Autoencoders Covering Space As A Life-Long Classifier

Rudolf Szadkowski, Jan Drchal, and Jan Faigl

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

Abstract. A life-long classifier that learns incrementally has many challenges such as concept drift, when the class changes in time, and catastrophic forgetting when the earlier learned knowledge is lost. Many successful connectionist solutions are based on an idea that new data are learned only in a part of a network that is relevant to the new data. We leverage this idea and propose a novel method for learning an ensemble of specialized autoencoders. We interpret autoencoders as manifolds that can be trained to contain or exclude given points from the input space. This manifold manipulation allows us to implement a classifier that can suppress catastrophic forgetting and adapt to concept drift. The proposed algorithm is evaluated on an incremental version of the XOR problem and on an incremental version of the MNIST classification where we achieved 0.9 accuracy which is a significant improvement over the previously published results.

Keywords: incremental learning, life-long learning, auto-encoder, catastrophic forgetting, concept drift

1 Introduction

Incremental learning is important for domains where the incoming data must be continually integrated into a classifier. Such classifier is expected to train and predict the incoming data as long as it is in operation; hence, a life-long classifier. In a life-long time scale, we expect that the target classes can change in time, where such change is called a 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 incremental learning [1].

In neural networks, concept representations are distributed throughout the network weights [2]. 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 life-long learning, these changes can accumulate, and concepts might get forgotten [1]. The existing approaches to the concept drift in

neural networks are based on preferring updates only a part of the network that is relevant to the new data increments [3, 4]. Probably, the most straightforward implementation of this idea is to use an ensemble of predictors [5].

Our proposed method is based on an ensemble of classifying autoencoders. An autoencoder is a neural network trained to approximate an identity transformation of the input. Training of the autoencoder is unsupervised because the loss, also called the reconstruction error, is defined as a difference between the input and its transformed image. This reconstruction error of the autoencoder has an alternative in anomaly detection. An anomalous input is detected by the autoencoder when the reconstruction error exceeds a given threshold [6, 7]. A related application of the same method is the novel class detection [8, 9] where the task is to detect unknown classes. In both applications, the anomaly and novel class detection, the autoencoder divides the input space into two regions according to the reconstruction error. Here, we call the region with low reconstruction error θ-wrap, where θ represents the maximum reconstruction error threshold. Hence, each θ-wrap can be understood as a manifold which covers most of the samples on which the corresponding autoencoder was trained.

In this paper, we take advantage of θ-wrap interpretation of the autoencoder and present an incremental supervised training algorithm that suppresses the 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 θ-wrap. We present two methods to train θ-wrap that are refered COVER and UNCOVER. The methods train the θ-wrap to cover or uncover the given set of points while suppressing the catastrophic forgetting. The UNCOVER method is used for the case of the concept drift when a class starts intersecting with θ-wrap unrelated to that class. The conflicting θ-wrap is then trained to uncover the intersection with the UNCOVER method. Both UNCOVER and COVER methods are the building blocks for the incremental training algorithm for the whole autoencoder ensemble. For each given sample batch, the algorithm uncovers given samples by θ-wraps to which the samples do not belong, and then it covers the samples by their respective θ-wrap.

The proposed algorithm is designed under relaxing assumptions where classes are "crisp" manifolds that are sampled without noise and UNCOVER and COVER methods always work perfectly. We examine the robustness of the proposed algorithm in two dimensions, incrementally learning the XOR dataset in which we simulate the concept drift. We also test how the ensemble of classifying autoencoders withstand incremental training on the MNIST dataset.

2 Related Work

Autoencoder ensembles have been employed for detecting outliers or anomalies [10, 11] where all autoencoders are trained to classify the same task, but each autoencoder is trained to its specific task [12]. As a new task appears, the algorithm builds a new autoencoder on this task in order to capture and store its representative information. This storage helps the proposed algorithm to suppress the catastrophic forgetting that is enforced structurally, where the concepts are stored in their subnetworks, and thus the new concept does not

influence the previous ones. This structural solution to catastrophic forgetting is also used in our proposed algorithm. We additionally introduce a resampling algorithm that suppresses catastrophic forgetting within each subnetwork. The structural separations of concepts are not the only solution to catastrophic forgetting. Recently proposed Elastic Weight Consolidation (EWC) algorithm [3] selectively freezes the neural network weights that are important to the previously learned task/class. The EWC is compared to other approaches tackling the catastrophic forgetting in [13], where the authors report the EWC provides the best results. However, neither the algorithm [12] nor [3] can successfully deal with the concept drift problem where classes change in time. In particular, when a class is suddenly relabeled. In such a case, a classifier needs to implement a forgetting mechanism. We present such a mechanism in the following section.

3 Analysis

Let $X = (0,1)^N$ be the N-dimensional input space and let $C_i \subset X$ be the i-th class, where $i \in \{1 \dots M\}$. We assume that classes C_i form mutually disjoint manifolds. The manifold C_i is unknown, but at the time t, we get a finite point-set of class-samples $S_i^t \subset C_i$. We can only work with the given class-samples S_i^t at the time t and, moreover, any class C_i can change at the time ($C_i^t \neq C_i^{t'}$) due to the concept drift. We aim to find such a classifier $F^t : X \to \{1 \dots M\}$ at t that

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

We propose to use M trainable manifolds $P_i(\theta) \subset X$, which we call θ-wraps, to mimic their respective classes. Ideally, each class C_i is wrapped by its respective θ-wrap $P_i(\theta)$, while all wraps are mutually disjoint:

$$\forall i \in \{1 \dots M\} : C_i \subset P_i(\theta), \tag{2}$$

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

We propose the following straightforward algorithm to reach this desired state. At each iteration t, the given class-sample set $S_i^t \subset C_i^t$ is firstly uncovered by all θ-wraps $P_j^t(\theta); j \neq i$ and then covered by $P_i^t(\theta)$ θ-wrap (see Alg. 2). After the training, we say P *covers* S and P *uncovers* S that means $S \subset P$ and $P \cap S = \emptyset$, respectively. The classification can be then realized by the function $F(\boldsymbol{x}) = \arg\max_i [\![\boldsymbol{x} \in P_i(\theta)]\!]$. The proposed implementation of θ-wrap $P_i(\theta)$ and its training algorithm follows.

4 Method

We propose to implement θ-wraps with autoencoders, where θ-wrap $P_i(\theta)$ is defined by its underlying autoencoder $g_i : X \to X$. An autoencoder is usually trained to reconstruct some target subset $A \subset X$, i.e., $\forall \boldsymbol{a} \in A : g_i^*(\boldsymbol{a}) = \boldsymbol{a}$. The reconstruction error is measured by the Euclidean distance $\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x})) = ||\boldsymbol{x} - g(\boldsymbol{x})||$. θ-wrap corresponding to the i-th autoencoder is defined as $P_i(\theta) =$

Algorithm 1 Train autoencoder with positive and negative samples.

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

function TRAIN(g, S_+, S_-, θ)
  $g^0 \leftarrow g$
  for $e = 1$ **to** E **do**
    $S_-^e \leftarrow \{\boldsymbol{s} | \mathcal{L}(\boldsymbol{s}, g^{e-1}(\boldsymbol{s})) \leq \theta; \boldsymbol{s} \in S_-\}$ ▷ Filters S_- points with low $\mathcal{L}$.
    $S_+^e \leftarrow \{\boldsymbol{s} | \mathcal{L}(\boldsymbol{s}, g^{e-1}(\boldsymbol{s})) \geq \theta; \boldsymbol{s} \in S_+\}$ ▷ Filters S_+ points with high $\mathcal{L}$.
    $\epsilon^e \leftarrow \mathcal{J}(g^{e-1}, S_+^e, S_-^e)$
    $g^e \leftarrow$ `gradient-descent`(g^{e-1}, ϵ^e)
    if $\forall \boldsymbol{s} \in S_+ : \mathcal{L}(\boldsymbol{s}, g^e(\boldsymbol{s})) < \theta$ **and** $\forall \boldsymbol{s} \in S_- : \mathcal{L}(\boldsymbol{s}, g^e(\boldsymbol{s})) > \theta$ **then**
      break
    end if
  end for
  $g \leftarrow g^e$
end function

$\{x | \mathcal{L}(\boldsymbol{x}, g_i(\boldsymbol{x})) < \theta, \forall x \in X\}$, i.e., it is a manifold where all distances between points $\boldsymbol{x}$ and images $g_i(\boldsymbol{x})$ are smaller than some threshold $\theta \in (0, \mathcal{L}_{\max})$ where $\mathcal{L}_{\max}$ is the maximal reachable distance from the middle of the $(0,1)^N$ space

$$\mathcal{L}_{\max} = \min_{\boldsymbol{x}} \sup_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{N}/2; \boldsymbol{x}, \boldsymbol{y} \in (0,1)^N. \tag{4}$$

θ-wrap $P_i(\theta)$ is trained by training the related autoencoder g_i using the function TRAIN(g_i, S_+, S_-) depicted in Alg. 1, where the finite sets $S_+, S_- \subset X$ are called the positive and negative samples, respectively. Ideally, the positive samples should reside inside θ-wrap $P_i(\theta)$ while the negative samples should be outside:

$$g_i^t \leftarrow \text{TRAIN}(g_i^{t-1}, S_+^t, S_-^t, \theta), \tag{5}$$

where for samples S_+^t, S_-^t, and i-th θ-wrap $P_i^t(\theta)$ it holds that

$$(S_+^t \subset P_i^t(\theta)) \wedge (S_-^t \cap P_i^t(\theta)) = \emptyset, \tag{6}$$

and where t is the iteration index. The cost function $\mathcal{J}(g, S_+, S_-) \in \mathbb{R}^+$ in Alg. 1 gives the cost of the autoencoder g w.r.t. the positive and negative sets S_+, S_-

$$\mathcal{J}(g, S_+, S_-) = \frac{1}{|S_+|} \sum_{\boldsymbol{s} \in S_+} \mathcal{L}(\boldsymbol{s}, g(\boldsymbol{s}))^2 + \frac{1}{|S_-|} \sum_{\boldsymbol{s} \in S_-} (\mathcal{L}_{\max} - \mathcal{L}(\boldsymbol{s}, g(\boldsymbol{s}))^2. \tag{7}$$

The class is then predicted by

$$F(\boldsymbol{x}) = \arg\min_i \mathcal{L}(\boldsymbol{x}, g_i(\boldsymbol{x})). \tag{8}$$

Let SAMP(g_i) be a function that gives a set of random wrap-samples: it creates a set $\hat{X}_i$ of uniform random samples of X, and keeps only those that are in θ-wrap, SAMP$(g_i) = \{\boldsymbol{s} | \mathcal{L}(\boldsymbol{s}, g_i(\boldsymbol{s})) < \theta, \boldsymbol{s} \in \hat{X}_i\}$, where $|\hat{X}_i| = S_{\max}$. Ideally,

to satisfy the properties (2) and (3), it would be sufficient to set $\forall i : g_i^* \leftarrow \textsc{train}(g_i^t, S_i, \hat{P}_i^t, \theta)$, where $S_i, \hat{P}_i^t$ are large enough finite point-sets sampled from C_i and $\cup_{j \neq i} \textsc{samp}(g_j)$, respectively. However, in our case, the class-sample set $S_i^t \subset C_i$ can contain just a few samples (one being the worst case scenario). If the class-sample set S_i^t does not represent its class C_i well (e.g., it contains just one sample) the catastrophic forgetting may take place. In the context of θ-wraps, the catastrophic forgetting is an event when after training θ-wrap $P_i^{t-1}(\theta)$, new θ-wrap $P_i^t(\theta)$ ceases to cover a part of C_i that was covered by its predecessor, i.e., $(C_i \cap P_i^{t-1}(\theta)) \not\subset P_i^t(\theta)$. θ-wrap $P_i^{t-1}(\theta)$ is not preserved by Alg. 1 because the gradient descent method optimizes the cost that is calculated only from the given samples S_-^t, S_+^t at the time t. We propose to address the problem by adding wrap-samples $\textsc{samp}(g^{t-1})$ to the positive samples and define the method $g_i^t \leftarrow \textsc{cover}(g_i^{t-1}, S_i^t, \theta)$ as

$$\textsc{cover}(g_i^{t-1}, S_i^t, \theta) \coloneqq \textsc{train}(g_i^{t-1}, S_i^t \cup \textsc{samp}(g_i^{t-1}), \bigcup_{j \neq i} \textsc{samp}(g_j^{t-1}), \theta). \quad (9)$$

As discussed above, the algorithm needs to implement a forgetting mechanism to deal with the concept drift. An abrupt concept drift of class C_i is a discrete event when a class abruptly changes $C_i^t \neq C_i^{t-1}$ [13]. The class change that hurts the performance of the classifier happens when C_i^t intersects with θ-wrap of other class $P_j^{t-1}(\theta), j \neq i$. To keep the properties (2) and (3), the θ-wrap $P_j^{t-1}(\theta)$ must first uncover the intersection $P_j^{t-1}(\theta) \cap C_i^t; j \neq i$ before the θ-wrap $P_i^{t-1}(\theta)$ covers this intersection. Let class-sample set $\hat{S}_i^t$ be a subset of such conflicting intersection $P_j^{t-1}(\theta) \cap C_i^t; j \neq i$, then, before we cover $\hat{S}_i^t$ with $P_i^t(\theta)$ using (9) we must train $P_j^t(\theta)$ to uncover $\hat{S}_i^t$. We define the method $g_j^t \leftarrow \textsc{uncover}(g_j^{t-1}, \hat{S}_i^t, \theta)$, that trains g_j to uncover $\hat{S}_i^t$, as

$$\textsc{uncover}(g_j^{t-1}, \hat{S}_i^t, \theta) \coloneqq \textsc{train}(g_j^{t-1}, \textsc{samp}(g_j^{t-1}) - \mathcal{B}(\hat{S}_i^t, \varepsilon), \bigcup_{k \neq j} \textsc{samp}(g_k^{t-1}) \cup \hat{S}_i^t, \theta), \quad (10)$$

Algorithm 2 Update autoencoders with the labeled samples D.

Variables $\{g_i\}$: collection of M autoencoders, $i \in \{1 \dots M\}$;
$D = \{(\boldsymbol{s}, k)\}$: set of labeled samples where k indicates class of sample $\boldsymbol{s}$;
θ: threshold; $\mathcal{L}$: metric function;
Result $\{g_i\}$: updated autoencoders;

```
function UPDATE({g_i}, D, θ)
    for i = 1 to M do
        S_i ← {s | k = i; (s, k) ∈ D}
        for j = 1 to M where j ≠ i and ∃s ∈ S_i : L(s, g_j(s)) < θ do
            g_j ← UNCOVER(g_j, S_i, θ)                    ▷ See (10).
        end for
        g_i ← COVER(g_i, S_i, θ)                          ▷ See (9).
    end for
end function
```

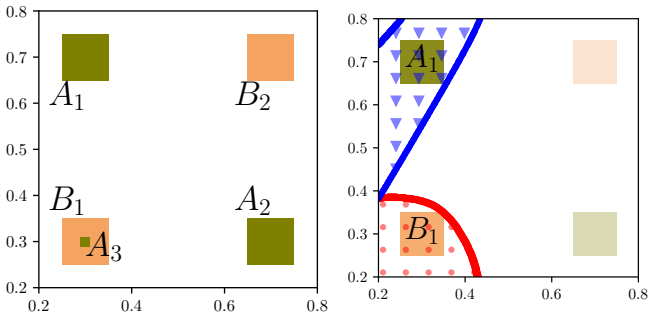
where $\mathcal{B}(\hat{S}_i^t, \varepsilon) = \bigcup_{\boldsymbol{s} \in \hat{S}_i^t} \{\boldsymbol{x} | \mathcal{L}(\boldsymbol{s}, \boldsymbol{x}) < \varepsilon; \boldsymbol{x} \in X\}$ is ε-neighbourhood of the whole set $\hat{S}_i^t$. With cost function (7), sample set covering (9), and uncovering (10) Alg. 1 and Alg. 2 are completely defined.

5 Results

The proposed approach has been empirically evaluated on two datasets. First, in Sec. 5.1 we study how the algorithm handles both the catastrophic forgetting and concept drift in the simple XOR problem. Then, in Sec. 5.2 we deploy the method in MNIST dataset [14], where the proposed approach demonstrates surprising benefits of the ensemble of almost-independent autoencoders. The utilized autoencoder is composed of four hidden ReLU layers with the sizes 800, 400, 400, and 800. The input and output layers have N units each where the output layer is composed of sigmoid units. The autoencoder is overcomplete for experiments in Sec. 5.1, but cost (7) prevents the autoencoder to become an identity function. ADAM with default parameters [15] is utilized as the `gradient-descent` method in Alg. 1. The subtraction parameter from (10) is set to $\varepsilon = 0.01$. All hyperparameters were selected empirically.

5.1 Incremental Training of Binary Classification

The principle of the proposed method listed in Alg. 2 is demonstrated in three classification scenarios where two autoencoders are trained to classify classes $C_1, C_2 \subset (0, 1)^2$. The input space dimension is $N = 2$, training hyperparameters are set to $\theta = \mathcal{L}_{\max}/2$, and the maximum number of epochs $E = 1000$. The two classes C_1, C_2 are composed of squares A_1, A_2, A_3, B_1, and B_2[1] depicted in Fig. 1a. Each scenario starts with $C_1 = A_1$ covered by $P_1(\theta)$ and $C_2 = B_1$ covered



(a) (b)

Fig. 1. (a) Squares used in the classification scenarios. Green squares are part of C_1 and orange squares are part of C_2, where $A_3 \subset B_1$. (b) Initially, the classifier is learned only on A_1 and B_1 squares. A_1 is covered by $P_1(\theta)$ (blue) and B_1 is covered by $P_2(\theta)$ (red). θ-wraps are visualized by taking samples from θ-wrap.

[1] The centers of 0.1×0.1 large squares A_1, A_2, B_1, and $B2$ are (0.3,0.7), (0.7,0.3), (0.3,0.3), and (0.7,0.7), respectively. A_3 is 0.02×0.02 large centered at (0.3,0.3).

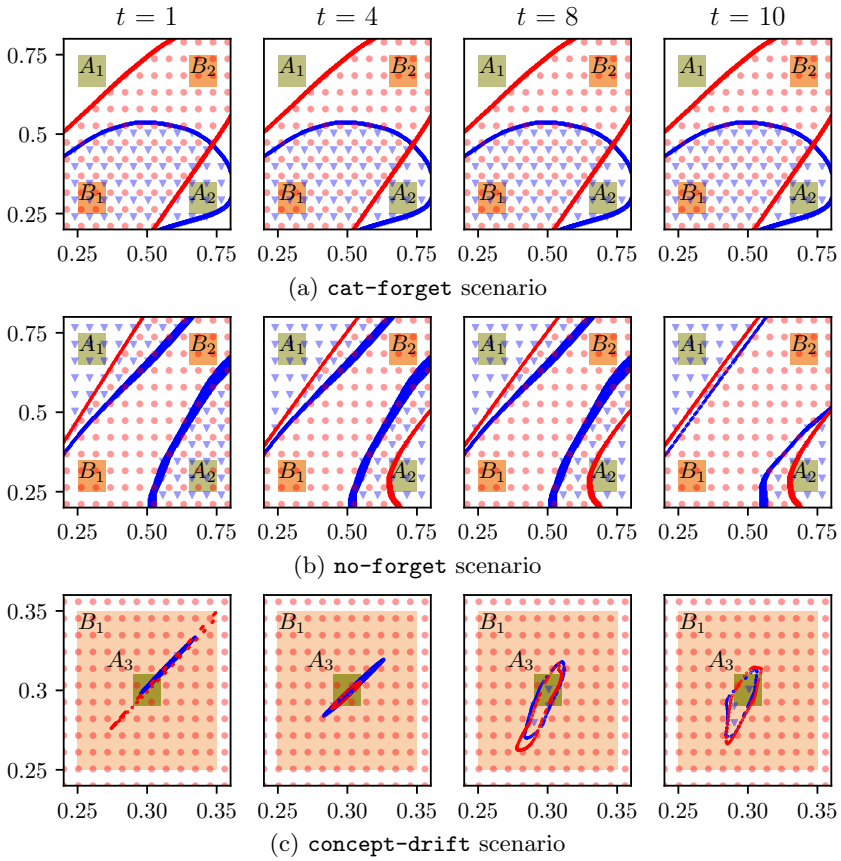
by $P_2(\theta)$ (see Fig. 1b). The `cat-forget` scenario demonstrates the catastrophic forgetting where the classes are extended to $C_1 = A_1 \cup A_2$ and $C_2 = B_1 \cup B_2$. The class-sample batches D^t of size 100 are sampled only from $A_2 \cup B_2$ which can result into "forgetting" the boxes A_1 and B_1. We set the number of the maximum wrap-samples $S_{\max} = 0$ to demonstrate the catastrophic forgetting. The `no-forget` scenario has the same setting except $S_{\max} = 5000$. The difference between the scenarios can be seen in Fig. 2a and Fig. 2b. Finally, in the `concept-drift` scenario, the new classes are $C_1 = A_1 \cup A_3$ and $C_2 = B_1 - A_3$ to study how the method handles the concept drift. The class-sample batches D^t are of the size one, and they are sampled only from A_3, and thus P_1 tries to cover each class-sample while P_2 tries to uncover it, see Fig. 2c. Results of the statistical evaluation are listed in Tab. 1. Accuracies were averaged from five 10-iteration long runs and evaluated on classes $C_1 = A_1 \cup A_2$ and $C_2 = B_1 \cup B_2$, except for `concept-drift` which was evaluated on $C_1 = A_3$ and $C_2 = B_1 - A_3$.

Table 1. Classifier evaluation on XOR test data (new samples taken from respective classes). Accuracy averages taken from 5 runs, with standard deviation to show the algorithm stability.

Scenario	`cat-forget`	`no-forget`	`concept-drift`
accuracy	0.62 ± 0.11	0.94 ± 0.09	0.78 ± 0.09

5.2 Catastrophic Forgetting Evaluation on MNIST

In the second scenario, we perform the evaluation introduced in [13] which measures the classifier performance on the incremental learning of the MNIST dataset. The dataset D is divided into $D_{\texttt{even}}$ and $D_{\texttt{odd}}$ containing even and odd number characters, respectively. The algorithm trains on $D_{\texttt{even}}$ first and then on $D_{\texttt{odd}}$ (with no access to $D_{\texttt{even}}$). During the training, the overall accuracy on the test set is measured. Each class-sample of MNIST dataset corresponds to a point in $[0, 1]^{784}$ space which is scaled to $[0.15, 0.85]^{784}$ space to avoid plateaus during training sigmoid output units. The training hyperparameters are set to $\theta = \mathcal{L}_{\max}/10$, and the maximal epoch number $E = 1000$. The maximum wrap-samples size is $S_{\max} = 0$ because the sampling method is not suitable for such a high-dimensional input, and thus there is no mechanism preventing catastrophic forgetting except for the fact that the autoencoders train almost independently on each other (see Alg. 2). There are $M = 10$ autoencoders training in their respective classes. Each iteration, we sample a random batch of the size 100 from the current dataset D. For the first $T_1 = 300$ iterations, only the odd numbers $D_{\texttt{odd}}$ are trained, and the next $T_2 = 300$ iterations, only the even numbers $D_{\texttt{even}}$ are trained. Here, we differ from [13] where the training parameters are $T_1 = T_2 = 2500$, and thus the algorithms are exposed to catastrophic forgetting for more iterations. However, it depends whether one iteration is equivalent to one parametric update because our algorithm performed 66.82 updates per iteration on average (calculated from six independent runs). Therefore about 40 092 updates are performed in the total, and thus the proposed algorithm withstands



(a) `cat-forget` scenario

(b) `no-forget` scenario

(c) `concept-drift` scenario

Fig. 2. Cover evolution in catastrophic forgetting, normal and concept drift scenarios. Initially (for $t = 0$), $P_1(\theta)$ (blue) and $P_2(\theta)$ (red) cover A_1 and B_1, respectively, see Fig. 1b. In (a) and (b), the goal is to cover A_2 and B_2 without uncovering A_1 and B_1. We train on batches containing 100 class-samples of A_2, B_2. (a) $S_{\max}$ is zero which leads to uncovered A_1 and B_1 being part of the both θ-wraps. (b) $S_{\max} = 5000$ and both θ-wraps managed to keep their respective squares A_1, B_1 covered. (c) A concept drift happened when the new classes became $C_1 = A_1 \cup A_3$ and $C_2 = B_1 - A_3$. We can see the detail of A_3 (green rectangle) which we want to uncover by $P_2(\theta)$ and cover by $P_1(\theta)$. We train on batches containing one class-sample taken from A_3. During ten iterations, $P_2(\theta)$ (red) formed a cavity which is covered by $P_1(\theta)$ at $t = 10$. Both θ-wraps are visualized with thick border and interior filled with sparse markers.

the catastrophic forgetting for even more updates than, e.g., EWC [3] with one update per iteration and thus updated just 5000 times. An example of the accuracy evolution is shown in Fig. 3.

The achieved accuracy on the MNIST testing dataset for six experimental runs of 600 iterations is 0.90 ± 0.02. For a rough comparison with existing approaches, the best result reported in [13] on even-odd numbers learning task is the accuracy 0.64, which was achieved using the EWC algorithm [3].

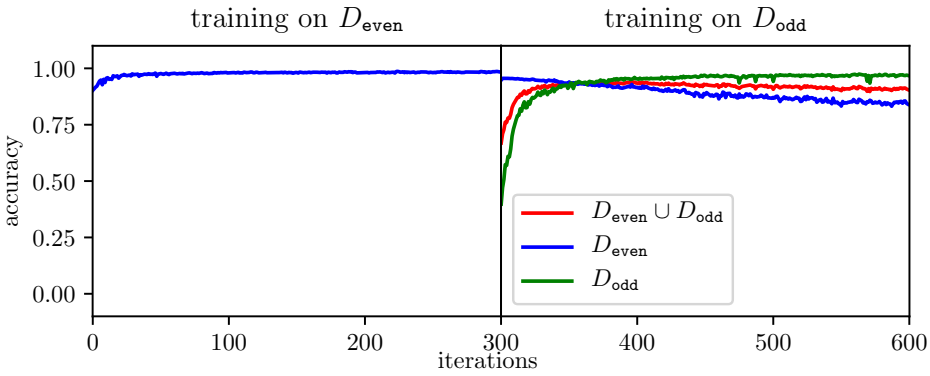


Fig. 3. Evolution of the accuracy during training on the MNIST dataset with even $D_{\texttt{even}}$ and odd $D_{\texttt{odd}}$ numbers. For each iteration, 100 random class-samples is taken from either $D_{\texttt{even}}$ or $D_{\texttt{odd}}$ and trained. On average, the UPDATE function processes 59.87 epochs at each iteration of Alg. 2.

5.3 Discussion

The idealized property (3) does not perfectly hold as can be seen in Fig. 2, where θ-wraps intersects for NO-FORGET and CONCEPT-DRIFT scenarios. The intersection is caused by TRAIN method (Alg. 1) which does not always ensure the property (6), where the positive samples should reside inside while the negative samples outside of the trained θ-wrap. θ-wrap then loses some positive samples or keeps covered some negative samples which can lead to intersections with other θ-wraps. However, these intersections seem to become smaller with increasing iterations, hinting the robustness of the proposed algorithm.

The bottleneck of the proposed algorithm is the SAMP method, which does not scale well with the increasing dimensionality[2]. For our future work, we aim to explore possibilities on how to implement SAMP method more efficiently. However, high-dimensionality might be the reason why the proposed classifier has surprisingly good results. Most images in MNIST have high contrast, and thus the samples are clustered close to the corners of the 784-dimensional unit hypercube. These clusters are covered by autoencoders with θ-wraps, i.e., manifolds of the Euclidean space. Euclidean distances between corners of the high-dimensional hypercube can be quite large (the maximum is $\sqrt{784} = 28$), θ-wraps are probably less likely to reach across such distances and cover other classes.

6 Conclusion

In this paper, we provide an analysis of the properties of classifying autoencoders in the context of the concept drift and catastrophic forgetting, where the autoencoders are represented as trainable manifolds called θ-wraps. With the θ-wrap representation and classes defined as subspaces of the input space, we describe the training, catastrophic forgetting, and concept drift using set operations. This allows us to design an algorithm that trains the ensemble of

[2] It is apparent from Sec. 5.2, as sampling a point that lies in θ-wrap is roughly equivalent to getting an actual image of digit by randomly sampling 28×28 pixels.

autoencoders to incrementally cover their respective classes with their respective θ-wraps. Even though the algorithm is designed under relaxing assumptions, the results support its feasibility and relatively robustness for two-dimensional input space, and the proposed approach produces competitive results on incremental learning of the MNIST dataset.

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

References

1. A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Symposium on Artificial Neural Networks (ESANN)*, 2016, pp. 357–368.
2. G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, ch. Distributed Representations, pp. 77–109.
3. J. Kirkpatrick and et. al, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
4. Z. Li and D. Hoiem, "Learning without forgetting," *CoRR*, vol. abs/1606.09282, 2016.
5. B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.
6. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, 2009.
7. A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Anomaly detection using autoencoders in high performance computing systems," *CoRR*, vol. abs/1811.05269, 2018.
8. E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, "Deep recurrent neural network-based autoencoders for acoustic novelty detection," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.
9. A. M. Mustafa, G. Ayoade, K. Al-Naami, L. Khan, K. W. Hamlen, B. Thuraisingham, and F. Araujo, "Unsupervised deep embedding for novel class detection over data stream," in *IEEE International Conference on Big Data*, 2017, pp. 1830–1839.
10. J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *SIAM International Conference on Data Mining*, 2017, pp. 90–98.
11. Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *CoRR*, vol. abs/1802.09089, 2018.
12. A. R. Triki, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," *CoRR*, vol. abs/1704.01920, 2017.
13. B. Pflb, A. Gepperth, S. Abdullah, and A. Kilian, "Catastrophic forgetting: Still a problem for dnns," in *International Conference on Artificial Neural Networks (ICANN)*, 2018, pp. 487–497.
14. Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010, cited on 2019-29-01. [Online]. Available: http://yann.lecun.com/exdb/mnist/
15. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.