Survey on Online Streaming Continual Learning

Nuwan Gunasekara[1], **Bernhard Pfahringer**[1], **Heitor Murilo Gomes**[2] and **Albert Bifet**[1,3]

[1]*AI Institute, University of Waikato*
[2]*Victoria University of Wellington*
[3]*LTCI, Télécom Paris*

ng98@students.waikato.ac.nz, {abifet,bernhard}@waikato.ac.nz, heitor.gomes@vuw.ac.nz

Abstract

Stream Learning (SL) attempts to learn from a data stream efficiently. A data stream learning algorithm should adapt to input data distribution shifts without sacrificing accuracy. These distribution shifts are known as "concept drifts" in the literature. SL provides many supervised, semi-supervised, and unsupervised methods for detecting and adjusting to concept drift. On the other hand, Continual Learning (CL) attempts to preserve previous knowledge while performing well on the current concept when confronted with concept drift. In Online Continual Learning (OCL), this learning happens online. This survey explores the intersection of those two online learning paradigms to find synergies. We identify this intersection as Online Streaming Continual Learning (OSCL). The study starts with a gentle introduction to SL and then explores CL. Next, it explores OSCL from SL and OCL perspectives to point out new research trends and give directions for future research.

1 Introduction

Stream Learning (SL) focuses on efficiently learning from streaming data by learning from one instance at a time. The requirements for Stream Learning are: to predict at any given moment, dynamically adapt to underlying data distribution changes (concept drifts), and be computationally efficient when learning and predicting [Bifet *et al.*, 2018]. Data streams are often assumed to be IID, but in most cases, data streams are often non-IID as the data distribution changes over time.

Continual Learning (CL), attempts to learn from a non-IID data stream to preserve and extend already accrued knowledge [Mai *et al.*, 2022]. The learning algorithm is expected to strike a balance between stability and plasticity as the stream undergoes distribution shifts. Furthermore, in Online Continual Learning (OCL) this learning happens online; thus, the learning algorithm is only allowed a single pass over the data [Mai *et al.*, 2022].

We identify Online Streaming Continual Learning (OSCL) as the intersection between Stream Learning and Online Continual Learning. OSCL allows well-researched SL fields such as efficient stream learners, concept drift detection, and adaptation to enhance or develop OCL methods. This survey attempts to understand the intersection of those two closely

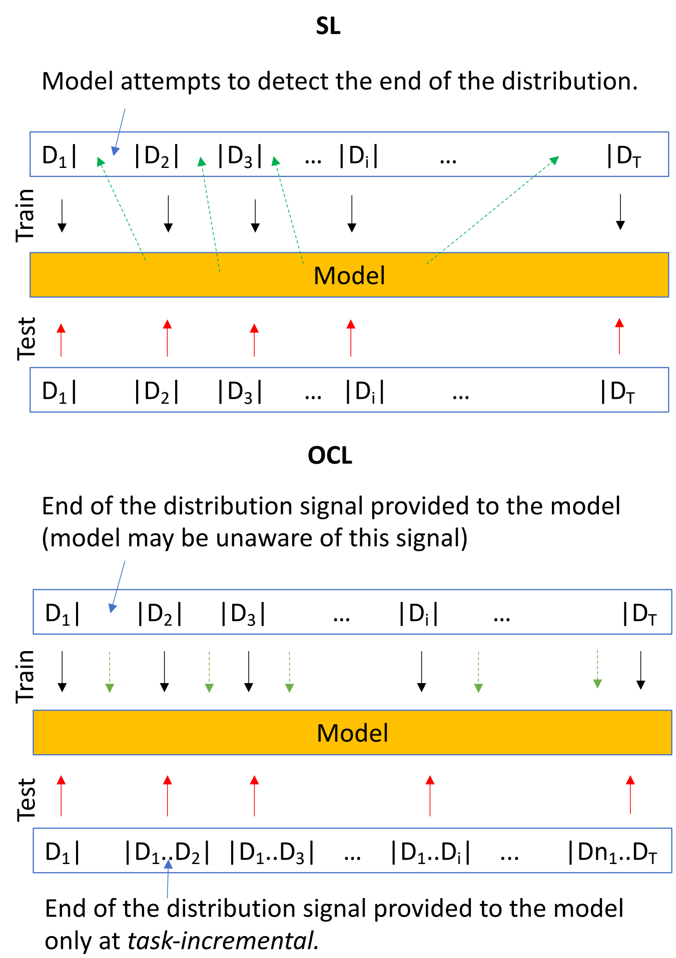

Figure 1: Comparison between SL and OCL settings. SL: uses drift detectors discussed in section 2.1 to detect distribution shifts, and evaluation is discussed in section 2.3. OCL: uses evaluation metrics (equations 1, 2, 3, and 4) discussed in section 2.3. The models in this setting do not detect distribution shifts. OSCL proposes some of the techniques used in SL to be used in OCL.

related fields, considering the underlying setting, evaluation methods, and applications. It also suggests future directions for OSCL taking into account recent advances in SL.

The rest of the paper is organized as follows. Section one explains Stream Learning considering the setting, drift adaptation, methods, evaluation, and applications. The next section explores Continual Learning considering the same criteria except for drift adaptation. Figure 1 compares Stream Learning (SL) and Online Continual Learning. Section 4 explains some of the intersection points between these two fields. These intersection points lay the ground for OSCL setting. This is further explained in this section considering new trends and future directions. Finally, we provide our conclusions in the last section.

2 Stream Learning

In Stream Learning, a model learns from an evolving data stream (non-IID data), processing one instance at a time. The learner must predict at any given moment using limited processing and memory [Bifet *et al.*, 2018; Gomes *et al.*, 2017a]. Also, it should adjust to distribution changes in the underlying input distribution [Bifet *et al.*, 2018; Bifet and Gavalda, 2007]. A shift in the data distribution is identified as a concept drift in literature.

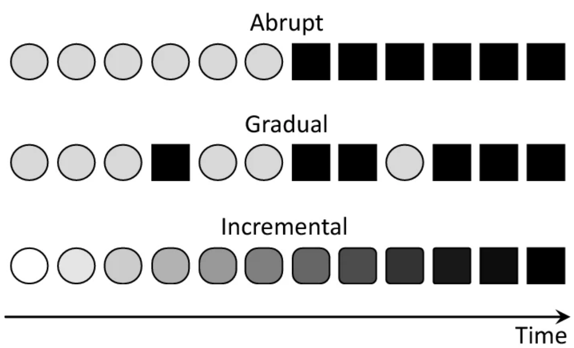

Figure 2: Evolution of different drift types under the "*Evolution of relationship between features and the target and the speed of change*" category: abrupt, gradual, and incremental. Source: [Souza *et al.*, 2020].

2.1 Concept drift

Concept drifts can be categorized according to their impact on the decision boundary, the evolution of the relationship between features and the target, speed of change, reach, and recurrence [Suárez-Cetrulo *et al.*, 2023].

- *Effect on the decision boundary (impact)*: the literature describes real and virtual concept drifts. The former effects the the decision boundary of the model. This affects the performance of the model. The latter does not affect the decision boundary. Hence the model performance is unaffected [Ramírez-Gallego *et al.*, 2017].
- *Evolution of the relationship between features and the target and the speed of change*: in the literature, drifts are categorized into sudden (abrupt), gradual, and incremental drifts, considering the evolution of the relationship between features and the target and the speed of change. With sudden or abrupt drifts, the current data distribution changes to a new one within a short period [Ramírez-Gallego *et al.*, 2017]. In the case of gradual drifts, this transition happens gradually [Suárez-Cetrulo *et al.*, 2023]. Here for a certain period, one could observe instances from both distributions. The transition time is very long with incremental drifts, and there may not be a statistical difference between adjacent instances [Ramírez-Gallego *et al.*, 2017]. Figure 2 shows how the drift types mentioned above evolve.
- *Reach of change*: drifts that affect all of the features are considered global drifts [Suárez-Cetrulo *et al.*, 2023], and drifts that affect some of the features are called local drifts [Khamassi *et al.*, 2015].
- *Recurrent concept drifts*: if a particular data distribution reoccurs in the stream after a given period, it is considered a recurrent concept drift [Suárez-Cetrulo *et al.*, 2023].
- *Random blips/outliers/noise*: are situations where, for a very short period of time, few instances which do not belong to the current distribution popup in the stream [Suárez-Cetrulo *et al.*, 2023].

Drift detectors

Many types of drift detectors are explained in the literature. [Souza *et al.*, 2020] explain three types of drift detectors for Stream Learning.

- *Methods based on differences between two distributions*: These methods compare the difference between two data windows. A reference window with old data and a detection window with recent data are compared using a statistical test to discard the null hypothesis that both data belong to the same distribution. Drift detectors based on fixed-size windows usually suffer from a delay in detection [Souza *et al.*, 2020]. Works such as ADaptive sliding WINdow (ADWIN) [Bifet and Gavalda, 2007] use dynamic windows.
- *Methods based on sequential analysis*: These are methods founded on the Sequential Probability Ratio Test (SPRT)[Wald, 1947]. CUSUM and Page–Hinkley [Page, 1954] are good examples of drift detectors of this type.
- *Methods based on statistical process control*: These methods consider the classification problem a statistical process and monitor the evolution of some performance indicators like error rate to apply heuristics to find change points. For example, DDM [Gama *et al.*, 2004] has three different states for the classification error evolution: *in-control* when the error is in the control level, *out-of-control* when the error is increasing significantly compared to the recent past, and *warning*, when the error is increasing but has not reached the out-of-control

level. Where DDM only looks at the magnitude of the errors, EDDM [Baena-Garcıa *et al.*, 2006] also considers the distance in time between consecutive errors.

We would like to direct the reader to work by [Khamassi *et al.*, 2018] and [Gama *et al.*, 2014] for a thorough review of drift detectors for Stream Learning.

2.2 Methods

Similar to batch learning, Stream Learning methods can be categorized into supervised, semi-supervised, and clustering. In supervised SL, it is assumed that target values are available for each instance. In semi-supervised Stream Learning, this assumption is relaxed for some instances. Target values may only become available at a later time or not be available at all. Clustering assumes the unavailability of target variables. Akin to batch learning, supervised SL has two main categories: classification and regression. There are quite a few popular classification methods proposed in SL. Starting with simple but effective learners like Naive Bayes (NB) and Hoeffding Tree (HT) to ensemble learners like Adaptive Random Forest (ARF), Streaming Random Patches (SRP), and **C**ontinuously **A**daptive **N**eural Networks for **D**ata Streams (CAND)[Gunasekara *et al.*, 2022c]. HT [Hulten *et al.*, 2001] builds a tree using the Hoeffding bound to control its split decisions with a given confidence. Later an adaptive version of it was introduced to replace the branches when the data stream is evolving [Bifet and Gavalda, 2009]. Ensemble methods have shown great success in Stream Learning [Gomes *et al.*, 2017a]. They allow the use of efficient SL base learners like HT in a bagging or random forest setting with efficient drift detectors like ADWIN [Gomes *et al.*, 2019]. ARF is an online random forest implementation for Stream Learning which uses effective re-sampling methods and drift adaptation mechanisms to cope with different types of concept drifts [Gomes *et al.*, 2017b]. SRP trains base models on random subsets of features and instances identified as patches [Gomes *et al.*, 2019]. It uses the same drift adaptation strategy as in ARF but produces better results than ARF. CAND trains a pool of simple NNs and uses the one with the smallest estimated loss for prediction. It employs ADWIN, an estimator to estimate each NN's loss. As CAND uses NNs as it base learners, it works well on high-dimensional data. We would like to direct the reader to [Gomes *et al.*, 2017a], which contains an extensive taxonomy of data stream ensemble classifiers.

Many data stream regression methods are explained in the literature [Choudhary *et al.*, 2021]. Hoeffding Tree Regressor (HTR) is an adaptation of the incremental tree algorithm HT for regression. Like HT, HTR uses the Hoeffding bound to control its split decisions. HTR relies on calculating the reduction of variance in the target space to find a split candidate. Fast Incremental Model Trees with Drift Detection (FIMT-DD) learn model trees from an evolving data stream with drift detection [Ikonomovska *et al.*, 2011]. It uses the variance reduction split criterion for splitting and the Page-Hinckley test for drift detection. More recent ensemble methods for streaming regression include Adaptive Random Forest Regressor (ARF-REG) [Gomes *et al.*, 2018], and Self-Optimising K-Nearest Leaves (SOKNL) [Sun *et al.*, 2022]. SOKNL claim to have superior accuracy compared to ARF-REG.

Label availability in a streaming setting can be categorized into four groups: (i) Immediate and fully labelled, (ii) Delayed and fully labelled, (iii) Immediate and partially labelled, (iv) Delayed and partially labelled [Gomes *et al.*, 2022]. The majority of data stream Semi-Supervised Learning (SSL) is devoted to understanding (iii). However, [Gomes *et al.*, 2022] highlights the importance of understanding the delayed and partially labelled (iv) setting. Furthermore, the authors categorize streaming SSL methods into: (i) *intrinsically* SSL, (ii) *self-training*, and (iii) *learning by disagreement*. *Intrinsically* SSL methods exploit the unlabelled instances directly as part of their objective function or optimization procedure [Gomes *et al.*, 2022]. *Self-training* methods are based on the idea that a classifier learns from its previous mistakes and then reinforces itself [Gomes *et al.*, 2022]. It can act as a wrapper algorithm that uses any arbitrary classifier. *Learning by disagreement* works by learners teaching other learners. Models are trained with multiple viewpoints of the same data[1], which results in disagreeing models. The key idea behind learning by disagreement is to generate multiple learners and let them collaborate to exploit the unlabelled data [Gomes *et al.*, 2022].

Data stream clustering can be categorized into: partition clustering, micro-cluster-based clustering, density-based clustering, and hierarchical clustering [Bahri *et al.*, 2021]. Instances from a stream are divided into segments without a class label. The objective of this type of SL is to discover patterns in the stream in an online fashion with a minimum amount of resources. Also, algorithms deployed in this setting should be able to cope with the evolving nature of the stream. The survey by [Zubaroğlu and Atalay, 2021] contains a recent and extensive study on this field.

2.3 Evaluation

Several methods are explained in the SL literature for evaluating a model. The most popular one is the *test-then-train* approach [Bifet *et al.*, 2018; Gama *et al.*, 2013]. As the name suggests, the evaluation uses the incoming instance to test the model first and later train the model. Here the current predictive evaluation is affected by the previous evaluations. This may be desirable when one is interested in the model's overall performance. Test-then-train is also known as prequential evaluation in the literature. The prequential evaluation may not be reliable in conveying the current predictive performance of the model. Therefore prequential evaluation can be equipped with a sliding window, or a fading factor, to gracefully forget the performance on instances from the distant past [Bifet *et al.*, 2018; Gama *et al.*, 2013]. For partly labelled data, prequential evaluation is still applicable as the loss can be calculated on just the labelled subset of instances [Gomes *et al.*, 2022]. Data stream *cross-validation* was introduced by [Bifet *et al.*, 2015] where models are trained and tested in parallel on different folds of the data. *Continuous re-evaluation* considers the verification latency in the streaming

[1]This could be achieved through techniques such as bootstrapping aggregation.

setting with partially delayed labels [Grzenda *et al.*, 2020b; Grzenda *et al.*, 2020a]. This evaluation attempts to evaluate how fast a model can transform from an initial possibly incorrect prediction to a correct prediction prior to the availability of the true label.

There are several metrics explained in the literature to measure the performance of an SL classification algorithm. The most popular one is accuracy. If the data stream is imbalanced, accuracy can be misleading; sensitivity and specificity are better measurement alternatives [Bahri *et al.*, 2021]. The *kappa* statistic compares the model's prequential accuracy against the chance classifier (one that randomly assigns to each class the same number of instances as the model under consideration) [Bifet *et al.*, 2018]. On the other hand, the *kappa M* compares the current model's performance against the majority class classifier [Bifet *et al.*, 2018]. *Kappa temporal* attempts to capture the temporal dependencies in a data stream by comparing the model performance against a "no-change" model, which predicts the next instance using the current instance's label [Bifet *et al.*, 2018][2]. For delayed label situations, when multiple predictions are made for a single instance, accuracy and kappa values can be aggregated to produce immediate measures until the true label is available [Grzenda *et al.*, 2020a; Gomes *et al.*, 2022].

Regression SL uses two main evaluation metrics: (i) Root mean squared error (RMSE) and (ii) Mean absolute error (MAE) [Bahri *et al.*, 2021]. We direct the reader to [Bifet *et al.*, 2018; Bahri *et al.*, 2021] for thorough reviews of regression evaluation methods and [Kremer *et al.*, 2011] for clustering evaluation methods. Furthermore, data stream evaluation also considers computing and memory usage [Bifet *et al.*, 2018].

2.4 Application

SL has been used in many situations where learning happens from an evolving data stream. [Souza *et al.*, 2020] used SL on data generated by optical sensors, which measure the flying behavior of insects to identify disease vector insects. Also [Gao and Lei, 2017] used SL methods for online crude oil price prediction. SL was used to predict power production considering environmental conditions by [Lobo *et al.*, 2020]. The study by [Žliobaitė *et al.*, 2016] contains some interesting applications of SL for monitoring and control problems. It includes application tasks such as traffic management, activity recognition, communication monitoring, controlling robots, intelligent appliances, intrusion detection, fraud detection, and insider trading. The study also contains some interesting areas where SL could provide solutions. We like to direct the reader to [Žliobaitė *et al.*, 2016] for a broader understanding of SL applications.

3 Continual Learning

The literature has thoroughly documented that an NN receiving non-IID data forgets past knowledge when confronted with a concept shift [Kirkpatrick *et al.*, 2017; Mai *et al.*, 2022]. CL attempts to learn with minimal forgetting of past concepts [Kirkpatrick *et al.*, 2017; Mai *et al.*, 2022]. In OCL, this learning happens online. Three main continual learning settings are described in the literature: task-incremental, class-incremental, and domain-incremental.

- *Task-incremental*: In this setting, output distributions are demarked by external task ids, available for training and testing. In this setting, the model can use the external task-id signal at test time [Mai *et al.*, 2022].
- *Class-incremental*: Each distribution consists of classes that are unavailable in other distributions (tasks). This setting adapts a single-head NN configuration. Here, output distributions differ from task to task [Mai *et al.*, 2022].
- *Domain-incremental*, on the other hand, assumes output distribution from one task to the other to be the same while having different input distributions [Mai *et al.*, 2022].

In both class-incremental and domain-incremental settings, an external task-id that separates one task from another is assumed to be unavailable at test time. The availability of this signal at training is optional. However, some CL methods rely on this signal during training. Online Class Incremental Continual Learning (OCICL) and Online Domain Incremental Continual Learning (ODICL) assume class-incremental and domain-incremental OCL settings, respectively.

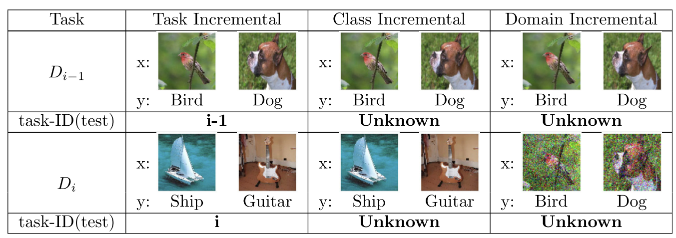

Figure 3: Three main CL settings discussed by [Mai *et al.*, 2022].*Task-incremental*: tasks are demarked by task id. Task id is available at the test time. *Class-incremental*: different classes are present at each task. Task id is not available at the test time. *Domain-incremental*: each task contains the same set of classes, but the input distribution changes from one task to another, e.g., blur vs. noise. Task id is not available at test time. Source: [Mai *et al.*, 2022].

3.1 Methods

CL algorithms use three popular approaches to avoid catastrophic forgetting in NNs: regularization, replay, and parameter isolation.

- *Regularization methods*: algorithms like Elastic Weight Consolidation (EWC) [Kirkpatrick *et al.*, 2017] and Learning without Forgetting (LwF) [Li and Hoiem, 2017] adjust the weights of the network in such a way that it minimizes the overwriting of the weights for the old concept. EWC uses a quadratic penalty to regularize updating the network parameters related to the past concept. It uses the Fisher Information Matrix's diagonal

[2] These measurements are thoroughly explained in [Bifet *et al.*, 2018]

to approximate the importance of the parameters [Kirkpatrick *et al.*, 2017]. EWC has some shortcomings: 1) the Fisher Information Matrix needs to be stored for each task, 2) it requires an extra pass over each task's data at the end of the training [Mai *et al.*, 2022]. Though different versions of EWC address these concerns [Mai *et al.*, 2022], [Chaudhry *et al.*, 2018] seems suitable for online CL by keeping a single Fisher Information Matrix calculated by a moving average. LwF uses knowledge distillation to preserve knowledge from past tasks. Here, the model related to the old task is kept separate, and a separate model is trained on the current task. When the LwF receives data for a new task (X_{new}, Y_{new}), it computes the output (Y_{old}) from the old model for the new data X_{new}. During training, assuming that $\hat{Y_{old}}$ and $\hat{Y_{new}}$ are predicted values for X_{new} from the old model and new model, LwF attempts to minimize the loss: $\alpha L_{KD}(Y_{old}, \hat{Y_{old}}) + L_{CE}(Y_{new}, \hat{Y_{new}}) + R$ [Mai *et al.*, 2022]. Here L_{KD} is the distillation loss for the old model, and α is the hyper-parameter controlling the strength of the old model against the new one. L_{CE} is the cross-entropy loss for the new task. R is the general regularization term. Due to this strong relation between old and new tasks, it may perform poorly in situations where there is a huge difference between old and new task distributions [Mai *et al.*, 2022].

- *Replay methods* present a mix of old and current concept's instances to the NN based on a given policy while training. This reduces forgetting as the training instances from the old concepts avoid complete overwriting of past concept's weights. GDumb [Prabhu *et al.*, 2020], Experience Replay (ER) [Chaudhry *et al.*, 2019], and Maximally Interfered Retrieval (MIR) [Aljundi *et al.*, 2019] are some of the most popular CL replay methods. GDumb attempts to maintain a class-balanced memory buffer using instances from the stream. At the end of the task, it trains the model using the buffered instances. ER uses reservoir sampling to sample instances from the stream to fill the buffer. Reservoir sampling ensures that every instance in the stream has the same probability of being selected to fill the buffer. ER uses random sampling to retrieve instances from the memory buffer. Despite its simplicity, ER has shown competitive performance in ODICL[Mai *et al.*, 2022]. Five (three buffer and two non-buffer) tricks have been proposed by [Buzzega *et al.*, 2021] to improve the accuracy of ER in the OCICL setting. The buffer tricks are independent buffer augmentation, balanced reservoir sampling, and loss-aware reservoir sampling. The two non-buffer tricks are bias control and exponential learning rate decay. Except for bias control which controls the bias of newly learned classes, these tricks can be used in ODICL to improve the performance of a replay method. MIR uses the same reservoir sampling as ER to fill the memory buffer. However, when retrieving instances from the buffer, it first does a virtual parameter update using the incoming mini-batch. Then it selects the top k randomly sampled instances with the most significant loss increases by the virtual parameter update for training. In the online implementation in [Mai *et al.*, 2022], this virtual update is done on a copy of the NN. Replay Using Memory Indexing (REMIND) [Hayes *et al.*, 2020] takes this approach to another level by storing the internal representations of the instances by the initial frozen part of the network and using a randomly selected set of these internal representations to train the last unfrozen layers of the network. REMIND can store more instance's representations using internal low-dimensional features. In general, these replay approaches are motivated by how the hippocampus in the brain stores and replays high-level representations of the memories to the neocortex to learn from them [Hayes *et al.*, 2020]. The empirical survey by [Mai *et al.*, 2022] suggests that ER and MIR perform better on OCICL and ODICL than other OCL methods. More recently [Zhang *et al.*, 2022] has proposed repeated augmented rehearsal to improve replay methods. The method utilize data argumentation for replayed instances to avoid over-fitting on replay buffer data[3]. The approach seems to improve all replay methods in general.

- *Parameter-isolation*: The intuition behind parameter-isolation methods is to avoid interference by allocating separate parameters for each task [Mai *et al.*, 2022]. There are two types of parameter-isolation-based methods: *fixed architecture* and *dynamic architecture*. Fixed architecture only activates the relevant part of the network without changing the NN architecture [Mai *et al.*, 2022]. Dynamic architecture, on the other hand, adds new parameters for the new task while keeping the old parameters [Yoon *et al.*, 2017; Mai *et al.*, 2022]. Continual Neural Dirichlet Process Mixture (CN-DPM) [Lin, 2013] trains a new model for each new task and leaves the existing models untouched so that at a later point, it can retain the knowledge of the past tasks. It composes of a group of experts where each expert contains a discriminative model and a generative model. Each expert is responsible for a subset of the data. The group is expanded based on the Dirichlet Process Mixture using Sequential Variational Approximation [Mai *et al.*, 2022].

We would like to direct the reader to a survey by [Mai *et al.*, 2022] for in-depth detail about those methods.

3.2 Evaluation

There are many evaluation metrics defined in the CL literature. On a stream with T tasks, after training the NN from tasks 1 to i, let $a_{i,j}$ be the accuracy on the held-out test set for task j. *Average accuracy* (A_i) at task i is defined as:

$$A_i = \frac{1}{i}\sum_{j=1}^{i} a_{i,j} \tag{1}$$

[3] A well-documented issue in replay methods [Zhang *et al.*, 2022].

[Chaudhry *et al.*, 2018]. Average forgetting (F_i) at task i is defined as:

$$F_i = \frac{1}{i-1}\sum_{j=1}^{i-1} f_{i,j} \tag{2}$$

, where

$$f_{k,j} = \max_{l \in \{1,\dots,k-1\}} (a_{l,j}) - a_{k,j} \forall j < k$$

Here $f_{k,j}$ is the best test accuracy the model has ever achieved on task j prior to learning task k. $a_{k,j}$ is the test accuracy on task j after learning task k [Chaudhry *et al.*, 2018]. The positive influence of learning a new task on previous tasks' performance is measured by *Positive Backward Transfer* (BWT):

$$BWT = \max\left(\frac{\sum_{i=2}^{T}\sum_{j=1}^{i-1} a_{i,j} - a_{j,j}}{\frac{T(T-1)}{2}}, 0\right) \tag{3}$$

[Mai *et al.*, 2022]. The positive influence of learning a given task on future tasks' performance is defined as *Forward Transfer* (FWT):

$$FWT = \frac{\sum_{i=1}^{T-1}\sum_{j=2}^{T} a_{i,j}}{\frac{T(T-1)}{2}} \forall i < j \tag{4}$$

[Mai *et al.*, 2022]. Further to the above metrics, run-time and memory usage are also considered when evaluating OCL methods [Mai *et al.*, 2022].

3.3 Applications

Recent research has focused on using ODICL methods to avoid costly retraining in practical situations where the model is confronted with a concept shift. ODICL has been used in X-ray image classification to avoid costly retraining on distribution shifts due to unforeseen shifts in hardware's physical properties [Srivastava *et al.*, 2021]. Also, it has been used to mitigate bias in facial expression and action unit recognition across different demographic groups [Kara *et al.*, 2021]. Furthermore, ODICL was used to counter retraining on concept shifts for multi-variate sequential data of critical care patient recordings [Armstrong and Clifton, 2021]. The authors highlight some replay method's infeasibility due to strong privacy requirements in clinical settings. This concern is further highlighted in the empirical study by [Mai *et al.*, 2022]. Practical implementations such as [Kara *et al.*, 2021] and [Armstrong and Clifton, 2021] use the end of the task signal to employ OCL methods such as EWC and LwF. However, on the other hand, practical implementation in [Srivastava *et al.*, 2021] assumes a gradual distribution shift in the input data distribution where instances from both the new and old tasks could appear in the stream for a certain period.

4 Online Streaming Continual Learning

In Stream Learning, the objective is to adjust to the current concept in the stream efficiently. On the other hand, OCL has dual learning objectives: adjust to the current concept while preserving knowledge about previous concepts. Both settings assume data is non-IID. In Stream Learning, it is assumed that model should detect distribution changes and adapt accordingly. However, in OCL, the end of the concept signal is provided at training time, even though some replay methods may not use it. This end-of-the-concept signal is only provided for task-incremental CL at test time. Figure 1 also shows the differences between these two settings. Contrary to the differences, these two fields have many intersection points. We identify these intersection points as Online Streaming Continual Learning (OSCL). In OSCL, we mainly identify how well-researched Stream Learning techniques and methods could be used to enhance OCL.

SL on recurrent concept drifts attempts to adjust to an evolving data stream where some concepts could reemerge at a later stage of the stream [Suárez-Cetrulo *et al.*, 2023]. The setting is similar to OCL but without the additional learning objective of preserving old knowledge explicitly. Hence evaluation in this setting does not consider how to measure forgetting of past knowledge. Most of the methods explored in this setting keep a fixed-size pool of classifiers [Suárez-Cetrulo *et al.*, 2023]. Various mechanisms are explored in the literature on how to maintain this pool [Suárez-Cetrulo *et al.*, 2023; Anderson *et al.*, 2019] and how to use it for prediction [Suárez-Cetrulo *et al.*, 2023; Almeida *et al.*, 2018]. This "pool of classifiers" is also known as "concept history", "concept list", and "concept repository" in the literature [Suárez-Cetrulo *et al.*, 2023]. Measures like *concept equivalence* and *concept similarity* were introduced to identify the current concept in the data stream from the concept pool.

- *Conceptual equivalence* assumes that when two classifiers behave similarly on a given time window, both describe the same concept [Yang *et al.*, 2006].
- *Concept similarity*: recognizes similar concepts using Euclidean distances between concept clusters [Li *et al.*, 2012]. Thus it can detect recurring drifts in unlabelled data.

Measures such as *concept equivalence* and *concept similarity* could be used for model selection and data retrieval from the replay buffer in OCL. When handling recurrent concepts, predicting the following concept is helpful so the learner can adjust to the incoming concept ahead of time. [Chen *et al.*, 2016] proposed a method that used a probabilistic network to predict future changes. [Maslov *et al.*, 2016] proposed a method to use patterns acquired during previous drifts to predict the time of the next drift. The method assumed a Gaussian distribution for the duration of the concepts. A recent survey by [Suárez-Cetrulo *et al.*, 2023] discusses the above and many more exciting topics on SL for recurrent concept drifts.

Most of the OCL methods rely on externally provided end-of-concept signals (task ids) at training[4]. It is critical for an autonomous learning agent to detect these concept shifts and adjust accordingly. While OCL research has explored different methods to preserve old knowledge when adjusting to

[4]Due to internal instance buffers, some OCL models may not need task ids at train time. The performance of these models is heavily dependent upon the size of this instance buffer [Mai *et al.*, 2022]

Topic	SL	OCL
Setting	Single learning objective: adjust to current concept efficiently.	Dual learning objective: adjust to current concept and preserve old knowledge.
Drift detection	Thoroughly studied	Can be used for task detection Some recent OCL work: [Gunasekara *et al.*, 2022a], and [Gunasekara *et al.*, 2022b].
Drift prediction.	Used when dealing with recurrent concept drifts.	Can be used for task prediction. Some SL work: [Chen *et al.*, 2016], [Suárez-Cetrulo *et al.*, 2023]
Missing labels	Some methods have been proposed to tackle this [Gomes *et al.*, 2022].	Yet to be fully explored. Can employ some of the SL approaches discussed in [Gomes *et al.*, 2022].
Recurrent concept drifts	Similar to OCL, without explicit learning objective to preserve old knowledge. For latest research refer to [Suárez-Cetrulo *et al.*, 2023].	SL concept pool maintenance techniques [Suárez-Cetrulo *et al.*, 2023] can be useful in maintaining references to different NN structures in OCL parameter-isolation methods. Concept equivalence and concept similarity can be used to retrieve relevant instances or NN structures. Many more techniques are discussed in [Suárez-Cetrulo *et al.*, 2023].
Evaluation	Frameworks can employ OCL dual learning objective and metrics discussed in section 3.2. So SL methods and techniques can be evaluated under OCL setting.	Employs dual learning objective.
Application	Suitable for applications which needs to adjust to current concept very quickly.	Suitable for applications which needs to adapt to current concept very quickly while preserving old knowledge.

Table 1: Synergies and differences between SL and OCL.

new concepts, SL has done an excellent job of understanding how to detect distribution shifts, especially through different drift detection methods on streams with different drift types (abrupt, gradual, incremental, and recurrent) and different label conditions (available for all instances/ delayed label/ no label). OCL could utilize well-researched drift detectors to detect the end-of-concept signal. Recent research in that direction includes Online Domain Incremental Pool (ODIP) [Gunasekara *et al.*, 2022b] and Online Domain Incremental Networks (ODIN) [Gunasekara *et al.*, 2022a], which use ADWIN as an end-of-task signal generator in an ODICL setting. These methods use this internal signal to adjust to newly perceived concepts automatically. Recent work by [Davalas *et al.*, 2022] has explored the use of drift detectors to identify when to use the data from the instance buffer and how to use it in an OCL setting. Having well-researched SL knowledge on different distribution shifts and different drift detectors would allow OCL algorithms to be more effective in practical OCL scenarios, like the gradual distribution shifts in x-ray images [Srivastava *et al.*, 2021].

Furthermore, semi-supervised SL could enhance OCL to be more practical in situations where label data is not always immediately available. Works like ORDisCo [Wang *et al.*, 2021] and CURL [Rao *et al.*, 2019] have already started exploring this research area. Semi-supervised SL methods under *Self-training* and *learning by disagreement* categories [Gomes *et al.*, 2022] could easily be deployed on real world OCL settings where labels are not always present. However, many opportunities exist considering the breadth of semi-supervised SL methods discussed by [Gomes *et al.*, 2022].

Streaming clustering is another exciting area that could be explored in future OCL work. Here, well-established streaming clustering algorithms [Zubaroğlu and Atalay, 2021] could solve interesting OCL problems as streaming clustering algorithms extract patterns from evolving data. One could use a streaming clustering algorithm to extract tasks from an unsupervised OCL setting. The extracted information, such as task information, could be used for model selection and data retrieval from the replay buffer. The most exciting aspect of the streaming clustering algorithms for OCL is that they are well-studied for evolving data streams.

On the other hand, SL could adapt the dual learning objective in OCL (adjust to the current concept while preserving knowledge about previous concepts). This would allow one to evaluate the breadth of well-studied SL methods for OCL. There is some emerging work in this area where catastrophic forgetting is explored in HTs [Korycki and Krawczyk, 2021]. Implementing the OCL evaluation discussed in [Mai *et al.*, 2022] on popular SL platforms such as MOA [Bifet *et al.*, 2010] and river [Montiel *et al.*, 2021] would speed up this area of research.

Table 1 summarizes the above-discussed synergies and differences between Stream Learning and OCL. It points out the differences in the settings and lists some future research directions in Online Streaming Continual Learning.

5 Conclusion

SL attempts to adjust to an evolving data stream with multiple concepts efficiently. The primary learning objective of the model is to perform well on the current concept. For the same setting, OCL, on the other hand, attempts to do well on the current concept while preserving the knowledge of past concepts. Drift detection techniques could be used to detect new concepts in OCL. So that OCL models could be trained without an external end-of-concept signal (task id). Also, mechanisms explored in SL for recurrent concept drifts could give new insights into predicting future drifts real world in OCL settings. Some of the techniques and methods explored in semi-supervised SL could allow OCL to be applicable in situations where labels are not always available. Furthermore, popular SL frameworks can employ the dual learning objective discussed in OCL. This would allow the OCL community to evaluate numerous SL methods and techniques for OCL.

References

[Aljundi *et al.*, 2019] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *NeurIPS*, 2019.

[Almeida *et al.*, 2018] Paulo RL Almeida, Luiz S Oliveira, Alceu S Britto Jr, and R Sabourin. Adapting dynamic classifier selection for concept drift. *Expert Syst. Appl.*, 2018.

[Anderson *et al.*, 2019] Robert Anderson, Yun Sing Koh, G Dobbie, and A Bifet. Recurring concept meta-learning for evolving data streams. *Expert Syst. Appl.*, 2019.

[Armstrong and Clifton, 2021] Jacob Armstrong and D Clifton. Continual learning of longitudinal health records. *IEEE EMBS (ITAB)*, 2021.

[Baena-Garcıa *et al.*, 2006] Manuel Baena-Garcıa, José del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method. In *IWKDDS*. Citeseer, 2006.

[Bahri *et al.*, 2021] Maroua Bahri, Albert Bifet, João Gama, Heitor M Gomes, and Silviu Maniu. Data stream analysis: Foundations, major tasks and tools. *WIRE: Data Min. Knowl. Discov.*, 2021.

[Bifet and Gavalda, 2007] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM (SDM)*. SIAM, 2007.

[Bifet and Gavalda, 2009] Albert Bifet and Ricard Gavalda. Adaptive learning from evolving data streams. In *IDA*. Springer, 2009.

[Bifet *et al.*, 2010] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 2010.

[Bifet *et al.*, 2015] Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In *21th ACM SIGKDD*, 2015.

[Bifet *et al.*, 2018] Albert Bifet, Ricard Gavaldà, G Holmes, and Bernhard Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT, 2018.

[Buzzega *et al.*, 2021] Pietro Buzzega, M Boschini, A Porrello, and S Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *ICPR*. IEEE, 2021.

[Chaudhry *et al.*, 2018] Arslan Chaudhry, P K Dokania, T Ajanthan, and P H S Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

[Chaudhry *et al.*, 2019] Arslan Chaudhry, Marcus Rohrbach, M Elhoseiny, T Ajanthan, P K Dokania, P H S Torr, and M Ranzato. On tiny episodic memories in continual learning. *arXiv:1902.10486*, 2019.

[Chen *et al.*, 2016] Kylie Chen, Yun S Koh, and P Riddle. Proactive drift detection: Predicting concept drifts in data streams using probabilistic networks. In *IJCNN*, 2016.

[Choudhary *et al.*, 2021] Ajay Choudhary, Preeti Jha, Aruna Tiwari, and Neha Bharill. A brief survey on concept drifted data stream regression. *SocProS*, 2021.

[Davalas *et al.*, 2022] Charalampos Davalas, Dimitrios Michail, Christos Diou, Iraklis Varlamis, and Konstantinos Tserpes. Computationally efficient rehearsal for online continual learning. In *ICIAP*. Springer, 2022.

[Gama *et al.*, 2004] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *SBIA*. Springer, 2004.

[Gama *et al.*, 2013] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Mach. Learn*, 2013.

[Gama *et al.*, 2014] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM (CSUR)*, 2014.

[Gao and Lei, 2017] Shuang Gao and Yalin Lei. A new approach for crude oil price prediction based on stream learning. *GSF*, 2017.

[Gomes *et al.*, 2017a] Heitor M Gomes, J P Barddal, F Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM (CSUR)*, 2017.

[Gomes *et al.*, 2017b] Heitor M Gomes, Albert Bifet, Jesse Read, J P Barddal, F Enembreck, Bernhard Pfahringer, G Holmes, and T Abdessalem. Adaptive random forests for evolving data stream classification. *ML*, 2017.

[Gomes *et al.*, 2018] Heitor M Gomes, J P Barddal, Luis Eduardo Boiko Ferreira, and Albert Bifet. Adaptive random forests for data stream regression. In *ESANN*, 2018.

[Gomes *et al.*, 2019] Heitor M Gomes, Jesse Read, and Albert Bifet. Streaming random patches for evolving data stream classification. In *IEEE (ICDM)*. IEEE, 2019.

[Gomes *et al.*, 2022] Heitor M Gomes, M Grzenda, Rodrigo Mello, Jesse Read, Minh Huong Le Nguyen, and Albert Bifet. A survey on semi-supervised learning for delayed partially labelled data streams. *ACM Comput. Surv.*, 2022.

[Grzenda *et al.*, 2020a] Maciej Grzenda, H M Gomes, and A Bifet. Performance measures for evolving predictions under delayed labelling classification. In *IJCNN*, 2020.

[Grzenda *et al.*, 2020b] Maciej Grzenda, Heitor M Gomes, and Albert Bifet. Delayed labelling evaluation for data streams. *Data Min. Knowl. Discov.*, 2020.

[Gunasekara *et al.*, 2022a] Nuwan Gunasekara, Heitor Gomes, Albert Bifet, and Bernhard Pfahringer. Adaptive neural networks for online domain incremental continual learning. In *DS 2022*. Springer, 2022.

[Gunasekara *et al.*, 2022b] Nuwan Gunasekara, Heitor Gomes, Albert Bifet, and Bernhard Pfahringer. Adaptive online domain incremental continual learning. In *ICANN 2022*. Springer, 2022.

[Gunasekara *et al.*, 2022c] Nuwan Gunasekara, Heitor Murilo Gomes, Bernhard Pfahringer, and Albert Bifet. Online hyperparameter optimization for streaming neural networks. In *IJCNN*, 2022.

[Hayes *et al.*, 2020] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *ECCV*. Springer, 2020.

[Hulten *et al.*, 2001] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *ACM SIGKDD*, 2001.

[Ikonomovska *et al.*, 2011] Elena Ikonomovska, Joao Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.*, 2011.

[Kara *et al.*, 2021] Ozgur Kara, N Churamani, and H Gunes. Towards fair affective robotics: Continual learning for mitigating bias in facial expression and action unit recognition. *arXiv:2103.09233*, 2021.

[Khamassi *et al.*, 2015] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. Self-adaptive windowing approach for handling complex concept drift. *Cogn. Comput.*, 2015.

[Khamassi *et al.*, 2018] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. Discussion and review on evolving data streams and concept drift adapting. *Evol. Syst.*, 2018.

[Kirkpatrick *et al.*, 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, G Desjardins, Andrei A Rusu, K Milan, J Quan, T Ramalho, Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

[Korycki and Krawczyk, 2021] Łukasz Korycki and Bartosz Krawczyk. Streaming decision trees for lifelong learning. In *ECML PKDD*. Springer, 2021.

[Kremer *et al.*, 2011] Hardy Kremer, Philipp Kranen, Timm Jansen, T Seidl, Albert Bifet, G Holmes, and Bernhard Pfahringer. An effective evaluation measure for clustering on evolving data streams. In *17th ACM SIGKDD*, 2011.

[Li and Hoiem, 2017] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[Li *et al.*, 2012] Peipei Li, Xindong Wu, and Xuegang Hu. Mining recurring concept drifts with limited labeled streaming data. *ACM (TIST)*, 2012.

[Lin, 2013] Dahua Lin. Online learning of nonparametric mixture models via sequential variational approximation. *NeurIPS*, 2013.

[Lobo *et al.*, 2020] Jesus L Lobo, Igor Ballesteros, Izaskun Oregi, Javier Del Ser, and Sancho Salcedo-Sanz. Stream learning in energy iot systems: a case study in combined cycle power plants. *Energies*, 2020.

[Mai *et al.*, 2022] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 2022.

[Maslov *et al.*, 2016] Alexandr Maslov, Mykola Pechenizkiy, Indreė Žliobaitė, and Tommi Kärkkäinen. Modelling recurrent events for improving online change detection. In *SIAM (SDM)*. SIAM, 2016.

[Montiel *et al.*, 2021] Jacob Montiel, Max Halford, S M Mastelini, G Bolmier, R Sourty, R Vaysse, A Zouitine, Heitor M Gomes, Jesse Read, T Abdessalem, et al. River: machine learning for streaming data in python. 2021.

[Page, 1954] Ewan S Page. Continuous inspection schemes. *Biometrika*, 1954.

[Prabhu *et al.*, 2020] Ameya Prabhu, P H S Torr, and P K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*. Springer, 2020.

[Ramírez-Gallego *et al.*, 2017] Sergio Ramírez-Gallego, B Krawczyk, Salvador García, Michał Woźniak, and F Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 2017.

[Rao *et al.*, 2019] Dushyant Rao, Francesco Visin, A Rusu, Razvan Pascanu, Yee Whye Teh, and R Hadsell. Continual unsupervised representation learning. *NeurIPS*, 2019.

[Souza *et al.*, 2020] Vinicius MA Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Min. Knowl. Discov.*, 2020.

[Srivastava *et al.*, 2021] Shikhar Srivastava, M Yaqub, K Nandakumar, Zongyuan Ge, and D Mahapatra. Continual domain incremental learning for chest x-ray classification in low-resource clinical settings. In *DART, FAIR*. Springer, 2021.

[Sun *et al.*, 2022] Yibin Sun, Bernhard Pfahringer, Heitor M Gomes, and A Bifet. Soknl: A novel way of integrating k-nearest neighbours with adaptive random forest regression for data streams. *Data Min. Knowl. Discov.*, 2022.

[Suárez-Cetrulo *et al.*, 2023] Andrés L. Suárez-Cetrulo, David Quintana, and Alejandro Cervantes. A survey on machine learning for recurring concept drifting data streams. *Expert Syst. Appl.*, 2023.

[Wald, 1947] Abraham Wald. Sequential analysis. 1947.

[Wang *et al.*, 2021] Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *CVPR*, 2021.

[Yang *et al.*, 2006] Ying Yang, X Wu, and X Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data Min. Knowl. Discov.*, 2006.

[Yoon *et al.*, 2017] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv:1708.01547*, 2017.

[Zhang *et al.*, 2022] Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Yunzhe Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. *NeurIPS*, 2022.

[Žliobaitė *et al.*, 2016] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, 2016.

[Zubaroğlu and Atalay, 2021] Alaettin Zubaroğlu and Volkan Atalay. Data stream clustering: a review. *AI Review*, 2021.