Noname manuscript No.
(will be inserted by the editor)

Continually Trained Life-Long Classification

Rudolf Szadkowski · Jan Drchal · Jan Faigl

Received: date / Accepted: date

Abstract Two challenges can be found in a life-long classifier that learns continually: the concept drift, when the probability distribution of data is changing in time, and catastrophic forgetting when the earlier learned knowledge is lost. There are many proposed solutions to each challenge, but very little research is done to solve both challenges simultaneously. We show that both the concept drift and catastrophic forgetting are closely related in our proposed description of the life-long continual classification. We describe the process of continual learning as a wrap modification, where a wrap is a manifold that can be trained to cover or uncover a given set of samples. The notion of wraps and their cover/uncover modifiers are theoretical building blocks of a novel general life-long learning scheme, implemented as an ensemble of variational autoencoders. The proposed algorithm is examined on evaluation scenarios for continual learning and compared to state-of-the-art algorithms demonstrating the robustness to catastrophic forgetting and adaptability to concept drift but also showing the new challenges of the life-long classification.

Keywords continual learning, life-long learning, autoencoder, catastrophic forgetting, concept drift

1 Introduction

Continual learning is essential for domains where the incoming data must be continually integrated into a classifier. Such a classifier is expected to train and predict the incoming data as long as it is in operation; hence, it is a life-long classifier. On a life-long time scale, we expect that the probabilistic distribution of incoming data can change in time, where such a change is called concept drift. Moreover, there is also a problem of catastrophic forgetting: as the classifier is continually trained, it can forget some knowledge it learned earlier. Both the concept drift and catastrophic forgetting are the main challenges of continual learning [7].

In neural networks, concept representations are distributed throughout the network weights [8]. In such distributed representations, during each training iteration, a slight change of a single parameter can lead to changes in multiple concepts at once. During continual learning, these changes can accumulate, and concepts might get catastrophically forgotten [5]. There are four approaches to preventing catastrophic forgetting [19]:

- weight regularization to prevent overfitting of the current task [13] ;
- storing selected samples that are used for rehearsal (incorporating the old samples into training dataset) [30, 27];
- learning a generative model that produces samples used for rehearsal [29, 26];
- designing a network architecture that (partially) isolates concepts in subnetworks [21, 9].

A straightforward implementation of the architecture design approach is an ensemble of predictors [15], which we combine with Variational AutoEncoder (VAE) [12] used as a generative model.

Our proposed method is based on an ensemble of classifying and generating autoencoders. An autoencoder is a neural network trained to approximate an identity transformation of the input. The autoencoder training is unsupervised because the loss, also called the reconstruction error,

This work was supported by the Czech Science Foundation (GAČR) under research project No. 18-18858S.

Department of Computer Science, Faculty of Electrical Engineering
Czech Technical University in Prague
Technick 2, 166 27, Prague 6, Czech Republic
E-mail: {szadkrud,drchajan,faiglj}@fel.cvut.cz,
WWW home page: `https://comrob.fel.cvut.cz/`

is defined as a difference between the input and its transformed image. This reconstruction error of the autoencoder has an alternative to anomaly detection. An anomalous input is detected by the autoencoder when the reconstruction error exceeds a given threshold [3,2]. A related application of the same method is the novel class detection [20,23], where the task is to detect unknown classes. In both applications, the anomaly and the novel class detection, the autoencoder divides the input space into two regions according to the reconstruction error. Here, we call the region with the error below a certain threshold: a wrap.

In this paper, we take advantage of the wrap interpretation of the autoencoder and present a continually supervised training algorithm that suppresses catastrophic forgetting and adapts the classifier to concept drift. The classifier is implemented as an autoencoder ensemble, where each autoencoder is trained to cover its respective class by a wrap.

We present two mechanisms modifying the wrap: cover, and uncover, where the former includes given samples, and the latter excludes given samples from the wrap. Both methods modify the wrap while suppressing catastrophic forgetting.

The wrap and its two modifiers, cover and uncover, are the building blocks for the general life-long learning algorithm: For each given sample batch, the algorithm uncovers given samples by wraps to which the samples do not belong. Then it covers the samples by their corresponding wrap.

The robustness of the proposed algorithm is intuitively demonstrated with four basic scenarios of incremental learning in a three-dimensional domain. Besides, the scalability to higher dimensional domains is evaluated using scenarios constructed from the MNIST and CIFAR10 datasets.

The paper is structured as follows. In the following Section 2, a brief introduction to the related work on continual learning is provided. The herein addressed continual learning is formulated in Section 3, where we present an analysis of the challenges based on which we propose a general life-long classifier. In Section 4, the classifier is implemented as an ensemble of generating VAEs. The results are reported and discussed in Sections 5 and 6, respectively. The paper is concluded in Section 7.

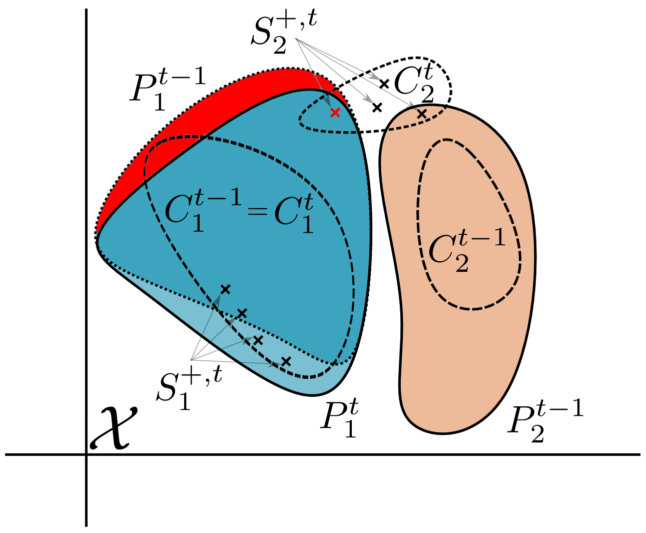


Fig. 1 Illustration of wraps P_1 and P_2 during iterations $t-1$ and t, and their relation to classes C_1 and C_2 (dashed borders) that provide the class-samples $S_1^{+,t}$ and $S_2^{+,t}$ (marks), respectively. Even though the samples $S_1^{+,t}$ do not represent the class C_1, the corresponding wrap P_1 must retain its shape. However, between the iteration $t-1$ (dotted border) and t (solid border), the wrap loses its part (red region) due to catastrophic forgetting. The classes can change in time due to the concept drift, like the class C_2 in the picture, and such drift can be detected only through new class-sample distribution. Notice that without any assumption about the probabilistic distribution of samples, the only way to detect the concept drift is by having a class-sample that is a part of the wrap of another class (on top of P_1 in the red).

2 Related Work

The continual learning is a paradigm where the predictor trains on a stream of labeled samples. The training on stream is constrained by limited storage. Thus, it is impossible to save all labeled samples; rather, the predictor must incrementally aggregate the knowledge. In a non-static environment, which produces a stream of labeled samples, the aggregated knowledge can become obsolete as the concept, described by the knowledge, changes. Two challenges arise as we assume a non-static environment and limited model size: the concept drift and catastrophic forgetting.

In continual connectionist learners, catastrophic forgetting stems from the strength of the connectionist models: distributed representation of concepts [5]. The concepts are represented by a large number of weights, which are shared with other concept representations. Therefore, updating part of the network influences overall performance [8]. There are four approaches countering catastrophic forgetting: regularization, rehearsal, generative replay, and network architecture [19].

In the regularization approach, the training is regularized to prevent the overfitting of the currently processed batch of data. Such as the Elastic Weight Consolidation (EWC) [13], where the algorithm selectively freezes the neural network weights that are important to the previously learned task/class.

In the rehearsal approach, the predictor stores a limited amount of samples reintroduced into the next training. In the algorithm presented in [27], the limited memory clusters and stores selected samples, which are then used for rehearsal.

The generative replay (also called pseudo-rehearsal) is similar to the rehearsal approach, where instead of saving the samples, the predictor trains a generator to generate the samples. In [29], the authors use the Generative Adversarial Network (GAN) as the generator, where the GAN generates samples that are mixed with the input samples. Thus the predictor is trained on current and reconstructed samples. Similarly, the VAE is the generator network used for memory replay in [1].

The generative replay is often combined with the architecture approach, where the architecture of the network is designed to minimize the interaction between concept representations during training. The FearNet [9] has a fast-slow memory system, where the short-term memory learns to replay recent experience while the long-term memory aggregates the short-term memory samples. Our presented classifier is also a hybrid of generative replay and dynamic architecture approaches; however, unlike the algorithms mentioned above, we also consider the concept drift.

By considering the concept drift, we assume the existence of concepts, a hidden part of the environment, which provides us samples. The concepts can be described with a joint distribution $p(X, y)$ between input X (samples) and output y (labels) variables. The concept drift is then defined as a change of the joint distribution in time: $p_{T_1}(X, y) \neq p_{T_2}(X, y)$, where T_1 and T_2 are consequent iterations.

The authors of [6] distinguish two types of the concept drift: (i) the virtual concept drift, where the distribution of the input data $p(X)$ changes; and (ii) the real concept drift, where the posterior class probability $p(y|X)$ changes. The learning that reacts to the concept drift and integrates the change into the learned model is called adaptive learning, which can be understood as a special case of continual learning.

The key feature of adaptive learning is selective forgetting. A voting ensemble of classifiers is proposed in [4] as an adaptive learning algorithm, where each classifier can be learned on a different task and weighted in the case of the concept drift. The authors of [24] propose a rehearsal-based classifier that uses a window of variable length that slides on the data stream and provides the training data. When the concept drift is detected, the window is shortened, and thus the model forgets the past data. More complex sample weighting is presented in [16], where the authors weight the stored data by its relevancy. The window forgetting unlearns the concepts basing on how old the concept is; however, such a strategy can forget old concepts that are still consistent with new data. Our algorithm unlearns the old concepts that are in a conflict with the current concept.

In contrast to the probabilistic description of the concept, the authors of [22] present a clustering algorithm that assumes that the clustered samples are points lying on an unknown manifold living within the feature space. In this work, we describe concepts (classes) as disjoint manifolds in the feature space. The definition of classes as manifolds is our basis for the formulation of continual learning presented in the following section.

3 Analysis

Let $\mathcal{X}$ be a metric *feature space* with an arbitrary metric d, and let a manifold $C_i \subset \mathcal{X}$ be the i-th *class*, where we assume that classes C_i are mutually disjoint. Each class C_i is unknown, but at the iteration t, we get a finite point-set of *class-samples* $S_i^{+,t} \subset C_i$. In the continual learning, we can only work with the given class-samples $S_i^{+,t}$ at the iteration t. Moreover, any class C_i can change during two iterations ($C_i^t \neq C_i^{t'}$) due to the concept drift; see Fig. 1. The goal of continual learning is finding such a classifier $F^t : \mathcal{X} \to \{1 \dots M\}$ at t that

$$\forall i \in \{1 \dots M\}, \forall x \in C_i^t : F^t(x) = i. \tag{1}$$

We propose to use M trainable manifolds $P_i \subset \mathcal{X}$, which we call *wraps*, to mimic their respective classes. Ideally, each class C_i is a subset of its respective *wrap* P_i, while all wraps are mutually disjoint:

$$\forall i \in \{1 \dots M\} : C_i^t \subset P_i^t, \tag{2}$$

$$\forall i, j \in \{1 \dots M\}, i \neq j : P_i^t \cap P_j^t = \emptyset. \tag{3}$$

Since the class C_i^t is unknown, the wrap P_i^t is modified w.r.t. *positive class-samples* $S_i^{+,t}$ and *negative class-samples* $S_i^{-,t} = \cup_{j \neq i}^{M} S_j^{+,t}$, finite sets of samples that are part of the class C_i^t or disjoint with the class, respectively. In continual learning, we consider that the class samples given at the t-th iteration do not necessarily represent the classes, i.e., a wrap that subsumes positive class samples taken from one iteration does not necessarily subsume the whole class.

Thus, the wraps have to aggregate the positive class-samples over time. Such aggregation is realized by keeping the wrap persistent over time; therefore, the trained wrap P_i^t should be disjoint with negative class-samples but it should also contain the positive class-samples, and a part of the original wrap P_i^t

$$\forall i \in \{1 \dots M\} : (P_i^{t-1} - V_i^t) \subset P_i^t, \tag{4}$$

$$\forall i \in \{1 \dots M\} : S_i^{+,t} \subset P_i^t, \tag{5}$$

$$\forall i \in \{1 \dots M\} : P_i^t \cap S_i^{-,t} = \emptyset, \tag{6}$$

where the *forget set* $V_i^t \subset P_i^{t-1}$ is a part of the i-th wrap that can be forgotten during the transition to the t-th iteration. The forget set is closely related to catastrophic forgetting; indeed, in the worst-case scenario, the forget set contains the entire wrap $P_i^{t-1} \subset V_i^t$. Thus, the whole aggregated knowledge from the past can be lost, and the wrap must be able to forget when the concept drift is detected.

Without any assumption about the probabilistic distribution of class samples, the only way to detect the concept drift is when negative class samples intersect the wrap $S_i^{-,t} \cap P_i^{t-1} \neq \emptyset$. The intersection of the wrap and negative class-samples must be a part of the forget set $S_i^{-,t} \cap P_i^{t-1} \subset V_i^t$ to stay consistent with the conditions (4) and (6).

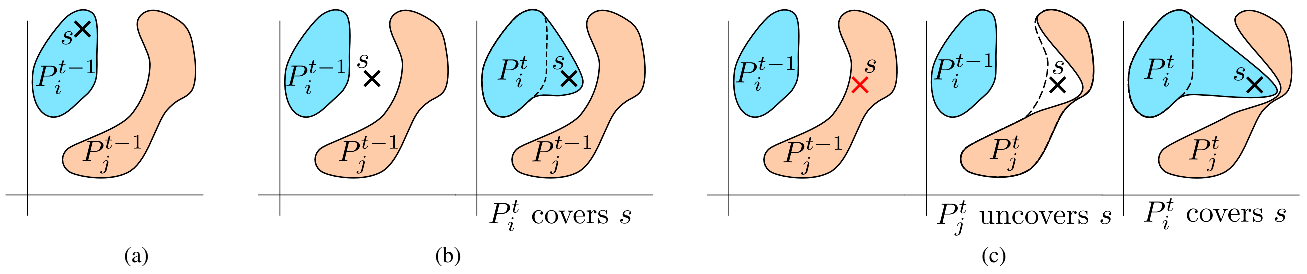


Fig. 2 Three possible scenarios depending on the membership of the class-sample $s \in S_i^{+,t}$. (a) The class-sample is inside its own wrap $s \in P_i^{t-1}$ and no wrap modification is needed. (b) If the class-sample is incident with none of the wraps $s \notin P_j^{t-1}; j \in \{1 \dots M\}$ then, P_i^t covers s. (c) If the class-sample falls in another wrap $s \in P_j^{t-1}; j \neq i$ then P_j^t uncovers s that is later covered by P_i^t.

The conditions (3), (4), (5), and (6) capture the problem of continual learning and provide constrains for the wrap training. In the context of sequential processing, only one wrap can be modified at a time. Therefore, if a class-sample of the class i ends up in the j-th wrap of a different class $j \neq i$, then, it is impossible to stay consistent with (3) and (6) with just a single modification. We present two elementary wrap modifiers called the cover and uncover that are used in construction of continual learning algorithm compliant with (3), (4), (5), and (6).

Both modifiers change the wrap w.r.t. different wraps and class samples. For each class-sample $s \in S_i^{+,t}$, three possible scenarios depend on whether the class-sample ends up either in the i-th wrap, in neither wrap, nor a different j-th wrap. The scenarios are depicted in Fig. 2.

In the first case $s \in P_i^{t-1}$, the sample is inside its wrap, and thus the wrap does not have to be changed w.r.t. the sample s. In the second case of $s \notin P_j^{t-1}; j \in \{1 \dots M\}$, the i-th wrap must be modified, and therefore, in the next iteration $s \in P_j^t$, we refer to such modification as P_i^t *covers* s. Note that by covering a class-sample s, the i-th wrap expands more than by a single class-sample ($P_j^{t-1} \cup \{s\} \subsetneq P_j^t$). The wrap is a manifold living in the metric space; therefore, every point s from the wrap is a center of some ball $B_d(s, \varepsilon_s)$ inside the wrap. Thus, if a sample becomes a part of the wrap during the cover, there is a ball centered at the covered sample that becomes a part of the wrap. Finally, in the third case $s \in P_j^t; j \neq i$, the i-th wrap must cover s, but also the j-th wrap must be modified to not include s at the iteration t to comply (3); and we refer such a modification as P_j^t *uncovers* s. Similarly to the cover, the wrap loses more than a single class sample during the uncover. After the uncover, s is not in P_j^t; therefore, in the metric space, there must be such a ball $B_d(s, \varepsilon_s)$ that is disjoint with P_j^t. The ball $B_d(s, \varepsilon_s)$ must intersect P_j^{t-1} at more than one point s; therefore, the intersection between the ball and the wrap must be a part of the forget set V_j^t.

We propose the following straightforward training scheme to modify the wraps w.r.t. class samples and consistency with (3), (4), (5), and (6). At each iteration t, the given class-sample set $S_i^{+,t}$ is firstly uncovered by all wraps $P_j^t; j \neq i$ and then covered by the P_i^t wrap. The classification can be then realized by the function

$$F^t(x) = \arg\max_i [\![x \in P_i^t]\!]. \tag{7}$$

The proposed implementation of the θ-wrap $P_i^t(\theta)$ and its training algorithm follows.

4 Method

The continual learning scheme described in the previous section is implemented as an ensemble of discriminating VAEs, each corresponding to its respective wrap. The autoencoders are trained to replay the samples of their respective wraps, used for further training, and discriminate between positive and negative samples. In this section, we present the training algorithm with two objectives for training $\mathcal{L}^+$ and untraining $\mathcal{L}^-$, see Alg. 1. The generative replay, training, and untraining loss functions are used to implement COVER and UNCOVER wrap modifiers, which are called in the update procedure summarized in Alg. 2.

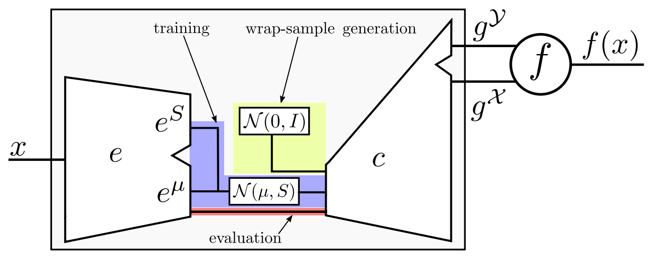
We define the wrap as a wrap function preimage of the interval $[0, \theta)$:

$$P = f^{-1}([0, \theta); g) = \{x | f(x; g) < \theta; x \in \mathcal{X}\}, \tag{8}$$

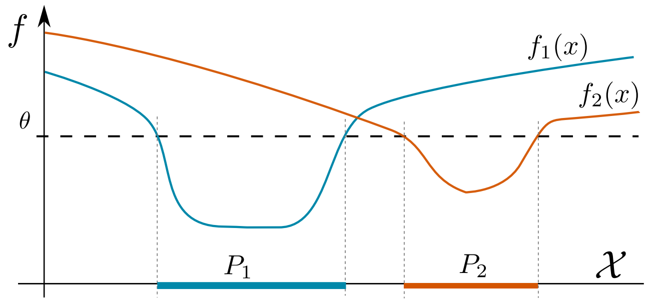
where $\theta \in (0, 1]$ is a threshold parameter and $f : \mathcal{X} \to \mathbb{R}^+$ is a continuous *wrap function* parametrized by a VAE $g : \mathcal{X} \to \mathcal{X} \times \mathcal{Y}$ with the output domain extended by the *discriminative space* $\mathcal{Y} = (0, 1)$.

The VAE is composed of the encoder $e : \mathcal{X} \to \mathcal{Z}_\mu \times \mathcal{Z}_S$ and decoder $c : \mathcal{Z} \to \mathcal{X} \times \mathcal{Y}$, where $\mathcal{Z} = \mathbb{R}^D$ is the *latent space* and $\mathcal{Z}_\mu \times \mathcal{Z}_S$ is the space of multivariate normal distribution parameters that VAEs are trained to approach $\mathcal{N}(0, I)$ for given samples [12]. The VAE architecture and the wrap representation are depicted in Fig. 3.

The autoencoder $g = (g^{\mathcal{X}}, g^{\mathcal{Y}})$ is trained to reconstruct positive samples $g^{\mathcal{X}}(x^+) \approx x$ and to discriminate positive $\sigma(g^{\mathcal{Y}}(x^+)) \approx 1$ and negative $\sigma(g^{\mathcal{Y}}(x^-)) \approx 0$ samples;



(a)



(b)

Fig. 3 (a) The Variational AutoEncoder (VAE) g is composed of two neural networks: encoder e and decoder c. The encoder maps the input x to multivariate normal distribution parameters of the latent space, where $e^\mu(x)$ and $e^S(x)$ give the mean and variance, respectively. The decoder then maps the latent space onto feature space $\mathcal{X}$ extended by discriminative space $\mathcal{Y}$. The VAE is used in three modes (use cases): as a wrap-sample generator (yellow), where a random sample is taken from the normal distribution and mapped into the feature space; as trainable network (blue), where for each input x a random sample is taken from the normal distribution parametrized by $e^S(x)$ and $e^\mu(x)$, and the result is evaluated by the loss $\mathcal{L}^+$ or $\mathcal{L}^-$, depending on whether the sample was positive or negative, respectively; as the wrap-function evaluator (red), where the input x is encoded by $e^\mu(x)$, decoded and evaluated by the wrap-function f. (b) The i-th wrap-function and threshold θ determines the i-th wrap manifold.

where σ is the sigmoid function. In this work, we use the wrap function

$$f(x; g) = \frac{1}{\dim(\mathcal{X})}||g^\mathcal{X}(x) - x||^2 + (1 - \sigma(g^\mathcal{Y}(x))), \quad (9)$$

where the first term corresponds to reconstruction objective training the network to reconstruct the input from the latent space [14]; the objective is normalized by the feature space dimension. We extend the reconstruction error by the second term of (9), which optimizes the discriminative output $g^\mathcal{Y}(x)$ to distinguish positive samples from negative ones.

The advantage of using the discriminative term is twofold. First, the autoencoder initialized by the LeCun initialization proposed in [18] maps the feature space $g^\mathcal{Y}(x) \approx 0.5$, and therefore, if $\theta < 0.5$, the wrap is mostly empty initially. The second advantage of the discriminative term is that, unlike the reconstruction error, the discriminative term is lower and upper bounded. Thus the term cannot diverge, which is utilized in the objective for untraining $\mathcal{L}^-$ (12).

Algorithm 1 Train autoencoder with positive and negative samples.

Variables g: autoencoder; X^+, X^-: positive and negative samples; θ: threshold; E: max epoch; $\mathcal{J}$: cost function;
Result g: trained autoencoder;

```
1: function TRAIN($g$, $X^+$, $X^-$)
2:     $g_0 \leftarrow g$
3:     for $k = 1$ to $E$ do
4:         $\epsilon_k \leftarrow \mathcal{J}(g_{k-1}, X^+, X^-)$
5:         $g_k \leftarrow \texttt{gradient-descent}(g_{k-1}, \epsilon_k)$
6:         if $\forall x \in X^+ : f(x; g_k) < \theta$ and $\forall x \in X^- : f(x; g_k) > \theta$ then
7:             break
8:         end if
9:     end for
10:    $g \leftarrow g_k$
11: end function
```

The autoencoder is trained under two objectives $\mathcal{L}^+$ and $\mathcal{L}^-$ used for positive and negative samples, respectively

$$\texttt{reg}(x) = D_{\mathbf{KL}}(\mathcal{N}(e^\mu(x), e^S(x))||\mathcal{N}(0, I)), \quad (10)$$

$$\mathcal{L}^+(x) = ||g^\mathcal{X}(x) - x||^2 + (1 - \sigma(g^\mathcal{Y}(x))) + \texttt{reg}(x), \quad (11)$$

$$\mathcal{L}^-(x) = 1 - \sigma(g^\mathcal{Y}(x)), \quad (12)$$

where `reg(x)` is the VAE regularization of that improves generative properties of the autoencoder [12] by optimizing the Kullback-Lieber divergence $D_{\mathbf{KL}}$. The relationship between the objectives and wrap function is $\mathcal{L}^-(x) \leq f(x) \leq \mathcal{L}^+(x)$. Thus, by minimizing the loss of positive samples $\mathcal{L}^+(x)$, the wrap value $f(x)$ is decreased, while by maximizing the loss of negative samples $\mathcal{L}^-(x)$, the wrap value $f(x)$ increases. The total *cost function* for the positive samples X^+ and negative samples X^- is then

$$\mathcal{J}(X^+, X^-) = \frac{1}{|X^+|}\sum_{x \in X^+} \mathcal{L}^+(x) - \frac{1}{|X^-|}\sum_{x \in X^-} \mathcal{L}^-(x), \quad (13)$$

which is used in the training, depicted in Alg. 1.

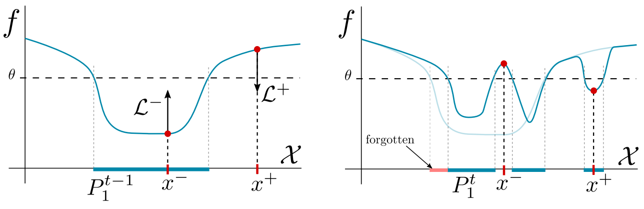


Fig. 4 Illustration of the wrap-function (blue curve) modified during training the positive sample x^+ and untraining negative sample x^- (red). The positive sample x^+ is optimized by the $\mathcal{L}^+$ minimization while x^- by the $\mathcal{L}^-$ maximization. On the right, the training is finished after the wrap-value of the positive sample (red disc) is below θ and the negative sample is above θ. The unoptimized part of $\mathcal{X}$ is uncontrolled, which might result in catastrophic forgetting, here visualized as the wrap P_1 (blue strip) losing its part (red strip) during training.

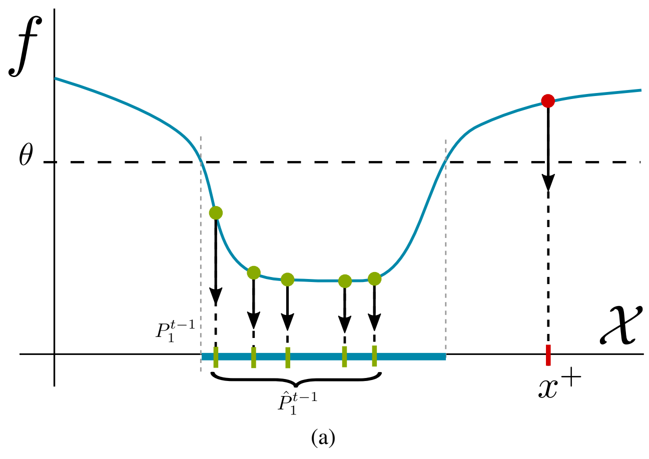


Fig. 5 Illustration of the cover and uncover wrap modifications. (a) When the wrap P_1 (blue strip) covers the sample x^+ (red), the wrap-function (blue curve) is minimized for the sample x^+ along with the wrap-samples $\hat{P}_1^{t-1}$ (green). (b) When the wrap P_1 uncovers the sample x^-, the wrap function is maximized for the sample x^- and minimized for the wrap-samples except for those that are close to x^-.

During the training, the i-th autoencoder is optimized until $X^+ \subset P_i$ and $X^- \cap P_i = \emptyset$, or until the max epoch is reached. The training optimizes the wrap value for each given positive or negative sample; however, it is generally unknown how the wrap value is changed in the rest of the feature space $\mathcal{X} - (X^+ \cup X^-)$; see Fig. 4. Therefore, some arbitrary regions of the unoptimized space $\mathcal{X} - (X^+ \cup X^-)$ might become or stop being a part of the optimized wrap, where the former can violate (3) and the latter violates (4).

Intuitively, the positive samples should contain a representation of the trained wrap from the previous iteration, so the new wrap subsumes the wrap from the previous iteration. Likewise, the negative samples of the trained wrap must contain a representation of other wraps so that the trained wrap stays disjoint.

The wraps are represented by samples, which we call *wrap-samples*. The wrap-samples $\hat{P}_i^t$ are extracted from the VAE decoder

$$Z = \{\mathcal{N}(0, I)\}_N, \tag{14}$$
$$\hat{P}_i^t = \{c_i^t(z) | f_i^t(c_i^t(z)) < \theta, z \in Z\}, \tag{15}$$

where the multivariate distribution $\mathcal{N}(0, I)$ is used to generate N random latent samples, which are decoded into the feature space. We preserve the learned knowledge through iterations by introducing the extracted wrap-samples into the training set.

The wrap-sample introduction has a straightforward use in the definition of the cover wrap modifier

$$\begin{aligned}&\text{COVER}(g_i^{t-1}, S_i^t) := \\ &\text{TRAIN}(g_i^{t-1}, S_i^{+,t} \cup \hat{P}_i^{t-1}, S_i^{-,t} \cup \bigcup_{j \neq i} \hat{P}_j^{t-1}),\end{aligned} \tag{16}$$

where the wrap-samples are simply added to class-samples, which supports (4), and the wrap-samples of other wraps are added as negative samples to ensure (3) and (6); see Fig. 5a. The COVER modifier allows the wrap to finish the second scenario shown in Fig. 2a, where the new class samples are not a part of any wrap.

However, in this work, we also consider the concept drift, which is detected by a class-sample being inside a wrap of another class as it is illustrated in Fig. 2c, where the wrap must uncover the foreign class sample. Assuming that the autoencoder is a continuous function, by uncovering point s^-, the wrap must forget some neighborhood $B_d(s^-, \varepsilon_{s^-})$, which intersects with the wrap P_i^{t-1}. Therefore, the intersection $B_d(s^-, \varepsilon_{s^-}) \cap \hat{P}_i^{t-1}$ cannot be part of negative samples. We propose to approximate the neighborhood $B_d(s^-, \varepsilon_{s^-})$ by the Euclidean ε_0-ball $B_d(s^-, \varepsilon_0)$ for each negative class-sample and subtract it from wrap-samples

$$\begin{aligned}&\text{UNCOVER}(g_j^{t-1}, S_j^t) := \\ &\text{TRAIN}(g_j^{t-1}, \hat{P}_j^{t-1} - B_d(S_j^{-,t}, \varepsilon_0), S_j^{-,t} \cup \bigcup_{k \neq j} \hat{P}_k^{t-1}),\end{aligned} \tag{17}$$

where ε_0 is a hyperparameter; see Fig. 5b. Practically, the ball subtraction is implemented as deletion of such $\hat{P}_j^{t-1}$ wrap-samples in ε_0-neighborhood of any negative class-sample.

The UNCOVER method is consistent with (4), (3), and (6), but not (5) because the positive class-samples can be inside wraps of other classes (in such situation (3) and (5) cannot be true at the same time). Thus, the UNCOVER of all negative class-samples transforms the eventual third scenario (see Fig. 2c) into the second scenario (Fig. 2b) for which we can use the COVER wrap modifier. With covering (16) and uncovering (17), the update of the autoencoder is completely defined, and the procedure is depicted in Alg. 2.

Algorithm 2 Update autoencoders with the dataset $\mathcal{D}$.

```
Variables $\{g_i\}$: collection of $M$ autoencoders, $i \in \{1 \dots M\}$;
$\mathcal{D} = \{(s, k)\}$: dataset of labeled samples where $k$ indicates a label of the sample $s$;
$\theta$: threshold; $f$: wrap function;
Result $\{g_i\}$: updated autoencoders;
1: function UPDATE($\{g_i\}$, $\mathcal{D}$, $\theta$)
2:     for $i = 1$ to $M$ do
3:         $S_i \leftarrow \{s | (s, i) \in \mathcal{D}\}$
4:         for $j = 1$ to $M$ where $j \neq i$ do
5:             if $\exists s \in S_i : f(s; g_j) < \theta$ then
6:                 $g_j \leftarrow$ UNCOVER($g_j, S_i$)        ▷ See (17).
7:             end if
8:         end for
9:         $g_i \leftarrow$ COVER($g_i, S_i$)        ▷ See (16).
10:    end for
11: end function
```

For the evaluation, we relax (7) by using

$$F'^t(x) = \arg\min_i f_i^t(x), \tag{18}$$

which maps each point of the feature space onto a single label.

5 Experiments

In this section, we report on the empirical evaluation of the proposed approach in two parts. First, we demonstrate the qualitative differences between continual learners on intuitive examples. Second, the proposed continual learner is compared with two state-of-the-art continual learners, and we show the selective forgetting is needed for scenarios where the concept drift is possible.

We assess the performance of continual learners by evaluating their cover and uncover mechanisms. The cover mechanism corresponds to the ability to learn without forgetting, while the uncover mechanism relates to selective forgetting needed for the concept drift adaptation. In this section, the performance of cover and uncover mechanisms is evaluated by evaluation scenarios. The scenario is a sequence of training tasks T_i constructed from subclasses, where each subclass has an assigned label (subclasses with the same label form a class).

Three domains are used as sources of the subclasses. In the first domain, further referred to as `gauss`, we use three clusters generated from the multivariate Gaussian distribution in the three-dimensional feature space to demonstrate how the proposed classifier works intuitively. The second domain is the MNIST [17] dataset (denoted `mnist`) containing ten subclasses (digit is a subclass), where we evaluate how the proposed classifier performs in complex domains. Finally, the CIFAR10 dataset [33] (denoted `cifar`) containing ten image subclasses as `mnist`; however, unlike `mnist`, `cifar` is challenging even in non-incremental classification scenarios. All three domains provide subclasses from which a sequence of training tasks can be constructed.

Each task T, the classifier is trained to label samples of the subclass G with the assigned label L; i.e., the classifier trains over the dataset $\{(s, L) | s \in G\}$. Multiple subclass-label assignments are possible during a single task $T = \{(G_i, L_j), (G_{i'}, L_{j'}), \dots\}$. The notion of evaluation scenarios generalizes the commonly used catastrophic forgetting benchmarks [13,10,26], allowing us to construct benchmarks for concept drift. Thus, various sequences of the training tasks can test different qualities of the continual learner.

We adopt four basic scenarios [32] evaluating the essential properties of continually trained classifiers in two tasks. The first two scenarios evaluate the robustness against catastrophic forgetting where a new class is added in the `ADD` scenario, and an existing class is expanded in the `EXP` scenario. The scenarios `INC` and `SEP` evaluate the adaptation to the concept drift by changing the labels of already presented samples.

All four scenarios are constructed from three subclasses, G_1, G_2, and G_3, from which the classes are created. In the case of the gauss domain, we use clusters generated from the Gaussian distribution, and for the `mnist` and `cifar` domains, the subclasses are digits and vehicle/animal images, respectively; see Table 1. In each scenario, the subclasses are presented to the classifier in two iterations T_1 and T_2 that learns to classify the subclass samples with one of two labels L_1 or L_2. The target labels assigned to each subclass and the iteration in which the subclasses are presented to the classifiers are unique for each scenario, as described in Table 2. For the `mnist` and `cifar` domains, we additionally examine the robustness of the classifiers in complex scenarios, which are composed of multiple tasks and multiple subclasses.

Table 1 The sources of subclass samples G_i. For the `gauss` domain, the clusters `R`, `G`, and `B`, corresponding to the red, green and blue colors in the RGB space, are used. For the `mnist` and `cifar` domains, three selected digits and vehicles/animals are used as subclasses, respectively.

Assignment	G_1	G_2	G_3
`gauss`	`R`	`B`	`G`
`mnist021`	0	2	1
`mnist197`	1	9	7
`cifar021`	airplane	bird	automobile
`cifar197`	automobile	truck	horse

We refer to the proposed algorithm as `ensgendel`[1] from *ens*emble of *gen*erative models with mechanism of sample *del*etion. The utilized VAE comprises hidden layers of

[1] Implementation of the evaluation framework and the `ensgendel` algorithm is available at https://github.com/comrob/ensgendel.

Table 2 Configurations of four basic scenarios. In each scenario, we present a subclass G_i of the samples (with the labels L_j described in the corresponding cell) to each classifier during two iterations T_1 and T_2; if there is no label, the subclass is not presented. The last row configures the testing iteration, where the accuracy of the predicted labels G_1 shows the robustness to catastrophic forgetting, and the accuracy of G_3 shows the adaptability to the concept drift in the `INC` and `SEP` scenarios.

	`ADD`			`EXP`			`INC`			`SEP`		
Iteration	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
T_1	L_1	-	-	L_1	-	-	L_1	-	L_2	L_1	-	L_1
T_2	-	L_2	-	-	L_2	L_1	-	L_2	L_1	-	L_2	L_2
test	L_1	L_2	-	L_1	L_2	L_1	L_1	L_2	L_1	L_1	L_2	L_2

rectified linear units (ReLU), except for the output layer for the encoder and decoder that are just linear units. The network is trained by ADAM with default parameters [11]. The neural network architecture and classifier hyperparameters are specified for each domain, where all hyperparameters were selected empirically.

5.1 Three Flavors of Continual Learning

The difference between the continual learners might not be just quantitative, where one learner has better results in certain metrics, but also qualitative, where the learner has fundamental limitations. We demonstrate such limitations on two different flavors of the proposed `ensgendel` learner.

Similarly to [32], we modify the proposed classifier into two continual learning algorithms without the ability to control the forgetting. The first is the ensemble of discriminators generating (recalling) samples (`ensgen`), which we implement by omitting Line 6 in Alg. 2. The `ensgen` classifier is a hybrid continual learner with generative replay and ensemble architecture approach, but it is unable to untrain already learned knowledge. The second classifier is an ensemble of discriminators (`ens`), which is different from `ensgen` by setting the maximum number of samples $N = 0$. Therefore, the classifier `ens` is unable to rehearse on generated samples. The classifiers `ens` and `ensgen` are designed to resist the concept drift; however, we demonstrate that the continually trained classifiers cannot adapt the concept drift without selective forgetting.

5.1.1 Gaussian Clusters in 3-Dimensional Space

In the `gauss` domain, we compare the classifiers on a toy problem in the three dimensional space $\mathcal{X} = \mathbb{R}^3$ that contains three clusters `R`, `G`, and `B` generated from the Gaussian distribution with the means $\mu_{\mathrm{R}} = (1, -1, -1), \mu_{\mathrm{G}} = (-1, 1, -1), \mu_{\mathrm{B}} = (-1, -1, 1)$, respectively, and with the common covariance matrix $0.001\mathbb{I}$. The classifiers are trained to label the cluster by one of two labels, L_1 or L_2. The cluster labeling in each scenario during both tasks is described in Table 2. The proposed classifier contains two VAEs, each with encoder and decoder layer sizes 3-15-30-4 and 2-15-30-4, respectively. The threshold parameter is set to $\theta = 0.2$, the subtraction radius $\varepsilon_0 = 0.1$, the maximum epochs $E = 1000$, and the number of latent samples $N = 200$. The results of the scenario evaluations are depicted in Table 3 and visualized in Fig. 6.

5.1.2 MNIST evaluation

In the `mnist` domain, we use MNIST [17] to construct the basic scenarios and three complex scenarios spanning five iterations. MNIST is a dataset containing ten handwritten digits from zero to nine, which are saved as 28×28 greyscale pictures; therefore, the feature space is $\mathcal{X} = \mathbb{R}^{784}$. The dataset contains 7000 samples per class (digit), where the samples are divided into training and testing sets with a ratio of six to one. The testing set is then used for the accuracy evaluation in each scenario.

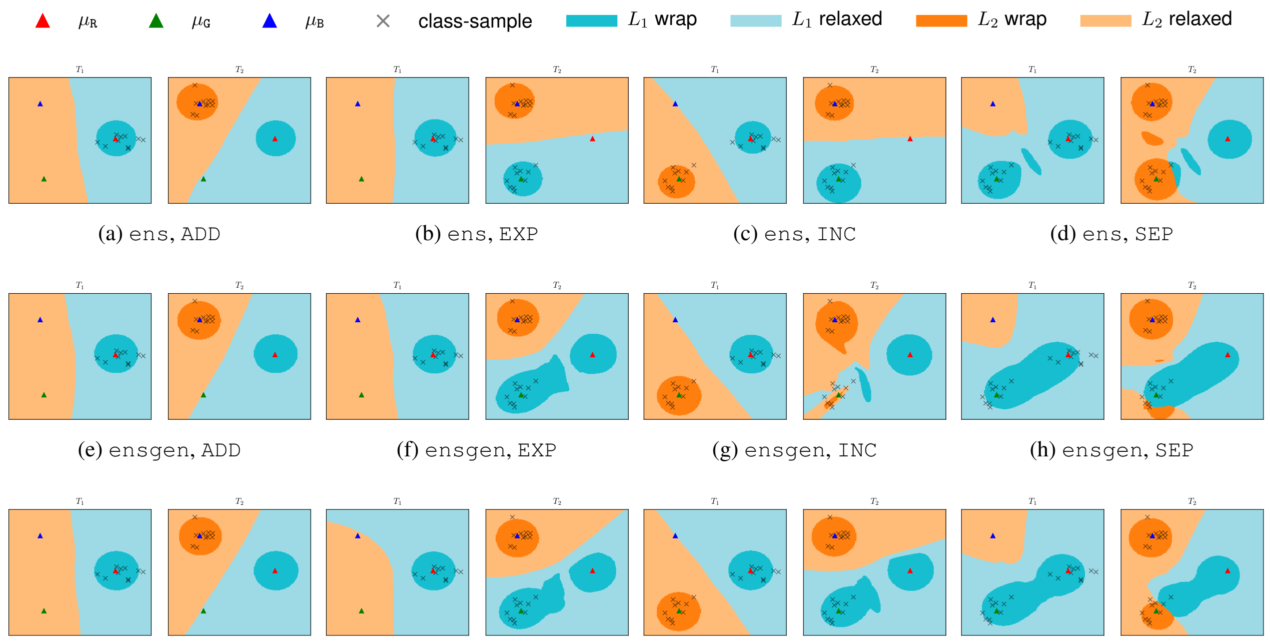
In the basic scenarios, the digit classes serve as subclasses from which we construct tasks for each scenario, similarly to the `gauss` domain in the previous section; see Table 2. Moreover, we examine the classifiers in three complex scenarios `ADD5`, `EXP5`, and `SEP5`, which are similar to basic scenarios but are five iterations long.

The used architecture of the VAE is 784-250-500-16, 8-250-500-785 for the encoder and decoder, respectively. The threshold parameter is set to $\theta = 0.1$, the subtraction radius $\varepsilon_0 = 4$, the maximum epochs $E = 100$, and the number of latent samples $N = 200$. The hyperparameters are tuned to `mnist021`. The evaluation results for basic scenarios are depicted in Table 3.

The three complex scenarios are extensions of the basic scenarios that examine the classifier robustness in the scenarios with more iterations, classes, and subclasses. `ADD5` is a scenario where the classifier trains to classify multiple digits, from zero to four, with their corresponding labels $L_1, L_2, \ldots, L_5$, where the samples of each digit are presented to the classifier in a separate iteration $T_1, T_2, \ldots, T_5$. In `EXP5`, the classifiers are trained in five iterations to distinguish between odd and even digits. Each iteration T_i, the classifier trains to classify $2i-2$ digit as even (L_1) and $2i-1$ as odd (L_2). Finally, in the scenario `SEP5`, there are two la-

Table 3 Classifier accuracy calculated on the testing task before the iteration T_2 and after it.

Assignment	Classifier	ADD		EXP		INC		SEP	
		T_1	T_2	T_1	T_2	T_1	T_2	T_1	T_2
`gauss`	`ens`	1.0	**1.0**	0.66	0.91	0.51	0.85	0.67	0.92
	`ensgen`	1.0	**1.0**	0.66	**1.0**	0.51	0.89	0.66	0.74
	`ensgendel`	1.0	**1.0**	0.54	**1.0**	0.53	**1.0**	0.67	**0.94**
`mnist021`	`ens`	0.49	0.94	0.67	0.69	0.5	0.68	0.31	0.87
	`ensgen`	0.49	**0.97**	0.67	0.96	0.46	0.92	0.31	0.9
	`ensgendel`	0.49	**0.97**	0.67	**0.97**	0.51	**0.98**	0.31	**0.97**
`mnist197`	`ens`	0.53	**0.99**	0.68	0.85	0.67	0.84	0.36	0.9
	`ensgen`	0.53	0.97	0.68	**0.96**	0.67	0.88	0.36	0.87
	`ensgendel`	0.53	0.98	0.68	**0.96**	0.65	**0.95**	0.36	**0.93**



(a) `ens`, `ADD` (b) `ens`, `EXP` (c) `ens`, `INC` (d) `ens`, `SEP`

(e) `ensgen`, `ADD` (f) `ensgen`, `EXP` (g) `ensgen`, `INC` (h) `ensgen`, `SEP`

(i) `ensgendel`, `ADD` (j) `ensgendel`, `EXP` (k) `ensgendel`, `INC` (l) `ensgendel`, `SEP`

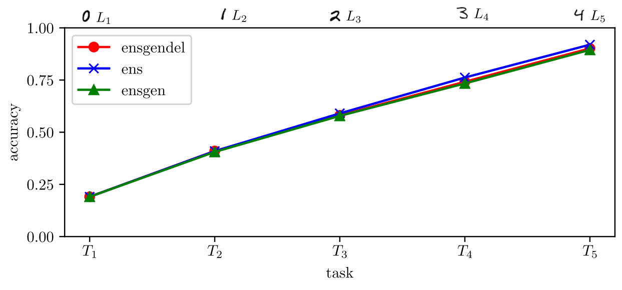
Fig. 6 The `gauss` domain projected onto the 2D plane by Principal Component Analysis (PCA) decomposition. The 2D plane is then uniformly sampled and projected back into feature space, where the projected samples are labeled by classifiers with (18) function. The region predicted as L_1 is shown in light blue while the L_2 is light orange. Moreover, the (7) is used to get the wraps of the L_1 (orange) and L_2 (blue) classes. The black marks represent the given class samples during the particular iteration, while the triangles are means of `R` (red), `G` (green), and `B` (blue) clusters.

bels, where at T_1, digits 1 to 5 are associated with the label L_1, and zero digits are associated with L_2. Each following iteration $T_i; 1 < i < 6$, the classifiers train on digits of i that are relabeled to L_2 label. After the T_5 iteration, the classifiers should label digits 0 to 4 as L_2 and remaining five digits as L_1. The results of three complex scenarios are visualized in Fig. 7.

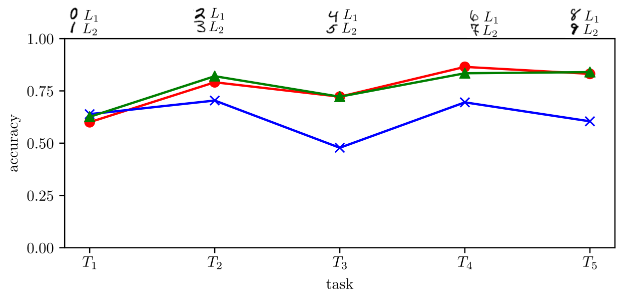
5.2 Comparative Study

In this part of the evaluation results, we assess how state-of-the-art (SotA) continual learners perform in scenarios that induce catastrophic forgetting and concept drift. The proposed algorithm, `ensgendel`, is compared to two continual classifiers: Growing Dual-Memory (`GDM`) learner [26] and Closed Loop Memory Generative Adversarial Network (`CloGAN`) [27]. All compared algorithms use a generative replay as a counter-measure to catastrophic forgetting. However, they are also hybridized by one other incremental learning mechanic: an architecture change for `ensgendel` and `GDM`, and (true) rehearsal technique for `CloGAN`.

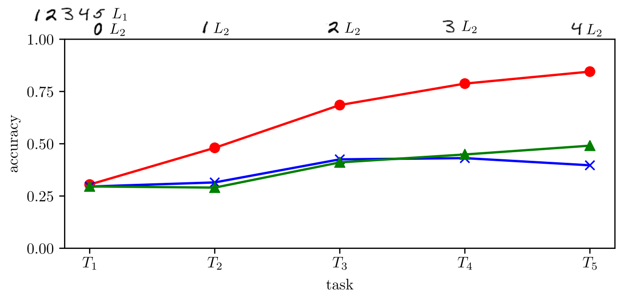
We compared the continual classifiers in the previously described MNIST evaluation scheme and the more challenging CIFAR10 domain. Since the CIFAR10 subclasses are high-dimensional, the neural network architecture of `ensgendel`



(a) `ADD5`



(b) `EXP5`



(c) `SEP5`

Fig. 7 Accuracy evolution during five training iterations in the complex scenarios `ADD5`, `EXP5`, and `SEP5`. For each iteration, the evaluated learners are exposed to subclass samples (digit images) with labels L_i. The accuracy is evaluated w.r.t. to the target final state.

was adapted for image classification to improve the performance. The `cifar` scenarios and the setup of compared learners are described in the following paragraphs.

5.2.1 `cifar` *scenarios*

The CIFAR10 dataset [33] is a set of roughly eighty million 32×32 RGB images. For each scenario, the training set is randomly subsampled to size 1000, which emulates the underrepresentation expected in continual learning, where datasets are experienced continually. Each image belongs to one of the ten subclasses: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck. Since `cifar10` and `mnist` have the same number of subclasses, we use the alphabetic order as the label (airplane and truck correspond to 0 and 9, respectively) to construct the same scenarios as we constructed for the `mnist` domain, see Table 1. Thus, the only difference between `cifar` and `mnist` scenarios are the input images.

5.2.2 *Growing Dual-Memory Learner (*`GDM`*)*

The approach of the `GDM` learner to continual learning is a hybrid of generative replay approach and dynamic architecture [26]. The architecture contains two layers of memory: the episodic memory that incrementally encodes the inputs, and the semantic memory, which extracts high-level knowledge from the episodic memory. Both layers are growing self-organizing maps (SOMs) [21], which grow with the complexity of the input data; thus, the architecture encodes the information incrementally. Moreover, SOM of the episodic memory is used as a generator of pseudo-samples, further improving the robustness against catastrophic forgetting. Due to the nature of SOM, which work in the Euclidean space, the authors propose to preprocess image data with VGG model [31] pre-trained on image dataset [28].

We have integrated the authors' implementation of the `GDM`[2] and preprocessed the MNIST 28×28 and CIFAR $32 \times 32 \times 3$ images into 256 dimensional feature vectors using the pre-trained VGG model. The `GDM` model trains for 80 epochs, where the learning rates for the episodic and semantic growing SOMs are set to 0.2 and 0.001, respectively.

5.2.3 *Closed Loop Memory Generative Adversarial Network (*`CloGAN`*)*

The `CloGAN` learner, introduced in [27], is a hybrid of generative replay and (true) rehearsal approaches. The generative replay is provided by the Auxiliary Conditional Generative Adversarial Network (ACGAN) [25] trained to generate pseudo-samples. In contrast to the `ensgendel` learner, where each generative model (the VAE in our case) is trained to represent one class, the `CloGAN`'s generative model is trained to represent all classes.

However, the main difference stems from the `CloGAN` learning algorithm's rehearsal mechanism, where `CloGAN` stores selected samples into limited memory. The authors propose to cluster the samples into multiple sets, from which `CloGAN` draws an equal number of the selected samples to ensure the heterogeneity amongst the stored samples. The number of selected samples depends on the memory capacity, which is limited. Therefore, if the memory is filled up, the size of the stored clusters is decreased by pruning the stored samples to free the storage space for the current batch of the selected samples. The stored, generated, and input samples partake in the learning of the classifier.

[2] https://github.com/giparisi/GDM

The implementation of CloGAN[3] was integrated with our evaluation framework. The storage was set to the capacity of 90 samples, which are divided into ten clusters. The classifier and generative model were trained within hundred epochs per task with the learning rate 0.0002.

5.2.4 *Convolutional network in* `ensgendel`

For high-dimensional inputs from `cifar` we implemented convolutional VAE, where we use convolutional layers instead of fully connected layers. The image is encoded into latent mean and variance by four convolutional layers mapping the $32 \times 32 \times 3$ image into $2 \times 2 \times 256$ tensor, which is then processed by three fully connected layers 1024-256-256 into 128-dimensional latent space. The 128-dimensional feature is then decoded by mirrored architecture: first processed by fully-connected layers 128-256-1024 and then mapped back into image by four deconvolutional layers. The last fully connected layer of the decoder is also used for getting the discriminative part of the output $g^{\mathcal{Y}}$, where the layer 1024-1 is added. For reconstruction error training, we used the negative log-likelihood of Bernoulli distribution.

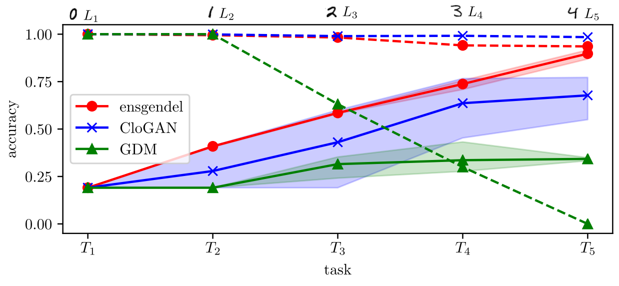
The change of the underlying VAE implementation does not change the `ensgendel` algorithm itself. The `ensgendel` hyperparameters are set to $\theta = 0.04$ and $\varepsilon_0 = 1$.

5.2.5 *Comparison of Proposed Method with SotA*

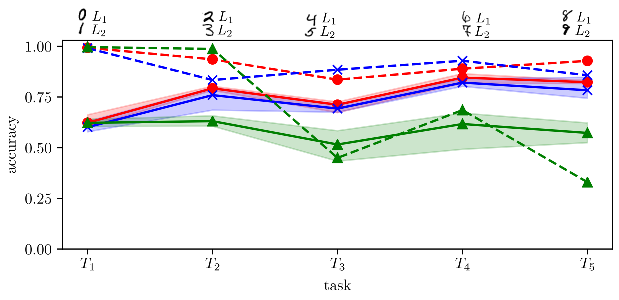
All three learners, `ensgendel`, `GDM`, and `CloGAN`, were evaluated in the same set of the scenarios described in Section 5.1.2. Similarly to `ensgendel`, we made an effort to tune hyperparameters of the compared learners w.r.t. to the scenario sets `mnist021` and `cifar021` for `mnist` and `cifar` domains, respectively. The comparison results are averaged over five experimental runs. The evaluation results of the learners in the basic scenarios of `mnist` and `cifar` domains are shown in Table 4. The results of the complex scenarios `ADD5`, `EXP5`, and `SEP5` are shown in Figs. 8 and 9.

It is evident that CloGAN has an uncover mechanism, and thus the algorithm untrains the knowledge selectively. The mechanism is not reported by its authors in [27]; however, we hypothesize that the unlearning mechanism is a consequence of the clustered storage in CloGAN. Nevertheless, the cluster storage cannot immediately forget outdated sample labels, as it is apparent from the `INC4` scenario shown in Fig. 8d. `INC4` is a four tasks long scenario similar to inclusion, which tests the continual learner's ability to retrain changed knowledge immediately. At T_1, the learner is trained only on zero digits with the assigned L_1 labels. At T_2, two labels are presented to the learner: ones labeled as L_1, and twos and zeroes labeled as L_2 (note that the label

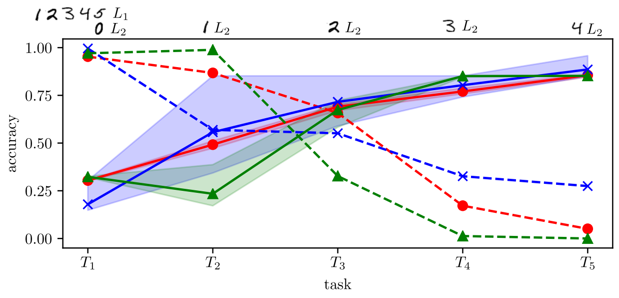
[3] https://github.com/fikavw/CloGAN



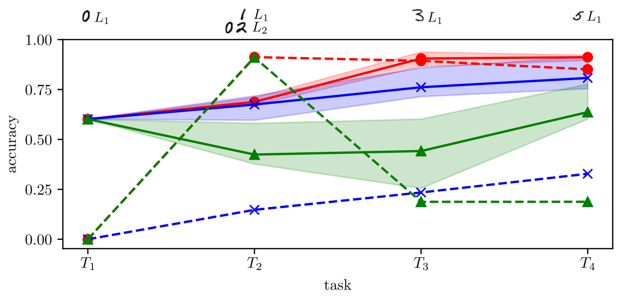
(a) `mnist ADD5`



(b) `mnist EXP5`



(c) `mnist SEP5`



(d) `mnist INC4`

Fig. 8 Accuracy evolution during multiple iterations of training in the complex scenarios `ADD5`, `EXP5`, `SEP5`, and `INC4` in `mnist` domain. The accuracy (full lines), averaged over five experimental runs, is evaluated w.r.t. to the target final state. Shaded areas indicate the maximum and minimum of accuracy. The dashed line tracks the accuracy of a particular subclass whose label can be gradually forgotten during subsequent tasks. In `ADD5` and `EXP5` scenarios, we track the accuracy of zeroes being labeled as L_1. In `SEP5`, we track the accuracy of fives being labeled as L_1, and in `INC4`, the accuracy of zeroes being labeled as L_2.

Table 4 Classifier accuracies averaged over five runs with the standard deviation shown in brackets. The accuracies are calculated on the testing task before the iteration T_2 and after it. The last column shows accuracy of G_1 and G_3 discrimination trained within one task, i.e., a regular binary classification accuracy.

Assignment	Classifier	`ADD`		`EXP`		`INC`		`SEP`		Discr.
		T_1	T_2	T_1	T_2	T_1	T_2	T_1	T_2	
`mnist021`	`GDM`	0.49(0.00)	0.82(0.17)	0.67(0.00)	0.80(0.10)	0.44(0.03)	0.73(0.09)	0.31(0.00)	0.64(0.08)	0.98(0.00)
	`CloGAN`	0.49(0.00)	**0.97**(0.02)	0.67(0.00)	**0.98**(0.02)	0.45(0.04)	0.92(0.03)	0.31(0.00)	0.94(0.10)	0.98(0.03)
	`ensgendel`	0.49(0.00)	0.96(0.01)	0.67(0.00)	0.97(0.01)	0.50(0.05)	**0.97**(0.01)	0.31(0.00)	**0.96**(0.01)	**0.99**(0.00)
`mnist197`	`GDM`	0.53(0.00)	0.81(0.07)	0.68(0.00)	0.74(0.04)	0.60(0.01)	0.64(0.03)	0.36(0.00)	0.69(0.06)	0.89(0.01)
	`CloGAN`	0.53(0.00)	**0.97**(0.04)	0.68(0.00)	0.91(0.05)	0.67(0.01)	0.93(0.04)	0.36(0.00)	**0.98**(0.01)	**0.98**(0.01)
	`ensgendel`	0.53(0.00)	0.96(0.00)	0.68(0.00)	**0.96**(0.00)	0.65(0.03)	**0.94**(0.00)	0.36(0.00)	**0.98**(0.01)	**0.98**(0.01)
`cifar012`	`GDM`	0.50(0.00)	0.63(0.02)	0.67(0.00)	**0.68**(0.02)	0.45(0.02)	0.48(0.02)	0.33(0.00)	0.59(0.07)	**0.73**(0.02)
	`CloGAN`	0.50(0.00)	0.56(0.06)	0.67(0.00)	0.67(0.00)	0.41(0.06)	0.57(0.10)	0.33(0.00)	0.62(0.14)	0.59(0.06)
	`ensgendel`	0.50(0.00)	**0.64**(0.03)	0.67(0.00)	0.61(0.03)	0.52(0.02)	**0.60**(0.03)	0.33(0.00)	**0.66**(0.01)	0.72(0.02)
`cifar197`	`GDM`	0.50(0.00)	0.66(0.05)	0.67(0.00)	0.69(0.02)	0.50(0.04)	0.56(0.03)	0.33(0.00)	0.64(0.04)	**0.62**(0.03)
	`CloGAN`	0.50(0.00)	0.50(0.00)	0.67(0.00)	0.67(0.00)	0.57(0.11)	**0.70**(0.04)	0.33(0.00)	0.47(0.16)	0.52(0.04)
	`ensgendel`	0.50(0.00)	**0.69**(0.04)	0.67(0.00)	**0.76**(0.01)	0.49(0.02)	0.62(0.01)	0.33(0.00)	**0.66**(0.01)	**0.62**(0.01)

of the zero digits changed). Then, at T_3 and T_4, the learner trains to label threes and fives as L_1 in their respective tasks. After the final task T_4, the continual learner should classify ones, threes, and fives as L_1, and twos and zeroes as L_2.

All compared continual learners show resistance to catastrophic forgetting; however, selective forgetting is still challenging. The performance metrics of the proposed `ensgendel` are competitive to `CloGAN` in most scenarios, although the former learner seems to be more stable. The purpose of the herein presented evaluation results is to find fundamental differences between the continual learners, limiting the performance in certain scenarios. We discuss the found qualitative differences in the following section.

6 Discussion

Catastrophic forgetting and concept drift are usually considered two separate problems; however, both can occur during continual learning. We related both problems by two identified mechanisms of continual learning: cover and uncover. The cover mechanism is implemented, explicitly or implicitly, by learners that can learn incrementally without forgetting. However, the robustness to forgetting hinders the concept drift adaptation, and only the algorithms with the uncover mechanism were able to adapt. Thus the cover and uncover mechanisms are fundamental for continual learning.

6.1 Measured Qualities of `ensgendel`

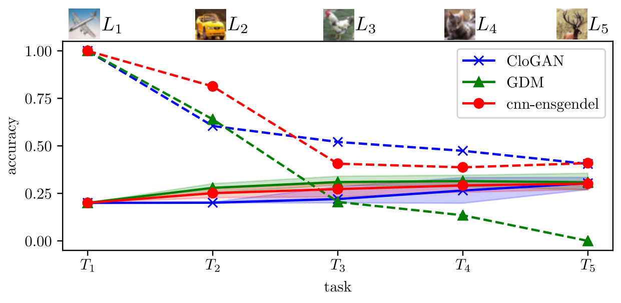
The cover and uncover are explicitly implemented in the proposed `ensgendel` classifier. `ensgendel` and its different configurations `ensgen` and `ens` were examined in two domains: `gauss` and `mnist`.

The evaluation of the classifiers in the `gauss` domain illustrates the properties and weaknesses of each evaluated classifier. The evaluated classifiers perform well in the `ADD` scenario since all of them have one independent discriminator network per class; therefore, during T_2, the training on f_{L_2} does not influence f_{L_1}.

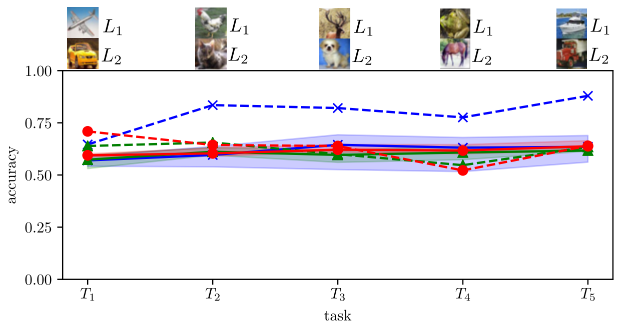
However, in the `EXP` scenario, L_1 is expanded at the iteration T_2, and thus influences the earlier learned subclass `R`. In Fig. 6b, the L_1 wrap catastrophically forgets the `R` cluster at the iteration T_2. The catastrophic forgetting does not occur for the `ensgen` and `ensgendel` classifiers, see Fig. 6f and Fig. 6j, which preserved the knowledge due to wrap-samples. The persistence is, however, harmful when the adaptation to the concept drift is needed.

In the `SEP` scenario, the `G` cluster switches the label from L_2 to L_1, and thus the L_2 wrap must uncover `G`, which the `ensgen` classifier cannot; see Fig. 6g. The `ens` classifier is able to forget `G`, but the forgetting is uncontrolled, and `ens` forgets the `R` cluster as well. In the `SEP` scenario, interestingly `ens` does not forget `R` because L_1 does not change during T_1, and it also forgets the `G` cluster as required. However, as it can be seen in Fig. 6d, the L_1 and L_2 wraps overlap at the `G` cluster as for some $x \in$ `G`, both $f_{L_1}^{T_2}(x)$ and $f_{L_2}^{T_2}(x)$ are below the threshold θ. The proposed classifier `ensgendel` performs well in all the scenarios since it has catastrophic forgetting persistence of `ensgen`, but it can also forget like `ens` during the concept drift.

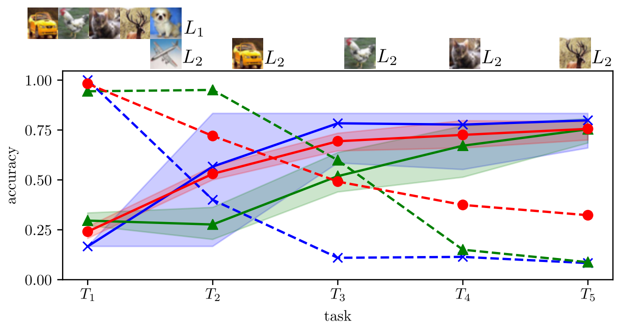
The basic scenarios were also examined within the `mnist` domain with two different sets of the basic scenarios, where we can observe a similar behavior of the classifiers as in the `gauss` domain, see Table 3. The `ensgen` classifier outper-



(a) `cifar ADD5`



(b) `cifar EXP5`



(c) `cifar SEP5`

Fig. 9 Accuracy evolution during five iterations of training in the complex scenarios `ADD5`, `EXP5`, and `SEP5` in the `cifar` domain. For each task, learners are exposed to labeled subclass samples (animal/vehicle images) as illustrated above each plot. The accuracy (full lines), averaged over five experimental runs, is evaluated w.r.t. to the target final state. Shaded areas indicate the accuracy maximum and minimum. The dashed line tracks the accuracy of a particular subclass whose label can be gradually forgotten during subsequent tasks. In `ADD5` and `EXP5` scenarios, we track the accuracy of airplanes (subclass 0) being labeled as L_1. In `SEP5`, we track the accuracy of dogs (subclass 5) being labeled as L_1.

forms `ens` in the `EXP` and `INC` scenarios, while in the `SEP` scenario, `ensgen` underperforms `ens`.

Similar trend appears in the complex scenarios `ADD5`, `EXP5`, and `SEP5`. In the `ADD5` scenario shown in Fig. 7a, all the classifiers perform well due to the architecture that separates weights of each class; thus, in each iteration, the trained weights are independent of the weights updated in the previous iterations. However, the separation of trained weights does not hold for the `EXP5` scenario, where the same class is expanded multiple times. Besides, the `ens` classifier underperforms in this scenario as shown in Fig. 7b. Finally, in the `SEP5` scenario, only `ensgendel` is able to retrain the subclasses trained at T_1; see Fig. 7c. Except for the `ADD` scenario in the `mnist197` assignment, the proposed classifier `ensgendel` outperforms both `ens` and `ensgen`, because it is able to selectively forget learned knowledge.

6.2 Measured Qualities of the Compared Algorithms

The proposed classifier was compared with the `GDM` and `CloGAN` learners in two domains: `mnist` and `cifar`.

6.2.1 The `mnist` Evaluation

The `mnist` domain provides high-dimensional images distinguished with high accuracy by all learners. Rather than generalization capabilities, the `mnist` scenarios evaluate the catastrophic forgetting robustness and concept drift adaptation.

The results of minimal scenarios reported in Table 8 show that all the evaluated algorithms can cover samples, i.e., learn without forgetting, although with different successfulness. The accuracy of the `GDM` learner in the `ADD` and `EXP` scenarios improves, and we can conclude that the `GMD` has a covering mechanism, which, however, performs worse than the covering mechanisms of `CloGAN` and `ensgendel`. The uncovering mechanism, the ability to selectively forget, is tested in the `INC` and `SEP` scenarios. Only the `GDM` learner appears to be not able to uncover, which is apparent in the `INC` scenario with `mnist197`, where the learner consistently does not improve. Surprisingly, the `CloGAN` learner appears to have the uncovering mechanism, and overall it has competitive results to the proposed `ensgendel`. `CloGAN` performs slightly better in covering, see the results for `ADD` and `EXP`, while the `ensgendel` learner performs slightly better in uncovering as can be seen in the `INC` and `SEP` scenarios.

A similar performance of `CloGAN` and `ensgendel` can be observed in the complex scenarios `ADD5`, `EXP5`, and `SEP5`. `ensgendel` has the best performance in the `ADD5` scenario, where the learners train a new class each layer. The good performance of `ensgendel` comes naturally from the learner's architecture, where each class updates a separate network, while the two other learners have to update the same network and thus risk forgetting. The ability to keep the previously trained knowledge can be directly observed in the tracked accuracy of subclasses shown as dashed lines in Fig. 8. The `GDM` seems to forget over a longer period as the learner gradually forgets the tracked subclass in all complex scenarios. `ensgendel` and `CloGAN` perform well in the `ADD5` and `EXP5` scenarios, and thus we can conclude that both continual learners have a robust covering mechanism.

The results of `SEP5` shown in Fig. 8c are harder to interpret since the overall model accuracy (full line) seems to have similar progress, yet the similar evolution has different causes. The `GDM` covers the tracked subclass (the accuracy of fives being labeled as L_1) perfectly during the tasks T_1 and T_2, as it is shown Fig. 8c by the green dashed line. However, the overall accuracy of the `GDM` (see full green line in Fig. 8c) does not improve at T_2 because the learner is unable to uncover, i.e., it cannot forget selectively. After the task T_2, the `GDM` forgets the knowledge similarly to uncontrolled forgetting observed in `ADD5` and `EXP5`. We assume that the forgetting of the `GDM` in `SEP5` is probably due to catastrophic (uncontrolled) forgetting. It is not the case of the `CloGAN` and `ensgendel` learners, where the accuracy improvement is monotone (see blue and full red lines in Fig. 8c). Both learners can gradually untrain the previously learned knowledge, even though they have robust covering mechanisms (as we see in `ADD5` and `EXP5` scenarios). Yet none of the learners performs perfectly in `SEP5`, the tracked subclass is gradually forgotten by all the learners. The selective untraining of samples from a class while keeping the rest of the class un-forgotten is still challenging.

The results of `CloGAN` and `ensgendel` are similar; furthermore, the `CloGAN`'s tracked accuracy in the challenging scenario `SEP5` is the highest among the evaluated learners. Therefore, we can assume that `CloGAN` has an uncovering mechanism as well. Since the generative model of `CloGAN` does not implement the negative sample loss maximization $\mathcal{L}^-$ (see (12)), we assume that the uncovering is implemented within the `CloGAN`'s clustered storage.

The rehearsal storage keeps samples representing the clusters. In a perfect case, the clustering algorithm identifies subclasses (digits of MNIST) as clusters. Then, the clustering storage would contain the representation of the subclasses and their labels. During the evaluation scenario, the clustered storage then incrementally adds subclasses with their labels and thus supports the covering mechanism. However, since the storage is limited, the stored subclasses must decrease the number of their representing samples to free the space for a new cluster representation. We hypothesize that the uniform decrease of the subclass-label representation size is what makes the uncovering behavior possible.

During the concept drift, one subclass, G, is represented in the memory with two different labels. For example, let (G, L_1) and (G, L_2) be the subclass-label assignment before and after the drift. The size of (G, L_1) representation is decreased; thus, if the size of the new (G, L_2) is large, the rehearsed classifier is more likely to assign L_2 labels to G samples. Therefore, `CloGAN` selectively forgets the (G, L_1) assignment.

However, such uncovering mechanism depends on the balance between the size of the old and new representations. Suppose the decreased size of the old (G, L_1) representations is still larger than the new representation of (G, L_2). In that case, the rehearsed classifier is more likely to classify the samples G as L_1 and thus ignore the concept drift. This memory balance problem is exposed by the `INC4` scenario, where the (G_{zeros}, L_1) representation is probably larger than the new (G_{zeros}, L_2), which results in the concept drift ignoration by `CloGAN` observed at T_2 in Fig. 8d. The tracked accuracy of (G_{zeros}, L_2) is correctly zero at T_1 for all algorithms, but at T_2, `CloGAN` does not adapt to the concept drift. The `GDM` learner adapts to the concept drift, but just for T_1 and only the proposed `ensgendel` learner can permanently adapt the concept drift.

The behaviors of `ensgendel` and `CloGAN` observed in Fig. 8d can be both correct under different paradigms. `ensgendel` is designed under the one-shot learning paradigm, where few examples can change the model abruptly, while `CloGAN` can adapt the concept drift gradually. These different approaches to continual learning are subtle yet critical in effect and should be researched in the future.

6.2.2 *The `cifar` Evaluation*

The `cifar` subclasses, in contrast to `mnist`, are discriminated by evaluated learners with low accuracy, see last column of Table 4. The low discrimination accuracy has impact on continual learning, where the overall results of all learners are worse in `cifar` than in `mnist`. The worse performance is evident from complex scenarios, see Fig. 9, where the `ensgendel` and `CloGAN` learners have worse accuracies than in the respective `mnist` scenarios as shown in Fig. 8. In fact, in the `cifar` scenarios, the learners show similar development of accuracy (full lines), unlike in the corresponding `mnist` scenarios. `CloGAN` and `ensgendel` perform similarly to the `GDM`; therefore, the uncovering mechanism does not provide any advantage as in the `mnist` scenarios.

The low scenario accuracy is a consequence of low discrimination accuracy for two reasons. First, the cover mechanism does not cover all given samples (violating (5)); therefore, the resulting wrap does not represent the given subclass. Second, since the wrap does not represent the subclass, the generated samples do not represent the subclass either. Thus, the knowledge is not transferred between tasks. The poor quality of generated samples can be indirectly observed in Fig. 9, where the tracked accuracies (dashed lines) in the `ADD5` scenario decrease (unlike in Fig. 8). The ability of the learner to discriminate between subclasses is important as the discrimination quality influences the covering mechanism.

The discrimination and reconstruction qualities can be improved by increasing the size of the training dataset. However, the point of continual learning is that the learners experience the dataset continually, i.e., the learners train on small datasets that might not represent the subclass. The underrepresentation is the essence of continual learning. All continual learners work under the assumption that a training set given at T does not represent the domain, class, or even the subclass. In the most extreme setup, the learner can be given just one training sample, which must be remembered. Without any prior knowledge, it is unreasonable to assume that the learner will generalize from that one sample.

The generalization then seems like an orthogonal quality to qualities of continual learning, where the overfitting can be interpreted as a perfect memory (or perfect cover). This dichotomy of generalization and continual learning capabilities is present in learners FearNET [9] and `GDM`, where the one subnetwork generalizes knowledge from other episodic subnetwork. Architectures combining the advantages of generalizing and continually learning neural networks should be more researched, as the generalization capability can improve the continual learning.

6.3 Limitations and Future Work

In this part, we discuss shortcomings of our theoretical description and proposed method, which we consider are worth addressing in future work.

6.3.1 Domain Dependent Hyperparameters

The first and foremost limitation of the proposed approach is the empirical tuning of the hyperparameters needed for each domain (such as the MNIST or Gauss domains). The domain itself can be subjected to the concept drift, and thus the tuned hyperparameters can become suboptimal. Aside from the VAE hyperparameters, there are three relevant hyperparameters of `ensgendel`, the maximum number of the wrap-samples N, threshold θ, and subtracting radius ε_0.

The parameter N influences the persistence of the wrap. The `ens` classifier with $N = 0$ catastrophically forgets while `ensgen` with $N = 200$ does not as it can be seen in the `EXP` scenario results in Table 3. A large value of N results in longer training because more samples have to be processed, but more importantly, the imbalance between class samples and wrap-samples can hurt the performance. If there is just one class sample to be covered (16), the larger N is, the lesser is the marginal error of the class-sample in (13) during the optimization, and the optimization of the class-sample will become slower. Moreover, if the number of classes is high, the unbalance between negative and positive samples can grow with the number of new classes since the ratio (without class samples) is N to N times the number of classes. Such an increasing unbalance between positive and negative samples makes scaling of the proposed classifier with new classes difficult.

Setting the threshold θ is also not straightforward. The parameter should be $\theta < 0.5$ to make it less likely that some sample is a part of the wrap in the initialized VAE because the LeCun initialization [18] initializes the VAE weights. Thus the samples are initially mapped $g(x) \approx 0.5$. However, the lesser θ is, the harder it is to learn the VAE to cover new samples.

Finally, the subtracting radius ε_0 defines the size of the ball $B_d(x, \varepsilon_0)$, which is subtracted from the sample during uncover (17). For high dimensions, setting the ball radius is not intuitive, as we can see for the results reported in Section 5.1.2, where we set $\varepsilon_0 = 4$, which in the 3D space would contain the whole unit cube, but in the 784D space, the volume of Euclidean 784D ball with the same radius is a tiny fraction of the unit hypercube volume. Moreover, the parameter ε_0 mainly depends on the VAE ability to cover the space that is a subset of $\mathcal{B}_d(x, \varepsilon_0)$. There is probably a technical limit on how small $\mathcal{B}_d(x, \varepsilon_0)$ can be, which is unknown to us. The dynamic adaptation of θ, ε_0, and N should be explored in the future.

6.3.2 Limited Capacity of Classifier

Although `gauss` class samples are classified correctly with the relaxed prediction (18), some class samples are not covered by their respective wraps. For example in Fig. 6i at T_1, there is one class-sample that is labeled as L_1 but it is not in the wrap $P_{L_1}^{T_1}$. Such outliers, both in the sense of the normal distribution and wrap, are probably caused by the bottleneck architecture, where the latent space of the dimensionality $\mathtt{dim}(\mathcal{Z}) = 2$ is mapped into the three-dimensional feature space. A latent space image is then a plane trained to fit the class samples of the three-dimensional normal distribution. Intuitively, it is possible to fit a finite number of such samples by plane, which is "carefully folded." [4] However, finding such folding requires prolonged training. With the limited number of training epochs, the VAE architecture imposes a restriction to the wrap, which should adapt to the unknown class structure. The structure of each class can differ in complexity; e.g., in the 3D space, the class can be zero to the three-dimensional manifold. Therefore, we believe that each VAE should adapt to the structure of its modeled class. In our future work, we aim to extend the proposed approach with a dynamic architecture adapting the structure of the modeled class.

[4] Non-intuitively, the famous Peano space-filling curve shows that there exists a continuous function that maps a continuous curve to a two-dimensional square that can be generalized to the n-dimensional cubes.

6.3.3 Non-probabilistic description of the continual learning

The continual learning description and analysis provided in Section 3 assumes crisp classes and the perfect cover/uncover modifications. However, `ensgendel` sometimes stopped at the maximum epoch E before it covered/uncovered given samples (e.g., the samples outside of the wrap in Fig. 6l). Moreover, the classes C_i can be so close to each other that maintaining the wraps disjoint is practically impossible. Even if the cover/uncover modifiers would be perfect, the class samples are noisy in practice. Thus without consideration of noise, the classifier would overfit (perfectly cover) the noisy data. Our presented theoretical framework for continual learning is insufficient to describe noisy datasets and imperfect training, and it is a subject of our future work.

7 Conclusion

In this paper, we describe continual learning, where we relate the concept drift and catastrophic forgetting problems in the proposed framework of trainable manifolds called wraps. The wraps are modified to imitate sampled classes with modifiers cover and uncover, which include and exclude given samples, respectively. The proposed framework is implemented as an ensemble of variational autoencoders, where each autoencoder represents the wrap, and the optimization of the autoencoder implements the cover and uncover modifiers. We have evaluated the algorithm with basic incremental learning scenarios and compared it with four continually learned classifiers demonstrating the robustness to catastrophic forgetting and utilization of the controlled forgetting needed for the concept drift adaptation. As for future work, we identify two challenges: (i) the influence of data topology on the continual learning parameters and (ii) the orthogonal relation between generalization and continual learning capabilities.

The herein proposed formal description can be applied both in practice and theory. In practice, the proposed continual learning framework provides tools for the design and evaluation of continual learners. Besides, the presented formalism can eventually lead to deepening theoretical understanding of learning in a continually observed environment.

Acknowledgements This work is an extension of the paper presented at the 13th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM 2019), where it received the Best Student Paper award.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Achille, A., Eccles, T., Matthey, L., Burgess, C.P., Watters, N., Lerchner, A., Higgins, I.: Life-long disentangled representation learning with cross-domain latent homologies. In: International Conference on Neural Information Processing Systems, pp. 9895–9905 (2018)
2. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: Anomaly detection using autoencoders in high performance computing systems. pp. 9428–9433 (2019). DOI 10.1609/aaai.v33i01.33019428
3. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys **41**(3), 15:1–15:58 (2009). DOI https://doi.org/10.1145/1541880.1541882
4. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10), 1517–1531 (2011). DOI https://doi.org/10.1109/TNN.2011.2160459
5. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences **3**(4), 128–135 (1999). DOI https://doi.org/10.1016/S1364-6613(99)01294-2
6. Gama, J.a., Žliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (2014). DOI https://doi.org/10.1145/2523813
7. Gepperth, A., Hammer, B.: Incremental learning algorithms and applications. In: European Symposium on Artificial Neural Networks (ESANN), pp. 357–368 (2016)
8. Hinton, G.E., McClelland, J.L., Rumelhart, D.E.: Distributed representations. In: D.E. Rumelhart, J.L. McClelland, C. PDP Research Group (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, pp. 77–109. MIT Press, Cambridge, MA, USA (1986)
9. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. In: International Conference on Learning Representations ICLR (2018)
10. Kemker, R., McClure, M., Abitino, A., Hayes, T.L., Kanan, C.: Measuring catastrophic forgetting in neural networks. In: S.A. McIlraith, K.Q. Weinberger (eds.) AAAI Conference on Artificial Intelligence, pp. 3390–3398 (2018)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, Conference Track Proceedings (2015). URL http://arxiv.org/abs/1412.6980
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR, Banff, AB, Canada, Conference Track Proceedings (2014). URL http://arxiv.org/abs/1312.6114
13. Kirkpatrick, J., et. al: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017). DOI https://doi.org/10.1073/pnas.1611835114
14. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal **37**(2), 233–243 (1991). DOI https://doi.org/10.1002/aic.690370209
15. Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M.: Ensemble learning for data stream analysis: A survey. Information Fusion **37**, 132–156 (2017). DOI https://doi.org/10.1016/j.inffus.2017.02.004
16. Krawczyk, B., Woźniak, M.: One-class classifiers with incremental learning and forgetting for data streams with concept drift. Soft Computing **19**(12), 3387–3400 (2015). DOI https://doi.org/10.1007/s00500-014-1492-5
17. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010). URL http://yann.lecun.com/exdb/mnist/. Cited on 2019-29-01
18. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012)

19. Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., Daz-Rodrguez, N.: Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. Information Fusion **58**, 52 – 68 (2020). DOI https://doi.org/10.1016/j.inffus.2019.12.004
20. Marchi, E., Vesperini, F., Squartini, S., Schuller, B.: Deep recurrent neural network-based autoencoders for acoustic novelty detection. Computational Intelligence and Neuroscience **2017** (2017). DOI https://doi.org/10.1155/8483
21. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. Neural networks : the official journal of the International Neural Network Society **15**(8-9), 10411058 (2002). DOI https://doi.org/10.1016/s0893-6080(02)00078-3
22. McInnes, L., Healy, J., Saul, N., Groberger, L.: Umap: Uniform manifold approximation and projection. Journal of Open Source Software **3**(29), 861 (2018). DOI https://doi.org/10.21105/joss.00861
23. Mustafa, A.M., Ayoade, G., Al-Naami, K., Khan, L., Hamlen, K.W., Thuraisingham, B., Araujo, F.: Unsupervised deep embedding for novel class detection over data stream. In: IEEE International Conference on Big Data, pp. 1830–1839 (2017). DOI https://doi.org/10.1109/BigData.2017.8258127
24. Nguyen, T.T.T., Nguyen, T.T., Liew, A.W.C., Wang, S.L.: Variational inference based bayes online classifiers with concept drift adaptation. Pattern Recognition **81**, 280 – 293 (2018). DOI https://doi.org/10.1016/j.patcog.2018.04.007
25. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: International Conference on Machine Learning (ICML), pp. 2642–2651 (2017)
26. Parisi, G.I., Tani, J., Weber, C., Wermter, S.: Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. Frontiers in Neurorobotics **12**, 78 (2018). DOI https://doi.org/10.3389/fnbot.2018.00078
27. Rios, A., Itti, L.: Closed-loop memory gan for continual learning. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 3332–3338 (2019). DOI https://doi.org/10.24963/ijcai.2019/462
28. Russakovsky, O., et. al: Imagenet large scale visual recognition challenge. Int J Comput Vis **115**, 211–252 (2015). DOI https://doi.org/10.1007/s11263-015-0816-y
29. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Advances in Neural Information Processing Systems, pp. 2990–2999 (2017)
30. Silver, D.L., Mercer, R.E.: The task rehearsal method of life-long learning: Overcoming impoverished data. In: R. Cohen, B. Spencer (eds.) Advances in Artificial Intelligence, pp. 90–101. Springer Berlin Heidelberg, Berlin, Heidelberg (2002). DOI https://doi.org/10.1007/3-540-47922-8_8
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). URL http://arxiv.org/abs/1409.1556
32. Szadkowski, R., Drchal, J., Faigl, J.: Basic evaluation scenarios for incrementally trained classifiers. In: International Conference on Artificial Neural Networks (ICANN), pp. 507–517 (2019). DOI https://doi.org/10.1007/978-3-030-30484-3_41
33. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(11), 1958–1970 (2008). DOI https://doi.org/10.1109/TPAMI.2008.128