

## Highlights

### **A Practical Guide to Streaming Continual Learning**

Andrea Cossu, Federico Giannini, Giacomo Ziffer, Alessio Bernardo, Alexander Gepperth, Emanuele Della Valle, Barbara Hammer, Davide Bacciu

- We provide a short background on Streaming Machine Learning (SML) and Continual Learning (CL), highlighting their similarities and differences.
- We define the Streaming Continual Learning (SCL) scenario, which offers the opportunity to merge ideas from CL and SML into a unified paradigm.
- We conduct an empirical evaluation showcasing the need for Streaming Continual Learning (SCL).

# A Practical Guide to Streaming Continual Learning

Andrea Cossu<sup>a,\*</sup>, Federico Giannini<sup>b,\*</sup>, Giacomo Ziffer<sup>b</sup>, Alessio Bernardo<sup>e</sup>, Alexander Gepperth<sup>c</sup>, Emanuele Della Valle<sup>b</sup>, Barbara Hammer<sup>d</sup>, Davide Bacciu<sup>a</sup>

<sup>a</sup>*Computer Science Department, University of Pisa, Largo Bruno Pontecorvo, 3, Pisa, 56127, Italy*

<sup>b</sup>*DEIB, Politecnico di Milano, Via Giuseppe Ponzio, 34, Milano, 20133, Italy*

<sup>c</sup>*University of Applied Sciences Fulda, Germany*

<sup>d</sup>*Bielefeld University, Germany*

<sup>e</sup>*Motus ml, Milano, Italy*

---

## Abstract

Continual Learning (CL) and Streaming Machine Learning (SML) study the ability of agents to learn from a stream of non-stationary data. Despite sharing some similarities, they address different and complementary challenges. While SML focuses on rapid adaptation after changes (concept drifts), CL aims to retain past knowledge when learning new tasks. After a brief introduction to CL and SML, we discuss Streaming Continual Learning (SCL), an emerging paradigm providing a unifying solution to real-world problems, which may require both SML and CL abilities. We claim that SCL can i) connect the CL and SML communities, motivating their work towards the same goal, and ii) foster the design of hybrid approaches that can quickly adapt to new information (as in SML) without forgetting previous knowledge (as in CL). We conclude the paper with a motivating example and a set of experiments, highlighting the need for SCL by showing how CL and SML alone struggle in achieving rapid adaptation and knowledge retention.

*Keywords:* continual learning, streaming machine learning, nonstationary environments, concept drift, forgetting

---

<sup>\*</sup>Equal contribution

*Email addresses:* [andrea.cossu@unipi.it](mailto:andrea.cossu@unipi.it) (Andrea Cossu), [federico.giannini@polimi.it](mailto:federico.giannini@polimi.it) (Federico Giannini)

## 1. Introduction

Over the last decade, the exponential growth of interconnected digital systems, driven by platforms such as social media and the Internet of Things (IoT), has resulted in massive, continuous flows of information. This results in the production of massive amounts of data, often generated by unbounded flows called data streams [1]. Data arrives sequentially and continuously over time, making it impractical to store or revisit them in full.

This streaming scenario challenges traditional machine learning paradigms, which often assume stationary data and bounded datasets. One of the most critical obstacles is concept drift [2], a phenomenon where the statistical properties of the data evolve due to underlying changes in the generative process (*concept*). Unlike sporadic anomalies, concept drift reflects structural, often unforeseeable changes that can render previously learned models unsuitable.

Due to concept drifts, the solution cannot train a model offline on historical data, since the trained model may soon become obsolete after concept drifts. Conversely, one must continuously learn from the data stream, monitoring and detecting changes, and adapting to them as rapidly as possible. Moreover, when learning new concepts, the model may forget what it learned regarding the previous ones. This issue is particularly critical when the new concept just expands the learning problem in a new subdomain, without contradicting what has been observed during the previous concepts. In this case, the model should be able to encapsulate the knowledge associated with the new concept while retaining the previously acquired one. This idea is associated with the well-known **stability-plasticity dilemma** [3, 4, 5], according to which the model should achieve a trade-off between the ability to learn new knowledge (plasticity) and the ability to remember the previous (stability). Too much plasticity leads to forgetting, too much stability prevents learning new knowledge.

However, forgetting may be desirable in specific situations. When the new concept contradicts a previously learned one, then the model should not preserve both. In this situation, avoiding forgetting takes on a more nuanced meaning. It does not require remembering the entire history of previous concepts, but rather preserving an up-to-date representation of the current knowledge while adapting dynamically to changes. Accessing prior knowledge remains a useful skill, as new changes in the data stream might require quick remembering of previously seen concepts.

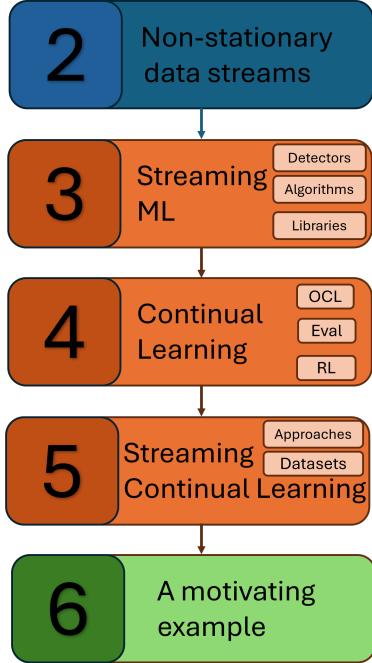


Figure 1: Overview of the structure of this manuscript. We start with an introduction to non-stationary environments, followed by a short background on streaming and continual learning, before discussing the streaming continual learning paradigm together with an empirical example motivating why it is needed.

own, they are not sufficient to provide a complete solution to real-world problems that demand continuous learning, rapid adaptation, and the ability to avoid forgetting (we further elaborate on this point in Section 6). **Streaming Continual Learning (SCL)** aims at unifying, without replacing, CL and SML. Recently, the ideas and challenges of SCL are gaining traction and fostering a positive discussion in the research community. SCL has been initially suggested in [7], followed up in [8], and the topic of a dedicated special session at the European Symposium on Artificial Neural Networks (ESANN)

in 2025 [9].

This paper builds upon the ideas proposed in the tutorial paper from the aforementioned special session [9]. In this extended version, we provide a more comprehensive overview of the scenarios, the objectives, and the evaluation protocols commonly used nowadays in CL and SML. We also expand the discussion about available datasets that can be directly used or repurposed to match the SCL framework. Finally, we added a motivating example showing the need for SCL: popular CL and SML approaches struggle in achieving both the CL objective (knowledge consolidation) and the SML objective (quick adaptation).

The paper also introduces the concept of a unified approach and discusses how this framework could evolve in the future by following specific research directions. Particularly, Section 2 presents the streaming scenario and the associated problem of concept drift. Section 3 presents SML, while Section 4 describes CL, by also illustrating the applications of Reinforcement Learning (RL) in streaming scenarios involving concept drifts. Section 5 introduces the vision of SCL and its motivation, while Section 6 illustrates an example supporting the motivations behind SCL. Finally, Section 7 discusses the conclusion and the future directions we envision for SCL. Figure 1 provides an overview of the structure of the paper.

## 2. Non-stationary data streams

The notion of a *data stream* emerges in two distinct scenarios, both involving a data source that is never fully accessible all at once. The first scenario, which we call **natural streams**, involves data that is generated continuously and in real-time. A typical example is the IoT, where sensors continuously gather measurements. In such cases, the model processes the current data instance (possibly along with a small buffer of recent past instances) and generates predictions accordingly. The second scenario, which we refer to as **artificial streams**, arises due to the huge volume of data available. Here, the model cannot access the entire dataset simultaneously. Instead, data is accessed incrementally in chunks or batches, effectively simulating a streaming process where data points arrive over time.

The most studied case in literature is the classification problem. In this case, the data stream can be defined as an ordered and unbounded sequence of data points  $DS : d_1, d_2, \dots, d_t, d_{t+1}, \dots$ , where  $d_t$  is the data point at time  $t$ , which is associated with a feature vector  $X_t$  and a label  $y_t$  (which is initially

**unknown**). The goal is to predict the label  $\hat{y}_t$  given the feature vector  $X_t$ , and, thus, learning the decision boundary defined by the probability distribution  $P(y_t|X_t)$ . The data stream results from a process called **concept** [10] which generates data points according to a probability distribution  $P(X_t, y_t)$ . Notably, this process is hidden and inaccessible.

The crucial problem that makes the traditional machine learning paradigm inapplicable is that  $P(X_t, y_t)$  may change over time. As a result, the model learned using the previous concept becomes obsolete and requires updates or changes. In particular, the **concept drift** is the need for a change in the learned model due to a change in the concept [2, 11, 12, 13]. Usually, these changes are unpredictable since the concept is hidden.

The literature distinguishes between virtual drifts and real drifts when considering the impact on the distribution  $P(X_t, y_t)$ . A real drift is defined as a change in the decision boundary  $P(y_t | X_t)$  [14]. Conversely, the definition of a **virtual drift** has received many interpretations over the years. It was originally defined as a change in the input distribution  $P(X_t)$  that necessitates a model update [2, 12, 13, 15]. Later works [14, 16, 17, 11, 18] described it as a change in  $P(y_t)$  or  $P(X_t | y_t)$  that does not affect  $P(y_t | X_t)$ . However, a necessary definition must be explicitly made to capture whether the new concept directly undermines or merely extends the observed classification problem. Although introducing a new  $P(X_t)$  or new classification labels may theoretically induce a change in  $P(y_t | X_t)$ , our distinction focuses on whether the classification rules for the input space observed so far remain unchanged. In this work, we refer to a virtual drift as a change in  $P(X_t)$  that exposes the model to a new input space, potentially a previously unseen subspace or an extension of an existing one. The crucial aspect is that it preserves the decision boundary associated with all instances in the previously observed input space. This means that all past instances retain their original classification after the drift. Such a drift may even introduce new labels, as long as it does not alter the classification of earlier instances. Virtual drift requires the model to update its representation to generalize in new regions of the input space, without contradicting previously acquired knowledge. Conversely, we consider a **real drift** as a drift that changes the decision boundary associated with the input distribution observed in earlier concepts. In this case, the same instance may be classified differently before and after the drift. A real drift, therefore, modifies the relationship between features and target outputs, effectively changing the nature of the problem and invalidating past knowledge. In some cases, virtual and real drift may

occur simultaneously, introducing both additional information in a new feature space and a transformation of the underlying problem associated with a feature subspace encountered in earlier concepts.

Notably, virtual drifts are common, though not exclusive, in artificial streaming settings. While these settings do not involve data generated in real time, shifts in the input distribution  $P(X_t)$  can still occur due to the order in which data is accessed or processed. For instance, large datasets read sequentially in batches may exhibit internal structure that leads to gradual distributional changes, even if the underlying task remains the same [15].

Virtual drifts are often treated superficially in the existing literature. The nuances and implications of virtual drift have received relatively little attention. This work provides, for the first time, a more detailed analysis of virtual drift. We propose a categorization that distinguishes between different types of virtual drift, depending on how the input distribution changes and how those changes affect model performance. This finer-grained perspective allows for a better understanding of the learning dynamics involved and opens the door to a better understanding of SML and CL settings. fig. 2 shows our categorization of concept drifts. For simplicity, we consider the case when the drift does not introduce new classes, but the problem is always defined on the same label set during the whole data stream. The red line represents the theoretical decision boundary of the problem, which the model is expected to learn. Note that this boundary is not known *a priori*. The model must infer it from the observed data. fig. 2a represents the original distribution and fig. 2b depicts a real drift, which changes the boundary on the previously observed feature subspace. The remaining categories are described below.

A change in the input distribution  $P(X_t)$  can, in some cases, reflect a shift toward a specific region of the feature space that was already present during the previous concept, though only sparsely represented. We refer to this scenario as **zoom-in virtual drift** (fig. 2c). In such cases, the model trained on the earlier concept may underperform because it learned only a rough approximation of the decision boundary within that region, due to limited exposure to data points from it. After the drift, the input distribution becomes concentrated in this previously underrepresented subspace, revealing that the earlier model fails to accurately represent the true decision boundary there. As a result, the model must be updated, even if the actual underlying function has not changed.

Another notable case of virtual drift occurs when the input distribution  $P(X_t)$  shifts to include a completely new region of the feature space that was

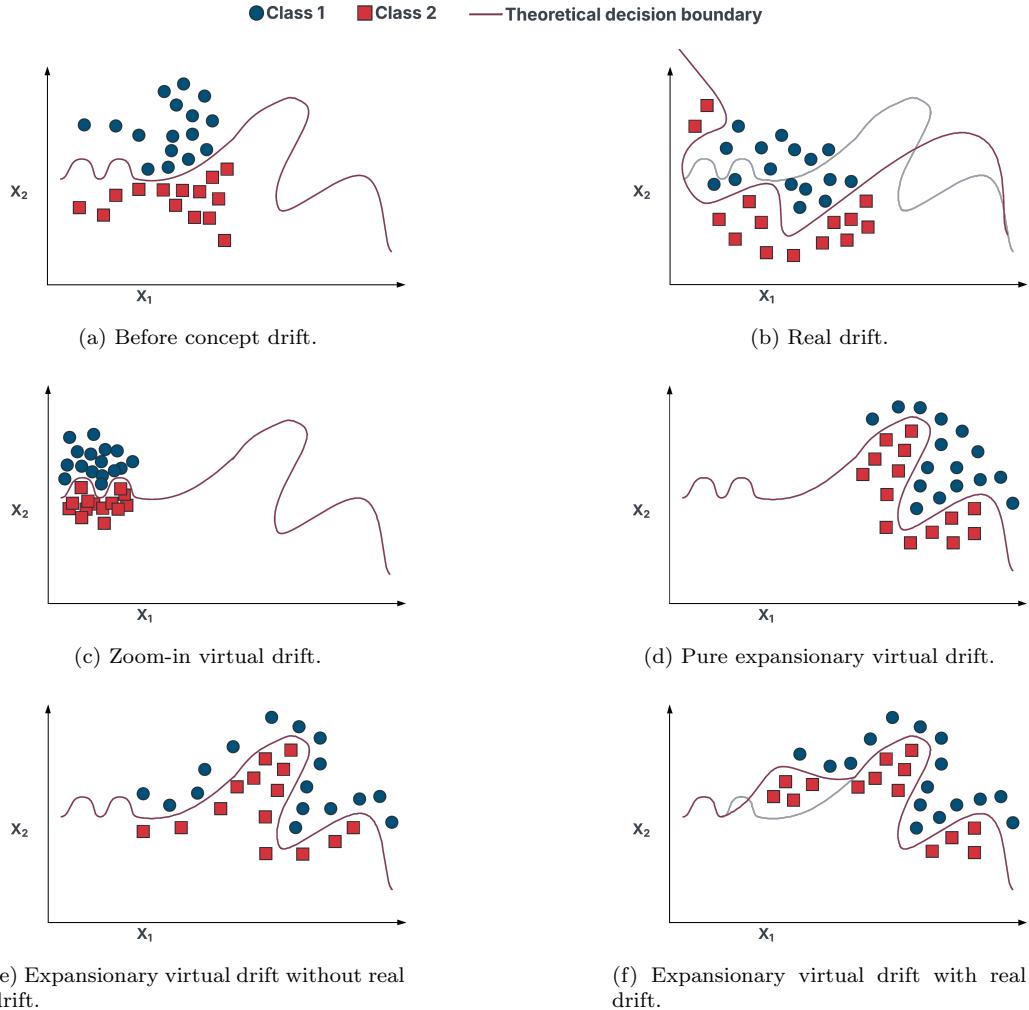


Figure 2: Illustrations of various concept drift types. The red line represents the theoretical decision boundary of the problem that the model must fit. 2b depicts a real drift where the decision boundary shifts. 2c represents a *zoom-in* virtual drift that concentrates on a subset of the initial feature space. 2d shows a *pure expansionary* virtual drift introducing a new, previously unseen feature subspace. 2e displays a drift involving both familiar and new subspaces without altering the original decision boundary. Finally, 2f illustrates a combination of virtual and real drift, resulting in changes to the decision boundary within the previously known subspace.

not previously observed. We define this as **expansionary virtual drift**. In such scenarios, the existing model is unlikely to perform well, as it has no prior experience with data from this newly introduced subspace. Expansionary virtual drift can be *pure* (fig. 2d), where the new concept generates data solely in the novel subspace. Alternatively, the new concept may include data from both the previously seen and newly introduced subspaces. In this mixed case, the new concept may also alter the decision boundary within a portion of the original feature space, resulting in the coexistence of both virtual and real drift (fig. 2f).

Along with the type of change, **severity** is an important marker for quantifying how the concept has changed with respect to the previous one [19, 11, 20]. Formally, severity measures the discrepancy between the joint distributions  $P(X_t, y_t)$  of features and labels before and after the drift. Associated with the severity, the **influence zone** of the drift indicates whether the change affects a limited region or the entire feature space [19, 11]. In the case of real drifts, analyzing the influence zone can help understand what has changed and must be relearned. A low-severity drift, or *mild* drift, will probably be associated with a limited influence zone, thus requiring updating the learned decision boundary with respect to a narrow region of the input space. Conversely, *severe* real drifts may change the decision boundary associated with a significant region of the input space, thus demanding substantial relearning and forgetting of past knowledge.

Additionally, concept drifts vary in their **speed of occurrence** [14, 11]. *Abrupt drifts* happen suddenly, causing immediate shifts in the data distribution. *Gradual drifts* involve a slow transition with overlap between old and new concepts, whereas *incremental drifts* proceed through intermediate stages of gradual change. Finally, *recurring drifts* occur when previously seen concepts reappear, requiring the model to efficiently recall past knowledge.

### 3. Streaming Machine Learning

SML [1] is usually concerned with natural streams, often involving real-time analytics where data is continuously collected over time. In this situation, the emphasis is on continuously training a model and quickly adapting to changes as new data points arrive. These scenarios often require several strict constraints. Each data point is processed only once, as only a single pass over the stream is permitted. To maintain responsiveness, the computational time required to process each instance must be minimal. Similarly,

memory usage is constrained and should grow at most sublinearly with respect to the total length of the data stream. Moreover, the model is expected to be available for prediction at any point during the stream.

The primary objective of an SML system is to promptly detect and adapt to concept drifts (both virtual and real). Such systems focus exclusively on the current data distribution and the active concept, aiming to provide accurate predictions under present conditions. The main challenge lies in minimizing the adaptation time following a drift, as the model’s outputs during this period are typically unreliable. Consequently, knowledge from past concepts is often discarded once a drift occurs and is no longer used to support future predictions.

*SML Setting and Evaluation.* A data stream is defined by SML literature as an ordered and unbounded sequence of data points  $\mathcal{S} = d_1, d_2, \dots, d_t, d_{t+1}, \dots$ , where  $t$  is the timestamp. As already mentioned in Section 2, in the classification problem, each data point  $d_t$  is represented by a feature vector  $X_t$  and a label  $y_t$ . SML assumes that, at timestamp  $t$ , the model receives the feature vector  $X_t$  and must predict the label  $\hat{y}_t$ . Before receiving the new feature vector  $X_{t+1}$ , the real label  $y_t$  is given, and the model can update its performance score using the pair  $(X_t, y_t)$  and learn from it. This paradigm is referred to as **sequential evaluation** or test-then-train evaluation [21, 22].

*Drift Detectors.* The design of drift detectors has attracted substantial interest within the SML community. Gama et al. [11] propose a taxonomy of the main approaches, broadly distinguishing three families. *Error rate-based methods*, such as ADWIN [23], the Early Drift Detection Method [24], EWMA [25], and EWMA for Concept Drift Detection [26], track the online classification errors of base learners to identify noteworthy deviations. These techniques are particularly suited to real and contradictory drifts, where changes in classification rules lead to performance degradation. *Data distribution-based* approaches assess discrepancies between historical and incoming data, making them appropriate for detecting input or virtual drift. Notable examples include Concept Drift via Competence Models (CM) [27], as well as techniques relying on information-theoretic measures [28] and statistical bounds on data streams [29]. *Multiple hypothesis testing* strategies employ sets of statistical tests to capture drifts, as exemplified by Just-in-Time (JIT) adaptive classifiers [30], hierarchical three-layer schemes [31], and

lightweight detection ensembles [32]. Selecting an appropriate drift detector is therefore a key challenge in the streaming context.

*SML Algorithms.* SML classification algorithms can generally be categorized into five main methodological families, each adopting distinct mechanisms to handle the constraints of data streams [33]. They usually start from offline machine learning approaches and produce online and streaming versions able to learn continuously from an unbounded data stream and adapt to changes.

Frequency-based approaches estimate posterior class probabilities by incrementally collecting statistics on the frequency of features. These methods assume conditional independence among attributes given the class label and typically employ Bayes' theorem to perform predictions. A canonical example is the Naïve Bayes (NB) classifier [34], including several revised versions designed to improve its performance in streaming contexts, like Sketch-Based NB [35], Ageing-Based Multinomial NB [36], and Incremental Weighted NB [37].

Neighborhood-based methods, such as variants of the k-Nearest Neighbors (KNN) algorithm, base their predictions on proximity in the feature space. To make this computationally feasible in a streaming setting, they maintain a sliding window of recent samples rather than storing the entire history. Notable examples are KNN with Self Adjusting Memory [38] and Efficient KNN Graph Construction [39].

Tree-based strategies adapt classic decision tree algorithms for incremental learning. These methods typically employ the Hoeffding bound [40] to select split attributes with high statistical confidence from streaming data. Early implementations [41] do not handle concept drifts effectively, whereas later extensions such as the Hoeffding Adaptive Tree (HAT) [42] incorporate change detection mechanisms like ADWIN at each node to dynamically adjust the tree structure.

Ensemble-based classification represents another prominent category. These models combine the predictions of multiple base learners, often trained on resampled subsets of the data [43, 44, 45]. One of the most well-known solutions is the streaming version of Random Forest called Adaptive Random Forests (ARF) [46]. By introducing diversity among learners, ensemble techniques aim to improve generalization performance [47].

Neural network-based approaches have started to receive increasing attention due to their capacity for high-level abstraction. However, their application in streaming scenarios is still poorly investigated since they present chal-

lenges related to real-time learning, memory efficiency, and parameter tuning. Notable examples are Continuously Adaptive Neural Networks for Data Streams [48], Autonomous Deep Learning [49], Incremental LSTM [50], Continuous LSTM [51, 52].

*SML Libraries.* The landscape of SML libraries has evolved from traditionally Java-centered ecosystems toward increasingly Python-oriented solutions, reflecting the changing needs of both researchers and practitioners. *MOA (Massive Online Analysis)*<sup>1</sup> [1] remains the reference library for SML in Java. Building on the WEKA project, MOA provides a mature and comprehensive framework with a wide range of algorithms for online classification, clustering, and outlier detection, as well as standardized evaluation protocols that ensure experimental rigor and reproducibility.

In the Python ecosystem, River<sup>2</sup> [53] initially emerged as the main reference library for incremental learning on data streams, following the merger of Creme and scikit-multiflow. River adopts a native streaming paradigm in which data are processed one instance at a time and emphasizes flexibility through lightweight, dictionary-based data structures that support evolving feature spaces.

CapyMOA<sup>3</sup> [54] bridges these two ecosystems by exposing MOA’s robust Java-based algorithms through a Python interface. By directly leveraging MOA’s highly optimized implementations, CapyMOA achieves substantially higher efficiency than fully Python-native libraries such as River, particularly in large-scale or high-throughput scenarios. Combined with seamless integration into the Python machine learning ecosystem (e.g., PyTorch and scikit-learn) and support for advanced settings such as complex concept drift simulation, semi-supervised learning, and online continual learning, CapyMOA is emerging as a strong candidate reference library for SML in Python.

#### 4. Continual Learning

Similar to SML, CL also studies data streams subject to concept drift. Unlike SML, CL focuses more on knowledge retention rather than quick adaptation. Historically, this is due to the first empirical observation of

---

<sup>1</sup><https://moa.cms.waikato.ac.nz/>

<sup>2</sup><https://riverml.xyz/dev/>

<sup>3</sup><https://capymoa.org/>

the *catastrophic forgetting* phenomenon in artificial neural networks [55, 56]. Forgetting is exacerbated on data streams with sudden, virtual drifts, where new concepts replace previous ones [57]. Given the large degree of plasticity in artificial neural networks trained with backpropagation and gradient descent, whenever a concept is not observed for a sustained period, it is quickly forgotten to learn new concepts.

Since forgetting is more crucial in the presence of virtual drifts, CL originally started by focusing on data streams where concepts never overlap, and come one after the other [58]. This aligns well with the artificial stream perspective, where a huge amount of data is scanned sequentially and, when observing new distributions (due to the rise of virtual drifts), one aims to learn it without forgetting what has been learned in the past. This allowed researchers to design ad-hoc strategies to mitigate forgetting [57, 59].

The *class-incremental* scenario is one of the first and most studied CL scenarios [58]. The learning model, usually an artificial neural network, is required to i) solve a classification task where ii) classes are presented sequentially one after the other, and iii) once a class disappears after a drift, it never occurs again (no recurring drifts). Class-incremental scenarios essentially partition a given data stream into subsets with non-overlapping classes.

Class-incremental scenarios can be easily derived from any classification dataset. For example, one can take the MNIST dataset composed of images of digits, and build a class-incremental scenario by considering only pairs of digits at once. For instance, the model is first trained on all 0 and 1 digits (the first “task”), then on all 2 and 3 digits, and so on up until all 8 and 9 digits. At any point during training, the network should be able to correctly classify all digits seen up to that point. However, due to forgetting, a simple fine-tuning of the model leads to almost zero accuracy on previous digits, as the model only predicts the most recent digit pair.

Sometimes, a class-incremental scenario also provides task labels during inference, to help the model understand which task the data comes from. This scenario allows building modular architectures with task-specific components. It is therefore called task-incremental learning [60].

If class-incremental is the most challenging scenario for forgetting, it is by no means the only CL scenario. One can easily introduce repetitions of previous concepts (i.e., classes) in the data stream. Alternatively, one can only introduce new examples of existing classes. This scenario is usually called *domain-incremental* [60]. Continuing with our example based on MNIST, a domain-incremental scenario might require the network to classify digits as

either even or odd. If the data stream presents examples of two classes at a time, the model has to continuously adjust its idea of what an even or odd digit is. This scenario is usually associated with a lower amount of forgetting, as it does not introduce completely new concepts.

Notably, all the aforementioned scenarios present virtual drift, since each new experience introduces a new input distribution that was not observed during the past. Particularly, the situation presented in Figures 2c and 2d explicitly maps the domain-incremental setting. The class-incremental setting, in addition to introducing a new input distribution, also complicates the decision boundary by adding new classes in the new input distributions. CL rarely studies streams with real drifts. Such streams would require not only to forget outdated information, but also to *selectively* forget knowledge, a challenging endeavour with deep neural networks, where representations are usually entangled.

Given the type of scenarios and data streams faced by a CL model, researchers derived different ideas on how to combat forgetting. While a comprehensive overview of CL approaches is beyond the scope of our work, we summarize here some of the most important classes of approaches. This will also be useful when introducing a motivating example for Streaming Continual Learning, later in Section 6.

As we have seen, a class-incremental scenario presents new concepts sequentially without repetition of previous ones. This breaks the *i.i.d.* assumption, which many machine learning approaches rely on. To better approximate an i.i.d. distribution, *replay strategies* [61] keep a storage of previously seen examples (e.g.,  $K$  examples per class). At each training iteration, the model receives the current examples from the stream and a sample from the replay memory. Intuitively, replay mitigates forgetting as it restores knowledge from previous concepts. However, CL requirements usually do not allow for storing the entire data stream in memory, thus effectively limiting the replay memory size. Recent works explore settings with unlimited memory but limited computational budget, as they model some real-world cases where storage cost is often negligible.

Replay is by far the most popular CL approach for class-incremental scenarios. Regularization approaches [62] are also among the most studied ones. The idea is to penalize changes in the current model that would drive it away from the optimum for previous concepts. This high-level idea is instantiated differently depending on the context: i) each weight can bring an importance

value related to previous tasks and receive an update inversely proportional to its importance; ii) the current model might be encouraged to approximate its predictions on previous tasks through distillation, to remain functionally similar to the previously reached optimum; iii) the update direction might be constrained to remain (exactly or approximately) orthogonal to previous update directions to mitigate interference. Other classes of CL approaches, called *architectural strategies*, dynamically expand the model upon receiving new data, usually leading to multi-module models with task-specific components [63, 64]. Naturally, different CL approaches can be combined to take the best of several possible options (hybrid approaches).

*Evaluation in CL.* Unlike SML, prequential evaluation is rarely used in CL. The CL framework includes a stream  $\mathcal{S} = (s_1, s_2, \dots)$  of a potentially infinite amount of experiences (or tasks, or concepts). Each  $s_i$  provides a set of examples, and drifts usually occur between any experiences. The model is not subject to strong constraints on the allotted training time on each experience, and it can therefore be trained for multiple epochs until convergence. At any point in time, the model is evaluated on a set of unseen examples (the test set), which are held out from the data in each experience. When training on  $s_i$ , the evaluation on the test set of  $s_i$  measures the generalization performance of the model. Differently, when training on  $s_i$ , evaluating on the test set of  $s_j, j < i$  measures the ability of the model to retain previous knowledge. That is, the forgetting. More formally, let us consider the supervised classification task (a similar reasoning applies to other tasks, like regression). The evaluation metric is the prediction accuracy (percentage of correctly classified examples). The metric ACC [65] is computed as  $\frac{\sum_i a_{i,N}}{N}$ , where  $N$  is the total number of experiences encountered by the CL agent and  $a_{i,j}$  is the average accuracy on the test set of  $s_i$  after training on  $s_j$ . ACC measures the overall predictive performance of a CL agent on a set of  $N$  experiences.

The Backward Transfer (BWT) [65] measures the knowledge retention. It is therefore equal to negative forgetting ( $BWT = -F$  with  $F$  forgetting). For a given experience  $s_i$ , its BWT is computed as  $a_{i,j} - a_{i,i}, j > i$ . The first term measures the performance on experience  $i$  after training on the (current) task  $j$ , while the second term measures the original performance on experience  $i$  right after training on it (it is the reference value). A positive value indicates that learning new experiences benefits the performance on  $s_i$ .

On the contrary, a negative value denotes forgetting. When averaged over a set of tasks, the forgetting metric is called Average Forgetting.

This short overview of evaluation in CL highlights how the model always has access to a set of held-out examples from each experience, even though they are only used for evaluation and never for training.

*Online continual learning (OCL).* As we have seen, CL does not impose strict constraints on the allotted training time or on the number of examples the model accesses at a time. We call the setup presented so far “batch CL”, to highlight that each experience carries a large amount of examples, and the model can be trained over multiple epochs until convergence. In contrast, OCL [66] focuses on a scenario much more similar to SML. Data arrives one “minibatch” at a time. In practice, this means that each experience carries only a few examples. The model can be trained for a few iterations on the same minibatch. However, this setup is much more restrictive than a multi-epoch training on a large batch of data. OCL preserves the evaluation setup where the model can be queried at any time on data belonging to previous experiences. One popular evaluation metric in OCL requires computing the predictive accuracy at any point, even after each iteration. The average of these measurements over the entire stream is called the Average Anytime Accuracy [67].

The CL scenarios presented before still apply. For example, the class-incremental scenario can be easily repurposed to OCL by splitting each experience into smaller experiences with only a few examples. As a result, OCL streams are composed of more experiences than CL ones.

Even though some of the challenges of SML and CL meet in the OCL scenario, the current literature is still dominated by the CL view: learning models are usually deep artificial neural networks, mitigating forgetting remains the main focus, and virtual drift is the most common type of drift.

However, OCL highlights the need for fast adaptation to new knowledge, as new data points arrive quickly, and training cannot be performed over multiple epochs. In addition to these strict time constraints, OCL also obeys the memory constraints of CL: no storage of the entire history of the data stream is allowed. Replay is still used in OCL, by leveraging a memory buffer of *fixed-length*. All these characteristics will be shared by the SCL framework.

*Continual Reinforcement Learning.* In Reinforcement Learning (RL), an agent implements a continuous, reciprocal relationship with the environment it is

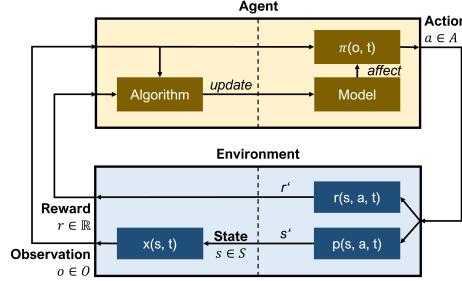


Figure 3: An RL control loop: as a reaction to actions  $a(t)$  taken by the agent, it receives observations  $o(t)$  and rewards  $r(t)$ .

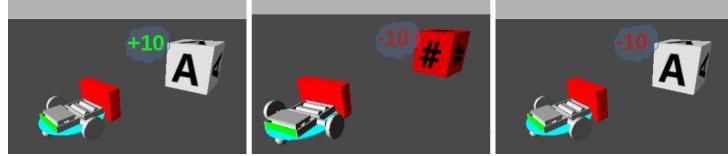


Figure 4: Virtual and real drift in Continual Reinforcement Learning (CRL). In this example scenario, a robot is rewarded or punished when touching certain objects, depending on form, color, and displayed symbol. Left: an object that should be touched, giving +10 reward. Middle: virtual drift, introducing a different object that should not be touched (or else: +10 reward). Right: real drift, changing the reward previously obtained for touching the object to punishment.

situated in: the agent continuously takes actions, which impact the environment and thus the observations and rewards obtained from it, see fig. 3. This involves learning from non-stationary data distributions even when environments themselves can be considered stationary. This is mainly because the assessment of different actions will evolve as learning progresses, leading to different actions being taken in similar situations, and thus to different observations. As such, this would mainly represent virtual drift (observations change, but not the optimality of actions in the same circumstances).

However, in Continual Reinforcement Learning (CRL), we consider environments themselves to be non-stationary. This implies that, for the same observation, different actions may be required for maximum reward at different points in time, which corresponds to a real shift, or even a combination of both (fig. 4).

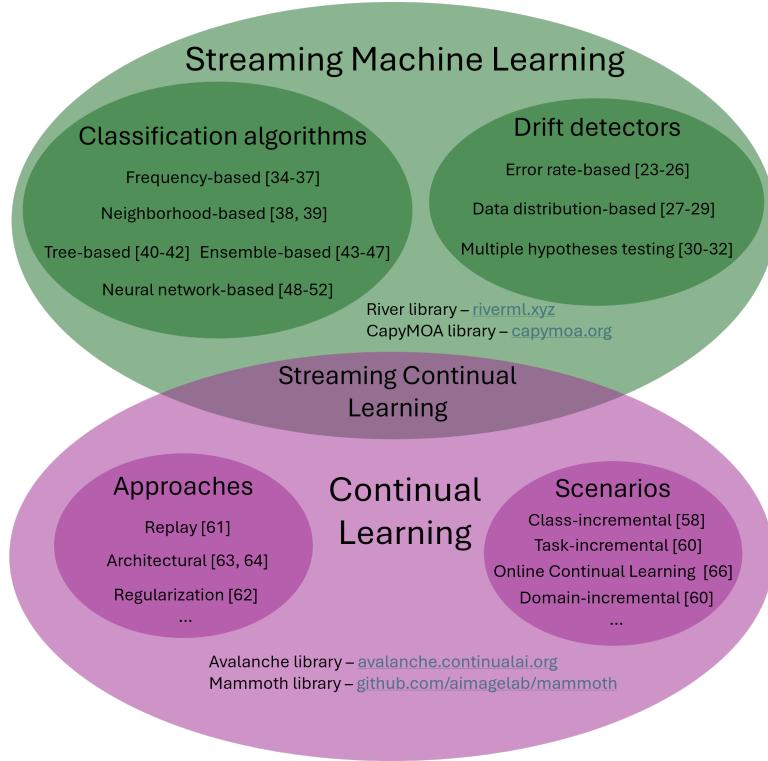


Figure 5: Summary of SML and CL, with some examples of popular state-of-the-art approaches and popular libraries. We refer to existing reviews for an in-depth treatment of either field.

## 5. Streaming Continual Learning

SCL emerges from the need to unify the complementary goals of SML and CL (summarized in Figure 5) when operating on evolving data streams, which may experience both real and virtual drifts. SML focuses on the current concept and aims to perform well on the currently observed feature subspace. SML does not address the issue of forgetting and makes no effort to assess whether knowledge from previously observed subspaces remains valid. Its main objective is to quickly adapt to new concepts, even at the cost of losing prior information. Since both real and virtual drifts require model updates, SML literature has often claimed that distinguishing between them is of limited practical relevance [2, 12, 68, 13, 18]. Conversely, CL typically assumes that concept drifts are only virtual (often expansionary), meaning that previously acquired knowledge remains valid.

The goal in this case is to retain previously learned knowledge while incorporating new information.

This distinction is also reflected in the evaluation protocols used in each field. SML usually adopts prequential evaluation (Section 3), assessing performance only on the current concept. This fits natural streaming scenarios, where data are generated in real time, and the focus is on immediate adaptation. In contrast, CL evaluates performance across all previously encountered concepts (Section 4), which is more suited to artificial streaming scenarios, where the full problem cannot be observed at once. Instead, it is split into subproblems (CL experiences), and the model is expected to remember past knowledge while learning new tasks.

Blending these two views into a unified framework is at the core of SCL. From this perspective, the model should maintain a general understanding of the entire stream while actively relying only on knowledge that remains valid for the current concept, or on reinterpreting prior knowledge in a way that supports current learning. Any knowledge that is not immediately useful should be preserved and made accessible for potential future reuse.

Crucially, as discussed in [8], the environment itself dictates which knowledge should be retained or discarded. New data may introduce patterns that either complement (virtual drifts) or replace existing ones (real drifts). In some situations, prior knowledge becomes irrelevant and should be suppressed to prevent interference. In others, it may still be valuable, either because it partially overlaps with new concepts or because it is likely to recur. Moreover, real drifts may not invalidate prior knowledge entirely: mild drifts or those with limited influence zones might still leave large portions of the input space unaffected.

Furthermore, in many real contexts, forgetting should not be seen solely as a drawback but as a necessary strategy for managing limited resources. The model should retain general, reusable knowledge while dynamically adapting to changes in the environment. This implies a more selective and context-aware form of forgetting, where information is not erased indiscriminately but evaluated based on its relevance to the ongoing data stream.

*SCL and Online CL.* As we mentioned in Section 4, research in OCL already took an important step in the SCL direction. Firstly, OCL considers data streams where only a few examples are available at a time. Secondly, OCL

incorporates anytime evaluation at a high frequency (up to every training iteration). Thirdly, OCL shares the tight computational constraints of SCL. However, the current direction of OCL research is still heavily oriented towards the mitigation of catastrophic forgetting, where neural networks and class-incremental learning scenarios are prominent. SCL aims at tackling more general scenarios, with different types of drifts possibly appearing in the data stream (e.g., real drifts). SCL offers the opportunity to build a shared vocabulary, connecting (O)CL and SML researchers, expanding the scope of their application domains, and leveraging existing works to design hybrid approaches.

*SCL approaches.* SCL paradigms should exhibit five essential capabilities [8]: (1) rapid adaptation to both real and virtual drifts, (2) autonomous detection of concept drifts, (3) ability to learn effectively from single (or a few) data points, (4) capacity to form hierarchical, structured representations, and (5) selective retention of relevant knowledge to mitigate forgetting. Notably, the ability to build hierarchical representations contributes directly to better knowledge selection, as it enables the model to structure the knowledge and identify which components could be reused or adapted.

In this context, SML methodologies could play an important role in monitoring and detecting changes and quickly adapting to them, while learning continuously from single data points. On the other side, CL and OCL are explicitly meant to structure and organize latent representations using deep learning models and apply strategies to mitigate forgetting.

Notably, Replay and regularization CL strategies may show limitations in streaming settings characterized by real concept drifts. These approaches typically attempt to retain and jointly optimize for all previously seen concepts. However, when past and current concepts conflict (as is common with real drifts), this simultaneous optimization can introduce incompatible learning signals, ultimately preventing convergence.

In contrast, **architectural strategies** (another popular and varied set of CL approaches) offer a more modular and scalable alternative. By structurally decoupling the parameters associated with different concepts, these methods can mitigate interference and allow for selective reuse of relevant components. Such architectures enable the system to isolate, preserve, and repurpose knowledge through mechanisms like dynamic module expansion or masking mechanisms. This design allows the learner to focus adaptively on the current concept while retaining access to past knowledge, aligning

closely with the core principles of SCL, where adaptability and scalability are crucial.

Regarding the evaluation procedure, one may apply both prequential evaluation to measure how the model is performing on the current concept and classical CL evaluation to estimate how the model retains the past concepts. Tracking how the model forgets over time provides valuable insight into the effects of newly acquired knowledge. Even when a previously learned concept loses its relevance, analyzing shifts in its accuracy can offer important information. Ideally, the model should preserve high-level knowledge that is common across concepts, as this can aid in solving current tasks [8].

Following this direction, we presented a proposal for an architecture that can embody SCL solutions: **fast and slow learners** [8] and takes inspiration from the Complementary Learning Systems (CLS) theory [69, 70]. The CLS theory distinguishes between a fast-learning system for rapid adaptation and a slow-learning system for stable, long-term consolidation. This duality naturally maps onto the SCL paradigm, where a fast learner (SML) quickly adapts to incoming data, while a slow learner (CL) retains generalizable knowledge and mitigates forgetting. The interaction between the two is bidirectional: the slow learner benefits from relevance signals and recurrence detected by the fast learner, while the fast learner can exploit structured representations built over time by the slow learner. This coordination enables SCL to adapt rapidly to change while preserving useful knowledge across the stream.

*Datasets for SCL.* As demonstrated by the quick rise of class-incremental scenarios in CL, leveraging existing datasets and benchmarks is fundamental to driving research and innovation in a field. SCL is no exception. [We briefly discuss existing datasets that can be quickly repurposed for experiments in SCL.](#) In particular, all the datasets listed below can be used within an SCL stream, provided that the evaluation protocol used to optimize the learning models accounts for both quick adaptation and mitigation of forgetting (e.g., by using prequential evaluation and the BWT metric). In general, any dataset including a temporal dimension and shifts in the data distribution is a suitable candidate for an SCL stream.

CLEAR [71] targets gradual drifts in the temporal evolution of visual concepts. CLEAR can be used for semi-supervised learning, as it includes labelled and unlabeled data [adapted from the YFCC100M dataset \[72\]](#). Interestingly, SCL data streams from CLEAR can easily include real drifts.

In fact, a given concept changes shapes and characteristics over time, thus requiring updating the learned representations. Together with the introduction of new concepts over time, CLEAR can be readily used to create SCL streams mixing real and virtual drifts.

Wild-Time [73] focuses on temporal distribution shifts. Wild-Time provides time stamps for each example, enabling models to build on previously encountered patterns for quick adaptation. The benchmark includes five different real-world datasets on a set of different domains, such as drug discovery, patient prognosis, and news classification.

Similarly, CLAD [74] and KITTI [75] capture temporal drifts in vision-based tasks (e.g., object detection). In particular, KITTI includes 6 hours of real-world traffic recorded through various sensors.

CLOC [76] is a dataset of geolocalized, time-stamped images from various locations around the world. The original paper uses CLOC in an OCL setting, therefore being already compatible with SCL approaches.

EGO4D [77] is a large-scale dataset of daily-life activities from egocentric videos of several users. It is therefore a suitable candidate for personalization tasks where the temporal aspect is key.

Natural language processing is another crucial test-bed for SCL. Like EGO4D, Firehose [78] allows for benchmarking personalization approaches, as it provides tweets from multiple users over time.

TemporalWiki [79] is a dataset of time-stamped Wikipedia and Wikidata pages, allowing for simulation of a data stream where knowledge is continuously updated as some information becomes outdated while new information is included.

Recently, [80] introduced a benchmark for land use classification from satellite images specifically designed with SCL in mind. Interestingly, predictive models deployed directly on satellites do not have access to large computational resources, making this use case fit for several SCL applications.

*Streaming/Continual Reinforcement Learning.* When using the tools developed for CL in CRL, several conceptual problems immediately arise. First of all, CL techniques are often intrinsically tied to the notion of *tasks*, that is, similarly to the definition of concepts, periods of stationary data statistics whose onsets are known. This idea, when transferred to RL, would imply periods of stationary *environment* statistics. Thus, drift would not occur during tasks, but only at task onsets. Such a notion is already quite artificial in RL,

especially the a priori knowledge about task onsets, and totally incompatible with streaming learning. Task onsets, if they were abrupt, could be detected by dedicated methods, especially for real drifts, where the well-known TD error (the difference between the expected and obtained reward for an action that was executed) should increase. Gradual drift could be detected as well, but this would invalidate the entire concept of a task itself.

The notion of a task is often very useful. For example, replay methods in CL [9] partition the memory buffer into fixed-size slots. Each slot gathers examples from the same class (for supervised classification problems) or from the same task (if a task identifier is available). Similarly, replay methods in CRL can exploit information about task labels to partition the replay buffer. However, replay can sometimes be deleterious. In particular, when dealing with an SCL stream where real drifts occur, performing replay would actually preserve the information that the real drifts has now made obsolete.

In the absence of a clear separation between tasks in the stream, replay can still work considering boundary-free approaches, like the works in [81, 82, 83, 84].

## 6. A motivating example for SCL

In this Section, we present a simple example to motivate the need for SCL and to empirically prove that SML and CL struggle to find a joint solution to the challenges of quickly adapting to concept drifts while avoiding forgetting. The data and code are available here for *reproducibility* (in the case of acceptance, they will be uploaded to a public repository on GitHub).<sup>4</sup>

### 6.1. Experimental setting

We compare SML and OCL methodologies in two different scenarios by leveraging the MNIST dataset to build our data streams.

The first scenario (**virtual drift scenario**) involves only virtual abrupt drifts. Each experience contains images from two digits (one odd digit and one even digit per experience). The learning model is required to classify each digit as odd or even (binary classification). After each experience, a

---

<sup>4</sup><https://drive.google.com/drive/folders/1nrotkzWlrUtYuTtyE34LD0QF9H2sArCV?usp=sharing>

new couple of digits is introduced. This drift alters the data distribution  $P(X_t)$ .

To better understand the nature of virtual drifts, consider a simplified representation of each digit using a 7-dimensional binary input space, similar to the structure of a seven-segment digital display (fig. 6). Each dimension corresponds to one segment that can be either on (1) or off (0). For example, the digit 1 activates only two segments, whereas 7 adds the top horizontal segment  $x_7$ . The digit 0 activates all segments except the central one ( $x_7$ ), while 8 turns on all seven. In this representation, each digit occupies a distinct region of the input space. Although some digits share identical values on a subset of features. For instance, even if 1 and 7 differ only in  $x_1$ , they still occupy different regions of the input space. In other words, when projected onto the subset of the features defined by  $x_2, \dots, x_7$ , their representations are indistinguishable, but they lie on different hyperplanes along  $x_1$  (e.g.,  $x_1 = 0$  for 1,  $x_1 = 1$  for 7). Even when digits are similar or lie close in the input space, they are still defined by different combinations of active dimensions and thus reside in separate subregions.

This toy example is not meant to faithfully reproduce the structure of real MNIST images. Instead, it provides an intuitive explanation of how virtual drifts operate. New experiences introduce inputs that occupy previously unseen regions of the input space, while the task definition (odd vs. even classification) remains unchanged. Thus, the labels assigned to the previously observed digits do not change over time. Each drift simply adds new types of instances that extend the problem without contradicting what has been observed so far. The model must learn to extend the previously learned classification rules to previously unexplored regions while retaining the knowledge it has acquired. This example simplifies the much more complex pixel-based input space of MNIST, which operates in grayscale and high dimensionality. In the actual MNIST setting, digits are represented in a much higher-dimensional grayscale pixel space, and although the geometry is far more complex, images belonging to the same digit class still tend to cluster in distinct regions. The simplified 7-segment example is used purely for conceptual illustration and is not used in the experiments.

The second scenario (**real drift scenario**) involves real abrupt drifts. It consists of five distinct binary classification tasks: (i) even vs. odd, (ii)  $> 4$  vs.  $\leq 4$ , (iii) multiple of three vs. not multiple of three, (iv) prime number (including one) vs. non-prime number, (v)  $\in [2, 5]$  vs.  $\notin [2, 5]$ .

Each experience uses the same input distribution, as all examples from all

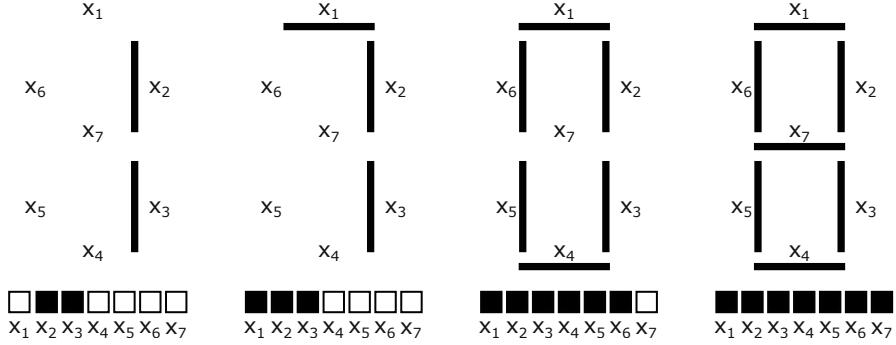


Figure 6: Illustration of digits 1, 7, 0, and 8 using a seven-segment display representation. Each digit is encoded as a binary vector over seven dimensions ( $x_1, \dots, x_7$ ), where each  $x_i$  corresponds to one segment (on = 1, off = 0). Although some digits share subsets of active segments (e.g., 1 and 7), their full encodings place them in distinct regions of the input space. This simplified scenario reflects the nature of virtual drifts in our experiments, where new digits introduce non-overlapping patterns in the input distribution. While MNIST uses grayscale pixel values in a high-dimensional space, the same principle holds: each digit activates a different subset of features, occupying a unique region in the space.

digits are present. However, each experience asks the model to solve a separate task, thus defining a different classification problem. In this case, the distribution  $P(X_t)$  remains fixed, while  $P(y_t|X_t)$  changes, since the same digit may be assigned different labels depending on the binary task. Additionally, some tasks may share class labels for a subset of the digits.

Our **research hypotheses** are as follows. **H1** states that mitigating forgetting slows down the adaptation to new concepts. **H2** states that SML models are not stable and forget the previous concepts after both real and virtual drifts. **H3** states that CL strategies struggle to avoid forgetting after real drifts. **H4** states that CL strategies struggle to efficiently learn the new concept after real drifts.

To verify these hypotheses, aligning with our SCL perspective, we apply both prequential and CL evaluations. Since some classification problems of the real drift scenarios are imbalanced, we consider Cohen’s Kappa Score [85]. Cohen’s Kappa is a metric that measures the agreement between predicted and true labels while correcting for the agreement that could occur by chance. Unlike plain accuracy, it provides a more reliable indication of model performance, especially in imbalanced settings. A value of 1 indicates perfect agreement, meaning every prediction is correct; a value of 0 suggests that the model’s predictions are no better than random guessing; negative values

imply systematic disagreement. In general, values between 0.6 and 0.8 reflect substantial agreement, and those above 0.8 are interpreted as near-perfect, indicating that the model is consistently making accurate predictions beyond chance.

As explained in Section 3, the prequential evaluation updates the score incrementally whenever a new digit  $d_t = (X_t, y_t)$  is generated.  $X_t$  is the feature vector representing the image, and  $y_t$  is the real binary label. The model receives the feature vector  $X_t$  and predicts the associated label  $\hat{y}_t$ . Then, the score is updated considering  $y_t$  and  $\hat{y}_t$ . The prequential evaluation returns an updated score after each data point using a *rolling window* of 1k data points. Thus, at timestamp  $t$ , it considers the data points from  $d_{t-999}$  to  $d_t$ . After a concept drift, it resets the window to focus on the adaptation to the new concept, avoiding considering the previous concept’s last data points. For the first 1k data points following a drift, it only considers the predictions of the data points from the drift onwards. It measures how the model is performing on the recent data points.

During prequential evaluation, a checkpoint of each model is stored after each drift, to be used for the CL evaluation. When the prequential evaluation ends, the CL evaluation computes the average Cohen’s Kappa ( $K_{avg}$ ) and backward transfer ( $BWT$ ) metrics using a test set for each concept, which is held out during training (see Section 4). For both metrics, we consider the complete evolution over the stream by averaging across all drifts. Formally, after the concept  $i$  ends, the two metrics are computed as in Equations 1 and 2, where  $K_{i,j}$  represents the Cohen’s Kappa score achieved by the checkpoint stored after the end of concept  $i$  and tested on the test set of concept  $j$ .

$$K_{avg} = \frac{\sum_{i=1}^N \sum_{j=1}^i K_{i,j}}{\frac{N \cdot (N+1)}{2}}, \quad (1)$$

$$BWT = \frac{\sum_{i=2}^N \sum_{j=1}^{i-1} (K_{i,j} - K_{j,j})}{\frac{N \cdot (N-1)}{2}}. \quad (2)$$

To investigate the effects of CL in streaming scenarios, we evaluate a set of representative replay and regularization strategies. As highlighted in Section 5, these methods are widely used but may struggle with real concept drifts, where conflicting information from past and current tasks can hinder convergence. We consider the following SML models and CL strategies.

- ARF [46] (see Section 3) is one of the most well-known models in SML. It consists of a streaming version of Random Forest and includes internal drift detectors to autonomously adapt to changes in the data stream, without requiring external drift signals.
- Experience Replay (ER) [61] is a widely used strategy in CL that mitigates forgetting by storing a small subset of past examples in a memory buffer and periodically revisiting them during training, allowing the model to reinforce previous knowledge while learning new tasks.
- AGEM [86] prevents forgetting by projecting the current gradient update onto the closest direction that does not increase the loss on a small memory of past examples. Instead of retraining directly on stored data, it uses these samples to constrain updates, ensuring that new learning does not interfere with previously acquired knowledge.
- Naive classifier: it is the base learning model without any CL strategy. The model is fine-tuned over the data stream continuously.

The methodologies’ choice is directly tied to the nature of real drifts and to the hypotheses formulated in this work. Replay and regularization CL strategies (represented by ER and AGEM) are included because they jointly optimize past and current tasks. While this approach is effective in virtual drift scenarios, it is fundamentally misaligned with real drifts, where past and current concepts may be incompatible. Under these conditions, the gradient signals introduced by replay or regularization can become contradictory, preventing convergence and slowing down the adaptation process. For this reason, ER and AGEM provide ideal representatives to test whether these mechanisms fail under real drift, as predicted by H1, H3, and H4.

The Naive classifier is included as a fully plastic model without any strategy to avoid forgetting. When compared to CL strategies, it represents a lower-bound baseline for stability evaluation and an upper-bound baseline for plasticity evaluation. In contrast, ARF is selected as a representative of SML approaches due to its well-established ensemble mechanism and robustness in streaming settings.

CL learning strategies are trained using the OCL approach with mini-batches containing ten images. The concept boundaries are given to the model. During each step of the prequential evaluation, ARF performs inference on a single data point and is trained on the same data point. OCL

methods follow the same protocol, but with minibatches of 10 examples at a time. No method performs multiple passes on the same data points. To perform a fair comparison between decision tree-based and deep learning-based models, images are represented using a common feature space of 30 components given by a UMAP model [87]. The UMAP model is trained offline on the whole MNIST dataset.

OCL strategies are implemented using the Avalanche library[88]<sup>5</sup> with a base learner consisting of a neural network with one hidden layer and size 512. All the models use a Stochastic Gradient Descent optimizer with learning rate 0.001 and momentum 0.9.<sup>6</sup> ARF is implemented using the River library with default parameters<sup>7</sup>. ER uses a replay buffer of ten data points for each mini-batch, sampled from a memory of 500 images.

## 6.2. Results

For the virtual drift scenario, we average results across ten runs with different orders and pairs of digits. For the real drift scenario, we follow the same protocol by changing the order of the classification tasks.

Results are shown in fig. 7 (prequential evaluation) and fig. 8 (CL evaluation). We recall that the main difference lies in the fact that the CL evaluation tests the models after each drift on the test sets of the concepts encountered so far. Table 1 shows the results for  $K_{avg}$  and  $BWT$  metrics.

When analyzing the virtual drift scenario, the prequential evaluation (fig. 7a) reveals that the decision tree model performs worse than the neural network. ARF, in fact, generally achieves lower prequential accuracy than the Naive classifier. ARF can adapt more quickly to the very first data points following a drift, whereas the Naive classifier performs very poor. However, it is important to note that the Naive classifier is trained incrementally on mini-batches, which introduces a delay in adapting to new data. Both ER and AGEM are outperformed by the Naive classifier, indicating that applying specific CL strategies can slightly hinder the learning of the current concept. This outcome is further clarified by the CL evaluation (fig. 8a and Table 1). ER exhibits the best performance in terms of mitigating forgetting, highlighting its stability. However, this stability comes at the expense of reduced

---

<sup>5</sup><https://avalanche.continualai.org/>

<sup>6</sup><https://docs.pytorch.org/docs/stable/generated/torch.optim.SGD.html>

<sup>7</sup><https://riverml.xyz/0.11.1/api/ensemble/AdaptiveRandomForestClassifier/>

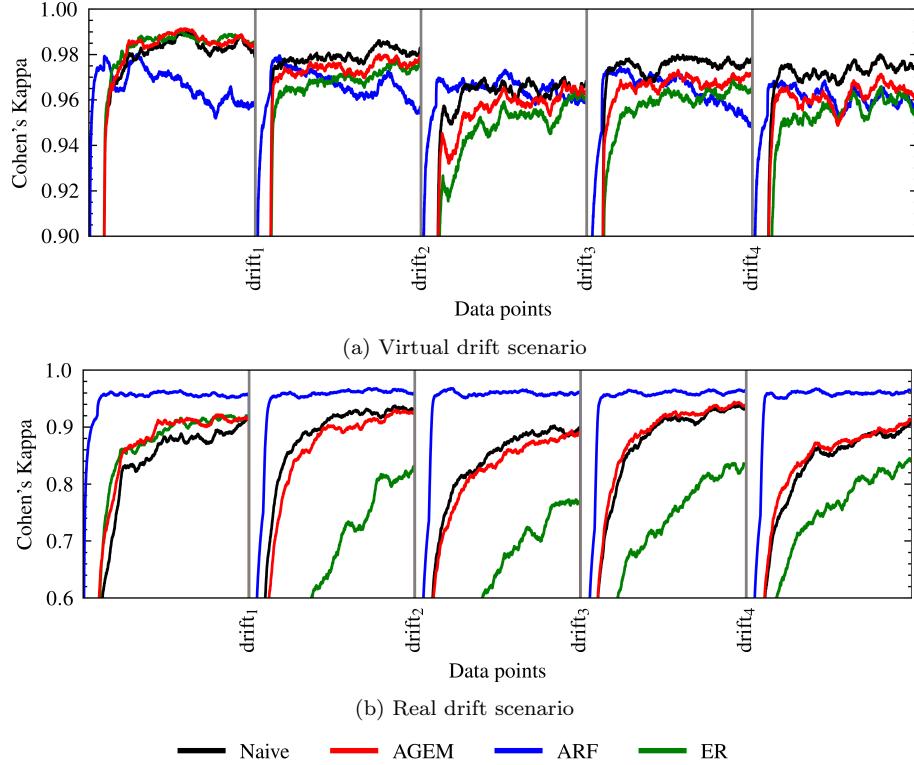


Figure 7: Prequential evaluation results averaged over the ten executions (considering Cohen's Kappa score). In the virtual drift scenario (8a), the Naive classifier performs well. ER and AGEM obtain slightly lower performance, since they balance between plasticity and stability. In the real drift scenarios (8b), the Naive classifier and AGEM perform equally, since the AGEM strategy has no impact. ER has strong difficulties to learn the new concept since it tries to mitigate forgetting. ARF is plastic in both scenarios.

plasticity, in line with the well-known stability-plasticity trade-off [3]. In contrast, Naive and ARF show strong online performance but suffer from catastrophic forgetting of previous concepts, since they do not apply specific strategies to mitigate it. AGEM, while slightly less stable than ER, achieves better online performance and approaches the Naive classifier.

Results are significantly different in the real drift scenario. As shown in the prequential evaluation (fig. 7b), decision trees outperform neural networks. AGEM performs on par with the Naive classifier, while ER struggles to learn the current concept effectively. This outcome is further explained

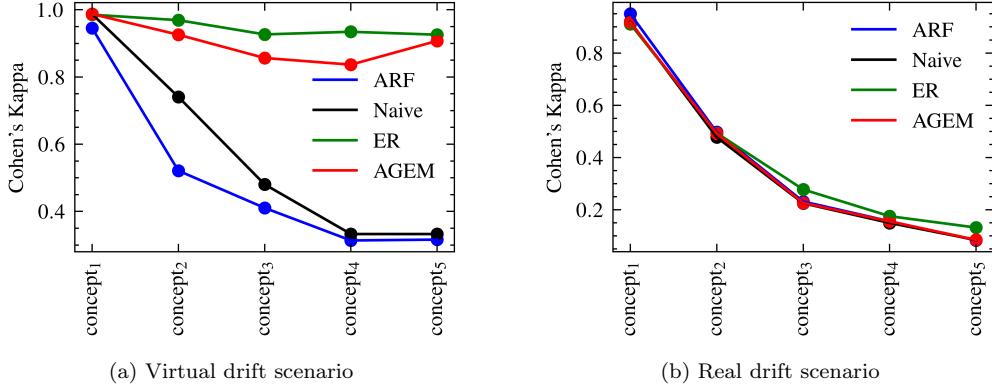


Figure 8: CL learning evaluation. After each drift, the models are tested over the current and all the previous test sets, and the average Cohen's Kappa score is computed. Finally, results are averaged over the ten executions. On the virtual drift scenario (8a), ER and AGEM are stable, while ARF and the Naive classifier catastrophically forget previous concepts. On the virtual drift scenario (8b), all the models cannot avoid forgetting. This is due to the fact that the concepts are in contradiction, and the model cannot solve both of them jointly.

by the CL evaluation results (fig. 8b and Table 1). AGEM and the Naive classifier achieve similarly poor scores, indicating that AGEM fails to provide any meaningful improvement and behaves essentially like a Naive classifier. ARF once again catastrophically forgets all previously learned concepts. ER performs slightly better in terms of retaining past knowledge, but still fails to prevent forgetting effectively. The underlying issue lies in ER's training approach: it mixes the current mini-batch with replayed examples from past concepts. However, in the presence of real drifts, these past examples are associated with previous decision boundaries that contradict the new target concept. As a result, the model receives contradictory training signals, which not only prevent it from preserving old knowledge but also hinder its ability to learn the new concept. Consequently, training fails to converge, and ER is unable to adapt properly in this setting.

To summarize, all the research hypotheses are verified. **H1:** the Naive classifier is more plastic than CL strategies. **H2:** ARF exhibits a poor CL performance in both scenarios. **H3:** in the real drift scenario, AGEM and ER cannot avoid forgetting since previous concepts are in contradiction with the current ones, and the model cannot solve all of them jointly. **H4:** in

	Virtual drift scenario		Real drift scenario	
	$K_{avg}$	BWT	$K_{avg}$	BWT
<b>AGEM</b>	$0.89 \pm 0.06$	$-0.13 \pm 0.10$	$0.24 \pm 0.01$	$-0.99 \pm 0.05$
<b>ARF</b>	$0.40 \pm 0.09$	$-0.83 \pm 0.14$	$0.24 \pm 0.00$	$-1.06 \pm 0.00$
<b>ER</b>	$0.94 \pm 0.02$	$-0.04 \pm 0.02$	$0.27 \pm 0.02$	$-0.85 \pm 0.08$
<b>Naive</b>	$0.46 \pm 0.09$	$-0.77 \pm 0.14$	$0.24 \pm 0.01$	$-1.00 \pm 0.04$

Table 1: Mean and standard deviation of the CL metrics over the ten executions using Cohen’s Kappa score.

the real drift scenario, AGEM is not effective and performs on par with the Naive classifier. Similarly, ER struggles with learning the current concept.

## 7. Conclusion and Future Directions

We discussed the Streaming Continual Learning scenario, where a model is required to quickly adapt to new information while retaining previous knowledge. We motivated the need for SCL with an empirical evaluation of existing SML and CL strategies, briefly showing how the two objectives are currently not met by any of the approaches alone (Section 6). While CL focuses on knowledge consolidation to avoid forgetting, SML focuses on rapid adaptation. This also results in different evaluation protocols, hiding the opportunities for a positive exchange of ideas and solutions.

We do not believe that SCL will replace SML or CL. Rather, we strive for an integration of the two approaches. Throughout this paper, we claimed that such opportunities are already available, and that they are currently being neglected.

While SCL ideas are gaining traction along several directions (e.g., online continual learning, SML with deep representation learning), a unified approach is still missing. Bringing together the SML and CL communities will speed up the process.

We foresee potential applications of SCL in areas where i) temporal correlations play a role and ii) virtual and real drift can occur at any time.

The presence of *temporal dependencies* influences many real-world scenarios, including the Internet of Things, robotics, object tracking, video surveillance, and satellite imagery analysis [9]. Despite its crucial relevance [52],

this problem is usually ignored by both research areas. As highlighted by M5 challenge [89], combining statistical and deep learning approaches can be effective in modelling temporal patterns. Moving towards an SCL perspective, integrating SML and CL methodologies could provide a viable solution. Since Recurrent Neural Networks models are widely applied in this case, they can also be used online in an SML way to learn continuously and handle temporal dependencies. Furthermore, architectural CL strategies may be applied on top of them to avoid forgetting and exploit transfer learning. Continuous Progressive Neural Networks [51] represent one of the first pioneering solutions that apply this rationale.

One example satisfying both temporal dependence and the presence of real and virtual drifts is the case of *personalization*. When a predictive model needs to be tailored to each user’s needs, it is important to consider how the user’s preferences evolve over time, what is relevant, and what is instead becoming outdated.

Equipped with the intuitions and the results from Section 6, SCL approaches would enable rapid adaptation to novel needs (virtual drifts), while at the same time preventing forgetting of old preferences. Selective forgetting (e.g., via model editing or similar techniques) would still be required in the presence of real drift, where old preferences become outdated. As we have seen, carefully balancing these two dimensions is still a challenge for existing approaches, and one of the most important and pressing motivations behind the study of SCL scenarios.

## Acknowledgments

This work has been partially funded by the EU-EIC EMERGE project (Grant No. 101070918) and by the HEU CoEvolution project (Grant No. 101168560).

## References

- [1] A. Bifet, R. Gavaldà, G. Holmes, B. Pfahringer, Machine learning for data streams: with practical examples in MOA, MIT press, 2018.
- [2] A. Tsymbal, The problem of concept drift: definitions and related work, Computer Science Department, Trinity College Dublin 106 (2) (2004) 58.

- [3] G. Carpenter, S. Grossberg, Adaptive Resonance Theory: Stable self-organization of neural recognition codes in response to arbitrary lists of input patterns, in: Proceedings of the Eight Annual Conference of the Cognitive Science Society, Hillsdale, NJ: Erlbaum, 1986, pp. 45–62.
- [4] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of learning and motivation, Vol. 24, Elsevier, 1989, pp. 109–165.
- [5] M. Mermilliod, A. Bugaiska, P. Bonin, The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects (2013).
- [6] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N. D. Rodríguez, Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, *Inf. Fusion* 58 (2020) 52–68.
- [7] N. Gunasekara, B. Pfahringer, H. M. Gomes, A. Bifet, Survey on Online Streaming Continual Learning, in: Thirty-Second International Joint Conference on Artificial Intelligence, Vol. 6, 2023, pp. 6628–6637. doi: [10.24963/ijcai.2023/743](https://doi.org/10.24963/ijcai.2023/743).
- [8] F. Giannini, G. Ziffer, A. Cossu, V. Lomonaco, Streaming Continual Learning for Unified Adaptive Intelligence in Dynamic Environments, *IEEE Intell. Syst.* 39 (6) (2024) 81–85.
- [9] A. Cossu, D. Bacciu, A. Bernardo, E. D. Valle, A. Gepperth, F. Giannini, B. Hammer, G. Ziffer, Don't Drift Away: Advances and Applications of Streaming and Continual Learning, in: ESANN 2025 Proceedings, Ciaco - i6doc.com, Bruges (Belgium) and online, 2025. doi: [10.14428/esann/2025.ES2025-23](https://doi.org/10.14428/esann/2025.ES2025-23).
- [10] J. Read, R. A. Rios, T. Nogueira, R. F. de Mello, Data Streams Are Time Series: Challenging Assumptions, in: BRACIS (2), Vol. 12320 of Lecture Notes in Computer Science, Springer, 2020, pp. 529–543.
- [11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under Concept Drift: A Review, *IEEE Trans. Knowl. Data Eng.* 31 (12) (2019) 2346–2363.

- [12] S. J. Delany, P. Cunningham, A. Tsymbal, L. Coyle, A case-based technique for tracking concept drift in spam filtering, *Knowl. Based Syst.* 18 (4-5) (2005) 187–195.
- [13] T. R. Hoens, R. Polikar, N. V. Chawla, Learning from streaming data with concept drift and imbalance: an overview, *Prog. Artif. Intell.* 1 (1) (2012) 89–101.
- [14] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, *ACM Comput. Surv.* 46 (4) (2014) 44:1–44:37.
- [15] G. Widmer, M. Kubat, Effective learning in dynamic environments by explicit context tracking, in: *ECML*, Vol. 667 of *Lecture Notes in Computer Science*, Springer, 1993, pp. 227–243.
- [16] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: A survey, *IEEE Comput. Intell. Mag.* 10 (4) (2015) 12–25.
- [17] I. Khamassi, M. Sayed Mouchaweh, M. Hammami, K. Ghédira, Discussion and review on evolving data streams and concept drift adapting, *Evol. Syst.* 9 (1) (2018) 1–23.
- [18] S. Wares, J. Isaacs, E. Elyan, Data stream mining: methods and challenges for handling concept drift, *SN Applied Sciences* 1 (11) (2019) 1–19.
- [19] A. Costa, R. Giusti, E. M. dos Santos, Analysis of descriptors of concept drift and their impacts, *Informatics* 12 (1) (2025) 13. [doi:10.3390/informatics12010013](https://doi.org/10.3390/informatics12010013).  
URL <https://doi.org/10.3390/informatics12010013>
- [20] G. I. Webb, R. Hyde, H. Cao, H. Nguyen, F. Petitjean, Characterizing concept drift, *Data Min. Knowl. Discov.* 30 (4) (2016) 964–994. [doi:10.1007/s10618-015-0448-4](https://doi.org/10.1007/s10618-015-0448-4).  
URL <https://doi.org/10.1007/s10618-015-0448-4>
- [21] J. Gama, R. Sebastião, P. P. Rodrigues, Issues in evaluation of stream learning algorithms, in: *KDD*, ACM, 2009, pp. 329–338.

- [22] J. Gama, R. Sebastião, P. P. Rodrigues, On evaluating stream learning algorithms, *Mach. Learn.* 90 (3) (2013) 317–346.
- [23] A. Bifet, R. Gavaldà, Learning from Time-Changing Data with Adaptive Windowing, in: *SDM*, SIAM, 2007, pp. 443–448.
- [24] M. Baena-Garcia, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: *Fourth international workshop on knowledge discovery from data streams*, Vol. 6, 2006, pp. 77–86.
- [25] H. Raza, G. Prasad, Y. Li, Ewma model based shift-detection methods for detecting covariate shifts in non-stationary environments, *Pattern Recognition* 48 (3) (2015) 659–669.
- [26] G. J. Ross, N. M. Adams, D. K. Tasoulis, D. J. Hand, Exponentially weighted moving average charts for detecting concept drift, *Pattern recognition letters* 33 (2) (2012) 191–198.
- [27] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, *Artificial Intelligence* 209 (2014) 11–28.
- [28] T. Dasu, S. Krishnan, S. Venkatasubramanian, K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in: *Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*, Vol. 12, Citeseer, 2006, pp. 1–22.
- [29] D. Kifer, S. Ben-David, J. Gehrke, Detecting change in data streams, in: *VLDB*, Vol. 4, Toronto, Canada, 2004, pp. 180–191.
- [30] C. Alippi, M. Roveri, Just-in-time adaptive classifiers—part i: Detecting nonstationary changes, *IEEE Transactions on Neural Networks* 19 (7) (2008) 1145–1153.
- [31] Y. Zhang, G. Chu, P. Li, X. Hu, X. Wu, Three-layer concept drifting detection in text data streams, *Neurocomputing* 260 (2017) 393–403.
- [32] B. I. F. Maciel, S. G. T. C. Santos, R. S. M. Barros, A lightweight concept drift detection ensemble, in: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2015, pp. 1061–1068.

- [33] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, S. Maniu, Data stream analysis: Foundations, major tasks and tools, *WIREs Data Mining Knowl. Discov.* 11 (3) (2021).
- [34] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, *Mach. Learn.* 29 (2-3) (1997) 131–163.
- [35] M. Bahri, S. Maniu, A. Bifet, A sketch-based naive bayes algorithms for evolving data streams, in: *IEEE BigData*, IEEE, 2018, pp. 604–613.
- [36] S. Wagner, M. Zimmermann, E. Ntoutsi, M. Spiliopoulou, Ageing-based multinomial naive bayes classifiers over opinionated data streams, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2015, pp. 401–416.
- [37] C. Salperwyck, V. Lemaire, C. Hue, Incremental weighted naive bays classifiers for data stream, in: *Data Science, Learning by Latent Structures, and Knowledge Discovery*, Springer, 2015, pp. 179–190.
- [38] V. Losing, B. Hammer, H. Wersing, KNN classifier with self adjusting memory for heterogeneous concept drift, in: *ICDM*, IEEE Computer Society, 2016, pp. 291–300.
- [39] W. Dong, M. Charikar, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, in: *WWW*, ACM, 2011, pp. 577–586.
- [40] W. Hoeffding, Probability inequalities for sums of bounded random variables, in: *The collected works of Wassily Hoeffding*, Springer, 1994, pp. 409–426.
- [41] P. M. Domingos, G. Hulten, Mining high-speed data streams, in: *KDD*, ACM, 2000, pp. 71–80.
- [42] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, in: *IDA*, Vol. 5772 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 249–260.
- [43] N. C. Oza, Online bagging and boosting, in: *SMC*, IEEE, 2005, pp. 2340–2345.

- [44] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: ECML/PKDD (1), Vol. 6321 of Lecture Notes in Computer Science, Springer, 2010, pp. 135–150.
- [45] H. M. Gomes, J. Read, A. Bifet, Streaming random patches for evolving data stream classification, in: ICDM, IEEE, 2019, pp. 240–249.
- [46] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfahringer, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, *Mach. Learn.* 106 (9-10) (2017) 1469–1495.
- [47] H. M. Gomes, J. P. Barddal, F. Enembreck, A. Bifet, A survey on ensemble learning for data stream classification, *ACM Comput. Surv.* 50 (2) (2017) 23:1–23:36.
- [48] N. Gunasekara, H. M. Gomes, B. Pfahringer, A. Bifet, Online hyperparameter optimization for streaming neural networks, in: IJCNN, IEEE, 2022, pp. 1–9.
- [49] A. Ashfahani, M. Pratama, Autonomous deep learning: Continual learning approach for dynamic environments, in: SDM, SIAM, 2019, pp. 666–674.
- [50] Á. C. Lemos Neto, R. A. Coelho, C. L. d. Castro, An Incremental Learning Approach Using Long Short-Term Memory Neural Networks, *Journal of Control, Automation and Electrical Systems* (2022) 1–9.
- [51] F. Giannini, G. Ziffer, E. Della Valle, cPNN: Continuous Progressive Neural Networks for Evolving Streaming Time Series, in: PAKDD (4), Vol. 13938 of Lecture Notes in Computer Science, Springer, 2023, pp. 328–340.
- [52] G. Ziffer, F. Giannini, E. Della Valle, Tenet: Benchmarking Data Stream Classifiers in Presence of Temporal Dependence, in: 2024 IEEE International Conference on Big Data (BigData), 2024, pp. 1187–1196. [doi:10.1109/BigData62323.2024.10825670](https://doi.org/10.1109/BigData62323.2024.10825670).
- [53] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, et al.,

River: machine learning for streaming data in python, *Journal of Machine Learning Research* 22 (110) (2021) 1–8.

- [54] Y. Sun, H. M. Gomes, A. Lee, N. Gunasekara, G. Weigert Cassales, J. J. Liu, M. Heyden, V. Cerqueira, M. Bahri, Y. S. Koh, et al., Machine learning for data streams with capymoa, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2025, pp. 438–443.
- [55] R. M. French, Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks, in: *In Proceedings of the 13th Annual Cognitive Science Society Conference*, Erlbaum, 1991, pp. 173–178.
- [56] A. Robins, Catastrophic Forgetting; Catastrophic Interference; Stability; Plasticity; Rehearsal., *Connection Science* 7 (2) (1995) 123–146. [doi:10.1080/09540099550039318](https://doi.org/10.1080/09540099550039318).
- [57] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, J. van de Weijer, Class-Incremental Learning: Survey and Performance Evaluation on Image Classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (5) (2023) 5513–5533. [doi:10.1109/TPAMI.2022.3213473](https://doi.org/10.1109/TPAMI.2022.3213473).
- [58] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, iCaRL: Incremental Classifier and Representation Learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.
- [59] E. Verwimp, R. Aljundi, S. Ben-David, M. Bethge, A. Cossu, A. Gephardt, T. L. Hayes, E. Hüllermeier, C. Kanan, D. Kudithipudi, C. H. Lampert, M. Mundt, R. Pascanu, A. Popescu, A. S. Tolias, J. van de Weijer, B. Liu, V. Lomonaco, T. Tuytelaars, G. M. van de Ven, Continual Learning: Applications and the Road Forward, *Transactions on Machine Learning Research* (2024).
- [60] G. M. van de Ven, A. S. Tolias, Three scenarios for continual learning, *Continual Learning Workshop NeurIPS* (2018). [arXiv:1904.07734](https://arxiv.org/abs/1904.07734).
- [61] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, C. Kanan, Replay in Deep Learning: Current Approaches

and Missing Biological Elements, *Neural computation* 33 (11) (2021) 2908–2950. [doi:10.1162/neco\\_a\\_01433](https://doi.org/10.1162/neco_a_01433).

- [62] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, *Neural Networks* 113 (2019) 54–71. [doi:10.1016/j.neunet.2019.01.012](https://doi.org/10.1016/j.neunet.2019.01.012).
- [63] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive Neural Networks, *arXiv* (2016). [arXiv:1606.04671](https://arxiv.org/abs/1606.04671).
- [64] G. Sokar, D. C. Mocanu, M. Pechenizkiy, SpaceNet: Make Free Space for Continual Learning, *Neurocomputing* 439 (2021) 1–11. [doi:10.1016/j.neucom.2021.01.078](https://doi.org/10.1016/j.neucom.2021.01.078).
- [65] D. Lopez-Paz, M. A. Ranzato, Gradient Episodic Memory for Continual Learning, in: *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc., 2017.
- [66] A. Soutif-Cormerais, A. Carta, A. Cossu, J. Hurtado, V. Lomonaco, J. Van de Weijer, H. Hemati, A Comprehensive Empirical Evaluation on Online Continual Learning, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2023, pp. 3518–3528.
- [67] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, E. Belilovsky, New Insights on Reducing Abrupt Representation Change in Online Continual Learning, in: *International Conference on Learning Representations*, 2022.
- [68] I. Zliobaite, Learning under concept drift: an overview, *CoRR* [abs/1010.4784](https://arxiv.org/abs/1010.4784) (2010).
- [69] D. Kumaran, D. Hassabis, J. L. McClelland, What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated, *Trends in Cognitive Sciences* 20 (7) (2016) 512–534. [doi:10.1016/j.tics.2016.05.004](https://doi.org/10.1016/j.tics.2016.05.004).
- [70] J. L. McClelland, B. L. McNaughton, A. K. Lampinen, Integration of New Information in Memory: New Insights from a Complementary Learning Systems Perspective, *Philosophical Transactions of the*

Royal Society B: Biological Sciences 375 (1799) (2020) 20190637. doi: 10.1098/rstb.2019.0637.

[71] Z. Lin, J. Shi, D. Pathak, D. Ramanan, The CLEAR Benchmark: Continual LEArning on Real-World Imagery, in: Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2022.

[72] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, L.-J. Li, YFCC100M: The new data in multimedia research, Commun. ACM 59 (2) (2016) 64–73. doi:10.1145/2812802.

[73] H. Yao, C. Choi, B. Cao, Y. Lee, P. W. W. Koh, C. Finn, Wild-time: A benchmark of in-the-wild distribution shift over time, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., 2022, pp. 10309–10324.  
URL [https://proceedings.neurips.cc/paper\\_files/paper/2022/file/43119db5d59f07cc08fc7ba6820179a-Paper-Datasets\\_and\\_Benchmarks.pdf](https://proceedings.neurips.cc/paper_files/paper/2022/file/43119db5d59f07cc08fc7ba6820179a-Paper-Datasets_and_Benchmarks.pdf)

[74] E. Verwimp, K. Yang, S. Parisot, L. Hong, S. McDonagh, E. Pérez-Pellitero, M. De Lange, T. Tuytelaars, Clad: A realistic continual learning benchmark for autonomous driving, Neural Networks 161 (2023) 659–669.

[75] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, The International Journal of Robotics Research 32 (11) (2013) 1231–1237.

[76] Z. Cai, O. Sener, V. Koltun, Online Continual Learning with Natural Distribution Shifts: An Empirical Study with Visual Data, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, 2021, pp. 8261–8270. doi:10.1109/ICCV48922.2021.00817.

[77] K. G. et al., Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, IEEE, 2024, pp. 19383–19400. doi:10.1109/CVPR52733.

2024.01834.

URL <https://doi.org/10.1109/CVPR52733.2024.01834>

- [78] H. Hu, O. Sener, F. Sha, V. Koltun, Drinking From a Firehose: Continual Learning With Web-Scale Natural Language, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (5) (2023) 5684–5696. [doi:10.1109/TPAMI.2022.3218265](https://doi.org/10.1109/TPAMI.2022.3218265).
- [79] J. Jang, S. Ye, C. Lee, S. Yang, J. Shin, J. Han, G. Kim, M. Seo, Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, Abu Dhabi, United Arab Emirates, December 7-11, 2022, Association for Computational Linguistics, 2022, pp. 6237–6250. [doi:10.18653/V1/2022.EMNLP-MAIN.418](https://doi.org/10.18653/V1/2022.EMNLP-MAIN.418)  
URL <https://doi.org/10.18653/v1/2022.emnlp-main.418>
- [80] L. Iovine, G. Ziffer, A. Proia, E. Della Valle, Towards streaming land use classification of images with temporal distribution shifts, in: *ESANN Proceedings*, 2025.
- [81] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, L. Page-Caccia, Online continual learning with maximal interfered retrieval, *Advances in neural information processing systems* 32 (2019).
- [82] A. Krawczyk, A. Gepperth, Adiabatic replay for continual learning, in: *International Joint Conference on Neural Networks (IJCNN)*, 2024.
- [83] A. Krawczyk, A. Gepperth, An analysis of best-practice strategies for replay and rehearsal in continual learning, in: *CVPR Workshop Paper (CLVISION)*, 2024.
- [84] Y. Denker, A. Gepperth, Continual reinforcement learning without replay buffers, in: *IEEE International Conference on Intelligent Systems (IS)*, 2024.
- [85] M. L. McHugh, Interrater reliability: the kappa statistic, *Biochemia medica* 22 (3) (2012) 276–282.
- [86] A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient Lifelong Learning with A-GEM, in: *ICLR (Poster)*, OpenReview.net, 2019.

- [87] L. McInnes, J. Healy, N. Saul, L. Großberger, UMAP: uniform manifold approximation and projection, *J. Open Source Softw.* 3 (29) (2018) 861.
- [88] A. Carta, L. Pellegrini, A. Cossu, H. Hemati, V. Lomonaco, Avalanche: A PyTorch Library for Deep Continual Learning, *Journal of Machine Learning Research* 24 (363) (2023) 1–6.
- [89] S. Makridakis, E. Spiliotis, V. Assimakopoulos, M5 accuracy competition: Results, findings, and conclusions, *IJOF* 38 (4) (2022) 1346–1364.