Evolving Spiking Neural Networks for online learning over drifting data streams

Jesus L. Lobo, Ibai Laña, Javier Del Ser, Miren Nekane Bilbao, Nikola Kasabov

PII: S0893-6080(18)30213-2
DOI: https://doi.org/10.1016/j.neunet.2018.07.014
Reference: NN 3997

To appear in: *Neural Networks*

Received date : 5 April 2018
Revised date : 11 June 2018
Accepted date : 25 July 2018

Evolving Spiking Neural Networks for Online Learning over Drifting Data Streams

Jesus L. Lobo[a,*], Ibai Laña[a], Javier Del Ser[a,b,c],
Miren Nekane Bilbao[b], Nikola Kasabov

[a]*TECNALIA, 48160 Derio, Spain.*
[b]*University of the Basque Country UPV/EHU, 48013 Bilbao, Spain*
[c]*Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain*
[d]*KEDRI - Auckland University of Technology (AUT), 1010 Auckland, New Zealand*

Abstract

Nowadays huges volumes of data are produced in the form of fast streams, which are further affected by non-stationary phenomena. The resulting lack of stationarity in the distribution of the produced data calls for efficient and scalable algorithms for online analysis capable of adapting such changes (concept drift). The online learning field has lately turned its focus on this challenging scenario, by designing incremental learning algorithms that avoid becoming obsolete after a concept drift occurs. Despite the noted activity in the literature, a need for new efficient and scalable algorithms that adapt to the drift still prevails as a research topic deserving further effort. Surprisingly, Spiking Neural Networks, one of the major exponents of the third generation of artificial neural networks, have not been thoroughly studied as an online learning approach, even though they are naturally suited to easily and quickly adapting to changing environments. This work covers this research gap by adapting Spiking Neural Networks to meet the processing requirements that online learning scenarios impose. In particular the work focuses on limiting the size of the neuron repository and making the most of this limited size by resorting to data reduction techniques. Experiments with synthetic and real data sets are discussed, leading to the empirically validated assertion that, by virtue of a tailored exploitation of the neuron repository, Spiking Neural Networks adapt better to drifts, obtaining higher accuracy scores than naive versions of Spiking Neural Networks for online learning environments.

Keywords: Spiking neural networks, data reduction, online learning, concept drift

Corresponding author: jesus.lopez@tecnalia.com (Jesus L. Lobo). TECNALIA. P. Tecnologico Bizkaia, Ed. 700, 48160 Derio, Spain. Tl: +34 946 430 50. Fax: +34 901 760 009.

1. Introduction

With the increasing number of applications based on fast-arriving information flows and applied to real scenarios (Zhou, Chawla, Jin & Williams, 2014; Alippi, 2014), concept drift has become a paramount issue for online learning environments. The distribution modeling data captured by sensor networks, mobile phones, intelligent user interfaces, industrial machinery and others alike is usually assumed to be stationary along time. However, in many real cases such an assumption does not hold, as the data source itself is subject to dynamic externalities that affect the stationarity of its produced data stream(s), e.g. seasonality, periodicity or sensor errors, among many others. As a result, possible patterns behind the produced data may change over time, either in the feature domain (new features are captured, part of the existing predictors disappear, or their value range evolves), or in the class domain (new classes emerge from the data streams, or some of the existing ones fade along time). This paradigm is what the literature has coined as *concept drift*, where the term *concept* refers to a stationary distribution relating a group of features to a set of classes.

When the goal is to infer the aforementioned class patterns from data (online classification), incremental models trained over drifting streams become obsolete when transitioning from one concept to another. Consequently, they do not adapt appropriately to the new emerged data distribution, unless they are modified to handle efficiently this unwanted effect. In order to minimize the impact of concept drift on the performance of predictive models, recent studies (Ditzler,Roveri, Alippi & Po
Webb, Hyde, Cao, Nguyen & Petitjean, 2016;
Khamassi, Sayed-Mouchaweh, Hammami & Ghédira, 2018) have been focused on the development of efficient techniques for continuous adaptation in evolving environments or, alternatively, in the incorporation of drift detectors and concept forgetting mechanisms (Žliobaitė, Pechenizkiy & Gama, 2016).

Indeed, online learning in the presence of concept drift has been a very hot topic during the last few years (Gama, Žliobaitė, Bifet, Pechenizkiy & Bouchachia, 2014; Ditzler, Roveri, Alippi & Polikar, 2015; Alippi, 2014; Lobo, Del Ser, Bilbao, Perfecto & Salcedo-Sanz, 2017), and still remains under active debate in the community (Khamassi, Sayed-Mouchaweh, Hammami & Ghédira, 2018) due to the fact that there are many relevant open challenges to tackle (Barddal, Gomes,Enembreck & Pfahı
Gomes, Barddal, Enembreck & Bifet, 2017a;
Krawczyk, Minku, Gama, Stefanowski & Woźniak, 2017; Wang, Minku & Yao,2018)
. Online learning environments impose a set of stringent computational restrictions that every new technique in the field should embrace (Domingos &Hulten, 2003)
, enforcing this technique to use mechanisms to adapt to the new concept when drift occurs with a fast recovery (plasticity) while, at the same time,

retaining the acquired knowledge of the old concept (stability). As dictated by the so-called *stability-plasticity dilemma* (Grossberg, 1988), a learning model should have the ability to retain acquired knowledge and also learn new concepts, but in no way could a model could be designed to do both equally well. While the ability to accumulate knowledge of the old concept is really treasured over stable data distributions (there is no drift), plasticity periods (right after drift occurs) require forgetting part or all previous concepts in order to capture the new upcoming concept as fast as possible.

The adaptation to the drift can be carried out proactively detecting first the concept drift, and only the model gets updated when a drift is detected (active approaches), or updating the model continuously every time new data samples are received (passive approaches). Focusing on active adaptation, three main mechanisms can be noted in the literature:

- Windowing: a sliding window over the last data instances is used as the exclusive training set for the learning algorithm (Bifet & Gavalda, 2007; Alippi & Roveri, 2008; Gama, Medas, Castillo & Rodrigues, 2004; Bifet & Gavalda, 2006).
- Weighting: all available samples are considered, but suitably weighted according to their age or relevance in terms of classification accuracy (Cohen& Strauss, 2003; Klinkenberg, 2004).
- Reservoir sampling: a subset of instances is selected for training, and randomly drawn examples from within the reservoir are discarded upon receiving new data (Vitter, 1985; Aggarwal, 2006; Ng & Dash, 2008).

Disregarding the particular active adaptation mechanism selected for concept drift tackling, the goal from a computing perspective is to develop efficient and scalable learners to meet online processing restrictions, namely(Domingos & Hul-ten, 2003)
:

- Each sample must be processed only once "on arrival".
- The processing time of each data sample must be small and constant.
- The algorithm should only use a preallocated amount of main memory.
- A valid model must be available at every scan of the data stream.
- The algorithm must produce a model that is equivalent to the one that would be produced by a batch processing algorithm.

Unfortunately, most off-the-shelf classification models need to be retrained if they are used in a changing environment and fail to scale properly. One of the most promising machine learning techniques in the field that can overcome this noted drawback is the Spiking Neural Network (SNN) (Gerstner & Kistler, 2002).

The advent of SNNs was propelled by the need for a better understanding of the information processing skills of the mammalian brain, for which the community committed itself to the development of more complex biologically connectionist systems. SNNs have revealed themselves as one of the most successful approaches to model the behavior and learning potential of the brain, and exploit them to undertake practical learning tasks. In essence SNNs leverage spike information representation so as to construct spike-time learning rules that capture temporal associations between a large number of temporal variables in streaming data. Among other applications (e.g. simulation of space-time systems), such a learned knowledge can be exploited to predict future events. In fact, SNNs must be regarded as a portfolio of models for different computational uses and applications, all inspired by the the same design principles (information encoding and neural processing based on time spikes). One of the successful SNNs is the Evolving Spiking Neural Networks (eSNNs) (Soltic & Kasabov, 2010; Schliebs & Kasabov, 2013), where the number of spiking neurons evolves incrementally in time to infer temporal patterns from data. First proposed in (Kasabov, 2007; Wysoski, Benuskova & Kasabov, 2006), eSNNs are based on the principles of evolving connectionist systems (ECOS) and Thorpe's neural model (Thorpe &Gautrais, 1998) . In SNNs, changes in the input stream data are encoded immediately as binary events - spikes. As we will motivate in forthcoming sections, they use one of the most suitable data encoding strategies for adapting to drifts.

Several works have focused on implementing SNNs for online learning environments. The most early attempt in this regard is SpikeProp, which was proposed for training SNNs and similar in concept to the backpropagation algorithm developed for traditional neural networks (Bohte, Kok & La Poutré, 2000). However, it is too slow to be used in an online setting, and prone to getting stuck in local minima as a result of its gradient-based learning algorithm. In (Belatreche, Maguire& McGinnity, 2007 the authors proposed a derivative-free supervised learning algorithm comprising an evolutionary strategy with a reportedly better performance than SpikeProp, but the training process was extremely time-consuming and hence, not suitable for online learning. ReSuMe (Ponulak, 2005, 2008; Ponu-lak & Kasi ński, 2010) integrated the idea of learning-windows with remote supervision; despite this method was claimed to be suitable for online learning, the network structure used in this method is fixed and does not adapt to incoming stimuli. In addition, the desired precise output spike timing is crucial to ReSuMe learning (Wang, Belatreche, Maguire & McGinnity, 2017). Other studies have addressed the online learning in a more realistic approach. The method proposed in (Wysoski, Benuskova & Kasabov, 2010) can perform learning in an online mode through synaptic plasticity and adaptive network structure. More recently, the

SpikeTemp method proposed in (Wang, Belatreche, Maguire & McGinnity, 2017) offers an enhanced rank-order-based learning method for SNNs with an adaptive structure where the precise times of incoming spikes are used to determine the required change in synaptic weights. With these few exceptions, there is a lack of efficient and scalable SNN-based algorithms in online learning scenarios.

Interestingly for the scope of this work, none of the contributions reviewed above takes into account the requirement of a limited size for the neuron repository, which is of utmost importance to meet the processing requirements of online learning. Should this crucial aspect not be taken into account, the number of neurons in the repository would increase every time step. Therefore, further efforts are still needed to devise new online learning mechanisms for SNNs and increase their applicability to real-world problems. To this end not only the SNN model should learn incrementally from the data stream, but the content of its neuron repository should also be limited in size and adapted when concept drift occurs. To the best of our knowledge, these two issues have not been addressed in the community. The contribution of our work can be summarized as follows:

1. We develop a new eSNN model that incorporates a set of novel ingredients for efficiently dealing with online learning applications, such as a limitation of the size of the neuron repository and the use of a sliding window. The developed model is able to classify inputs after just one presentation of the training samples, without requiring the entire training set to be available in advance. Regarding the limited size of the neuron repository, our work will embrace the adoption of Data Reduction Techniques (DRTs) (Triguero, García & Herrera, 2010), which aim to obtain a representative training set with a lower size when compared to the original one, and with similar or even higher generalization capability when fed to a predictive model. They can be divided into prototype selection (PS) techniques (Li & Wang, 2015; Meena & Devi, 2015), which consist of choosing a subset of the original training data; and prototype generation (PG) techniques (Triguero, Derrac, Garcia & Herrera, 2012; Hu & Tan, 2016; Escalante, Graff & Morales-Reyes, 2016), which build new artificial prototypes to better adjust the decision boundaries between classes.
2. We hybridize our proposed eSNNs approach with a drift detector, yielding a solid learning model to be deployed in a realistic online scenario.
3. We analyze the impact of different data reduction techniques in the newly devised eSNN model over a wide range of online learning datasets, with emphasis on the predictive performance of the model after a drift occurs, the data reduction percentage achieved by every DRT, and the implications of the proposed strategy in the future of the online learning field.
4. As a result of this research work, we provide an online learning technique

based on a single classifier, proven to perform competitively in comparison with other techniques based on ensembles while using less storage capacity. Single classifier approaches are widely regarded as an attractive solution for massive data streams due to their reduced computational cost when compared to their ensemble counterparts (Ditzler, Roveri, Alippi & Polikar, 2015).

The rest of the paper is organized as follows: first, Section 2 provides an general introduction to the evolving spiking neural networks and their relevance in online learning scenarios. Section 3 delves into the data reduction methods used in the proposed approach. Section 4 provides a detailed description of the proposed approach, while Section 5 presents the experimental setup designed to assess its performance. Sections 6 and 7 present and discuss the obtained results from such experiments and finally, Section 8 draws concluding remarks and outlines future research lines related to this work.

2. Evolving Spiking Neural Network (eSNN)

An eSNN consists of an encoding part, which transforms a real-valued vector into spikes generated over time, a neuron model, and a learning mechanism that calculates the connection weights between the input and the output neurons. eSNNs are now a part of a comprehensive SNN architecture for spatio-temporal data modeling, NeuCube (Kasabov, 2014), which is able to deal with a wide range of applications (Kasabov, Scott, Tu, Marks, Sengupta, Capecci, Othman, Dobor-jeh, Murli, Hartono
.

2.1. Architecture

As depicted in Figure 1, the architecture of the eSNN is composed by three layers (Wysoski, Benuskova & Kasabov, 2010, 2006; Kasabov, 2007; Kasabov, Dhoble, Nuntalid & Indiveri, 2013). The first layer corresponds to the input data. The second layer is for encoding purposes, where the real values of the features of every sample are encoded as trains of spikes, generating the pre-synaptic neurons and each of them having a receptive field. Receptive fields of neighboring neurons overlap with each other by adopting the shape of Gaussian or Logic functions, in all cases covering the whole range of the values of each feature (as explained below). The number of encoding neurons N (or receptive fields) may vary depending on the nature of the data at hand, and must be tuned for achieving a good predictive performance of the overall model. The third layer is the evolving output layer, where a repository of spiking neurons representing samples (divided in one subrepository per every class in the problem) evolves as new data arrive. Each output neuron is linked to all input neurons through connections whose weights are learned from the data instances fed to the model.

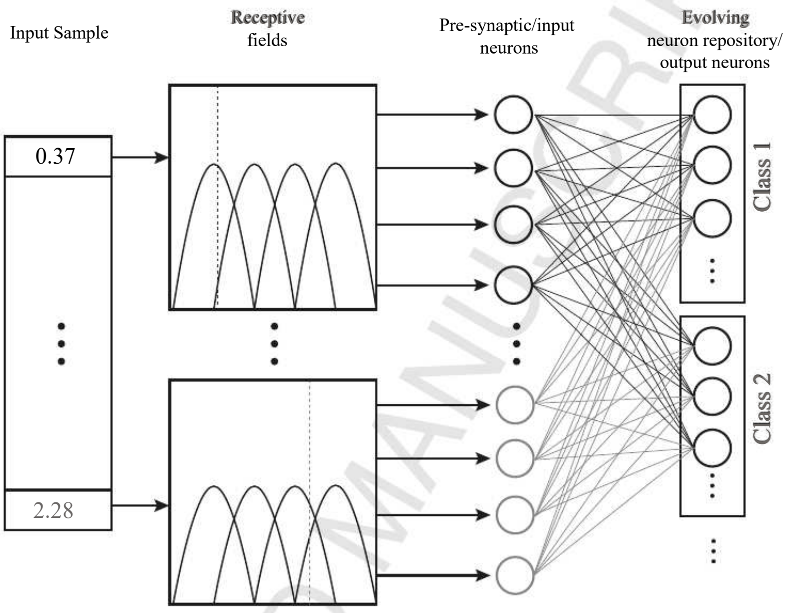

Input Sample
Receptive fields
Pre-synaptic/input neurons
Evolving neuron repository/ output neurons
0.37
2.28
Class 1
Class 2

Figure 1: Architecture of feed-forward eSNN (Kasabov, 2007).

2.2. *Neural Encoding*

In order to learn from real-valued data, each sample is encoded to a sequence of spikes over time (spike trains) by using a neural encoding technique, e.g. rank order population (Thorpe & Gautrais, 1997; Bohte, Kok & La Poutre, 2002) or any other encoding approach alike. The rank order population scheme works on the basis of the order of the spikes across all the synapses connected to the particular neuron. It creates the priority in the input spikes depending on the order of spike arrival to the neuron, which provides extra information to the network regarding the order of the spike (Thorpe & Gautrais, 1998). In this work we adopt the Gaussian Receptive Field (GRF) population encoding scheme, where the input can be distributed over several neurons with overlapping and graded sensitivity profiles by using Gaussian activation functions. Each encoding neuron is fired only once during the time coding interval T. As a result, each input sample is translated into a spatio-temporal spike pattern. Specifically, the center C_j and the width W_j of each GRF of pre-synaptic neuron j are computed as

$$C_j = I^n_{min} + \frac{2j-3}{2}\left(\frac{I^n_{max} - I^n_{min}}{N-2}\right) \tag{1}$$

and

$$W_j = \frac{1}{\beta}\left(\frac{I^n_{max} - I^n_{min}}{N-2}\right), \tag{2}$$

where N is the number of receptive fields, whose value impacts on the amplitude of the input neuron and must be optimized for the problem. The range of the n-th input variable is assumed to be $[I^n_{min}, I^n_{max}]$. Parameter $\beta \in [1, 2]$ (also referred to as overlap factor) establishes the width of receptive fields, thereby their amount of overlapping and ultimately, in the firing time of the pre-synaptic neuron j. The output of neuron j is defined as

$$output_j = \exp\left(-\frac{(x - C_j)^2}{2W_j^2}\right), \tag{3}$$

where x is the input value. The firing time of each pre-synaptic neuron j is defined as

$$T_j = \lfloor T(1 - output_j)\rfloor \tag{4}$$

where T is the simulation or spike interval. Figure 2 exemplifies the GRF encoding process for the feature of any given sample.

2.3. Neural Model

A simplified Leaky Integrate-and-Fire (LIF) model was formally proposed in (Thorpe & Gautrais, 1998), but the idea can be tracked to publications as early as 1990. LIF states that the spike response of a neuron depends only on the arrival time of pre-synaptic spikes, that is, the earlier the spike arrives to a neuron, the stronger its weight will be when compared to a later spike. Each neuron in this model can spike at most once, and a neuron fires when its Post-Synaptic Potential (PSP) reaches its threshold value. The PSP of a neuron i is defined as

$$PSP_i = \begin{cases} 0, & \text{if fired,} \\ \sum_j w_{ji} \cdot mod^{order(j)}, & \text{otherwise,} \end{cases} \tag{5}$$

where $w_{j,i}$ represents the weight of the synaptic connection between pre-synaptic neuron j to output neuron i; mod is the modulation factor with a range $[0, 1]$; and $order(j)$ defines the rank of the pre-synaptic neurons spike. The first rank is assigned as 0 and subsequently, rank is increased by 1 based on firing time of each pre-synaptic neuron.

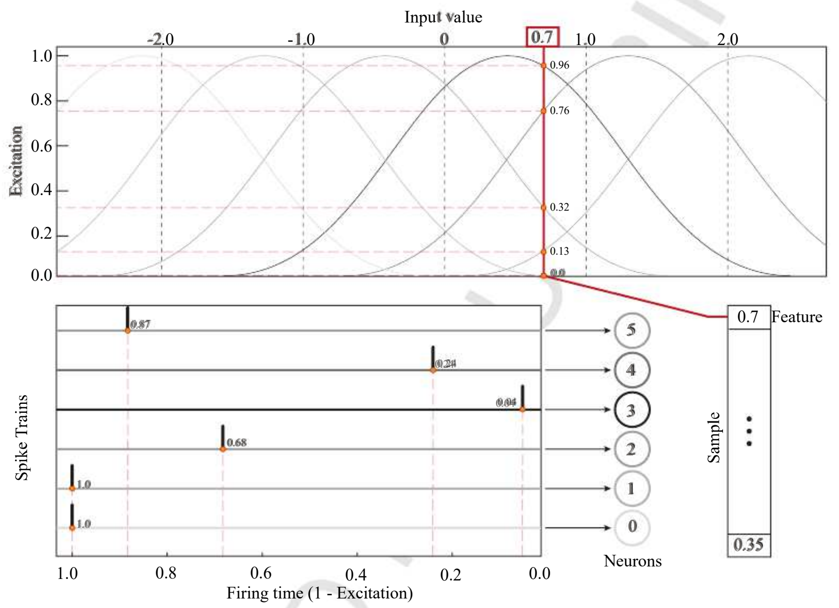


Figure 2: Example of population encoding based on 6 GRFs. For an input value of 0.7 (bold straight line) the intersection points with each GRF are computed $(0.96, 0.76, 0.32, 0.13, 0.0, 0.0)$, which are in turn translated into firing times $(0.04, 0.24, 0.68, 0.87, 1.0, 1.0)$.

2.4. Supervised Learning

When utilized in a supervised learning setting, the aim of the eSNN learning method is to produce and update a repository of output neurons, each of them labeled with a certain class label.

In this classification context, the eSNN training algorithm is algorithmically described in Algorithm 1. First, the model creates a repository of output neurons for the training patterns. For each pattern that belongs to a same given class, a new output neuron is created and connected to all pre-synaptic neurons in the previous layer through weights w_{ji} (see Figure 1). The value of $w_{j,i}$ is calculated based on the spike order through a synapse j as $w_{j,i} = mod^{order(j)}$, where j is the pre-synaptic neuron of the output neuron i (line 7). A numerical threshold γ_i is set for the newly created output neuron as the fraction $C \in (0, 1)$ of its maximum post-synaptic potential $PSP_{max,i}$, i.e. $\gamma_i = PSP_{max,i} \cdot C$. The weight vector of a newly created output neuron is then compared with the already trained output neurons in the repository. If the Euclidean distance between the newly created output neuron weight vector and that of anyone of the already trained output neurons is smaller than a similarity parameter (*SIM*), they are considered to be similar. As a result, their thresholds and weight vectors are merged according to

$$w_{j,i} = \frac{w_{new} + (w_{j,i} \cdot M)}{M+1}, \tag{6}$$

and

$$\gamma_i = \frac{\gamma_{new} + (\gamma_i \cdot M)}{M+1}, \tag{7}$$

where M is the number of previous merges of similar neurons through the learning history of the eSNN. After merging, the weight vector of the newly created output neuron is discarded, and the new pattern is presented to the model. If none of the already trained neurons in the repository is found to be similar (as per the *SIM* parameter) to the newly produced output neuron, then it is added to the repository.

The testing phase is carried out by propagating the spikes that encode the test sample to all trained output neurons. The class label for the test sample is assigned according to the class label of the output neuron which has fired first after reaching its threshold value γ_i.

2.5. eSNN in Online Learning

The neural model of eSNN models allows for a very fast real-time simulation of large networks and a low computational cost. These properties make eSNNs a very suitable candidate for online learning scenarios, where stringent restrictions on computational cost and processing time prevail. Besides, the evolving nature of the network makes it possible to accumulate knowledge as data become available, without the requirement of storing and retraining the model with past samples. We will later evince how this evolving nature is also useful for adapting the model to eventual drifts along the stream.

Several approaches have been developed so far in order to adapt eSNNs to online learning setups (Wysoski, Benuskova & Kasabov, 2006; Soltic, Wysoski & Kasabov, 2008; Alnajjar, Zin & Murase, 2008; Ponulak & Kasiński, 2010). However, most of them are unable to predict inputs after just one presentation of the training samples, hence requiring the entire training set to be available in advance. The online learning field has not certainly been as thoroughly addressed in the eSNN literature as its offline (batch) counterpart. The reason for this lack of research lies on the aforementioned computational restrictions, which require adapting the original eSNN learning algorithm to meet these design constraints.

In online learning scenarios tailored predictive scores metric are often proposed to shed light not only on the net accuracy of online learning models, but

Algorithm 1: eSNN algorithm

```
1  Initialize neuron repository, $NR = \{\}$
2  Set eSNN parameter $mod = [0, 1]$, $C = [0, 1]$, $SIM = [0, 1]$
3  Set eSNN encoding parameters $\beta, T, N$
4  Calculate $I_{max}, I_{min}$ for the training data set
5  for every sample s belonging to class c do
6      Calculate $C_j$ and $W_j$ for encoding s into firing time of multiple
         pre-synaptic neurons $j$
7      Create a new output neuron i and the connection weights as
         $w_{j,i} = mod^{order(j)}$
8      Calculate $PSP_{max(i)} = \Sigma_j w_{j,i} \cdot mod^{order(j)}$
9      Compute PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
10     if min distance(Newly output neuron weight vector,Neuron repository
         weight vectors in in NR)) $\leq$ SIM then
11         Update the weight vector and threshold of the most similar neuron
             $w_{j,i} = \frac{w_{new}+(w_{j,i} \cdot M)}{M+1}$ and $\gamma_i = \frac{\gamma_{new}+(\gamma_i \cdot M)}{M+1}$
12         Set $M = M + 1$
13     else
14         Add the weight vector and threshold of the new output neuron to NR
15     end
16 end
17 Go to 1 and repeat for all target classes
```

also to quantitatively analyze its reaction capability against changes in the data streams. Discussions in this paper will follow the common practice in this research area by embracing the so-called *prequential accuracy* metric, proposed in (Dawid, Vovk et al., 1999) and widely used by the community (Minku & Yao,2012) . This metric quantifies the sample-by-sample progression of the average accuracy obtained by a learning model in a test-then-train basis (i.e. null verification latency), and is given by

$$preACC(t)=\begin{cases} preACC_{ex}(t), & \text{if } t = t_{ref}, \\ preACC_{ex}(t\text{-}1) + \dfrac{preACC_{ex}(t) - preACC_{ex}(t\text{-}1)}{t - t_{ref} + 1}, & \text{otherwise,} \end{cases}$$

where t is the time between samples in the stream; $preACC_{ex}(t) = 0$ if the prediction of the test example at time t before its learning is wrong; and $preACC_{ex}(t) = 1$ if it is correct. Here, t_{ref} serves as a reference time that fixes the first time step used in the calculation. This reference time allows isolating the computation of the prequential accuracy before and after a drift has started, so that insight on the reaction of different drift handling strategies can be assessed.

3. Data Reduction Techniques

In eSNN, if the Euclidean distance between the newly created output neuron weight vector and that of anyone of the already trained output neurons is smaller than *SIM*, they are considered to be close and they are merged. Closeness is decided based on the similarity between the weight vectors $\{w_{j,i}\}$ of every output neuron i to the weight vector produced by the newly input sample. This distance-based prediction strategy can be regarded as that of the k-Nearest Neighbors (kNN), one of the most utilized algorithms in machine learning due to its simplicity and effectiveness. However, kNN models are known to suffer from several drawbacks (Kononenko & Kukar, 2007):

- The need for a high storage capacity to retain the set of training samples so as to perform the decision rule;
- the computational burden associated to the search for the closest example, due to multiple similarity computations between the test sample and the training examples; and
- the low tolerance of noisy samples within the training data.

The literature is rich in methods proposed to tackle the above drawbacks. In this work we will embrace data reduction techniques (DRTs, also referred to as instance selection approaches) to simultaneously face all such issues in an online setting. Data reduction techniques (DRTs) aim to obtain a representative training set with a lower size compared to the original one, yet with similar or even higher classification accuracy for new incoming data. DRTs can be classified depending on whether the method at hand is based on filtering and selecting samples from the training set (prototype selection, PS) or, alternatively, on synthesizing representative examples therefrom (prototype generation techniques, PG). For the sake of completeness, we next overview both categories, and we show how DRTs can be applied to reduce the number of output neurons in the repository providing an useful mechanism to limit the size of the repository.

3.1. Prototype Selection Techniques

Prototype selection techniques select a subset of the original training data for constructing the model, hence discarding the remaining data samples. The main advantage of these techniques is their capacity to discriminate relevant examples without synthesizing artificial data. A widely used categorization of PS techniques include edition, condensation, and hybrid methods (Garcia, Derrac, Cano& Herrera, 2012) . Edition methods remove noisy instances in order to increase the classification performance. Condensation methods remove superfluous samples that do not affect the classification performance. Hybrid methods are based on combining edition and condensation methods to yield a PS technique leveraging specific computational and/or performance aspects of both approaches.

The exhaustive study of PS techniques presented in (Garcia, Derrac, Cano & Herrera, 2012) shows the advantages and disadvantages of all DRT methods falling in this category. Indeed this work empirically proved that the choice of a certain method depends on diverse factors, in essence a multi-criteria decision that becomes even more crucial when dealing with online learning in the presence of concept drift. For example, an edition method usually outperforms a naive kNN in the presence of noise, but only a few instances will be removed (Garcia, Derrac, Cano & Herrera, 2012). However, although they allow for a high data reduction rate while preserving the model accuracy, edition-based PS techniques are usually the slowest ones due to their greedy search procedure. On the contrary, condensation PS approaches are fast and achieve high data reduction rates, but they usually render classification performance scores lower than those of naive kNN schemes (Garcia, Derrac, Cano & Herrera, 2012).

Figures 3 and 4 show the behavior of the PS techniques on a synthetic dataset, illustrating the nature of decision boundaries after applying the sample reduction techniques.

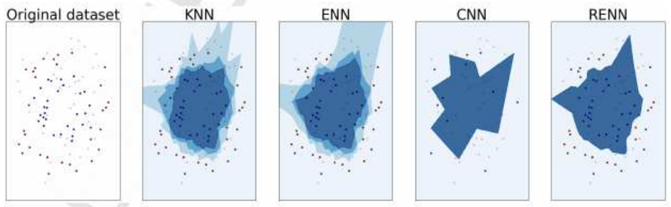


Figure 3: A comparison of the mean accuracies (95.00%, 82.50%, 87.50%, 77.50%) and data reduction percentages (0.00%, 11.67%, 78.33%, 16.67%) of kNN, ENN, CNN, and RENN techniques respectively on a synthetic data set. The figure shows training points in solid colors and testing points semi-transparent.

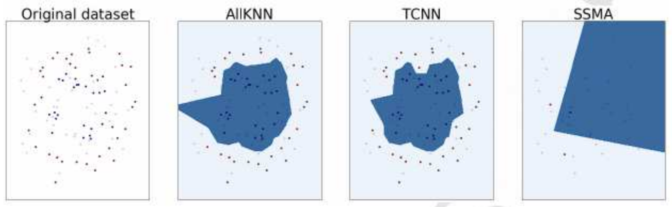


Figure 4: A comparison of the mean accuracies $(92.50\%, 90.00\%, 85.00\%)$ and data reduction percentages $(18.33\%, 10.00\%, 91.67\%)$ of AllKNN, TCNN, and SSMA techniques respectively on a synthetic data set. The figure shows training points in solid colors and testing points semi-transparent.

3.2. Prototype Generation Techniques

Prototype generation techniques build new artificial prototypes to better adjust the decision boundaries between classes in kNN classifiers. To this end, PG methods produce and replace training data samples with new artificial data filling regions in the domain of the problem lacking representative samples in the original dataset. The thorough categorization and empirical assessment of PG techniques reported (Triguero, Derrac, Garcia & Herrera, 2012) drew similar conclusions to those obtained for PS schemes in (Garcia, Derrac, Cano & Herrera, 2012): a categorical claim cannot be made in regards to the comparative performance of different PG techniques: the choice of one approach or another will roughly depend on the problem under consideration.

Figure 5 shows the behavior of the PG techniques on a synthetic dataset, illustrating the nature of decision boundaries after applying the sample reduction techniques.

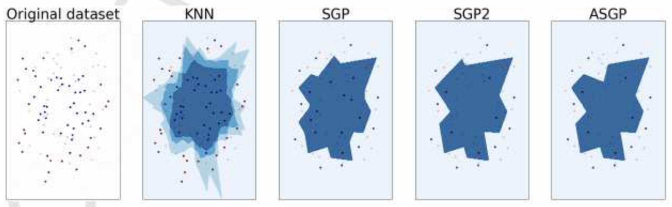


Figure 5: A comparison of the mean accuracies $(95.00\%, 92.50\%, 85.00\%, 90.00\%)$ and data reduction percentages $(0.00\%, 65.00\%, 81.67\%, 75.00\%)$ of kNN, SGP, SGP2 and ASGP techniques respectively on a synthetic data set. The figure shows training points in solid colors and testing points semi-transparent.

As it will be further explained, the reduction power of DRTs at the same time that their accuracy remains competitive will be used in favor of our proposed approaches.

4. Proposed Approach: Online Evolving Spiking Neural Network (OeSNN)

As in other online learning algorithms, our online version of eSNN (OeSNN) stores a reduced number of samples taken over a $\mathcal{W}$-sized sliding window. In our case, however, such samples are not used for statistical tests (Bifet & Gavalda,2007) , but rather for performing the neural encoding procedure described in Section 2.2. Every time a new sample arrives in the stream, the neural encoding is performed for the samples falling in $\mathcal{W}$. This encoding is also used for the prediction of the test sample, following a *test-then-train* scheme (Bifet, Holmes,Pfahringer & Gavald . The value of $\mathcal{W}$ can be 1) set fixed, e.g. whenever a new sample arrives it is stored in the memory and the oldest one is discarded; or 2) adjusted over time depending on e.g. the information provided by a drift detector analyzing the statistical characteristics of the stream. A fixed size is often adopted as a baseline scheme when evaluating new online learning algorithms (Gama, Žliobaitė, Bifet, Pechenizkiy & Bouchachia, 2014); our study will follow this common practice from the related literature.

As explained in previous sections, the learning procedure of the eSNN relies mostly on the neuron repository. Therefore, the size of this repository should be upper bounded in order to meet the restrictive storage constraints imposed in an online learning scenario. The neuron repository collects all the available knowledge in the form of output neurons, so that the more knowledge (neurons) is stored in this model stage, the more likely it will be to find an output neuron similar to the one under test, and ultimately the better accuracy will be achieved. Consequently, the proposed OeSNN approach utilizes a fixed size of the neuron repository NR_size: the new output neuron produced by the test sample through the eSNN structure is stored whenever there is room in the repository, i.e. its current occupation CNR_size is below the net capacity of the reservoir NR_size; if there is no free space (corr. $CNR\_size = NR\_size$), the oldest output neuron will be replaced by the new output neuron. Intuitively, this repository updating strategy addresses two different constraints in online learning under concept drift: the need for a limited size reservoir of output neurons, and an inherent forgetting mechanism to discard outdated concepts when data streams are non-stationary. To the best of our knowledge there is no prior study elaborating on this dual byproduct of the eSNN repository.

Other related studies usually divide the data set into training and testing phases, applying the online learning to the test part after once a well-trained eSNN classifier has been attained (Schliebs & Kasabov, 2013; Wang, Belatreche, Maguire & McGinnity, 2017; Wang, Belatreche, Maguire & Mcginnity, 2014). However, in fast streaming scenarios the algorithm must update itself one sample at a time from the beginning of the data streaming process. This is accomplished by adopting incremental methods for warm-start model training while predicting test samples in parallel. This will be the scheme adopted in this work.

Algorithm 2 reflects the adaptations made to the original eSNN in Algorithm 1 in order to deploy it in online learning scenarios by fulfilling with the restrictions described before. The aforementioned $\mathcal{W}$-sized sliding window yields a group of recent samples from which the encoding parameters are computed (lines 4, 8, and 9). The neuron repository is limited to a fixed size (line 5), which is checked in order to decide whether the recent sample can be stored directly or instead, replaces the oldest output neuron in the repository.

At this point it is important to underline that every time a merging process is performed, only two neurons are involved at most. Therefore, it is likely that the output neuron repository stores redundant information when processing a stream, with emphasis during those periods where the data distribution remains stable, i.e. before the drift occurs and long after the drift event. Here lies the rationale for further optimizing the information stored within the OeSNN neuron repository by applying data reduction techniques, which is exposed in the next subsection.

4.1. Data Reduction Techniques for OeSNN Models: OeSNN-PS and OeSNN-PG

Our proposed OeSNN approach sketched in Algorithm 2 is the basis for our proposed approaches hybridized with DRTs schemes (OeSNN-PS and OeSNN-PG), in which PS and PG techniques are applied respectively on the neuron repository. For this purpose we have selected a wide portfolio of DRTs:

- OeSNN-PS, all based on a majority voting between the k most similar instances to a given unseen observation (the value of k has to be defined beforehand):
  - Edited Nearest Neighbor (ENN) (Chang, Pei & Zhang, 2011), which is a modified editing version of the kNN rule, applies a NN algoritthm and *edits* the dataset by removing samples which do not agree *enough* with their neighborhood. For each sample in the class to be undersampled, the set of nearest neighbors are computed; if the selection criterion is not fulfilled, the sample is removed.
  - Repeated Edited Nearest Neighbor (RENN) (Wilson, 1972) extends ENN by repeating the algorithm multiple times so that more data samples are deleted.
  - Condensed Nearest Neighbor (CNN) (Hart, 1968) was suggested as a rule which retains the basic approach of the NN rule, but without imposing its stringent storage requirements. CNN picks out points near the boundary between the classes, achieving an important reduction of the sample size while

maintaining the underlying distribution. It uses a 1-NN rule to iteratively decide if a sample should be removed or not. Note that it is sensitive to noise and will add noisy samples.

- AllKNN (Tomek, 1976a) differs from RENN in the fact that the number of neighbors of the internal kNN algorithm is increased at each iteration so as to yield a smoother decision region.
- Tomek Condensed Nearest Neighbor (TCNN) (Tomek, 1976b) removes every pair of instances $\mathbf{x}$ and $\mathbf{x}$ of different class that form a Tomek link, i.e. whenever there is no other sample $\mathbf{z}$ such that $d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{x})$ or $d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{x})$, where $d(\cdot, \cdot)$ is the distance measure of the problem at hand.
- Steady-State Memetic Algorithm (SSMA) (Derrac, García & Herrera, 2010), which is an evolutionary prototype selection algorithm that uses a memetic algorithm in order to perform a local search. In this case, an additional parameter max_it sets the maximum number of iterations performed by the search algorithm.

• OeSNN-PG, all controlled by parameters $min\_size\_cluster$ and $error\_tol$, which determine the minimum size of the cluster and the error tolerance before splitting a group, respectively:

- Self-Generating Prototypes (SGP) (Fayed, Hashem & Atiya, 2007; Oliveira,Magalhaes, Cavalca
, which is a centroid-based prototype
generation algorithm that uses a space splitting mechanism to generate prototypes in the center of every cluster in which data can be grouped;
- Self-Generating Prototypes 2 (SGP2) (Fayed, Hashem & Atiya, 2007; Oliveira, Magalhaes, Cavalcanti & Ren, 2012) is the second version of the SGP algorithm. It has a higher generalization power, including merge and pruning procedures; and
- Adaptive Self-Generating Prototypes (ASGP) (Fayed, Hashem & Atiya, 2007; Oliveira, Magalhaes, Cavalcanti & Ren, 2012), which has been specially designed to cope with imbalanced data sets.

We propose two different strategies for the resulting hybrid approaches, hereafter labeled as OeSNN-DRT, where

$$DRT \in \{\text{ENN}, \text{RENN}, \text{CNN}, \text{AllKNN}, \text{SSMA}, \text{SGP}, \text{SGP2}, \text{ASGP}\}.$$

In the first strategy data reduction is carried out every time the neuron repository is full (Algorithm 3), whereas in the second approach a drift detector notifies when the data reduction process needs to be triggered (Algorithm 4). As will be further explained, the first one (passive strategy) will be used for those experiments with synthetic data where the drift moment is known beforehand, whereas the second one (active strategy) will be adopted for the experiments with real data, where the drift moment is unknown.

One of the differences between the proposed OeSNN approach (Algorithm 2) and the approaches hybridized with DRTs schemes (Algorithms 3 and 4) is that the neuron repository size is limited and the merging process is hence unnecessary. The main goal of the DRTs is to summarize the underlying characteristics of the neuron repository over long periods of time, such that every newly included neuron has relevant information from two perspectives: 1) timeliness, as it replaces old neurons when the knowledge base in the neuron repository has no free space; and 2) predictive representativeness, because the new neuron will be fused with other neurons in the repository if it provides no further information on the prevailing concepts along the stream. This process can be regarded as a merging strategy of the whole neuron repository rather than a fusion between any two output neurons. Furthermore, the knowledge stored in the repository becomes more optimally assigned and hence, leads to a more suitable model design for online learning environments.

Algorithms 3 (passive strategy) and 4 (active strategy) evince that the operation of the OeSNN-PS and OeSNN-PG variants is similar to that of the OeSNN baseline in Algorithm 2. In essence, when the neuron repository is full DRTs are used to summarize the content of the neuron repository (line 18 in Algorithms 3 and 4). In the case of an active strategy, a drift detector is used (lines 16 and 17) to infer the moment at which DRTs should be applied: while no drift is detected and the neuron repository is not full, produced output neurons are always stored in the repository. When the repository saturates, the oldest output neuron is removed and the new one is stored instead. When a drift is detected, a DRT is applied to the neuron repository so as to make more room to store samples of the newly arriving concept.

Our proposed OeSNN approach and those hybridized with DRTs schemes (OeSNN-PS and OeSNN-PG in both passive and active operation modes schematically depicted in Figure 6) have been specially devised for online learning purposes: on the one hand, windowing strategies are usually preferred for sudden drifts, while on the other hand, instance selection strategies are instead adopted to handle gradual drifts and reoccurring contexts (Žliobaitė, 2010). Both characteristics have been included in the proposed approaches by using a sliding window and neuron repository composed of prototypes.

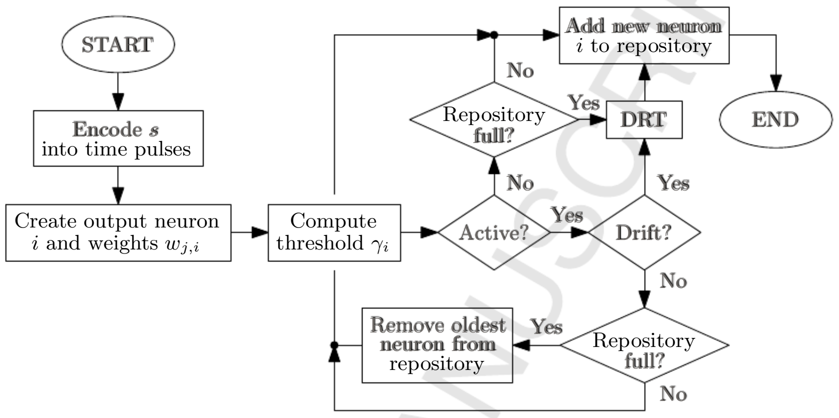

START
Encode s into time pulses
Create output neuron i and weights $w_{j,i}$
Compute threshold γ_i
Active?
Yes
No
Repository full?
Yes
No
Drift?
Yes
No
DRT
Add new neuron i to repository
END
Repository full?
Yes
No
Remove oldest neuron from repository

Figure 6: Schematic diagram of the proposed OeSNN-DRT schemes in both passive and active strategies.

Algorithm 3: OeSNN algorithm with DRTs: passive approach

```
Initialize neuron repository, $NR = \{\}$
Set eSNN parameter $mod = [0, 1]$, $C = [0, 1]$, $SIM = [0, 1]$
Set eSNN encoding parameters $\beta, T, N$
Set sliding window size $\mathcal{W}$
Set Neuron repository of size $NR\_size$
Set current neuron repository size $CNR\_size = 0$
for every sample s belonging to the class c do
    Update sliding window with sample s
    Calculate $I_{max}, I_{min}$ for the $\mathcal{W}$ samples in the sliding window
    Calculate $C_j$ and $W_j$ over the sliding window for their encoding into
      firing time of multiple pre-synaptic neurons $j$
    Create a new output neuron i and the connection weights as
      $w_{ji} = mod^{order(j)}$
    Calculate $PSP_{max(i)} = \Sigma_j w_{ji} \cdot mod^{order(j)}$
    Get PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
    if $CNR\_size < NR\_size$ then
        Add the weight vector and threshold of the new output neuron to NR
        $CNR\_size = CNR\_size + 1$
    else
        Apply a DRT over the neuron repository (PS or PG)
        Add the weight vector and threshold of the new output neuron to NR
        Update CNR_size
    end
end
Repeat above for all target classes
```

Algorithm 2: Proposed OeSNN algorithm

```
1  Initialize neuron repository, $NR = \{\}$
2  Set eSNN parameter $mod = [0, 1]$, $C = [0, 1]$, $SIM = [0, 1]$
3  Set eSNN encoding parameters $\beta, T, N$
4  Set sliding window size $\mathcal{W}$
5  Set neuron repository of size $NR\_size$
6  Set current neuron repository size $CNR\_size = 0$
7  for every sample s belonging to the class c do
8      Update sliding window with sample s
9      Calculate $I_{max}, I_{min}$ for the $\mathcal{W}$ samples in the sliding window
10     Calculate $C_j$ and $W_j$ over the sliding window for encoding s into firing
         time of multiple pre-synaptic neurons $j$
11     Create a new output neuron i and the connection weights as
         $w_{ji} = mod^{order(j)}$
12     Calculate $PSP_{max(i)} = \Sigma_j w_{ji} \cdot mod^{order(j)}$
13     Get PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
14     if min distance(Newly output neuron weight vector,Neuron repository
         weight vectors in in NR)) $\leq$ SIM then
15         Update the weight vector and threshold of the most similar neuron
             $w_{j,i} = \frac{w_{new}+(w_{j,i} \cdot M)}{M+1}$ and $\gamma_i = \frac{\gamma_{new}+(\gamma_i \cdot M)}{M+1}$
16         Set $M = M + 1$
17     else
18         if CNR_size < NR_size then
19             Add the weight vector and threshold of the new output neuron
                 to NR
20             CNR_size = CNR_size + 1
21         else
22             Remove the oldest weight vector and its threshold, and put the
                 new ones into NR
23         end
24     end
25 end
26 Repeat above for all target classes
```

Algorithm 4: OeSNN algorithm with DRTs: active approach

```
1  Initialize DriftDetector()
2  Initialize neuron repository, $NR = \{\}$
3  Set eSNN parameter $mod = [0, 1]$, $C = [0, 1]$, $SIM = [0, 1]$
4  Set eSNN encoding parameters $\beta, T, N$
5  Set sliding window size $\mathcal{W}$
6  Set Neuron repository of size $NR\_size$
7  Set current neuron repository size $CNR\_size = 0$
8  Set drift_detection $= False$
9  for every sample s belonging to the class c do
10     Update sliding window with sample s
11     Calculate $I_{max}, I_{min}$ for the $\mathcal{W}$ samples in the sliding window
12     Calculate $C_j$ and $W_j$ over the sliding window for their encoding into
         firing time of multiple pre-synaptic neurons $j$
13     Create a new output neuron i and the connection weights as
         $w_{ji} = mod^{order(j)}$
14     Calculate $PSP_{max(i)} = \Sigma_j w_{ji} \cdot mod^{order(j)}$
15     Get PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
16     drift_detection = DriftDetector()
17     if drift_detection $= True$ then
18         Apply a DRT over the neuron repository (PS or PG)
19         Add the weight vector and threshold of the new output neuron to NR
20         Update CNR_size
21         Set drift_detection $= False$
22     else
23         if CNR_size < NR_size then
24             Add the weight vector and threshold of the new output neuron
                 to NR
25             CNR_size = CNR_size + 1
26         else
27             Remove the oldest weight vector and its threshold, and insert
                 the new one into NR
28         end
29     end
30 end
31 Repeat above for all target classes
```

5. Computer Experiments

An extensive experimental benchmark has been designed to shed light on the performance of the proposed schemes over synthetic and real streaming datasets. Such experiments are divided into two main blocks:

- In the first set of experiments, the naive OeSNN approach (Algorithm 2) will be compared to *passive* OeSNN-PS and OeSNN-PG techniques (Algorithm 3) when they are applied over synthetic data sets, assuming that the drift moment is known beforehand.
- In the second set of experiments, the naive OeSNN will be compared to *active* OeSNN-PS and OeSNN-PG approaches over real data streams, where drifts are unknown and therefore motivates the use of a drift detector.

As a result of this experimental design, we will first analyze the performance of OeSNN-PS and OeSNN-PG approaches with synthetic data sets, and assess their benefits during the plasticity period (i.e. shortly after the drift). The main advantage of OeSNN-PS and OeSNN-PG is that while the neuron repository is not full, it can accept more neurons (the latest ones) inside. But when it is full, all the information is condensed in a reduced number of neurons (prototypes), letting more free space to accept more neurons inside until the neuron repository is full again. Therefore, we should expect OeSNN-PS and OeSNN-PG to react better (faster) to sudden drifts. Next, a set of experiments with real data sets and a drift detector are planned to confirm this benefit in a realistic setting. To numerically quantify the predictive performance of the proposed schemes during the stability and plasticity periods, the prequential accuracy will be measured at three different points in time: right before the drift occurs (BD), during the drifting period (D), and after the drift occurs (AD). The exact time ticks at which this score is computed will be set explicitly for every dataset in the benchmark, which are described in the following subsection.

5.1. Datasets

When working with real datasets, it is not possible to know exactly when a drift occurs, which type of drift arises when a drift is detected, or even if there is any drift. Consequently, it is not possible to perform a detailed analysis of the behavior of different algorithms in the presence of concept drift by using only real-world datasets. In order to analyze the effect of DRTs for facing concept drift and to complement the analysis of our proposed approaches, we first use the renowned set of synthetic datasets described in (Minku, White & Yao, 2010).

Results for 4 different problems (Minku, White & Yao, 2010) (labeled as `CIRCLE`, `LINE`, `SINEH` and `SINEV`) will be considered, each containing one drift simulated by varying among low and high severity, and low and high speed, resulting 4 different types of drift for each data set. Severity represents the amount of changes caused by a new concept, that is the percentage of the input space which has its label changed after the drift is complete. Speed is the inverse of the time taken for a new concept to completely replace the previous one. Each dataset consists of 2000 samples ($t \in \{1, \ldots, 2000\}$), 2 normalized ($[0, 1]$) continuous features $\{X_1, X_2\}$, and a binary target class $l \in \{1, 2\}$. Drift appears at $t = 1000$ in all datasets, and the drifting period finishes at $t = 1099$, being $BD = 999$, $D = 1099$, and $AD = 1999$. The number of samples that belong to class 1 and 2 is always the same.

As for real drifting scenarios we resort to three different datasets. The first two datasets are well-known in the online learning community, since they have been used in several relevant studies of online approaches (Minku & Yao, 2012; Bifet & Gavalda, 2007; Elwell & Polikar, 2011). The third one was published in Kaggle[1], a platform for predictive modeling and analytics competitions and challenges. It is a brand new dataset, recently used for the first time in a work on evolving data stream classification (Gomes, Bifet, Read, Barddal, Enembreck,Pfharinger, Holmes . More details on these three datasets are next provided:

- The Australian New South Wales Electricity Market (Harries & Wales, 1999;Gama , Medas, Castillo & Rodrigues, 2004) (labeled as ELEC2) contains $45,312$ instances dated from May 1996 to December 1998. Each example of the dataset refers to a period of 30 minutes, and has 5 fields (day of week, time stamp, NSW electricity demand, Vic electricity demand, and scheduled electricity transfer between states). The class label identifies the change of the price related to a moving average of the last 24 hours.
- The National Oceanic and Atmospheric Administration of the United States Department of Commerce[2] (USDC) has built a database (labeled as NOAA) with $18,154$ daily weather measurements (50 years) from over $7,000$ weather stations all around the world. Data samples include 8 features, such as temperature, dew point, sea level pressure, visibility, average wind speed, and other weather related predictors alike. These variables are used to infer whether each day was rainy or not.
- The Give Me Some Credit[3] data set (labeled as GMSC) is a credit scoring dataset

[1] https://www.kaggle.com

[2] Available at: ftp.ncdc.noaa.gov/pub/data/gsod. Last access in March 20th, 2018.

[3] Available at: https://www.kaggle.com/c/GiveMeSomeCredit. Last access in March 20th, 2018.

aimed at deciding whether a loan should be granted. This is a core decision for banks due to the risk of unexpected expenses and future lawsuits. The dataset comprises supervised historical data of $150,000$ borrowers described by 10 features.

5.2. Drift Detection

When dealing with real-world streaming data sets, the drift moment is unknown, and a drift detector is required for those active approaches that need to know this information as soon as possible in order to trigger their adaptation mechanisms. This is the case of our proposed approaches: to this aim, they have been hybridized with the so-called Early Drift Detection Method (EDDM) (Baena-García, del Campo-Ávila, Fidalgo, Bifet, Gavaldà & Morales-Bueno, 2006). Its simplicity and capacity to detect repeatedly occurring concept drifts even with very noisy data (Baena-García, del Campo-Ávila, Fidalgo, Bifet, Gavaldà & Morales-Bueno, 2006) has motivated the selection of EDDM for the experimental benchmark discussed in what follows. But there is the possibility of hybridizing with other drift detection methods in the literature.

5.3. Parameter Configuration

Parameter configuration for synthetic data sets `CIRCLE`, `LINE`, `SINEH` and `SINEV` data sets are presented in Table 1. Empirical experiments have been carried out to find the values.

	W	T	β	N	MOD	C	SIM	k	max_loop	$min\_size\_cluster$	$error\_tol$
OeSNN	100	20	1.5	10	0.85	0.75	0.15	-	-	-	-
OeSNN-TCNN	100	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-SSMA	100	20	1.5	10	0.85	0.75	-	3	50	-	-
OeSNN-ENN	100	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-RENN	100	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-AllKNN	100	20	1.5	10	0.85	0.75	-	5	-	-	-
OeSNN-CNN	100	20	1.5	10	0.85	0.75	-	1	-	-	-
OeSNN-SGP	100	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-SGP2	100	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-ASGP	100	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3

Table 1: Parameter values set for the proposed OeSNN approaches when applied over the `CIRCLE`, `LINE`, `SINEH` and `SINEV` datasets.

In a real scenario, there is no a priori knowledge about the streaming data that the algorithm will predict, thus we cannot assume any parameter configuration. Given this issue, an effective yet unrealistic workaround is to isolate a representative portion of the dataset to make offline assumptions about the distribution of the data and to assign a suitable parametric configuration through a heuristic wrapper.

We embrace this strategy to initialize the algorithm with a realistic configuration. Remarkably, the recent literature has widely acknowledged that the parametric optimization of predictive models for data streams while in operation still remains an open research problem (Krawczyk, Minku, Gama, Stefanowski & Woźniak, 2017).

Regarding the `ELEC2` dataset, we have used the first 12 months (38%) of the data set to tune the parameters of the algorithms, whereas the remaining months have been used for prediction and performance assessment. A period of 48 time steps that corresponds to one day was assumed to study the behavior of our approaches during the drift (D). The sliding window size $\mathcal{W}$ was set to 96 samples. As for the `NOAA` dataset we proceeded similarly: the first 5 years (10%) of data were used for parameter tuning, whereas the following years were used for testing purposes. A period of 2 time steps that corresponds to two days was assumed to study the behavior of our approaches during the drift (D). The value of $\mathcal{W}$ was set to 25 samples. Finally, in the `GMSC` data set the first 20000 samples (16%) of the dataset were used for model configuration, and the rest for performance assessment. The behavior of our approaches during the drift (D) was studied over 50 time steps. In this case, $\mathcal{W}$ was established to 25 samples.

	T	β	N	MOD	C	SIM	k	max_loop	$min\_size\_cluster$	$error\_tol$
OeSNN	20	1.5	10	0.85	0.75	0.15	-	-	-	-
OeSNN-TCNN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-SSMA	20	1.5	10	0.85	0.75	-	1	50	-	-
OeSNN-ENN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-RENN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-AllKNN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-CNN	20	1.5	10	0.85	0.75	-	2	-	-	-
OeSNN-SGP	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-SGP2	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-ASGP	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3

Table 2: Parameter values utilized for the real datasets `ELEC2`, `NOAA` and `GMSC`.

Table 2 summarizes the parameter values for experiments with the real datasets. The warning level α and the drift level threshold β of the EDDM detector were set to 0.95 and 0.90, respectively. In order to inspect the impact of the DRTs on the neuron repository, three storage capacities $NR\_size \in \{50, 100, 150\}$ have been simulated for all datasets (either synthetic or real).

6. Results

This section analyzes the behavior of the proposed OeSNN when hybridized with DRTs schemes (OeSNN-PS and OeSNN-PG), in which PS and PG data reduction techniques are applied respectively to the neuron repository. The analysis focuses not only on the accuracy of the approaches during the stability (BD and AD) and plasticity periods (D), but also on the data reduction percentage they achieve.

6.1. *Impact of the Neuron Repository with Synthetic Data*

To begin with, Table 3 shows the prequential accuracies of the naive OeSNN (i.e. the OeSNN without DRTs that has been detailed in Algorithm 2) when the neuron repository has a storage capacity of 50, 100, and 150, measured at points BD, D, and AD for all synthetic datasets. The table is complemented by Figure 7, which exemplifies the occupancy level of the neuron repository along the stream. Little variations of the occupancy correspond to those time instants at which the incoming sample is found to be similar to another one in the neuron repository, thus triggering the merging process. As evinced in this plot, this procedure occurs frequently depending on the *SIM* parameter. However, the main drawback is that this process only involves two neurons at every time, so there is no real contribution of this merging process to the optimized management of the neuron repository that stream processing clearly demands.

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	0.884/0.889/0.888	0.790/0.750/0.760	0.851/0.880/0.873	0.884/0.889/0.888	0.940/0.960/0.950	0.876/0.909/0.904
LINE	0.926/0.935/0.948	0.900/0.910/0.880	0.932/0.950/0.948	0.926/0.935/0.948	0.890/0.940/0.960	0.912/0.922/0.930
SINE	0.907/0.928/0.942	0.880/0.900/0.880	0.921/0.950/0.944	0.907/0.928/0.942	0.940/0.990/0.970	0.929/0.937/0.936
SINEH	0.792/0.811/0.827	0.760/0.770/0.670	0.753/0.795/0.794	0.792/0.811/0.827	0.840/0.840/0.830	0.736/0.781/0.797
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	0.884/0.889/0.888	0.810/0.730/0.740	0.831/0.845/0.834	0.884/0.889/0.888	0.880/0.890/0.910	0.791/0.834/0.823
LINE	0.930/0.940/0.949	0.850/0.690/0.650	0.912/0.901/0.893	0.930/0.940/0.949	0.890/0.940/0.950	0.879/0.882/0.866
SINE	0.927/0.947/0.956	0.820/0.680/0.610	0.917/0.918/0.880	0.927/0.947/0.956	0.890/0.910/0.920	0.878/0.884/0.865
SINEH	0.792/0.811/0.827	0.670/0.460/0.460	0.773/0.766/0.744	0.792/0.811/0.827	0.830/0.840/0.840	0.699/0.738/0.757

Table 3: Prequential accuracies of the proposed naive OeSNN model with neuron repository sizes 50, 100, and 150 for `CIRCLE`, `LINE`, `SINEH` and `SINEV` datasets.

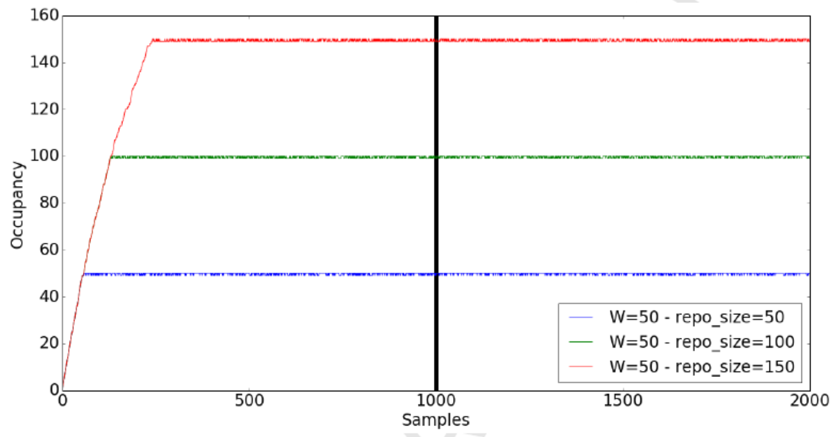


Figure 7: Evolution of the occupancy of the neuron repository for the proposed OeSNN in the `CIRCLE` dataset under low severity and high speed conditions. The averaged occupancy is 98.08% (50 neurons), 96.56% (100 neurons), and 94.57% (150 neurons).

We now turn the focus on the OeSNN incorporating DTRs in a passive strategy. Tables 4 and 5 summarize the results obtained for the proposed model hybridized with selective data reduction techniques (OeSNN-PS): OeSNN-TCNN, OeSNN-SSMA and OeSNN-ENN (Table 4), and OeSNN-RENN, OeSNN-AllKNN and OeSNN-CNN (Table 5) when the neuron repository has a storage capacity of 50, 100, and 150 neurons, measured in the points BD, D, and AD over synthetic datasets. Figure 8 depicts the evolution of the repository occupancy over the stream for all the aforementioned OeSNN schemes. Finally, the same set of results are shown in Table 6 and Figure 9 for OeSNNs based on generative data reduction techniques (OeSNN-PG).

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.848*/0.882/0.895	0.780/**0.770**/**0.894**	0.857/0.885/**0.915**	*0.848*/0.882/0.895	*0.870*/*0.940*/0.940	*0.845*/0.916/**0.922**
LINE	*0.911*/0.932/0.946	**0.920**/**0.930**/**0.900**	*0.910*/*0.927*/*0.925*	*0.911*/0.932/0.946	**0.920**/*0.910*/0.960	*0.888*/*0.910*/*0.913*
SINE	*0.892*/0.933/0.938	**0.900**/0.900/0.870	*0.888*/*0.915*/*0.901*	*0.892*/0.933/0.938	0.940/*0.940*/0.960	*0.893*/*0.910*/*0.921*
SINEH	*0.764*/0.802/0.836	*0.720*/*0.740*/**0.710**	0.751/0.801/0.797	*0.764*/0.802/0.836	*0.780*/**0.850**/0.850	0.746/**0.811**/**0.810**
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.848*/0.882/0.895	**0.840**/**0.750**/0.730	*0.777*/*0.780*/*0.769*	*0.848*/0.882/0.895	0.890/**0.920**/0.900	*0.764*/*0.772*/*0.752*
LINE	0.932/0.937/0.951	*0.650*/*0.620*/*0.610*	*0.755*/*0.709*/*0.697*	0.932/0.937/0.951	**0.910**/*0.910*/0.950	*0.780*/*0.759*/*0.748*
SINE	0.922/0.943/*0.942*	*0.680*/*0.630*/**0.630**	*0.710*/*0.707*/*0.701*	0.922/0.943/*0.942*	**0.910**/0.920/0.920	*0.779*/*0.763*/*0.748*
SINEH	*0.764*/0.802/0.836	*0.560*/0.460/*0.440*	*0.664*/*0.658*/*0.656*	*0.764*/0.802/0.836	*0.760*/0.840/0.840	*0.661*/*0.653*/*0.617*

(a) OeSNN-TCNN

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.832*/*0.861*/**0.888**	*0.770*/**0.780**/**0.850**	*0.803*/*0.836*/**0.888**	*0.812*/*0.864*/**0.882**	*0.890*/*0.890*/**0.960**	*0.824*/*0.879*/**0.895**
LINE	*0.902*/*0.922*/**0.944**	**0.890**/**0.900**/**0.890**	*0.889*/*0.932*/*0.932*	*0.898*/*0.918*/*0.932*	*0.860*/*0.870*/*0.910*	*0.866*/*0.900*/*0.912*
SINE	*0.886*/*0.904*/*0.917*	*0.840*/*0.860*/**0.870**	*0.892*/*0.922*/*0.924*	*0.870*/*0.904*/*0.916*	*0.880*/*0.950*/*0.940*	*0.871*/*0.909*/**0.928**
SINEH	*0.768*/*0.778*/*0.795*	**0.770**/**0.800**/**0.710**	*0.727*/**0.785**/**0.785**	*0.753*/*0.770*/*0.789*	*0.760*/*0.810*/**0.840**	**0.727**/*0.747*/*0.759*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.817*/*0.870*/**0.898**	*0.750*/**0.770**/**0.740**	*0.785*/*0.819*/**0.833**	*0.828*/*0.866*/*0.872*	**0.870**/**0.880**/*0.890*	*0.740*/*0.772*/**0.813**
LINE	*0.890*/*0.919*/*0.927*	**0.840**/**0.800**/**0.870**	*0.874*/**0.915/0.916**	*0.886*/*0.912*/*0.929*	*0.840*/*0.880*/*0.910*	*0.828*/**0.876**/**0.880**
SINE	*0.875*/*0.914*/*0.930*	**0.830**/**0.780**/**0.810**	*0.872*/**0.910**/**0.912**	*0.873*/*0.918*/*0.931*	*0.870*/*0.890*/*0.900*	*0.858*/*0.861*/**0.891**
SINEH	*0.731*/*0.766*/*0.784*	**0.750**/**0.710**/**0.660**	*0.724*/**0.776**/**0.773**	*0.743*/*0.766*/*0.784*	*0.700*/**0.840**/**0.830**	*0.670*/*0.709*/*0.716*

(b) OeSNN-SSMA

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.727*/*0.839*/*0.848*	*0.700*/**0.780**/**0.780**	*0.710*/*0.850*/0.871	*0.727*/*0.839*/*0.848*	*0.700*/*0.850*/*0.910*	*0.716*/*0.870*/*0.889*
LINE	*0.890*/**0.943**/0.938	0.900/0.900/**0.910**	*0.903*/*0.891*/*0.904*	*0.890*/**0.943**/0.938	**0.910**/0.940/*0.900*	*0.891*/*0.901*/*0.901*
SINE	*0.859*/*0.908*/0.932	*0.840*/*0.880*/**0.910**	*0.837*/*0.881*/*0.907*	*0.859*/*0.908*/0.932	*0.900*/*0.900*/*0.950*	*0.850*/*0.882*/*0.914*
SINEH	*0.724*/*0.784*/*0.807*	*0.650*/*0.740*/**0.720**	*0.691*/*0.749*/*0.735*	*0.724*/*0.784*/*0.807*	*0.750*/*0.790*/*0.810*	*0.723*/0.775/*0.751*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.727*/*0.839*/*0.848*	*0.630*/0.730/0.730	*0.677*/*0.717*/*0.713*	*0.727*/*0.839*/*0.848*	*0.740*/*0.830*/*0.880*	*0.672*/*0.711*/*0.717*
LINE	*0.884*/0.930/0.941	*0.660*/*0.640*/*0.630*	*0.656*/*0.652*/*0.659*	*0.884*/0.930/0.941	*0.850*/*0.840*/*0.920*	*0.698*/*0.696*/*0.718*
SINE	*0.885*/*0.907*/*0.922*	*0.630*/*0.640*/**0.640**	*0.672*/*0.666*/*0.658*	*0.885*/*0.907*/*0.922*	0.880/*0.890*/*0.900*	*0.736*/*0.739*/*0.736*
SINEH	*0.724*/*0.784*/*0.807*	*0.400*/*0.350*/*0.350*	*0.400*/*0.410*/*0.380*	*0.724*/*0.784*/*0.807*	*0.760*/*0.770*/*0.790*	*0.475*/*0.493*/*0.474*

(c) OeSNN-ENN

Table 4: Prequential accuracies obtained by (a) OeSNN-TCNN, (b) OeSNN-SSMA and (c) OeSNN-ENN working with repository sizes of 50, 100, and 150 neurons over the `CIRCLE`, `LINE`, `SINEH` and `SINEV` datasets. Prequential accuracies in bold denote an improvement greater than 0.01 in comparison with the traditional OeSNN for the same scenario setup. On the contrary, prequential accuracies in italics are declared to be worse than the traditional OeSNN if they are at least 0.01 below the prequential accuracies scored by the latter. Prequential accuracies in regular text stand for performance gaps less than 0.01 in absolute value.

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.723*/*0.825*/*0.833*	*0.690*/**0.770**/**0.810**	*0.718*/*0.842*/0.876	*0.723*/*0.825*/*0.833*	*0.720*/*0.840*/*0.880*	*0.727*/*0.851*/*0.881*
LINE	*0.869*/0.943/0.941	*0.780*/0.900/**0.910**	*0.835*/*0.891*/*0.916*	*0.869*/0.943/0.941	**0.910**/0.940/*0.910*	*0.847*/*0.901*/0.926
SINE	*0.859*/*0.908*/*0.924*	*0.840*/*0.880*/0.890	*0.837*/*0.881*/*0.894*	*0.859*/*0.908*/*0.924*	*0.900*/*0.900*/*0.930*	*0.850*/*0.882*/*0.907*
SINEH	*0.724*/*0.783*/*0.806*	*0.650*/*0.740*/**0.720**	*0.691*/*0.749*/*0.736*	*0.724*/*0.783*/*0.806*	*0.750*/*0.790*/*0.810*	*0.723*/0.772/*0.752*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.723*/*0.825*/*0.833*	*0.640*/**0.750**/0.750	*0.681*/*0.734*/*0.736*	*0.723*/*0.825*/*0.833*	*0.740*/*0.830*/*0.860*	*0.669*/*0.724*/*0.730*
LINE	*0.884*/0.930/0.940	*0.660*/*0.640*/*0.630*	*0.656*/*0.652*/*0.659*	*0.884*/0.930/0.940	*0.850*/*0.840*/*0.920*	*0.698*/*0.696*/*0.718*
SINE	*0.885*/*0.907*/*0.919*	*0.630*/*0.640*/**0.640**	*0.672*/*0.666*/*0.659*	*0.885*/*0.907*/*0.919*	0.880/*0.890*/*0.900*	*0.736*/*0.739*/*0.735*
SINEH	*0.724*/*0.783*/*0.806*	*0.400*/*0.350*/*0.350*	*0.400*/*0.407*/*0.380*	*0.724*/*0.783*/*0.806*	*0.760*/*0.770*/*0.790*	*0.475*/*0.491*/*0.474*

(a) OeSNN-RENN

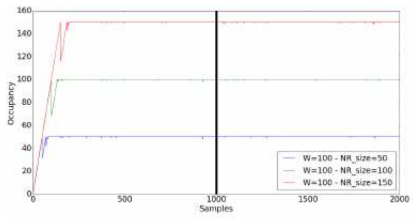
	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.794*/*0.828*/*0.852*	*0.770*/**0.800**/**0.820**	*0.747*/*0.856*/0.876	*0.794*/*0.828*/*0.852*	*0.770*/*0.830*/*0.880*	*0.803*/*0.850*/*0.886*
LINE	*0.870*/*0.894*/*0.929*	*0.880*/*0.860*/*0.860*	*0.882*/*0.891*/*0.911*	*0.870*/*0.894*/*0.929*	0.880/*0.890*/*0.900*	*0.855*/*0.874*/*0.892*
SINE	*0.879*/*0.906*/*0.918*	*0.780*/*0.840*/0.880	*0.839*/*0.873*/*0.904*	*0.879*/*0.906*/*0.918*	0.930/*0.900*/*0.950*	*0.847*/*0.890*/*0.915*
SINEH	*0.719*/*0.793*/*0.790*	*0.660*/0.780/0.680	*0.638*/0.785/*0.728*	*0.719*/*0.793*/*0.790*	*0.730*/*0.800*/*0.810*	*0.657*/**0.801**/*0.757*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.794*/*0.828*/*0.852*	*0.730*/**0.800**/**0.780**	*0.649*/*0.781*/*0.787*	*0.794*/*0.828*/*0.852*	*0.800*/*0.820*/*0.860*	*0.655*/*0.743*/*0.763*
LINE	0.922/0.930/0.940	*0.700*/*0.640*/0.650	*0.809*/*0.782*/*0.779*	0.922/0.930/0.940	*0.870*/*0.900*/*0.910*	*0.845*/*0.811*/*0.805*
SINE	*0.849*/*0.906*/*0.918*	*0.710*/*0.660*/**0.660**	*0.783*/*0.800*/*0.782*	*0.849*/*0.906*/*0.918*	*0.810*/*0.880*/*0.890*	*0.825*/*0.808*/*0.816*
SINEH	*0.719*/*0.793*/*0.790*	*0.460*/*0.440*/**0.500**	*0.488*/*0.600*/*0.613*	*0.719*/*0.793*/*0.790*	*0.720*/*0.820*/*0.810*	*0.565*/*0.617*/*0.615*

(b) OeSNN-AllKNN

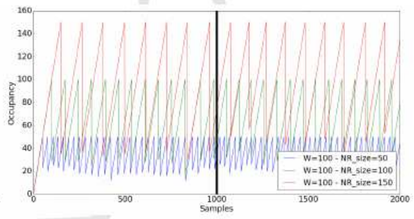
	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.795*/*0.841*/*0.840*	*0.740*/**0.770**/**0.750**	*0.807*/*0.823*/*0.810*	*0.837*/*0.846*/*0.862*	**0.930**/*0.890*/*0.870*	*0.825*/*0.840*/*0.826*
LINE	*0.903*/*0.924*/**0.938**	*0.860*/*0.860*/**0.910**	*0.913*/*0.909*/**0.937**	**0.916**/**0.929**/**0.945**	*0.860*/*0.880*/*0.910*	*0.851*/*0.871*/*0.879*
SINE	*0.889*/*0.916*/*0.928*	*0.840*/*0.860*/*0.860*	*0.902*/*0.918*/*0.908*	*0.870*/*0.902*/*0.927*	*0.900*/*0.960*/**0.960**	*0.887*/*0.916*/*0.907*
SINEH	*0.734*/*0.772*/*0.804*	**0.770**/*0.680*/**0.720**	**0.744**/*0.750*/*0.762*	*0.709*/*0.756*/*0.782*	*0.780*/*0.730*/**0.830**	*0.712*/*0.727*/*0.740*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.847*/*0.841*/*0.848*	*0.730*/*0.670*/*0.680*	*0.776*/*0.758*/*0.760*	*0.861*/*0.846*/*0.849*	*0.850*/*0.790*/*0.780*	*0.759*/*0.745*/*0.731*
LINE	*0.891*/*0.911*/*0.927*	*0.750*/**0.730**/**0.700**	*0.863*/*0.829*/*0.818*	*0.899*/*0.922*/*0.925*	**0.910**/*0.920*/*0.850*	*0.795*/*0.801*/*0.804*
SINE	*0.856*/*0.913*/*0.931*	*0.750*/**0.740**/**0.700**	*0.841*/*0.853*/*0.852*	*0.884*/*0.908*/*0.923*	*0.840*/**0.900**/**0.920**	*0.799*/*0.762*/*0.793*
SINEH	*0.726*/*0.773*/*0.809*	*0.550*/**0.540**/**0.560**	*0.676*/*0.693*/*0.688*	*0.733*/*0.787*/*0.793*	*0.690*/*0.780*/*0.820*	*0.651*/*0.662*/*0.631*

(c) OeSNN-CNN

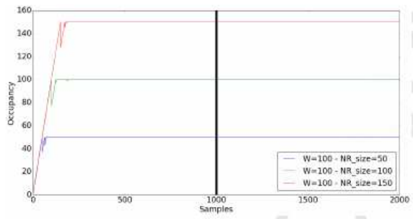
Table 5: Prequential accuracies obtained by (a) OeSNN-RENN, (b) OeSNN-AllKNN and (c) OeSNN-CNN working with repository sizes of 50, 100, and 150 neurons over the `CIRCLE`, `LINE`, `SINEH` and `SINEV` datasets. The same notational criteria hold in terms of statistical significance between these schemes and the naive OeSNN scheme.



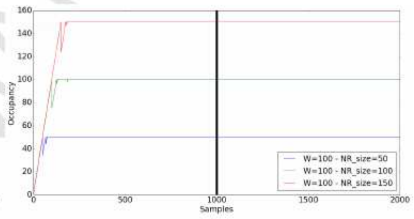
(a) OeSNN-AllKNN: $98.47\%, 97.19\%, 95.96\%$



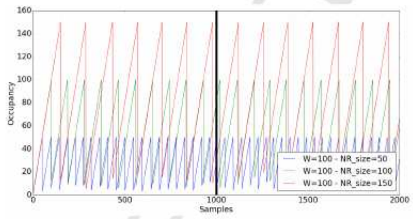
(b) OeSNN-CNN: $69.93\%, 67.60\%, 64.71\%$



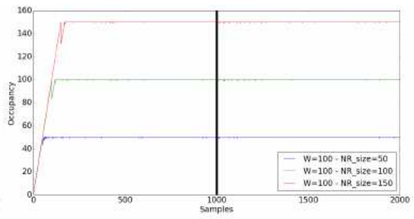
(c) OeSNN-ENN: $98.60\%, 97.32\%, 96.08\%$



(d) OeSNN-RENN: $98.57\%, 97.26\%, 95.99\%$



(e) OeSNN-SSMA: $55.83\%, 54.97\%, 53.40\%$



(f) OeSNN-TCNN: $98.64\%, 97.39\%, 96.16\%$

Figure 8: Evolution of the occupancy of the neuron repository for the OeSNN-PS approaches in the `CIRCLE` dataset under high severity and high speed conditions. Three percentages are displayed for every technique, corresponding to the occupancy (in %) for repository sizes equal to 50, 100 and 150 neurons.

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860*/*0.876*/*0.859*	*0.750*/**0.810**/**0.760**	*0.823*/*0.855*/*0.859*	*0.860*/*0.876*/**0.882**	*0.860*/*0.920*/**0.950**	*0.843*/*0.878*/*0.888*
LINE	**0.918**/**0.928**/*0.932*	**0.910**/**0.940**/**0.920**	**0.927**/**0.943**/**0.951**	**0.918**/**0.928**/*0.932*	**0.880**/*0.900*/*0.910*	*0.899*/**0.916**/**0.923**
SINE	**0.911**/**0.922**/*0.918*	**0.910**/**0.890**/*0.850*	**0.923**/*0.930*/**0.934**	**0.911**/**0.922**/*0.918*	**0.950**/*0.960*/*0.950*	*0.917*/*0.923*/**0.933**
SINEH	*0.766*/*0.780*/*0.785*	**0.760**/**0.770**/**0.760**	**0.754**/*0.766*/**0.787**	*0.766*/*0.780*/*0.785*	**0.830**/*0.800*/**0.830**	**0.742**/*0.757*/*0.762*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860*/*0.876*/**0.882**	*0.770*/**0.750**/**0.740**	*0.800*/*0.812*/*0.820*	*0.860*/*0.876*/**0.882**	*0.850*/*0.870*/*0.860*	*0.754*/*0.797*/*0.799*
LINE	**0.925**/*0.924*/*0.929*	*0.820*/**0.850**/**0.770**	*0.897*/**0.912**/**0.904**	**0.925**/*0.924*/*0.929*	*0.870*/*0.880*/*0.870*	*0.858*/*0.859*/**0.870**
SINE	*0.900*/*0.916*/*0.928*	**0.880**/**0.810**/**0.760**	**0.912**/**0.911**/**0.912**	*0.900*/*0.916*/*0.928*	**0.890**/**0.900**/*0.890*	**0.873**/**0.878**/**0.883**
SINEH	*0.766*/*0.780*/*0.785*	**0.690**/**0.710**/**0.690**	*0.741*/**0.960**/**0.767**	*0.766*/*0.780*/*0.785*	**0.850**/*0.770*/**0.830**	**0.707**/*0.712*/*0.734*

(a) OeSNN-SGP, OeSNN-SGP2

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860*/*0.876*/**0.882**	*0.750*/**0.810**/**0.760**	*0.822*/*0.855*/*0.859*	*0.860*/*0.876*/**0.882**	*0.870*/*0.920*/**0.950**	*0.834*/*0.878*/*0.888*
LINE	**0.918**/**0.928**/*0.932*	**0.910**/**0.940**/**0.920**	**0.927**/**0.943**/**0.951**	**0.918**/**0.928**/*0.932*	**0.880**/*0.900*/*0.910*	*0.889*/**0.916**/**0.923**
SINE	**0.911**/**0.922**/*0.918*	**0.910**/**0.890**/*0.850*	**0.923**/*0.930*/**0.934**	**0.911**/**0.922**/*0.918*	**0.950**/*0.960*/*0.950*	*0.917*/*0.923*/**0.933**
SINEH	*0.766*/*0.780*/*0.785*	**0.760**/**0.770**/**0.760**	**0.754**/*0.766*/**0.787**	*0.766*/*0.780*/*0.785*	**0.830**/*0.800*/**0.830**	**0.742**/*0.757*/*0.762*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860*/*0.876*/**0.882**	*0.770*/**0.750**/**0.740**	*0.807*/*0.821*/*0.823*	*0.860*/*0.876*/**0.882**	*0.860*/*0.870*/*0.860*	*0.752*/*0.780*/*0.798*
LINE	**0.925**/*0.924*/*0.929*	*0.820*/**0.850**/**0.770**	*0.897*/**0.912**/**0.904**	**0.925**/*0.924*/*0.929*	*0.870*/*0.880*/*0.870*	*0.858*/*0.859*/**0.870**
SINE	*0.900*/*0.916*/*0.928*	**0.880**/**0.810**/**0.760**	**0.912**/**0.911**/**0.912**	*0.900*/*0.916*/*0.928*	**0.890**/**0.900**/*0.890*	**0.873**/**0.878**/**0.883**
SINEH	*0.766*/*0.780*/*0.785*	**0.690**/**0.710**/**0.690**	*0.741*/**0.760**/**0.767**	*0.766*/*0.780*/*0.785*	**0.850**/*0.770*/**0.830**	**0.707**/*0.712*/*0.734*

(b) OeSNN-ASGP

Table 6: Prequential accuracies of the (a) OeSNN-SGP/OeSNN-SGP2 and (b) OeSNN-ASGP approach working with the neuron repository sizes of 50, 100, and 150 respectively for `CIRCLE`, `LINE`, `SINEH` and `SINEV` data sets. Where prequential accuracies are in bold, there is a notable improvement in comparison with the traditional OeSNN, whereas prequential accuracies in italics are worse than the traditional OeSNN. When prequential accuracies are in regular text means that there is no significant difference.

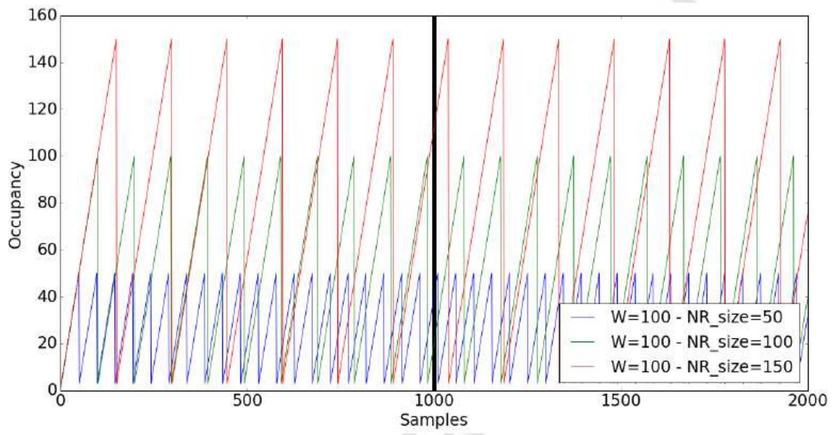


Figure 9: Averaged neuron repository occupancies (52.64%, 50.85%, 50.01%) of the all OeSNN-PG approaches for the sizes 50, 100, and 150 respectively with the CircleG data set under high severity and high speed conditions.

6.2. Impact of the Neuron Repository with Real Data

Once OeSNN-PS and OeSNN-PG approaches have been simulated with synthetic datasets, a second set of experiments has been performed with real datasets. The OeSNN-DRT hybridized with a drift detector (Algorithm 4) has been used in these simulations to be compared with the naive version of the proposed OeSNN because no a priori information about the drift moments is known. The drift detector serves as an active strategy to notify when it is necessary to trigger the DRT at hand. In this second experimental benchmark, averaged prequential accuracy scores for all cases (OeSNN, active OeSNN-PS and active OeSNN-PG) are jointly summarized in Table 7 for all real datasets, repository sizes (50, 100, and 150 neurons), and measured at points BD, D, and AD. Statistics during the same periods for the drift detector are also provided. The averaged prequential accuracies have been calculated over all BD, D, and AD periods that occur in all drift detections.

		BD	D	AD	# drifts
OeSNN approach					
OeSNN	`ELEC2`	0.817/0.808/0.806	0.802/0.793/0.784	0.810/0.802/0.796	n.a.
	`NOAA`	0.700/0.675/0.685	0.692/0.640/0.674	0.702/0.630/0.673	n.a.
	`GMSC`	0.907/0.919/0.908	0.901/0.916/0.898	0.912/0.923/0.905	n.a.
Active OeSNN-PS approaches					
OeSNN-TCNN	`ELEC2`	0.827/0.806/0.801	**0.822**/0.792/*0.763*	**0.834**/0.802/*0.776*	13/117/23
	`NOAA`	0.697/**0.689**/0.692	*0.604*/**0.689**/*0.623*	*0.669*/**0.686**/*0.646*	50/61/12
	`GMSC`	**0.907**/**0.911**/**0.907**	*0.888*/**0.908**/**0.906**	*0.896*/*0.912*/**0.906**	44/33/50
OeSNN-SSMA	`ELEC2`	**0.819**/**0.807**/**0.807**	**0.792**/**0.804**/**0.800**	**0.802**/**0.810**/**0.805**	5/106/66
	`NOAA`	*0.684*/**0.706**/**0.675**	*0.647*/*0.567*/**0.715**	*0.652*/*0.540*/**0.705**	85/4/22
	`GMSC`	**0.902**/**0.917**/**0.908**	**0.891**/**0.932**/**0.896**	*0.889*/**0.921**/**0.907**	45/7/19
OeSNN-ENN	`ELEC2`	0.817/0.804/0.800	0.800/0.784/0.774	0.805/0.801/0.792	106/111/99
	`NOAA`	0.697/**0.699**/0.695	*0.679*/**0.710**/*0.646*	0.705/**0.726**/**0.687**	56/17/57
	`GMSC`	**0.908**/**0.913**/**0.915**	**0.907**/**0.919**/**0.918**	**0.913**/**0.923**/**0.923**	51/49/47
OeSNN-RENN	`ELEC2`	0.811/0.804/0.804	*0.740*/0.784/0.788	*0.768*/0.801/0.800	5/111/100
	`NOAA`	0.693/0.684/0.681	*0.659*/**0.664**/*0.583*	*0.639*/**0.675**/*0.621*	56/71/15
	`GMSC`	**0.908**/**0.913**/**0.915**	**0.907**/**0.923**/**0.912**	**0.913**/**0.924**/**0.921**	51/49/46
OeSNN-AllKNN	`ELEC2`	0.815/0.811/0.802	0.792/0.784/0.785	0.808/0.801/0.803	106/100/126
	`NOAA`	**0.719**/**0.696**/**0.728**	**0.783**/**0.674**/**0.669**	**0.742**/**0.704**/**0.716**	12/19/8
	`GMSC`	**0.908**/**0.912**/**0.914**	**0.911**/**0.921**/**0.925**	**0.912**/**0.919**/**0.925**	51/46/45
OeSNN-CNN	`ELEC2`	**0.816/0.806/0.800**	**0.798**/**0.791**/**0.787**	**0.806**/**0.814**/**0.802**	60/26/114
	`NOAA`	*0.684*/**0.763**/**0.716**	**0.702**/**0.919**/*0.571*	*0.650*/**0.951**/*0.642*	55/1/3
	`GMSC`	**0.915**/**0.911**/*0.865*	*0.846*/**0.915**/*0.841*	*0.845*/*0.886*/*0.820*	5/6/21
Active OeSNN-PG approaches					
OeSNN-SGP	`ELEC2`	**0.817**/**0.813**/**0.813**	**0.815**/**0.807**/**0.809**	**0.814**/**0.810**/**0.811**	112/113/104
	`NOAA`	**0.695**/**0.674**/**0.683**	*0.657*/**0.739**/**0.720**	**0.698**/**0.623**/**0.698**	51/21/28
	`GMSC`	**0.911**/**0.920**/**0.908**	**0.914**/**0.931**/**0.911**	**0.913**/**0.935**/**0.912**	45/6/30
OeSNN-SGP2	`ELEC2`	**0.819**/**0.811**/**0.810**	**0.815**/**0.805**/**0.802**	**0.819**/**0.810**/**0.812**	113/112/98
	`NOAA`	**0.695**/**0.682**/**0.688**	*0.657*/**0.662**/**0.754**	**0.698**/**0.680**/**0.741**	51/59/39
	`GMSC`	**0.911**/**0.920**/**0.908**	**0.914**/**0.931**/**0.911**	**0.916**/**0.935**/**0.912**	45/6/30
OeSNN-ASGP	`ELEC2`	**0.816**/**0.812**/**0.805**	**0.811**/**0.806**/**0.797**	**0.815**/**0.809**/**0.802**	117/117/52
	`NOAA`	**0.697**/**0.678**/**0.687**	**0.713**/**0.638**/**0.693**	**0.696**/**0.658**/**0.686**	48/38/69
	`GMSC`	**0.912**/*0.904*/*0.896*	*0.875*/*0.893*/*0.873*	*0.882*/*0.880*/*0.880*	18/13/7

Table 7: Averaged prequential accuracies and number of detected drifts of the OeSNN-DRT approaches working with the neuron repository sizes 50, 100, and 150 for the real data sets. Where prequential accuracies are in bold, there is a notable improvement in comparison with the traditional OeSNN, whereas prequential accuracies in italics are worse than the traditional OeSNN. When prequential accuracies are in regular text means that there is no significant difference.

6.3. Comparison to Other Methods

In Table 8 a comparison with some of the most recent and well-known online learning methods in the presence of concept drift is presented, namely, Hoeffding Trees or also known as Very Fast Decision Trees (HTs, (Domingos & Hul-ten, 2000)), Random Forests (RFs, (Breiman, 2001)), and Hoeffding Naive Bayes Tree ensembles (HNBTs, (Gomes, Bifet, Read, Barddal, Enembreck, Pfharinger, Holmes & Abdessalem, 2017b)). Active implementations of some of the above detectors are also included in the paper, encompassing drift detectors such as DDM (Gama, Medas, Castillo & Rodrigues, 2004), EDDM (Baena-García, del

Campo-Ávila, Fidalgo, Bifet, Gavaldà & Morales-Bueno, 2006), ADWIN (Bifet& Gavalda, 2007) or EDIST2 (Khamassi, Sayed-Mouchaweh, Hammami & Gh édira, 2015). For the sake of fairness and comparability the table only reports experimental results of schemes published in the literature by other authors. Unfortunately, to the best of our knowledge no prior work for NOAA dataset has been carried out under similar conditions and evaluation criteria, hence this dataset has not been considered in this last experimental phase.

TECHNIQUES	TYPE	ELEC2	GMSC
OeSNN-DRT approaches			
OeSNN-TCNN	Single	0.822/0.792/0.763	0.888/0.908/0.906
OeSNN-SSMA	Single	0.792/0.804/0.800	0.891/0.932/0.896
OeSNN-ENN	Single	0.800/0.784/0.774	0.907/0.919/0.918
OeSNN-RENN	Single	0.740/0.784/0.788	0.907/0.923/0.912
OeSNN-AllKNN	Single	0.792/0.784/0.785	0.911/0.921/0.925
OeSNN-CNN	Single	0.798/0.791/0.787	0.846/0.915/0.841
OeSNN-SGP	Single	0.815/0.807/0.809	0.914/0.931/0.911
OeSNN-SGP2	Single	0.815/0.805/0.802	0.914/0.931/0.911
OeSNN-ASGP	Single	0.811/0.806/0.797	0.875/0.893/0.873
(Khamassi & Sayed-Mouchaweh, 2017)			
HTs ensemble + EDIST2	Ensemble (10)	0.848	-
(Gomes, Bifet, Read, Barddal, Enembreck, Pfharinger, Holmes & Abdessalem, 2017b)			
Adaptive RFs	Ensemble (100)	0.885	0.935
Online Bagging (HNBTs)	Ensemble (100)	0.825	0.935
Online Accuracy Updated Ensemble (HNBTs)	Ensemble (100)	0.863	0.935
Online Boosting (HNBTs)	Ensemble (100)	0.901	0.926
Online Smooth-boost (HNBTs)	Ensemble (100)	0.875	0.925
Leveraging Bagging (HNBTs)	Ensemble (100)	0.885	0.935

Table 8: Comparison during the drifting phase of the average prequential accuracies of OeSNN-DRTs and some of the most relevant ensemble based techniques in the literature. The same evaluation criteria was used in all works: a *test-then-train* online scheme and average of prequential accuracies during the drifting phase as the performance metric.

A first look at the results in this table reveals that the performance of several OeSNN-DRT approaches occurs to be very competitive with respect to the state of the art, above all after considering that our OeSNN-DRT approaches are based on a single model and the rest are based on ensemble models composed of 10 (Khamassi & Sayed-Mouchaweh, 2017) or 100 (Gomes, Bifet, Read, Barddal, Enembreck, Pfharinger, Holmes & Abdessalem, 2017b) base learners. Nevertheless, next Section will elaborate on this comparison in depth.

7. Discussion

A first look in Figures 8 and 9 reveals that DRTs are able to retain the knowledge with a reduced number of output neurons, but not all DRTs achieve a good classification performance and a relevant data reduction percentage at the same time. This is a key point in online learning scenarios where the storage capacity is limited.

We start the discussion by analyzing the results of the synthetic data, which are presented in Tables 3 to 6. The obtained scores for OeSNN-PS show that in general, OeSNN-TCNN, OeSNN-ENN, OeSNN-RENN and OeSNN-AllKNN render a degraded performance when compared to the naive OeSNN approach. Some exceptions deserve further attention at this point: the OeSNN-TCNN approach has a general better classification performance in the plasticity period D, mostly in those datasets with low severity and high speed. Similarly, OeSNN-ENN and OeSNN-RENN yield a better performance in the plasticity period D in datasets with low severity and high speed when the size of the repository is large (150 neurons). The OeSNN-AllKNN approach performs best in the plasticity period D over datasets with high severity and high speed, again for large repository sizes. This interesting result advocates for one of our postulated hypothesis: the larger the neuron repository is, the more important an optimized management of its contents is in an online learning setup; this fact becomes even more noticeable when the drift imprints deep changes on the stream data (high severity, high speed drifts), as the repository needs to be refreshed quickly so as to grasp and learn the newly evolving concept.

On the contrary, OeSNN-SSMA and OeSNN-CNN (those with higher data reduction percentages) offer further performance improvements in other periods over the simulated streams. OeSNN-SSMA has in general better classification performance in 1) the plasticity period D in datasets characterized by fast drifts, and 2) in the stable period AD for datasets with high severity and high speed. Besides, for stable periods BD and AD in many other cases, OeSNN-SSMA outperforms the proposed OeSNN, remarkably when the size of the repository is the largest one (150). On the other hand, the OeSNN-CNN approach produces in general better accuracy scores in the plasticity period D over datasets with high speed, and occasionally outperforms the naive OeSNN in other specific conditions. But it is the OeSNN-SMMA technique which provides better prequential accuracies and a more efficient use of the neuron repository capacity (more data reduction percentage) at the same time. Unfortunately, this comes along with a computation penalty, since the SMMA encompasses a heuristic search demanding for memory and processing resources that could eventually clash with stringent computational constraints of the online problem in question.

Regarding OeSNN-PG approaches whose results are compiled in Table 6, OeSNN-SGP and OeSNN-SGP2 approaches rendered the same results and show a general better classification performance than the OeSNN approach in the plasticity period D for high speed datasets. They were found to be also competitive in low speed datasets, even in stable periods. The OeSNN-ASGP approach has similar results than OeSNN-SGP and OeSNN-SGP2, but it performs better in the `CIRCLE` data set for low severity and high speed drifts in the stable period BD when the size of the neuron repository is 150. It also shows a general better classification performance than the OeSNN approach in the plasticity period D for high speed data sets, and it is also a competitive technique in the low speed data sets, even in stable periods.

Once analyzed the impact of applying DRTs to OeSNN, we proceed by analyzing the set of experiments with real datasets leveraging the use of a drift detector to trigger the application of the data reduction technique. Regarding OeSNN-PS approaches, in Table 7 one can observe that OeSNN-TCNN, OeSNN-ENN, and OeSNN-RENN perform competitively in the stability and plasticity periods for several datasets. However, OeSNN-SSMA, OeSNN-AllKNN and OeSNN-CNN are the approaches rendering the best overall accuracy scores, with OeSNN-AllKNN dominating all OeSNN-PS techniques specially in the `NOAA` and `GMSC` datasets. As for OeSNN-PG approaches, Table 7 elucidates that the three approaches under this category perform better than the naive OeSNN in a wider spectrum of datasets and regions than their OeSNN-PS counterparts. Nonetheless, OeSNN-SGP and OeSNN-SGP2 are the techniques producing better results in all real datasets.

All in all, from the above experiments insightful conclusions can be drawn: OeSNN-SMMA, OeSNN-CNN, and all OeSNN-PG approaches (OeSNN-SGP, OeSNN-SGP2, and OeSNN-ASGP) achieve very high data reduction ratios and a competitive classification performance in comparison with the naive OeSNN approach in plasticity periods (D). They allow for more space for newly produced output neurons to enter the repository, thus favoring a quicker adaptation to the drift. Besides, these approaches decrease the processing time when the newly output neuron is compared to the rest of the neurons in the repository in the training phase: the less neurons in the repository, the less the number of pairwise comparisons will be needed to find the most similar neuron (*SIM* parameter for the merging process), and ultimately the less processing time and less storing space will be required. Interestingly, OeSNN-PG approaches have revealed themselves as the most suitable models to be applied right after a drift occurs.

Finally, the results in Section 6.3 certify that OeSNN-DRT schemes perform very competitively in comparison with other methods from the state of the art, getting comparable accuracy scores to avant-garde schemes over the `ELEC2` and `GMSC` datasets. At this point the reader should not overlook the fact that most online learning methods are based on ensemble classifiers rather than single models. Ensemble classifier models are widely acknowledged to be more accurate due to their robustness to error variance. Furthermore, they are more flexible to assimilate new available data into their learning algorithm, and they provide more straightforward mechanisms to forget irrelevant knowledge once a drift has been detected (e.g. by simply discarding the oldest classifier from the ensemble). In contrast, single model approaches generally trade lower accuracy results for a reduced computational cost, reason for which they are often regarded as an attractive solution for massive data streams (Ditzler, Roveri, Alippi & Polikar, 2015). Bearing the previous observations and considering key performance factors such as the size of the window or the number of base learners utilized in the compared ensembles (10 in the case of HTs ensemble + EDIST2 (Khamassi & Sayed-Mouchaweh,2017) or 100 in the case of ARFs and HNBTs approaches in (Gomes, Bifet, Read, Barddal, Enembreck, Pfharinger, Holmes & Abdessalem, 2017b)), we conclude that OeSNN-DRTs are along with the state of the art related to online learning and concept drift in terms of accuracy and computational efficiency (since, as already argued in the introduction, single classifiers are considered more suitable for online scenarios with stringent timing constraints that jeopardize the adoption of ensemble-based approaches). This statement stimulates further research around different extensions of the proposed family of online classifiers, as will be next outlined.

8. Conclusions and Future Research Lines

This work has presented a portfolio of new adaptations of evolving Spiking Neural Networks (eSNNs) for online learning scenarios under concept drift. Firstly, we have adapted the traditional eSNN technique to be used on online data streams by limiting the size of the neuron repository, yielding the so-called Online eSNN (OeSNN). Secondly, we have embraced the use of selective and generative data reduction techniques (DRTs) to optimize the contents of the neuron repository so as to achieve a better adaptability of the model to changing concepts over the processed data stream. Both passive and active strategies have been defined to incorporate DRTs into the OeSNN learning procedure: the active comprises a drift detector that detects changes along the data stream and triggers the application of the DRT at hand to the neuron repository. Two different families of OeSNN models have been proposed: OeSNN-PS (using prototype-selection DRTs) and OeSNN (corr. using prototype-generation DRTs), both capable of operating in passive and active modes when processing data streams.

An extensive set of computer experiments over synthetic and real streaming datasets has been designed and discussed to find performance differences between the above approaches. Part of the OeSNN-PS variants (OeSNN-SMMA and OeSNN-CNN) and all OeSNN-PG approaches (i.e. OeSNN-SGP, OeSNN-SGP2, and OeSNN-ASGP) have been proven to attain better predictive scores during plasticity periods than the naive version of the OeSNN (i.e. the proposed OeSNN with no DRT included in its learning algorithm). The application of DRTs to the proposed OeSNN model also allows reducing the required space to store output neurons, thus decreasing the processing time needed to train with newly samples: the less neurons in the repository, the less similarity computations during the learning phase. This alleviation of the computational resources demanded by the model is of utmost importance in online learning, where processing times and storage should be kept as low as possible to process high stream data rates.

Those approaches that use DRTs achieving the lowest occupancy of the neuron repository retain the old concept by generating or selecting prototypes, and at the same time they adapt better to the new concept by storing more information (output neurons) of the new concept during the plasticity period after a drift occurs. This is possible due to the fact that high data reduction ratios lead to more available space in the neuron repository where to store neurons with more recent information. Thus, the adaptation to the new concept is better (a higher accuracy is achieved) and faster (less similarity computations in the training phase as argued previously). Although the selection of a data reduction technique is not straightforward (it depends on the characteristics of the dataset), we have found out empirical evidences which indicate that OeSNN-PG approaches feature a better adaptability right after the drift occurs, while performing very competitively in stability periods.

In summary, this work brings to light the natural capability of the proposed family of OeSNN-DRTs to:

1. simultaneously learn the new incoming concept while retaining the older one without incorporating specific forgetting mechanisms (is used just to compute encoding parameters, there is no windowed adaptation whatsoever as Section 2.2 clearly shows);
2. update the model without requiring a retraining mechanism;
3. use less capacity storage by reducing the amount of required neurons for the overall model to achieve a better balance between stability (performance over stationary data distributions) and plasticity (reaction against drift events in the processed data stream);
4. be faster than the traditional OeSNN approach by carrying out less similarity computations (less output neurons) in the training phase;
5. be competitive in terms of their balance between predictive performance and complexity in comparison with ensembles of classifiers, which makes OeSNN-DRTs a better match for scenarios under severe computational restrictions; and

6. to provide a realistic solution to be hybridize with a drift detector.

Future efforts can be invested in several research lines. On the one hand, it is necessary to address the need for incorporating a priori information of the expected dynamics of the drift (severity and velocity), which can be estimated in practice when drifts occur in a recurrent fashion, namely, changes that have multiple underlying modes and reappear periodically over the stream (as in e.g. financial prediction or dynamic control, among others (Gonçalves Jr & De Barros, 2013)). In this case the most suitable data reduction technique for the plasticity period could be chosen based on the predicted type of drift. Finally, it will be also interesting to explore the possibility of building ensembles of OeSNNs and study their behavior in comparison with other ensemble approaches, as well as their online parametric configuration.

9. Acknowledgements

This work was supported by the EU project Pacific Atlantic Network for Technical Higher Education and Research - PANTHER (grant number 2013-5659/004-001 EMA2); and by the Basque Government through the EMAITEK program.

Bibliography

Aggarwal, C. C. (2006). On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases* (pp. 607–618). VLDB Endowment.

Alippi, C. (2014). *Intelligence for embedded systems*. Springer.

Alippi, C., & Roveri, M. (2008). Just-in-time adaptive classifierspart ii: Designing the classifier. *IEEE Transactions on Neural Networks*, *19*, 2053–2064.

Alnajjar, F., Zin, I. B. M., & Murase, K. (2008). A spiking neural network with dynamic memory for a real autonomous mobile robot in dynamic environment. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (pp. 2207–2213). IEEE.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Proc. of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS06)* (pp. 77–86).

Barddal, J. P., Gomes, H. M., Enembreck, F., & Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, *127*, 278–294.

Belatreche, A., Maguire, L. P., & McGinnity, M. (2007). Advances in design and application of spiking neural networks. *Soft Computing*, *11*, 239–248.

Bifet, A., & Gavalda, R. (2006). Kalman filters and adaptive windows for learning in data streams. In *International Conference on Discovery Science* (pp. 29–40). Springer.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining* (pp. 443–448). SIAM.

Bifet, A., Holmes, G., Pfahringer, B., & Gavalda, R. (2009). Improving adaptive bagging methods for evolving data streams. In *Asian conference on machine learning* (pp. 23–37). Springer.

Bohte, S. M., Kok, J. N., & La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*, 17–37.

Bohte, S. M., Kok, J. N., & La Poutré, J. A. (2000). Spikeprop: backpropagation for networks of spiking neurons. In *ESANN* (pp. 419–424).

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Chang, R., Pei, Z., & Zhang, C. (2011). A modified editing k-nearest neighbor rule. *JCP*, *6*, 1493–1500.

Cohen, E., & Strauss, M. (2003). Maintaining time-decaying stream aggregates. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 223–233). ACM.

Dawid, A. P., Vovk, V. G. et al. (1999). Prequential probability: Principles and properties. *Bernoulli*, *5*, 125–162.

Derrac, J., García, S., & Herrera, F. (2010). Stratified prototype selection based on a steady-state memetic algorithm: a study of scalability. *Memetic Computing*, *2*, 183–199.

Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, *10*, 12–25.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 71–80). ACM.

Domingos, P., & Hulten, G. (2003). A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, *12*, 945–949.

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, *22*, 1517–1531.

Escalante, H. J., Graff, M., & Morales-Reyes, A. (2016). Pggp: prototype generation via genetic programming. *Applied Soft Computing*, *40*, 569–580.

Fayed, H. A., Hashem, S. R., & Atiya, A. F. (2007). Self-generating prototypes for pattern classification. *Pattern Recognition*, *40*, 1498–1509.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence* (pp. 286–295). Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*, 44.

Garcia, S., Derrac, J., Cano, J., & Herrera, F. (2012). Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE transactions on pattern analysis and machine intelligence*, *34*, 417–435.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017a). A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, *50*, 23.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., & Abdessalem, T. (2017b). Adaptive random forests for evolving data stream classification. *Machine Learning*, *106*, 1469–1495.

Gonçalves Jr, P. M., & De Barros, R. S. M. (2013). Rcd: A recurring concept drift framework. *Pattern Recognition Letters*, *34*, 1018–1025.

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, *1*, 17–61.

Harries, M., & Wales, N. S. (1999). Splice-2 comparative evaluation: Electricity pricing. *Technical Report, The University of South Wales*, .

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, *14*, 515–516.

Hu, W., & Tan, Y. (2016). Prototype generation using multiobjective particle swarm optimization for nearest neighbor classification. *IEEE transactions on cybernetics*, *46*, 2719–2731.

Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2013). Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, *41*, 188–201.

Kasabov, N., Scott, N., Tu, E., Marks, S., Sengupta, N., Capecci, E., Othman, M., Doborjeh, M., Murli, N., Hartono, R. et al. (2016). Design methodology and selected applications of evolving spatio-temporal data machines in the neucube neuromorphic framework. *Neural Networks*, *78*, 1–14.

Kasabov, N. K. (2007). *Evolving connectionist systems: the knowledge engineering approach*. Springer Science & Business Media.

Kasabov, N. K. (2014). Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, *52*, 62–76.

Khamassi, I., & Sayed-Mouchaweh, M. (2017). Self-adaptive ensemble classifier for handling complex concept drift. In *CEUR Workshop Proceedings*. volume 1958.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2015). Self-adaptive windowing approach for handling complex concept drift. *Cognitive Computation*, *7*, 772–790.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems*, *9*, 1–23.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent data analysis*, *8*, 281–300.

Kononenko, I., & Kukar, M. (2007). *Machine learning and data mining: introduction to principles and algorithms*. Horwood Publishing.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156.

Li, J., & Wang, Y. (2015). Prototype selection based on multi–objective optimisation and partition strategy. *International Journal of Sensor Networks*, *17*, 163–176.

Lobo, J. L., Del Ser, J., Bilbao, M. N., Perfecto, C., & Salcedo-Sanz, S. (2017). Dred: An evolutionary diversity generation method for concept drift adaptation in online learning environments. *Applied Soft Computing*, .

Meena, L., & Devi, V. S. (2015). Prototype selection on large and streaming data. In *International Conference on Neural Information Processing* (pp. 671–679). Springer.

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, *22*, 730–742.

Minku, L. L., & Yao, X. (2012). Ddd: A new ensemble approach for dealing with concept drift. *IEEE transactions on knowledge and data engineering*, *24*, 619–633.

Ng, W., & Dash, M. (2008). A test paradigm for detecting changes in transactional data streams. In *International Conference on Database Systems for Advanced Applications* (pp. 204–219). Springer.

Oliveira, D. V., Magalhaes, G. R., Cavalcanti, G. D., & Ren, T. I. (2012). Improved self-generating prototypes algorithm for imbalanced datasets. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on* (pp. 904–909). IEEE volume 1.

Ponulak, F. (2005). Resume-new supervised learning method for spiking neural networks. *Institute of Control and Information Engineering, Poznan University of Technology*, *42*.

Ponulak, F. (2008). Analysis of the resume learning process for spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, *18*, 117–127.

Ponulak, F., & Kasiński, A. (2010). Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural computation*, *22*, 467–510.

Schliebs, S., & Kasabov, N. (2013). Evolving spiking neural networka survey. *Evolving Systems*, *4*, 87–98.

Soltic, S., & Kasabov, N. (2010). Knowledge extraction from evolving spiking neural networks with rank order population coding. *International Journal of Neural Systems*, *20*, 437–445.

Soltic, S., Wysoski, S. G., & Kasabov, N. K. (2008). Evolving spiking neural networks for taste recognition. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (pp. 2091–2097). IEEE.

Thorpe, S., & Gautrais, J. (1998). Rank order coding. In *Computational neuroscience* (pp. 113–118). Springer.

Thorpe, S. J., & Gautrais, J. (1997). Rapid visual processing using spike asynchrony. In *Advances in neural information processing systems* (pp. 901–907).

Tomek, I. (1976a). An experiment with the edited nearest-neighbor rule. *IEEE Transactions on systems, Man, and Cybernetics*, (pp. 448–452).

Tomek, I. (1976b). Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, *6*, 769–772.

Triguero, I., Derrac, J., Garcia, S., & Herrera, F. (2012). A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*, 86–100.

Triguero, I., García, S., & Herrera, F. (2010). Ipade: Iterative prototype adjustment for nearest neighbor classification. *IEEE Transactions on Neural Networks*, *21*, 1984–1990.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, *11*, 37–57.

Wang, J., Belatreche, A., Maguire, L., & Mcginnity, T. M. (2014). An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing*, *144*, 526–536.

Wang, J., Belatreche, A., Maguire, L. P., & McGinnity, T. M. (2017). Spiketemp: an enhanced rank-order-based learning approach for spiking neural networks with adaptive structure. *IEEE transactions on neural networks and learning systems*, *28*, 30–43.

Wang, S., Minku, L. L., & Yao, X. (2018). A systematic study of online class imbalance learning with concept drift. *IEEE Transactions on Neural Networks and Learning Systems*, .

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*, 964–994.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (pp. 408–421).

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2006). Adaptive learning procedure for a network of spiking neurons and visual pattern recognition. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (pp. 1133–1142). Springer.

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2010). Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, *23*, 819–835.

Zhou, Z.-H., Chawla, N. V., Jin, Y., & Williams, G. J. (2014). Big data opportunities and challenges: Discussions from data analytics perspectives [discussion forum]. *IEEE Computational Intelligence Magazine*, *9*, 62–74.

Žliobaitė, I. (2010). Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, .

Žliobaitė, I., Pechenizkiy, M., & Gama, J. (2016). An overview of concept drift applications. In *Big Data Analysis: New Algorithms for a New Society* (pp. 91–114). Springer.