, pages 1–8, July 2020.

[21] Nikunj Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345, 2005.

[22] Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448, 2007.

[23] Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012.

[24] Chun Wai Chiu and Leandro L. Minku. Diversity-based pool of models for dealing with recurring concepts. In *2018 International Joint Conference on Neural Networks*, pages 1–8, July 2018.

[25] Chun Wai Chiu and Leandro L. Minku. A diversity framework for dealing with multiple types of concept drift based on clustering in the model space. *IEEE Transactions on Neural Networks and Learning Systems*, 33(3):1299–1309, 2022.

[26] Paulo Mauricio Gonçalves Jr and Roberto Souto Maior de Barros. RCD: A recurring concept drift framework. *Pattern Recognition Letters*, 34(9):1018–1025, 2013.

[27] Alberto Cano and Bartosz Krawczyk. Kappa updated ensemble for drifting data stream mining. *Machine Learning*, 109(1):175–218, January 2020.

[28] Boyu Wang and Joelle Pineau. Online Bagging and Boosting for Imbalanced Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3353–3366, December 2016.

[29] Weike Liu, Hang Zhang, Zhaoyun Ding, Qingbao Liu, and Cheng Zhu. A comprehensive active learning method for multiclass imbalanced data streams with concept drift. *Knowledge-Based Systems*, 215:106778.1–106778.15, 2021.

[30] Shupeng Gui, Haotao Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[31] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2285–2294, July 2015.

[32] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. *arXiv:1804.03235*, August 2020.

[33] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, 2018.

[34] Devesh Walawalkar, Zhiqiang Shen, and Marios Savvides. Online Ensemble Model Compression Using Knowledge Distillation. In *Computer Vision – ECCV 2020*, volume 12364, pages 18–35, 2020.

[35] Dihia Boulegane, Vitor Cerquiera, and Albert Bifet. Adaptive model compression of ensembles for evolving data streams forecasting. In *2022 International Joint Conference on Neural Networks*, pages 1–8, July 2022.

[36] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.

[37] R. Agrawal, T. Imielinski, and A. Swami. Database mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, 1993.

[38] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001.

[39] Jeffrey C. Schlimmer and Richard H. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, March 1986.

[40] Michael Harries. SPLICE-2 comparative evaluation: Electricity pricing. Technical report, University of New South Wales, School of Computer Science and Engineering, 1999.

[41] National Oceanic and Atmospheric Administration (NOAA). Fed. Climate Complex Global Surface Summary of Day Data - Version 7 - USAF Datsav3 Station n. 725540, 2012. [Online; accessed 2020-02-01].

[42] Indrė Žliobaitė. Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis*, 15(4):589–611, December 2011.

[43] Vinícius M. A. Souza, Diego F. Silva, João Gama, and Gustavo E. A. P. A. Batista. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 873–881, June 2015.

[44] Kevin Killourhy and Roy Maxion. Why did my detector do that?! Predicting keystroke-dynamics error rates. In *Proceedings of the 13th International Conference on Recent Advances in Intrusion Detection*, pages 256–276, 2010.

[45] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017.

[46] X. Zhu. Stream data mining repository, 2010. Accessed online on 2020-02-01, `http://www.cse.fau.edu/~xqzhu/stream.html`.

[47] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[48] Yanmin Sun, Mohamed Kamel, and Yang Wang. Boosting for learning multiple classes with imbalanced class distribution. In *Sixth International Conference on Data Mining*, pages 592–602, December 2006.

[49] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, December 2006.

[50] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[51] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, dec 2019.

[52] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 31, 2018.