Neurocomputing 597 (2024) 127960

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Drifting explanations in continual learning

Andrea Cossu [a,*], Francesco Spinnato [b,c], Riccardo Guidotti [a,c], Davide Bacciu [a]

[a] University of Pisa, Largo B. Pontecorvo, 3, Pisa, 56127, Italy
[b] Scuola Normale Superiore, Piazza Cavalieri, 7, Pisa, 56127, Italy
[c] ISTI-CNR, Via Giuseppe Moruzzi, 1, Pisa, 56124, Italy

ARTICLE INFO

Keywords:
Continual learning
Explainable AI
Lifelong learning
Recurrent neural networks

ABSTRACT

Continual Learning (CL) trains models on streams of data, with the aim of learning new information without forgetting previous knowledge. However, many of these models lack interpretability, making it difficult to understand or explain how they make decisions. This lack of interpretability becomes even more challenging given the non-stationary nature of the data streams in CL. Furthermore, CL strategies aimed at mitigating forgetting directly impact the learned representations. We study the behavior of different explanation methods in CL and propose CLEX (ContinuaL EXplanations), an evaluation protocol to robustly assess the change of explanations in Class-Incremental scenarios, where forgetting is pronounced. We observed that models with similar predictive accuracy do not generate similar explanations. Replay-based strategies, well-known to be some of the most effective ones in class-incremental scenarios, are able to generate explanations that are aligned to the ones of a model trained offline. On the contrary, naive fine-tuning often results in degenerate explanations that drift from the ones of an offline model. Finally, we discovered that even replay strategies do not always operate at best when applied to fully-trained recurrent models. Instead, randomized recurrent models (leveraging on an untrained recurrent component) clearly reduce the drift of the explanations. This discrepancy between fully-trained and randomized recurrent models, previously known only in the context of their predictive continual performance, is more general, including also continual explanations.

1. Introduction

Continual Learning (CL) [1,2] studies the challenges of training models in dynamic environments, where the data distribution drifts over time. When trained sequentially on non-stationary data, neural networks have been shown to *forget* previous knowledge [3]. The problem of forgetting is attributed to the inability of neural networks (and of models in general) to maintain a proper stability–plasticity trade-off [4]. On the one hand, learning new information requires the network to be plastic and to adapt over time. On the other hand, preserving previous knowledge requires the network to be stable and to avoid large, unnecessary changes. Over the years, CL researchers have developed specific strategies to mitigate forgetting by finding a better stability–plasticity trade-off than simply fine-tuning a model continuously on an incoming stream of data [5]. However, a drop in the final predictive performance is not the only pitfall of a model that forgets. The hidden representations the model learns during continual training are subject to changes depending on how the model is trained. Existing CL strategies aimed at mitigating forgetting directly impact on *what* the model is computing and how. Consequently, eXplainable AI (XAI) [6] techniques that provide insights on the learned hidden representations can also be affected by forgetting and applying CL strategies. In fact, XAI techniques are computed for trained, *fixed* models. In a CL setup, some questions arise. Is an explanation affected by the fact that a model has been trained continuously? How will the explanations differ with respect to a model trained offline on the same set of data? How can we clearly show the effect of non-stationarity and CL in an explanation?

Our main contribution is to quantify the degree of change of an explanation with respect to a non-CL (offline) model. The alignment between these two explanations is useful to understand the impact of continual training on XAI techniques. In fact, it is not clear whether we can still trust the explanations of a CL model. To this extent, we propose CLEX (ContinuaL EXplanations) an evaluation protocol to measure the performance of XAI strategies in a CL environment. CLEX provides a qualitative and quantitative analysis of this alignment by studying how much the explanations of the CL model drift with respect to the ones of the offline model. We measure this drift through a novel metric, the Explanation Drift (ED), and study its role with respect to different data domains (images vs. sequences), model architectures (feedforward/convolutional vs. recurrent), and CL strategies in 3 different class-incremental scenarios [7] (where new classes are encountered

* Corresponding author.
E-mail address: andrea.cossu@di.unipi.it (A. Cossu).

https://doi.org/10.1016/j.neucom.2024.127960
Received 12 January 2024; Received in revised form 3 April 2024; Accepted 29 May 2024
Available online 3 June 2024

over time) that are known to induce forgetting. The explanations are computed with two different XAI techniques: SHapley Additive exPlanations (SHAP) [8] and DeepLIFT [9,10].

This work is an extended version of the conference paper [11].

In this paper, we provide an extended experimental evaluation and a more thorough analysis of our proposed metrics with respect to the analysis carried out in [11]. In particular, we add an XAI technique to highlight that our findings are not limited to the SHAP technique. Moreover, we also benchmark a recently published randomized recurrent model [12]. The advantages discussed in [11] for randomized neural networks are also observed for this additional model.

The main results of our experimental evaluation are threefold:

1. Models with similar final predictive accuracy are still subject to generating different explanations. This is quite intuitive, considering that different optimization trajectories leading to a similar predictive performance can easily lead to different learned representations, even outside CL. Therefore, explanations can also differ. We observed this phenomenon in all our experiments.
2. Different types of replay strategies effectively align explanations with the ones of a model trained offline on the same data set. This happens naturally, without engineering replay strategies for a specific explanatory purpose.
3. Previous works already showed that recurrent neural networks need particular care in a CL setting [13]. However, this was never studied in the context of XAI. We show that explanations in fully-trained recurrent models are not well aligned with the explanations of an offline model, *even when using replay*. We largely mitigated this effect by using randomized recurrent models, where the recurrent component is untrained. Interestingly, and in agreement with the first point, the difference in the explanation drift is not reflected in the final accuracy, which is comparable across both fully-trained and randomized recurrent models.

The rest of the paper is organized as follows: Section 2 introduces the main concepts and definitions necessary to understand our proposal. Section 3 describes the current state-of-the-art and the main works in the field of XAI for CL. Section 4 details our proposal, CLEX, which is benchmarked in Section 5. Finally, Section 6 presents our conclusions and discusses future work.

2. Background

Before delving deeper into our analysis, it is convenient to formally introduce the topics of CL, XAI, and recurrent neural networks. For recurrent neural networks, we mostly focus on *randomized recurrent models* since they are usually less popular than fully-trained recurrent models, like Long Short-Term Memory networks [14].

Continual learning. A data stream in CL is defined by an ordered sequence of experiences (or tasks) $\mathcal{S} = (e_1, e_2, \ldots)$, where each experience e_i introduces a dataset D_i [15]. The CL model is trained on each experience sequentially. In the case of supervised CL, each dataset provides a set of input–target pairs, $D_i = \{x_j, y_j\}_{j=1,\ldots,N}$. We consider classification problems where y is the target class associated with the input x. Following existing literature [5], we call *Naive* the approach of simply fine-tuning a model on the data stream one experience at a time, without leveraging any CL-specific approach.

One of the most studied CL scenarios, also adopted in this paper, is the *class-incremental* scenario [7], where each experience introduces a new set of classes, never seen before. Class-incremental is a scenario well known to induce a large amount of forgetting on previous experiences. Therefore, class-incremental has been adopted as one of the standard scenarios in which to assess the performance of CL strategies, especially the ones aimed at mitigating forgetting.

In many cases, to mitigate forgetting, replay strategies have emerged as the most effective ones [16–18]. These strategies leverage a memory buffer to store a subset of previous examples. During each training iteration, the model is updated with a sample taken from the memory buffer and the new examples coming from the stream. Experience Replay (ER) [19,20] is a replay strategy that keeps a fixed-size memory buffer. Usually, the buffer's content is balanced across classes or experiences [21] (e.g., the buffer contains the same number of examples per class, reduced as new classes are added to the buffer). Gradient-based Sample Selection (GSS) [22] is a replay strategy with a custom selection policy. The policy selects the examples that satisfy a set of constraints; namely, the loss on previously seen examples should not increase. Besides ER, in this work, we adopt the greedy version of the GSS algorithm, which trades off the exact solution for the constraints in favor of a computationally efficient implementation.

Although other CL strategies exist [2], many of them are ineffective in class-incremental scenarios in the absence of replay [23]. For example, regularization strategies like Elastic Weight Consolidation [24] or Synaptic Intelligence [25] are among the most popular ones. Unfortunately, they are not able to overcome the drift in the classifier [23], and they often need to resort to other CL approaches (like replay). Therefore, we do not consider regularization strategies in our evaluation, since their performance would be the same as without any CL strategy.

Explainable AI. Explainable AI encompasses a wide range of explainers [6]. In this work, we focus on model-specific methods tailored for neural networks. These methods usually exploit gradients or activations to assign an importance score to each input feature [26]. Additionally, these approaches are local; that is, they are designed to explain the predictions of a neural network for individual instances. Only a few model-specific local approaches applicable to neural networks can be used across MLPs, CNNs, and RNNs. The two most notable among these are GradientSHAP [8] and DeepLIFT [9].

GradientSHAP is based on SHAP [8], which computes the importance of each input feature by perturbing a given input instance x using a mask $z' \in \{0,1\}^K$ to decide which of the K feature values to keep or replace in x. The contribution of each feature to the model output is computed by observing how the model output changes depending on different perturbations. Formally, the SHAP value, ϕ_k, for the kth feature in a model's prediction is the average marginal contribution of feature k across all possible combinations of features:

$$\phi_k = \sum_{P \subseteq K \setminus \{k\}} \frac{|P|!(|K| - |P| - 1)!}{|K|!} \left[f_x(P \cup \{k\}) - f_x(P) \right], \tag{1}$$

where P is a subset of features excluding k, $f_x(P)$ is the model prediction with only the features in set P, and $f_x(P \cup \{k\})$ is the model prediction with the features in P and feature k. Given that computing exact Shapley values can be computationally infeasible for models with many features, SHAP usually approximates these values using a weighted linear regression on a sampled subset of features. In this way, a positive or negative SHAP value ϕ is assigned to each feature value, and the resulting explanation is represented as an additive feature attribution method. Formally:

$$g(z') = \phi_0 + \sum_{k=1}^{K} \phi_k z'_k. \tag{2}$$

In other words, given an input x, the explanation model g tries to linearly approximate the output of a model f in the local neighborhood obtained by perturbing x, i.e., $g(z') \approx f(x)$. The term ϕ_0 denotes the base value, i.e., the default prediction for an "empty" instance. In a classification setting, a SHAP value close to 0 indicates that the feature is almost irrelevant to the classification of a given instance. A positive value indicates a contribution toward a specific class, while negative values indicate a contribution toward the other classes. In general, SHAP is model-agnostic, and can be used to explain any black-box model, with the drawback of a high computational cost. GradientSHAP [8] addresses this challenge by approximating the SHAP values of an instance through the expected gradients over a distribution of baselines. These baselines are reference instances provided by the user, intended to represent a "neutral" instance and serve as a benchmark for evaluating the influence of each feature in the input. Specifically, the method introduces Gaussian random noise to the instance under examination [27], then selects a baseline and a random instance along the interpolation path between this baseline and the original input. This process is repeated multiple times, and GradientSHAP calculates the gradient of the model outputs with respect to these randomly chosen interpolations. This approach effectively approximates the impact of each feature on the prediction by averaging over these gradients.

Unlike methods that rely on gradients, DeepLIFT [9] compares the activation of each neuron to its "reference activation" and assigns contribution scores based on their difference. This reference activation is typically determined by a baseline input, often a neutral or average input for the model. The core idea of DeepLIFT is to apply a modified chain rule to decompose each input feature's contribution across all network layers. This rule is adapted to handle the non-linearities in neural networks. DeepLIFT propagates the contribution scores back through the network layers, distributing the output difference across the input features. In practice, it calculates how much each input feature contributes to the final prediction compared to its reference value. This method is particularly useful for its ability to handle situations where small changes in the input feature lead to significant changes in the output, a scenario where gradient-based methods might struggle [9]. Similarly to SHAP, the feature importance scores output by DeepLIFT can be either positive, negative, or zero, with the same semantics as SHAP.

Randomized recurrent neural networks. Sequential data processing tasks [28] deal with inputs structured in ordered sequences of items. Each sequence $x = (x_1, x_2, \dots, x_T)$ has T steps and a generic input at step t is the feature vector $x_t \in \mathbb{R}^I$. Sequence classification tasks require to predict a target scalar class y associated to each input sequence x (e.g., predicting the heart condition of a patient given the electrocardiogram). We will focus on sequence classification tasks as they are the most popular ones in CL [13,29].

In machine learning, sequential data processing is often performed by means of Recurrent Neural Network (RNN) models [30]. RNNs comprise a recurrent state-update equation that allows learning a fixed-size memory state of previous time steps. The output is computed from the hidden memory state by a linear projection. Our findings rely on two different types of RNNs: *fully-trained* RNNs and *randomized* RNNs. Fully-trained RNNs leverage adaptive weights matrices in the state-update equation that are learned by backpropagation through time. In this work, we use the Long Short-Term Memory (LSTM) network [14] as our fully-trained RNN.

Randomized RNNs carefully initialize the parameters of the state-update equation and learn only the linear output projection (called *readout*). We consider the Echo-State Network (ESN) [31] and the Random Oscillators Network (RON) [12] as our randomized RNNs.

The ESN is defined by a simple state-update equation:

$$h_{t+1} = \sigma(W h_t + W^I x_{t+1} + b), \tag{3}$$

where W_h, W^I are the hidden-to-hidden and input-to-hidden matrices, respectively, b is the bias, x_t is the input at time t and σ is the hyperbolic tangent activation function. The recurrent state h_t represents the memory of past time steps up to time step t. The matrix W is initialized uniformly in $[-1, 1]$, and its spectral radius is scaled to take on value ρ. Usually, $\rho < 1$ is a necessary condition for the Echo-State Property [32], which guarantees that the asymptotic behavior of an ESN does not depend on its initial conditions. The matrix W^I is initialized uniformly in $[-2, 2]$ and then scaled by the input scaling ν. The bias b is randomly sampled in $[-1, 1]$.

The RON network is composed of a randomly connected set of damped oscillators. The state-update equation reads:

$$y_{t+1} = y_t + \tau z_{t+1}, \tag{4}$$

$$z_{t+1} = z_t + \tau(\sigma(W y_t + W^I x_{t+1} + b) - \gamma y_t - \epsilon z_t), \tag{5}$$

where y_t is the hidden state at time t, z_t is a latent state used to compute y_t, τ is a time-step constant, γ and ϵ are the frequency and damping vector coefficients, respectively. The matrices W and W^I are initialized as in an ESN and kept fixed, as is the bias.

3. Related works

The studies at the intersection of XAI and CL are still in an early phase. Most of the time, existing works focus on how to enhance CL performance by leveraging XAI techniques. For example, Ebrahimi et al. [33] combined a regularization approach with a replay approach, where the replay buffer stores both previous examples and the explanation associated with each example. During training, the model is regularized via minimization of the L1 distance between the explanation stored in the memory buffer and the explanation computed by the current CL model. The approach can be implemented on top of any replay-based CL strategies, and the authors showed that it contributes to mitigating forgetting on image classification tasks. The main idea is that producing explanations that do not change much over time helps improving the CL model's stability. Note that this is different from our approach, since we study the stability with respect to the optimal model trained offline on the entire data stream (the usual model considered by XAI approaches). Obviously, the offline model is unavailable during CL and cannot be exploited to build CL strategies.

More recently, Rymarczyk et al. [34] introduced a regularization-based CL strategy that does not rely on previous examples. The proposed approach relies on many components, one of the most important being a modified distillation loss that includes a regularization component to promote the stability of currently and previously generated explanations. Therefore, although the method does not use previous examples for training, it still requires a separate external memory like the replay approaches. Moreover, the approach is defined on a specific family of architectures based on prototypes. It is not obvious how to generalize it to different types of networks. The evaluation shows improvements in both the final accuracy (mitigation of forgetting) and the interpretability of explanations (measured through the intersection over union across the same explanation in different tasks).

A different line of work focused on producing interpretable explanations directly in a CL setting. For example, Ye and Bors [35] designed a generative adversarial protocol that produces interpretable representations by maximizing the mutual information between the generator output and the latent variables provided as input to the generator itself. The authors show that the resulting CL model, applied to vision tasks, can produce interpretable representations that are guided by the latent variables. A similar study has been conducted for concept drift with Generative Adversarial Networks in [36]. There, the focus on concept drift is highlighted by introducing a new prequential metric that monitors the ability of the generative model to adapt to concept drifts. Moreover, the authors also leverage the same metric to study the relationship between change in the explanation and change in the generative capabilities of the CL model.

Overall, we believe many directions exist to study the interaction between XAI and CL. Our paper explores the under-studied question of whether or not learning continuously impacts existing XAI techniques, especially when combined with specific CL strategies. In particular, we found novel evidence that the explanations computed by different neural architectures (from feedforward to recurrent networks) behave very differently in a CL setting. As a result, blindly applying XAI methods can largely deteriorate the resulting explanations.

4. Continual explanations

In this section, we describe our proposal, CLEX, an evaluation protocol to measure the performance of XAI strategies in a CL environment. Given an instance, x, and a classifier, f, trained following the CL paradigm as described in Section 2, CLEX assesses if the explanation, S, for the prediction, $f(x)$, suffers from drift. In particular, we use this protocol to understand whether XAI techniques designed for offline, static models behave as expected in the presence of dynamic CL models. Intuitively, good CL models should preserve the explanation computed by an optimal offline model. If this does not happen, we may observe *forgetting* at the level of explanations, in addition to forgetting at the level of predictive performance. This can be formalized by defining a distance metric to measure the divergence between the explanation for a CL model and for a "baseline" model. Thus, CLEX comprises three components: a baseline, an explainer, and the evaluation metrics, presented in the following paragraphs.

Baseline. To detect an explanation drift, we must first define a baseline, i.e., a model that produces a "correct" explanation, which must be compared to the explanation obtained in a CL setting. The most natural solution to this problem is to set a "static" explanation as the baseline, given that we are sure it does not contain any artifacts that could be caused by continual training. For this reason, beside the CL model, f, trained continuously on a data stream, one experience at a time, our proposed evaluation protocol considers as a baseline another (offline) model, with the same architecture as f, trained from random initialization on the union of all experiences. Formally, the offline model is trained on $D = \cup_i D_i$. Both the CL model and the offline baseline are trained for multiple epochs until convergence.

Explainer. Once the baseline is trained, we select an explainer to produce an explanation in terms of feature importance for the instance x, both for the prediction of the baseline model, i.e., $J \in \mathbb{R}^{C\times K}$, and for the prediction of the CL model, $S \in \mathbb{R}^{C\times K}$, where C is the number of classes, and K is the number of features. We focus on model-specific, local, XAI methods for neural networks. Given that neural networks can be structured in various ways, for example, dense, convolutional, or recurrent modules, to ensure maximum compatibility, we need to select the most flexible XAI approaches, which can produce an explanation for any of these specific types of networks. Two of the most notable approaches that fit this purpose are GradientSHAP [8] and DeepLIFT [9]. In particular, we can use any of these approaches to explain the contribution of each of the K input features in x toward the model prediction. We compute the explanations for both models at the end of the training phase. For the CL model, this corresponds to the end of training on all the experiences in the stream. CLEX considers the explanation for the offline model, J, jointly trained on all the data, as the "ground truth" reference. In fact, this is what is usually computed by XAI techniques. In such cases, the model is not expected to learn new knowledge in the future. Clearly, the offline model does not suffer from forgetting since it does not learn continually.

Metrics. To evaluate the quality of a continual explanation, we introduce the *Explanation Drift (ED)* metric, which measures the distance between the continual and baseline explanations. For a given reference class $c \in C$, ED is defined as follows:

$$\mathrm{ED}^{(c)}(S,J) = \frac{1}{K}\left(\sum_{i=1}^{K}\max(0,S_i^{(c)}) - \sum_{i=1}^{K}\max(0,J_i^{(c)})\right)^2, \tag{6}$$

where $S^{(c)} \in \mathbb{R}^K$ and $J^{(c)} \in \mathbb{R}^K$ contain K values that are the contribution values toward class c for a given CL strategy and for the baseline, respectively. We compute ED separately for each target class we are interested in, since the explanation for a given class usually differs from the explanation for another class. We will drop the c superscript when not necessary. Note that, since we compute the contributions toward each possible class, we consider only *positive* contribution values, clamping any negative ones to zero. This is important since we are interested in whether features that were relevant to predict a given class in the offline model are still relevant for the same class in the CL model. Positive contribution values, such as those produced via GradientSHAP and DeepLIFT, denote exactly these kinds of features. This allows us to investigate how much each input feature contributes to driving the prediction toward the candidate class.[1] We can compute these values and, consequently, the ED metric at the end of training on each experience. However, our primary focus is on the ED metric for the classes present in the *first* experience. This is because the first experience represents the most challenging setup for our experiments, as it is the most susceptible to forgetting. In general, older experiences are much more prone to be forgotten than recent ones.

With the ED metric, we aim to assess the similarity in magnitude of the explanations, i.e., the extent to which the sum of the explanation deviates from that of the baseline. In computer vision experiments, the metric ED of Eq. (6) may fail to capture the spatial dependencies between pixels. The sum, in fact, compares contribution values independently of their position in an image. However, at the opposite end of the spectrum, comparing explanations pixel-wise often results in overly localized comparisons. For this reason, we address spatial locality by proposing a second metric, ED_{pool}, allowing for a more direct comparison of the explanations' local properties:

$$\mathrm{ED}^{(c)}_{\text{pool}}(S,J) = \frac{1}{K'}\sum_{i=1}^{K'}(\text{pool}(\max(0,S_i^{(c)}),p) - \text{pool}(\max(0,J_i^{(c)}),p))^2, \tag{7}$$

where "pool" represents a 2D average pooling operation with a kernel of $p\times p$ pixels, and K', in this case, is the number of pixels produced by the pooling operation. Contribution values are normalized to have zero mean and unitary variance. The normalization allows us to compare the distribution of the positive values instead of the scale (which is considered by the ED metric in Eq. (6)). Metric $\mathrm{ED}_{\text{pool}}$ includes spatial locality since it compares neighborhoods of $p \times p$ pixels from one strategy against the corresponding neighbor from Joint Training. Note that this metric is not limited to images, but can also be used with univariate or multivariate sequences, by applying instead a $p\times 1$ pooling on the time domain.

5. Experiments

In this section, we describe our experimental setup, present the results of the experiments, and finally discuss their implications.

5.1. Experimental setup

For our experiments,[2] we benchmark CLEX using the three most popular and widely-used CL benchmarks from computer vision and signal processing, i.e., Split MNIST [37], Split CIFAR-10 [25] and Synthethic Speech Commands (SSC) [13,38]. Split MNIST and Split CIFAR-10 have 5 experiences, with 2 classes in each experience, while SSC is composed of pre-processed audio sequences of 101 steps with 40 Mel features.

We chose image classification benchmarks as they are commonly used in CL [5] given that recurrent models can operate on computer vision datasets by taking each image one pixel at a time. However, image classification benchmarks do not have a true notion of time. For this reason, we also selected a CL benchmark for audio processing, where the time series is the pre-processed audio signal (SSC). In this way, we prevent our results with recurrent models from being negatively affected by a lack of temporal correlation in the input data. We

[1] Experiments considering all explanation values can be found in Appendix.

[2] The code to reproduce all experiments can be found at https://github.com/AndreaCossu/explainable-continual-learning.

Table 1
Final classification accuracy on the entire dataset at the end of training on all experiences for MNIST, CIFAR-10 and SSC. The XAI method (SHAP or LIFT) does not affect the predictive performance.

ACC	MNIST	CIFAR-10	SSC			
	MLP	CNN	LSTM	ESN	RON	CNN1D
Joint	0.97	0.87	0.98	0.97	0.99	0.99
Naive	0.20	0.19	0.18	0.18	0.19	0.18
ER	0.85	0.51	0.90	0.94	0.93	0.92
GSS	0.78	0.33	0.84	0.81	0.82	0.85

observed similar results independently from the specific benchmark, thus highlighting the generality of our conclusions.

Specifically, we used a ReLU feedforward network with 1 hidden layer on MNIST [37] and a Reduced ResNet18 [39] on CIFAR-10. For SSC, since we are dealing with sequences, we used a fully-trained LSTM and randomized ESN and RON models [40]. All recurrent networks use one layer. In addition, for SSC, we also considered a one-dimensional Convolutional Neural Network (CNN). The architecture comprises a 3×1 convolutional layer with 32 filters, followed by a max pooling with a 3×1 filter and a 25% dropout. The same structure is repeated in a second layer with 64 filters and without dropout. The output is then flattened and given as input to the final classifier. We provide the configuration for each model in Appendix.

We employ two Replay strategies: GSS [22] and Experience Replay (ER). Also, we use Naive fine-tuning to control for the case in which no CL strategy is used. We ensure that CL models achieve a performance (average accuracy on the test sets of all experiences after training on the last experience) that is on par with the expected CL performance for each strategy. We report the final accuracy in Table 1. Note that, as expected, Naive fine-tuning causes catastrophic forgetting while ER and GSS achieve a much better performance.

For GradientSHAP and DeepLIFT, we used 600 examples as background and 50 per class as a test, taken from the test set of e_1. We used $p = 4$ for the experiments with $\text{ED}^{c}_{\text{pool}}$. For the implementation of the CL algorithms, we used the Avalanche library [41]. To implement XAI techniques, we used the Captum library.[3] In particular, we adopted the GradientSHAP and DeepLift implementations.

5.2. Results

Qualitative analysis of drifting explanations. To provide a qualitative understanding of the explanation drift phenomenon, we plot the relevance of each input feature on MNIST when using as a reference all the 10 candidate classes (Fig. 1). After the model has been trained on the entire data stream, the explanations are computed for Naive, ER, and GSS. We show 6 examples coming from the first 2 classes encountered by the CL model (first experience). The Appendix also shows the same type of visualization for CIFAR-10 with SHAP and SSC with DeepLIFT (using the RON model). The plots show that *training continuously without any CL technique clearly harms the explanations* computed by both GradientSHAP and DeepLIFT. Fig. 1 shows that the Naive strategy assigns large SHAP values to the last two classes (last two columns in the image), while the second and third columns corresponding to the true classes are associated with low SHAP values. Basically, naive fine-tuning results in explanations that suffer from the recency bias: recently seen classes dominate the feature relevance computed by both SHAP and LIFT. This well-known phenomenon connected to forgetting affects the predictive performance in CL, where last-seen classes are classified more accurately than previously seen ones [5]. We show that

[3] The Captum library is available at https://captum.ai/.

Table 2
ED metric (lower is better) for GradientSHAP (SHAP) and DeepLIFT (LIFT) on the first experience after training on all experiences. c_0 represent class 0 and c_1 class 1. Best values for each class, dataset, and model highlighted in bold.

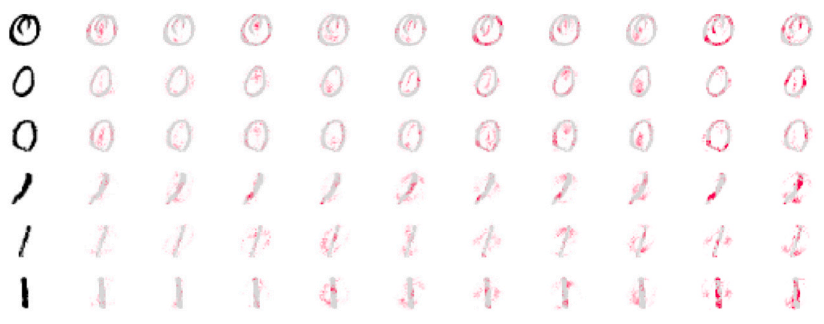
SHAP		MNIST	CIFAR	SSC			
ED		MLP	CNN	LSTM	ESN	RON	CNN1D
Naive	c_0	7.51	128.52	**0.38**	0.55	0.17	8.21
	c_1	2.47	63.51	0.67	0.46	0.18	**2.30**
ER	c_0	4.06	**0.01**	0.82	**3.2e−4**	0.02	**0.35**
	c_1	2.95	**0.20**	0.75	0.05	**9.1e−6**	3.73
GSS	c_0	**0.19**	6.65	1.47	0.05	**1.0e−3**	0.90
	c_1	**0.69**	19.53	**0.64**	**0.04**	0.03	10.47
LIFT							
Naive	c_0	1.55	45.91	0.35	0.12	**1.4e−3**	3.53
	c_1	0.44	163.02	0.24	0.12	0.03	0.90
ER	c_0	0.27	85.77	0.21	0.02	0.02	**8.0e−3**
	c_1	0.57	6.66	0.19	**1.3e−3**	**0.01**	0.81
GSS	c_0	**0.07**	**3.56**	**0.15**	**6.7e−3**	0.01	0.46
	c_1	**0.30**	**1.84**	**0.02**	0.03	**0.01**	**0.06**

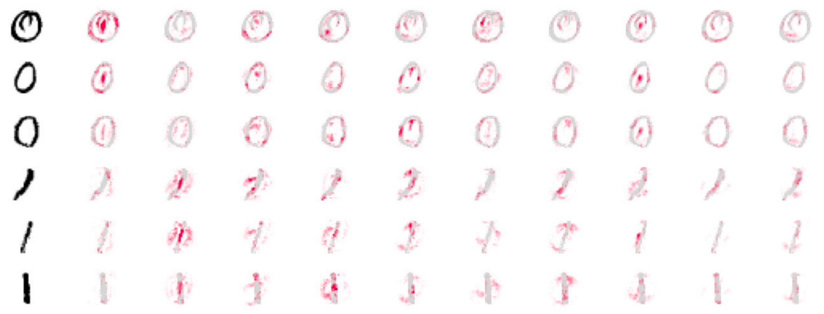
Table 3
ED_{pool} metric (lower is better) for SHAP and LIFT on the first experience after training on all experiences. Pooling is performed with a 4×4 kernel for MNIST and CIFAR and with a 1×4 kernel (on the time domain) for SSC. c_0 represent class 0 and c_1 class 1. Best values for each class, dataset, and model highlighted in bold.

SHAP		MNIST	CIFAR	SSC			
ED_{pool}		MLP	CNN	LSTM	ESN	RON	CNN1D
Naive	c_0	0.26	0.28	**0.52**	0.10	0.26	0.34
	c_1	0.31	0.36	0.52	**0.13**	0.26	0.40
ER	c_0	0.20	**0.26**	**0.52**	**0.08**	**0.24**	**0.11**
	c_1	**0.23**	**0.29**	**0.47**	0.15	**0.24**	**0.22**
GSS	c_0	**0.16**	0.36	0.58	0.14	0.24	0.21
	c_1	0.30	0.47	0.51	0.15	0.27	0.34
LIFT							
Naive	c_0	0.17	**0.22**	0.85	0.11	0.36	0.31
	c_1	0.24	0.27	0.74	0.31	0.22	0.33
ER	c_0	**0.08**	0.38	0.42	**0.10**	**0.14**	**0.13**
	c_1	0.10	**0.24**	0.68	**0.10**	**0.18**	**0.14**
GSS	c_0	0.09	0.26	**0.41**	0.16	0.20	0.17
	c_1	**0.09**	0.27	**0.58**	0.40	0.19	0.16

Table 4
ED_{pool} metric (lower is better) for SHAP and LIFT on the first experience after training on all experiences. Pooling is performed with a 8×8 kernel for MNIST and CIFAR and with a 1×8 kernel (on the time domain) for SSC. c_0 represent class 0 and c_1 class 1. Best values for each class, dataset, and model highlighted in bold.

SHAP		MNIST	CIFAR	SSC			
ED_{pool}		MLP	CNN	LSTM	ESN	RON	CNN1D
Naive	c_0	0.13	0.15	**0.27**	0.02	0.11	0.14
	c_1	0.20	0.22	0.29	0.03	0.10	0.18
ER	c_0	0.11	**0.14**	0.28	**0.01**	0.12	**0.05**
	c_1	**0.15**	**0.17**	**0.27**	**0.02**	**0.10**	**0.10**
GSS	c_0	**0.10**	0.18	0.28	0.03	0.11	0.09
	c_1	0.16	0.24	0.28	**0.02**	0.12	0.17
LIFT							
Naive	c_0	0.09	**0.11**	0.44	0.03	0.16	0.12
	c_1	0.09	0.17	0.38	**0.02**	0.08	0.13
ER	c_0	0.04	0.22	0.18	**0.02**	**0.05**	**0.05**
	c_1	**0.04**	**0.15**	0.35	0.03	0.06	**0.06**
GSS	c_0	**0.03**	0.13	**0.16**	0.03	0.08	0.07
	c_1	**0.04**	**0.15**	**0.29**	0.03	**0.05**	**0.06**


(a) Naive MLP - MNIST


(b) ER MLP - MNIST

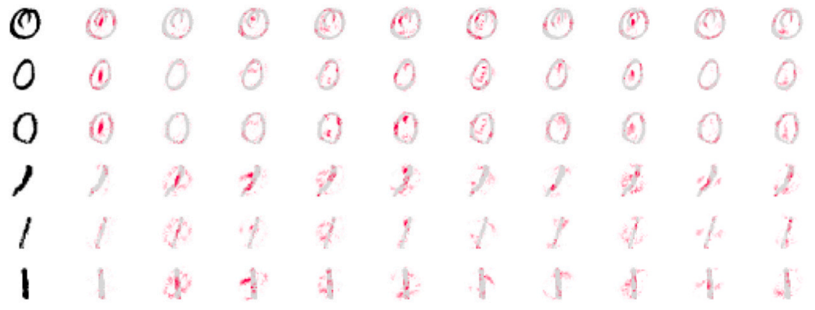

(c) GSS MLP - MNIST

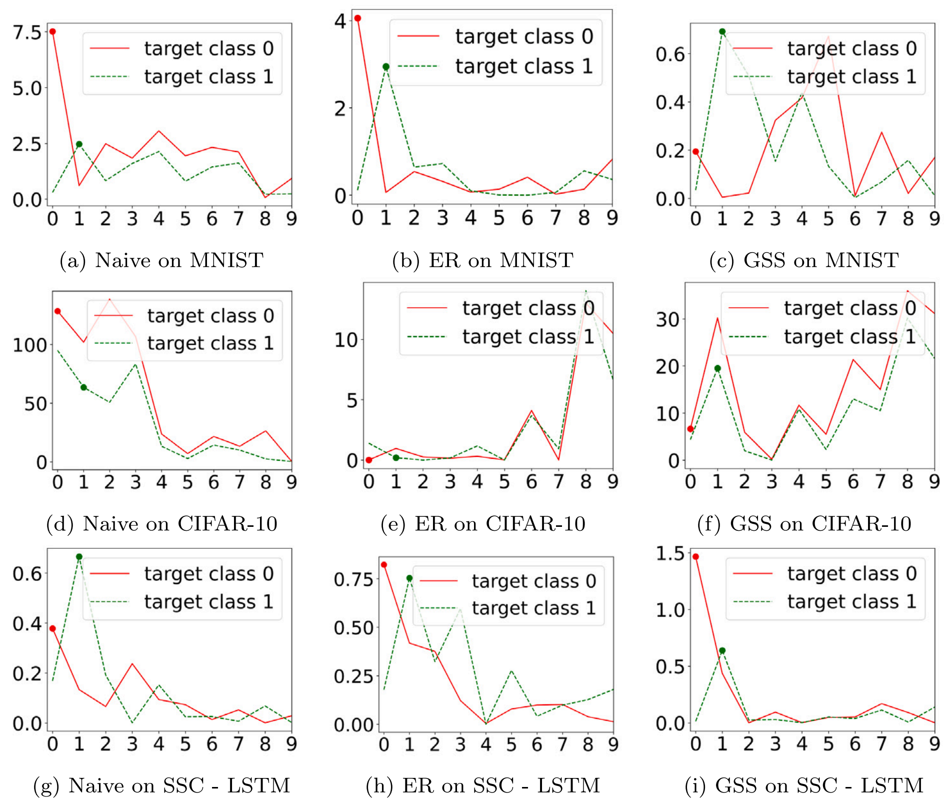
Fig. 1. Positive SHAP values after training on the last experience. The first column shows the input image, and the other ten columns show the SHAP values for each class (the more active, the higher the pixel's contribution toward the class). Replay and GSS are able to effectively preserve the SHAP values for class 0 and class 1, while Naive is mostly activated by class 8 and 9 (forgetting).

the same phenomenon occurs not only at the level of the predictions but also at the level of the explanations.

Interestingly, applying CL strategies like ER and GSS mitigates forgetting both at the level of predictive performance and at the level of the explanations. Fig. 1 shows that, overall, ER and GSS assign larger relevance values when computed with respect to the correct target class (second column for the first three examples and third column for the last three examples). However, this is only a qualitative analysis that prevents from understanding the degree to which forgetting is mitigated in the explanations. The evaluation protocol we propose, as well as the ED metric, is specifically designed to study this phenomenon quantitatively.

CL strategies reduce explanation drift. Tables 2–4 provide a detailed report on both ED and ED_{pool}, respectively, computed on the first experience and averaged over the first two classes (class 0 and 1). We report the value of both metrics separately for each class since different classes require different explanations, and it would not make sense to average them together. Each value is computed as the average metric on all the test examples used for that class. By looking at the results of Table 2, we can see that in most cases, the application of replay-based CL strategies results in a better ED. In the few cases where this does not hold (e.g., LSTM on SSC with SHAP, only for class 0), the difference between Naive and the best replay strategy is either very low or the best replay strategy takes on the same ED as Naive. This first result is quite intuitive and reflects the qualitative analysis of the visualization of the explanations presented at the beginning of this section. Still, it is useful to show quantitatively that an effective CL model behaves similarly to an offline model, not only in terms of predictive performance but also in terms of explanations. The same reasoning applies when using ED_{pool}. Table 3 shows results with a 4×4 pooling, while Table 4 shows results for a 8×8 pooling. We recall that the metric ED corresponds to a pooling equal to the size of the entire image/sequence. Our results are consistent with respect to all these metrics.

Models with similar predictive accuracy produce different explanations. Models that achieve a comparable predictive performance may also be expected to produce similar explanations since they build their predictions on relevant features that do not depend on the specific choice of the model. While this is usually true for XAI techniques applied to

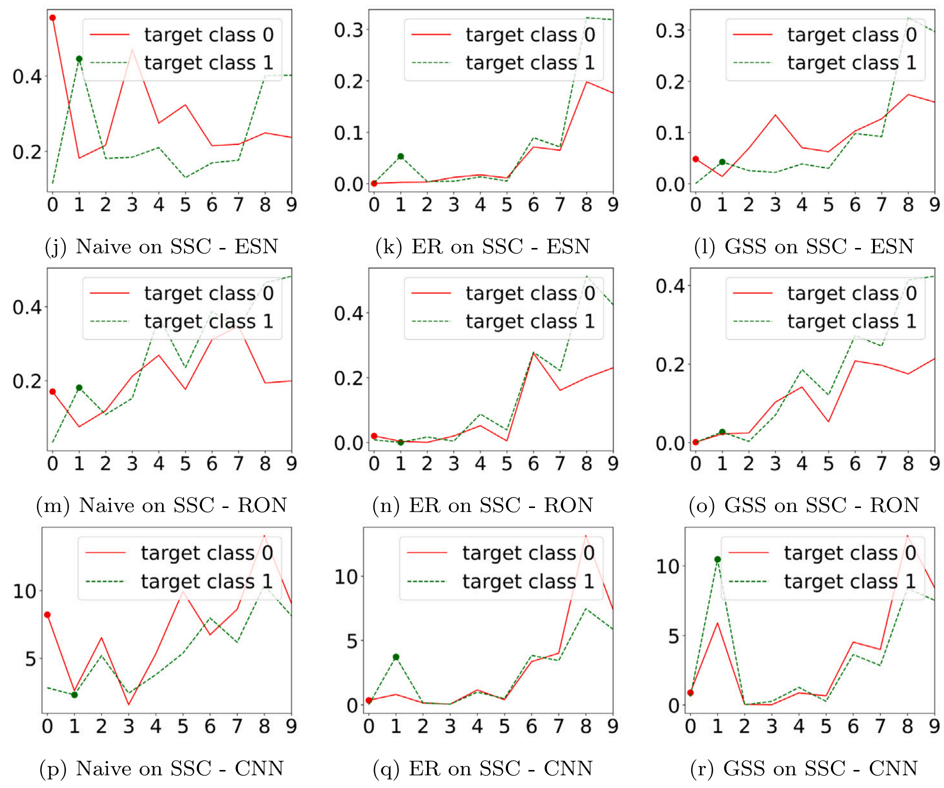


(a) Naive on MNIST (b) ER on MNIST (c) GSS on MNIST

(d) Naive on CIFAR-10 (e) ER on CIFAR-10 (f) GSS on CIFAR-10

(g) Naive on SSC - LSTM (h) ER on SSC - LSTM (i) GSS on SSC - LSTM

Fig. 2. ED metric for SHAP for each strategy and benchmark (lower is better). In each plot, the *x*-axis represents the number of classes in the dataset. The dots indicate the value of ED for the output unit corresponding to the target class. Each curve is averaged over 50 examples from the test set.

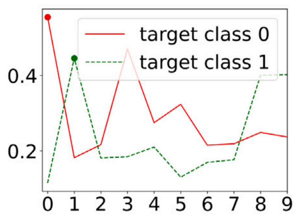


(j) Naive on SSC - ESN (k) ER on SSC - ESN (l) GSS on SSC - ESN

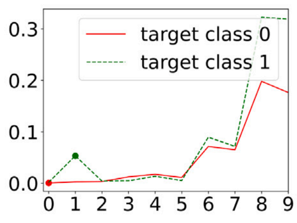
(m) Naive on SSC - RON (n) ER on SSC - RON (o) GSS on SSC - RON

(p) Naive on SSC - CNN (q) ER on SSC - CNN (r) GSS on SSC - CNN

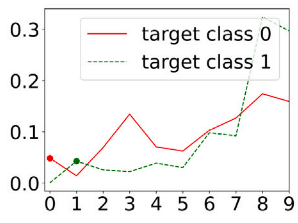
Fig. 2. (*continued*).



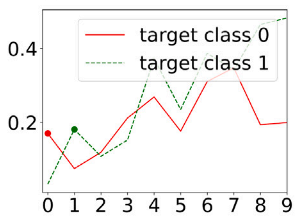
(j) Naive on SSC - ESN



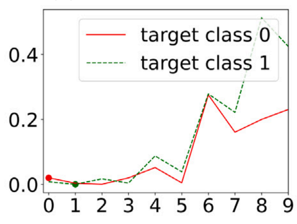
(k) ER on SSC - ESN



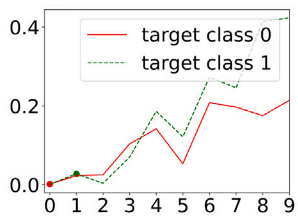
(l) GSS on SSC - ESN



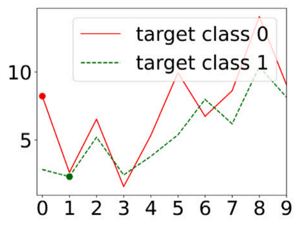
(m) Naive on SSC - RON



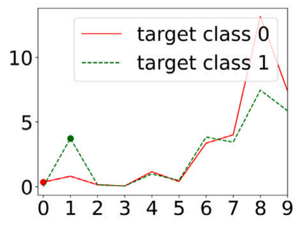
(n) ER on SSC - RON



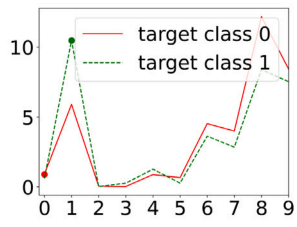
(o) GSS on SSC - RON



(p) Naive on SSC - CNN



(q) ER on SSC - CNN



(r) GSS on SSC - CNN

Fig. 2. (*continued*).

offline models, it does not hold in our experiments for CL models. In fact, in the SSC benchmarks, all models share a similar accuracy when using the same CL strategy (Table 1). The joint training performance is also very similar. However, the degree to which the models suffer from explanation drift differs greatly. This fact prevents from using accuracy as a proxy for the quality of the explanations. Correctly predicting the class for a given example does not entail that the prediction is based on a meaningful relationship or, conversely, that it is not based on spurious correlation. Ultimately, this means that evaluating and understanding the predictions made by a CL model requires looking at those explanations since there is no general conclusion that can be made by only looking at the model's predictive performance.

Fully-trained recurrent models are prone to explanation drift. The experiments on SSC highlight another interesting phenomenon. In all our experiments with both ED and ED_{pool}, fully-trained recurrent neural networks like LSTM show a worse performance with respect to randomized RNNs like ESN and RON when using replay strategies. This result contributes to an already existing line of work showing how fully-trained RNNs' behavior is often unexpected in CL. For example, Cossu et al. [13] showed that fully-trained RNNs are more prone to forgetting (i.e., losing accuracy on) previously seen classes than feedforward networks, even when using CL strategies. Moreover, Cossu et al. [40] showed that randomized RNNs like ESN are able to largely mitigate this effect. Our experiments in this paper stress the same point and extend it to the realm of XAI techniques. Interestingly, 1D CNNs are also often more effective than LSTM in reducing explanation drift, even though their performance is less consistent than randomized RNNs.

Explanation drift for all 10 candidate classes. One advantage of CLEX is that we can compare the explanation drift for any given class, not only for the true target class (the superscript c in the metrics definition of Eqs. (6) and (7)). All the points raised in this section can also be confirmed by looking at Fig. 2, which shows the value of ED for all 10 candidate classes (the true class is marked with a filled circle). In the Appendix, we provide the same type of plots also for DeepLIFT. We observed very distinct behaviors for classes different from the true target class, with an explanation drift that can either be much larger or smaller. This also supports the fact that models with similar accuracy cannot be automatically linked to similar explanations. The optimization trajectory followed by each model (hence, each local minimum) is relevant and affects how the model arrives at the final prediction.

6. Conclusion and future works

Dynamic environments present non-stationary data streams that are often challenging to learn with deep neural network models. With our proposal, CLEX, we showed that forgetting previous knowledge, a well-known phenomenon in deep neural networks, occurs not only at the level of predictive performance but also at the level of the explanations computed by XAI methods. In particular, we focused on the explanation drift between a CL model and a baseline offline model, jointly trained on the entire stream simultaneously. CL strategies that mitigate forgetting are also beneficial for the computed explanations, better aligning them with the ones computed by an offline model. We consistently observed this behavior for different data domains (images and sequences), with different models (feedforward, convolutional, and recurrent models), and for two XAI techniques (GradientSHAP and DeepLIFT). Replay-based strategies are effective in preserving the final accuracy of the CL models as well as reducing the drift in the computed explanations. Randomized recurrent models better exploit replay strategies, with a clear decrease in explanation drift with respect to fully-trained recurrent models.

Our experiments are a first step in understanding explanation drift and the challenge of interpreting a model's behavior in dynamic environments. GradientSHAP and DeepLIFT currently resort to some information about the neural network, like gradients and activations. Admittedly, other forms of explanations do not need to rely on such information, such as those obtained through intrinsically interpretable models or model-agnostic approaches [26]. However, model-specific approaches are usually much faster and thus can be used in real-world scenarios. Also, different kinds of model-specific approaches exist for other types of classifiers outside the family of artificial neural networks, for example, tree-based models such as Random Forests [42] and Boosted Trees [26]. Our empirical evaluation does not include such a family of models. The main reason is that, over the last decade, CL has mostly focused on deep neural network architectures [1]. Therefore, existing CL strategies mainly target this type of model. Future works can study the explanation drift phenomenon in environments similar to CL ones but using alternative models. For example, streaming learning (or online learning in non-stationary environments) [43,44] often employ statistical models or machine learning models like Random Forest. We see an opportunity to extend our empirical evaluation along these research lines, where the combination of XAI and CL models to obtain continual explanations is also promising.

Given the lack of existing ad-hoc solutions, there is space for future works to explicitly target explanation drift in CL models. Similar to what happens for the mitigation of forgetting [5], replay strategies look like one of the best-performing options also to improve the stability of explanations. However, all replay strategies optimize their memory buffer to preserve the model's predictive performance. A buffer designed to mitigate the explanation drift should probably include information about the explanation itself, by also taking into account that the model that computed the explanation is already changed. The combination of regularization strategies that mainly stabilize the feature extractor and replay strategies that mitigate forgetting at the level of the classifier might constitute a viable solution. The interplay between XAI and CL can also benefit CL itself, allowing to better understand (when possible) how the key relevance of input and latent features changes over time. Overall, to fully exploit the potential of CL, it is important to design a reliable pipeline that covers all steps of the model lifetime: from its design and training to its continuous monitoring.

CRediT authorship contribution statement

Andrea Cossu: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Francesco Spinnato:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Riccardo Guidotti:** Writing – review & editing, Validation, Supervision, Funding acquisition. **Davide Bacciu:** Writing – review & editing, Validation, Supervision, Resources, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andrea Cossu reports financial support was provided by European Union. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is publicly available.

Acknowledgments

Work supported by: EU EIC project EMERGE (Grant No. 101070918), EU NextGenerationEU programme under the funding schemes PNRR-PE-AI (PE00000013) FAIR - Future Artificial Intelligence Research, PNRR-SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics - Prot. IR0000013, H2020-INFRAIA-2019-1: Res. Infr. G.A. 871042 *SoBigData++*, G.A. 761758 *Humane AI*, G.A. 952215 *TAILOR*, ERC-2018-ADG G.A. 834756 *XAI*, and CHIST-ERA-19-XAI-010 SAI, and by the Green.Dat.AI Horizon Europe research and innovation programme, G.A. 101070416.

Table B.5
Mean and standard deviation over 3 random seeds for ED metric (lower is better) for GradientSHAP (SHAP) and DeepLIFT (LIFT) on the first experience after training on all experiences. c_0 represent class 0 and c_1 class 1. Best values for each class, dataset, and model highlighted in bold.

SHAP		MNIST	CIFAR	SSC	
ED		MLP	CNN	LSTM	ESN
Naive	c_0	9.32 ± 1.08	88.32 ± 10.1	0.41 ± 0.04	0.35 ± 0.1
	c_1	3.51 ± 0.98	75.42 ± 11.01	0.70 ± 0.08	0.53 ± 0.1
ER	c_0	4.2 ± 0.5	0.05 ± 0.02	0.8 ± 0.02	$2.1e{-3} \pm 8.5e{-4}$
	c_1	2.75 ± 0.8	0.30 ± 0.05	0.77 ± 0.03	0.02 ± 0.01
GSS	c_0	0.18 ± 0.02	6.33 ± 1.05	1.3 ± 0.5	0.1 ± 0.03
	c_1	0.52 ± 0.1	22.4 ± 5.7	0.67 ± 0.08	0.05 ± 0.01
LIFT					
Naive	c_0	1.22 ± 0.05	30.3 ± 9.01	0.35 ± 0.03	0.01 ± 0.03
	c_1	0.36 ± 0.03	80.4 ± 10.3	0.32 ± 0.06	0.08 ± 0.01
ER	c_0	0.20 ± 0.03	55.2 ± 9.4	0.23 ± 0.06	0.02 ± 0.001
	c_1	0.5 ± 0.08	5.3 ± 0.98	0.22 ± 0.02	$8.8e{-3} \pm 3.1e{-4}$
GSS	c_0	0.04 ± 0.003	3.55 ± 0.7	0.13 ± 0.01	$9.3e{-3} \pm 8.8e{-4}$
	c_1	0.4 ± 0.1	1.65 ± 0.13	0.09 ± 0.01	0.03 ± 0.01

Appendix A. Full experiments configuration

We report all model configurations and hyper-parameters that were used in the experiments presented in this paper. In the online repository, we provide the scripts to reproduce the experiments.

On MNIST, we used a feedforward ReLU network with 1 hidden layer and 256 units trained with SGD (learning rate 0.001 and momentum 0.9). The model is trained for 30 epochs during joint training (batch size 128) and for 10 epochs for each experience (batch size 64).

On CIFAR-10, the Reduced ResNet is trained with Adam (learning rate 0.001). The model is trained for 50 epochs during joint training (batch size 128) and for 30 epochs for each experience (batch size 64).

On SSC, all models have been trained with Adam. The LSTM has 1 hidden layer with 256 units, learning rate 0.001. The model is trained for 30 epochs during joint training (batch size 128) and for 10 epochs for each experience (batch size 64). The ESN has 1 layer with 2000 randomly initialized units, spectral radius 0.9 and input scaling 1. The training configuration is the same as the LSTM. The RON model has 1 layer with 500 randomly initialized units, spectral radius 0.99, input scaling 1, τ 0.1, $\gamma, \epsilon \in [0.5, 2]$. The joint learning rate is 0.001 and the learning rate for each experience is 0.001. The batch size is the same as the LSTM. Finally, the 1D CNN follows the LSTM training details.

The replay memory size is 300 examples for all experiments.

Appendix B. Statistical significance

In the experiments presented in the main text we never fixed the seed. Therefore, each experiment uses a different random seed. This avoids over-fitting on a given seed and shows that our results do not depend on the choice of a specific seed. To further test the robustness of our results, we ran experiments averaged over 3 random seeds. The explanations are therefore computed from a given trained model on different examples at each run (from the same test set, never seen during training). Table B.5 provides the average and standard deviations across these runs. The conclusions drawn in the main text still hold when assessed over multiple runs, showing their statistical significance.

Appendix C. Additional SHAP and LIFT plots

We report the same plot of Fig. 1 for MNIST also for CIFAR-10 (Fig. D.3) and SSC (Fig. D.4). Similar phenomena can be observed in

Table D.6
Element-wise squared difference between SHAP/LIFT from jointly trained model and CL model (lower is better). Only positive SHAP/LIFT values are considered. SHAP/LIFT is computed on the first experience after training on all experiences. c_0 represent class 0 and c_1 class 1. Best value for each class, dataset, and model highlighted in bold.

SHAP		MNIST	CIFAR	SSC			
eED+		MLP	CNN	LSTM	ESN	RON	CNN1D
Naive	c_0	2.3e−3	0.01	**1.6e−4**	3.5e−4	**6.8e−5**	6.7e−3
	c_1	**9.7e−4**	0.01	**5.9e−4**	5.9e−4	9.7e−5	**2.8e−3**
ER	c_0	4.3e−3	2.4e−3	1.7e−3	**3.2e−4**	9.3e−5	**4.7e−3**
	c_1	2.4e−3	3.5e−3	2.0e−3	4.8e−4	1.0e−4	4.3e−3
GSS	c_0	**2.0e−3**	**1.4e−3**	1.2e−3	4.1e−4	7.9e−5	6.9e−3
	c_1	1.7e−3	**2.0e−3**	1.0e−3	**4.7e−4**	**9.5e−5**	7.3e−3
LIFT							
Naive	c_0	**1.0e−3**	1.3e−2	6.5e−4	**2.7e−4**	7.7e−4	5.9e−3
	c_1	**6.1e−4**	2.2e−2	7.8e−3	3.4e−4	1.7e−4	**3.5e−3**
ER	c_0	1.1e−3	4.6e−2	2.3e−4	8.0e−4	**3.0e−4**	6.0e−3
	c_1	1.2e−3	5.1e−3	**4.7e−3**	**2.8e−4**	1.7e−4	5.3e−3
GSS	c_0	1.1e−3	**2.8e−3**	**1.9e−4**	6.7e−4	5.1e−4	**4.1e−3**
	c_1	1.1e−3	**2.5e−3**	4.8e−3	1.2e−3	**1.5e−4**	4.1e−3

Table D.7
Element-wise squared difference between SHAP/LIFT from jointly trained model and CL model (lower is better). Both positive and negative SHAP/LIFT values are considered. SHAP/LIFT is computed on the first experience after training on all experiences. c_0 represent class 0 and c_1 class 1. Best value for each class, dataset, and model highlighted in bold.

SHAP		MNIST	CIFAR	SSC			
eED		MLP	CNN	LSTM	ESN	RON	CNN1D
Naive	c_0	**3.0e−3**	2.8e−2	**3.2e−4**	**5.0e−4**	**1.9e−4**	1.2e−2
	c_1	**1.4e−3**	2.2e−2	**9.9e−4**	**7.3e−4**	2.3e−4	**6.0e−3**
ER	c_0	5.4e−3	6.0e−3	2.2e−3	8.5e−4	2.0e−4	**0.01**
	c_1	3.3e−3	8.8e−3	3.4e−3	7.9e−4	2.0e−4	0.01
GSS	c_0	3-1e−3	**3.5e−3**	2.4e−3	8.9e−4	**1.9e−4**	**0.01**
	c_1	2.4e−3	**4.2e−3**	1.8e−3	8.4e−4	**1.9e−4**	0.02
LIFT							
Naive	c_0	**2.8e−3**	0.03	1.5e−3	**7.8e−4**	1.9e−3	0.02
	c_1	**1.4e−3**	0.05	9.4e−3	1.4e−3	4.1e−4	**0.01**
ER	c_0	3.1e−3	0.10	**5.1e−4**	2.9e−3	**1.1e−3**	**0.01**
	c_1	2.8e−3	0.01	**5.4e−4**	**1.2e−3**	4.5e−4	0.02
GSS	c_0	3.0e−3	**7.0e−3**	1.0e−3	1.7e−3	1.6e−3	**0.01**
	c_1	2.6e−3	**6.2e−3**	6.9e−3	2.5e−3	**3.6e−4**	0.02

terms of explanation forgetting and recovery when using CL strategies like ER and SSC.

Fig. D.5 shows the same type of results as Fig. 2 but with LIFT instead of SHAP.

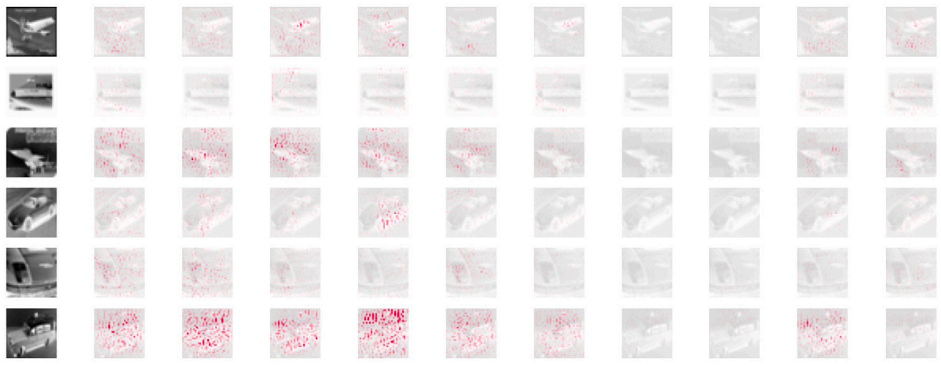
Appendix D. Additional SHAP and LIFT tables

We present two alternative versions of the ED and ED$_{\text{pool}}$ metric introduced in Section 4. We introduce these metrics to show that the choices we made in the design of the ED metric are based on reliable assumptions. In ED and its version with pooling, we either discard the spatial and temporal position of input feature relevance (ED) or include it for a local neighborhood of input features (ED$_{\text{pool}}$). We show that using a fine-grained spatial or temporal locality returns meaningless results. For example, comparing input relevance values on a pixel-by-pixel basis may (and, in fact, does) introduce a strong noise component that prevents the metric from capturing real patterns. To intuitively show this behavior, let us imagine an explanator (SHAP or LIFT) that takes the explanations computed by the joint model and swaps the relevance values for adjacent pixels (each pixel takes on the relevance of the pixel next/below/above it). This would result in a very large value for the element-wise metric, despite the fact that the overall explained pattern would not change significantly. Disregarding this effect (ED) or taking it into account for local regions (ED$_{\text{pool}}$) allows to really extract patterns at the level of a region (spatial domain) or of a consistent portion of a time series (time domain). The element-wise ED (eED+) is defined as:

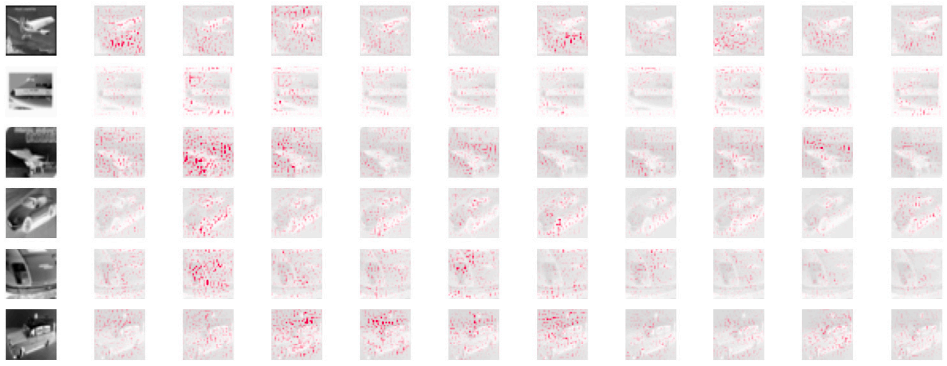
$$\text{ED}^{(c)}(S, J) = \frac{1}{K}\left(\sum_{i=1}^{K} \max(0, S_i^{(c)}) - \max(0, J_i^{(c)})\right)^2. \quad \text{(D.1)}$$

We can consider eED a version of ED$_{\text{pool}}$ with a 1×1 pooling.

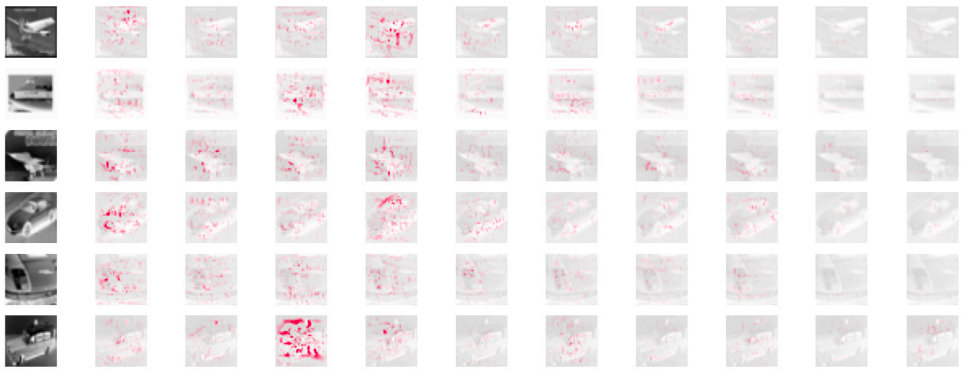
We can also define eED as eED+ with both positive and negative relevance values, i.e., dropping the *max* operator. We show in Tables D.6 and D.7 that both versions of eED do not capture the explanation forgetting effects discussed in Section 5.



(a) Naive CNN - CIFAR-10

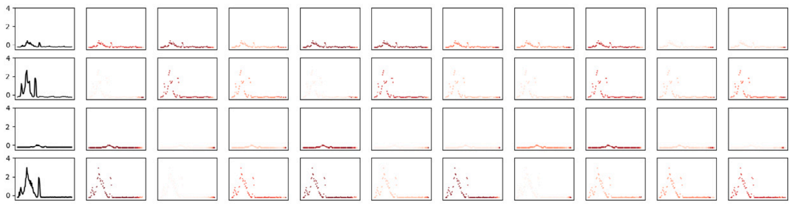


(b) ER CNN - CIFAR-10

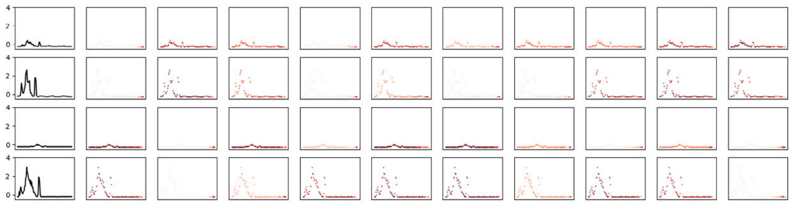


(c) GSS CNN - CIFAR-10

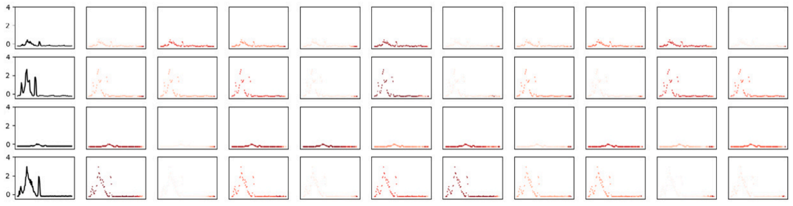
Fig. D.3. Positive SHAP values after training on the last experience the CNN model on CIFAR-10. The first column shows the input image (channels summed), and the other ten columns show the SHAP values for each class (the more active, the higher the time step's contribution toward the class). Replay and GSS are able to effectively preserve the SHAP values for class 0 and class 1, while Naive is mostly activated by class 8 and 9 (forgetting).



(a) Naive ESN - SSC

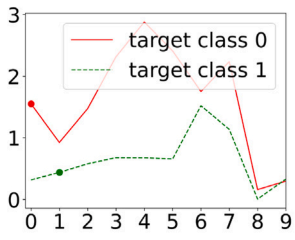


(b) ER ESN - SSC

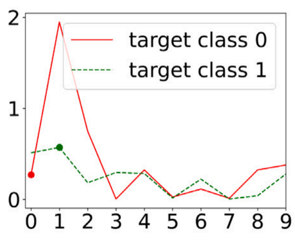


(c) GSS ESN - SSC

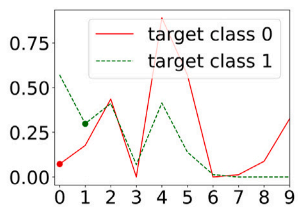
Fig. D.4. Positive SHAP values after training on the last experience the ESN model on SSC. The first column shows the input time series (time domain), and the other ten columns show the SHAP values for each class (the more active, the higher the time step's contribution toward the class). Replay and GSS are able to effectively preserve the LIFT values for class 0 and class 1, while Naive is mostly activated by class 8 and 9 (forgetting).



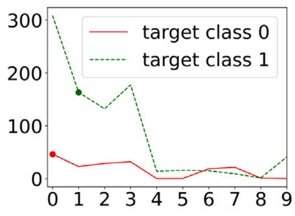
(a) Naive on MNIST



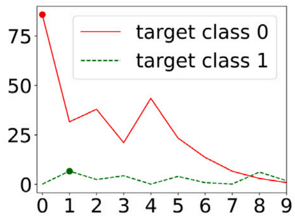
(b) ER on MNIST



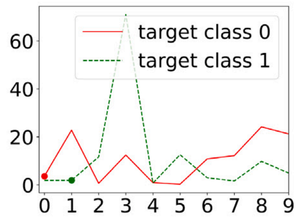
(c) GSS on MNIST



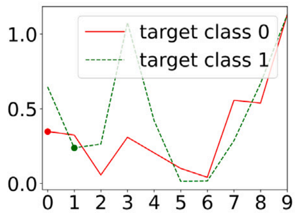
(d) Naive on CIFAR-10



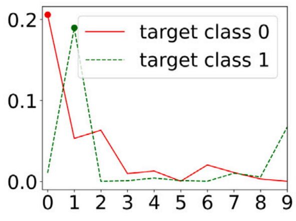
(e) ER on CIFAR-10



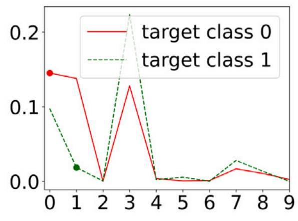
(f) GSS on CIFAR-10



(g) Naive on SSC - LSTM

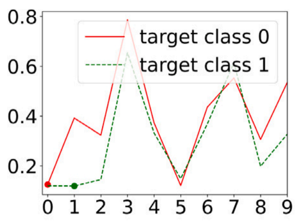


(h) ER on SSC - LSTM

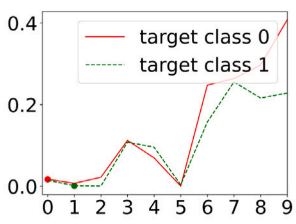


(i) GSS on SSC - LSTM

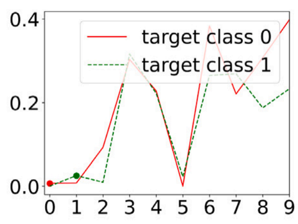
Fig. D.5. Metric ED for LIFT for each strategy and benchmark (lower is better). In each plot, the x-axis represents the number of classes in the dataset. The dots indicate the value of ED for the output unit corresponding to the target class. Each curve is averaged over 50 examples from the test set.



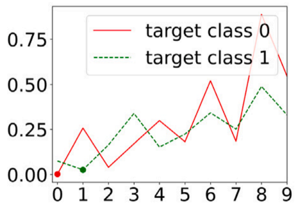
(j) Naive on SSC - ESN



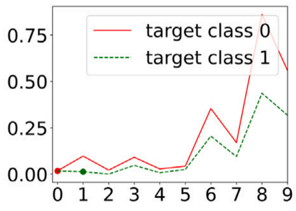
(k) ER on SSC - ESN



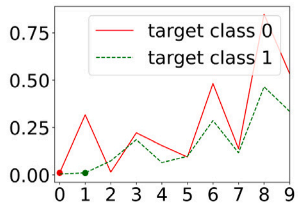
(l) GSS on SSC - ESN



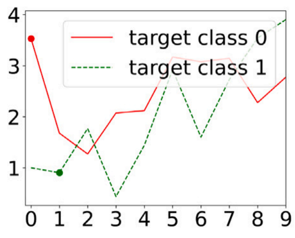
(m) Naive on SSC - RON



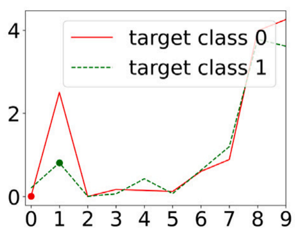
(n) ER on SSC - RON



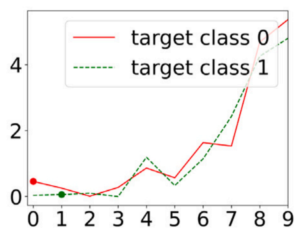
(o) GSS on SSC - RON



(p) Naive on SSC - CNN



(q) ER on SSC - CNN



(r) GSS on SSC - CNN

Fig. D.5. (*continued*).

References

[1] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N. Díaz-Rodríguez, Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, Inf. Fusion 58 (2020a) 52–68, http://dx.doi.org/10.1016/j.inffus.2019.12.004.

[2] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71, http://dx.doi.org/10.1016/j.neunet.2019.01.012.

[3] 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.

[4] 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, Erlbaum, Hillsdale, NJ, 1986, pp. 45–62.

[5] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A continual learning survey: Defying forgetting in classification tasks, IEEE Trans. Pattern Anal. Mach. Intell. (2021) 1, http://dx.doi.org/10.1109/TPAMI.2021.3057446.

[6] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A Survey of Methods for Explaining Black Box Models, CSUR, 2018.

[7] 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.

[8] S.M. Lundberg, S.I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).

[9] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 3145–3153.

[10] M. Ancona, E. Ceolini, C. Öztireli, M. Gross, Towards better understanding of gradient-based attribution methods for Deep Neural Networks, in: International Conference on Learning Representations, 2018.

[11] A. Cossu, F. Spinnato, R. Guidotti, D. Bacciu, A protocol for continual explanation of SHAP, in: European Symposium on Artificial Neural Networks, ESANN, 2023, http://dx.doi.org/10.14428/esann/2023.ES2023-41, arXiv:2306.07218.

[12] A. Ceni, A. Cossu, M. Stölzle, J. Liu, C. Della Santina, D. Bacciu, C. Gallicchio, Random oscillators network for time series processing, in: AISTATS 2024, 2024.

[13] A. Cossu, A. Carta, V. Lomonaco, D. Bacciu, Continual learning for recurrent neural networks: An empirical evaluation, Neural Netw. 143 (2021b) 607–627, http://dx.doi.org/10.1016/j.neunet.2021.07.021, URL https://www.sciencedirect.com/science/article/pii/S0893608021002847.

[14] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[15] V. Lomonaco, L. Pellegrini, A. Cossu, A. Carta, G. Graffieti, T.L. Hayes, M.De. Lange, M. Masana, J. Pomponi, M. van de Ven, Q. She, K. Cooper, J. Forest, E. Belouadah, S. Calderara, G.I. Parisi, F. Cuzzolin, A.S. Tolias, S. Scardapane, L. Antiga, S. Ahmad, A. Popescu, C. Kanan, T. van de Weijer, D. Bacciu, D. Maltoni, Avalanche: An end-to-end library for continual learning, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2021, pp. 3595–3605, http://dx.doi.org/10.1109/CVPRW53098.2021.00399.

[16] A. Robins, Catastrophic forgetting; Catastrophic interference; stability; plasticity; rehearsal, Connect. Sci. 7 (1995) 123–146, http://dx.doi.org/10.1080/09540099550039318.

[17] 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 Comput. 33 (2021) 2908–2950, http://dx.doi.org/10.1162/neco_a_01433.

[18] 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: ICCV Visual Continual Learning Workshop, 2023.

[19] D. Rolnick, A. Ahuja, J. Schwarz, T.P. Lillicrap, G. Wayne, Experience replay for continual learning, in: NeurIPS, 2019, pp. 350–360.

[20] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P.K. Dokania, P.H.S. Torr, M. Ranzato, On tiny episodic memories in continual learning, arXiv (2019).

[21] G. Merlin, V. Lomonaco, A. Cossu, A. Carta, D. Bacciu, Practical recommendations for replay-based continual learning methods, in: Workshop on Novel Benchmarks and Approaches for Real-World Continual Learning (CL4REAL), 2021, arXiv:2203.10317.

[22] R. Aljundi, M. Lin, B. Goujaud, Y. Bengio, Gradient based sample selection for online continual learning, in: NeurIPS, Curran Associates, Inc, 2019, pp. 11816–11825, URL http://papers.nips.cc/paper/9354-gradient-based-sample-selection-for-online-continual-learning.pdf.

[23] T. Lesort, A. Stoian, D. Filliat, Regularization shortcomings for continual learning, arXiv (2020b).

[24] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, D. Grabska-Barwinska, C. Clopath, D. Kumaran, R. Hadsell, Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. USA 114 (2017) 3521–3526, http://dx.doi.org/10.1073/pnas.1611835114.

[25] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in: International Conference on Machine Learning, 2017, pp. 3987–3995.

[26] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, S. Rinzivillo, Benchmarking and survey of explanation methods for black box models, Data Min. Knowl. Discov. (2023) 1–60.

[27] D. Smilkov, N. Thorat, B. Kim, F. Viégas, M. Wattenberg, Smoothgrad: removing noise by adding noise, 2017, arXiv preprint arXiv:1706.03825.

[28] T.G. Dietterich, Machine learning for sequential data: A review, in: T. Caelli, A. Amin, R.P.W. Duin, D. de Ridder, M. Kamel (Eds.), Structural, Syntactic, and Statistical Pattern Recognition, Springer, Berlin, Heidelberg, 2002, pp. 15–30, http://dx.doi.org/10.1007/3-540-70659-3_2.

[29] A. Cossu, A. Carta, D. Bacciu, Continual learning with gated incremental memories for sequential data processing, in: 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–8, http://dx.doi.org/10.1109/IJCNN48605.2020.9207550.

[30] J.L. Elman, Finding structure in time, Cogn. Sci. 14 (1990) 179–211, http://dx.doi.org/10.1207/s15516709cog1402_1.

[31] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comp. Sci. Rev. 3 (2009) 127–149, http://dx.doi.org/10.1016/j.cosrev.2009.03.005.

[32] M. Lukoševičius, A practical guide to applying echo state networks, in: G. Montavon, G.B. Orr, K.R. Müller (Eds.), Neural Networks: Tricks of the Trade, second ed., in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 659–686, http://dx.doi.org/10.1007/978-3-642-35289-8_36.

[33] S. Ebrahimi, S. Petryk, A. Gokul, W. Gan, J.E. Gonzalez, M. Rohrbach, T. Darrell, Remembering for the right reasons: Explanations reduce catastrophic forgetting, Appl. AI Lett. 2 (2021) e44, http://dx.doi.org/10.1002/ail2.44, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ail2.44.

[34] D. Rymarczyk, J. van de Weijer, B. Zielinski, B. Twardowski, ICICLE: Interpretable class incremental continual learning, arxiv (2023) http://dx.doi.org/10.48550/arXiv.2303.07811, URL http://arxiv.org/abs/2303.07811 arXiv:2303.07811.

[35] F. Ye, A.G. Bors, Lifelong learning of interpretable image representations, in: 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA), 2020, pp. 1–6, http://dx.doi.org/10.1109/IPTA50016.2020.9286663.

[36] F. Guzy, M. Woźniak, B. Krawczyk, Evaluating and explaining generative adversarial networks for continual learning under concept drift, in: 2021 International Conference on Data Mining Workshops (ICDMW), 2021, pp. 295–303, http://dx.doi.org/10.1109/ICDMW53433.2021.00044.

[37] G.M. van.de Ven, A.S. Tolias, Three scenarios for continual learning, in: Continual Learning Workshop NeurIPS, 2018, arXiv:1904.07734.

[38] B. Cramer, Y. Stradmann, J. Schemmel, F. Zenke, The heidelberg spiking data sets for the systematic evaluation of spiking neural networks, IEEE Trans. Neural Netw. Learn. Syst. 33 (2020) 2744–2757.

[39] D. Lopez-Paz, M.A. Ranzato, Gradient episodic memory for continual learning, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017.

[40] A. Cossu, D. Bacciu, A. Carta, C. Gallicchio, V. Lomonaco, Continual learning with echo state networks, in: ESANN 2021, 2021, http://dx.doi.org/10.14428/esann/2021.ES2021-80, URL http://arxiv.org/abs/2105.07674.

[41] A. Carta, L. Pellegrini, A. Cossu, H. Hemati, V. Lomonaco, Avalanche: A PyTorch library for deep continual learning, arxiv (2023) http://dx.doi.org/10.48550/arXiv.2302.01766, URL http://arxiv.org/abs/2302.01766 arXiv:2302.01766.

[42] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[43] D. Dell'Aglio, E. Della Valle, F. van Harmelen, A. Bernstein, Stream reasoning: A survey and outlook, Data Sci. 1 (2017) 59–83.

[44] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: A survey, IEEE Comput. Intell. Mag. 10 (2015) 12–25, http://dx.doi.org/10.1109/MCI.2015.2471196.

Andrea Cossu has a Ph.D. in Data Science from Scuola Normale Superiore, Pisa. He is currently a researcher at the Computer Science Department of the University of Pisa. His research interests revolve around continual/lifelong learning and recurrent neural networks. Andrea graduated with honors in Computer Science from the University of Pisa.

Francesco Spinnato is a Ph.D. candidate in Data Science at the Scuola Normale Superiore, and a researcher at the University of Pisa. His research focuses on Explainable AI (XAI) for sequential data, particularly on interpreting black-box models for univariate and multivariate time series. In 2017, he earned a bachelor's degree in Economics and Management from the University of Padua, and in 2020, he obtained a master's degree in Data Science from the University of Pisa.

Riccardo Guidotti is an Assistant Professor at the Department of Computer Science at the University of Pisa and a member of the Knowledge Discovery and Data Mining Laboratory (KDDLab), a joint research group with ISTI-CNR of Pisa and with Scuola Normale Superiore. Riccardo Guidotti graduated with honors in Computer Science from the University of Pisa, where he also earned his Ph.D. in Computer Science with a thesis on "Personal Data Analytics". He won the IBM fellowship program in 2015, the DSAA New Generation Data Scientist Award in 2018, and the Marco Somalvico Award for Artificial Intelligence in 2021. He has been the principal investigator of the FIS 2021 MIMOSA project. His research interests include explainable artificial intelligence, interpretable machine learning, personal data mining, clustering, and the analysis of transactional data and time series.

Davide Bacciu has a Ph.D. in Computer Science and Engineering from IMT Lucca. He is Full Professor at the Computer Science Department, University of Pisa, where he heads the Pervasive AI Lab. His research interests include machine learning for structured data, Bayesian learning, deep learning, reservoir computing, distributed and embedded learning systems. He has been the coordinator of the HE-EIC EMERGE and H2020 TEACHING projects. He is the Chair of the IEEE Neural Network Technical Committee and a Senior Editor of the IEEE Transactions on Neural Networks and Learning Systems.