Basic Evaluation Scenarios for Incrementally Trained Classifiers

Rudolf Szadkowski[0000−0003−4075−116X], Jan Drchal[0000−0003−0466−275X], and Jan Faigl[0000−0002−6193−0792]

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. Evaluation of incremental classification algorithms is a complex task because there are many aspects to evaluate. Besides the aspects such as accuracy and generalization that are usually evaluated in the context of classification, we also need to assess how the algorithm handles two main challenges of the incremental learning: the concept drift and the catastrophic forgetting. However, only catastrophic forgetting is evaluated by the current methodology, where the classifier is evaluated in two scenarios for class addition and expansion. We generalize the methodology by proposing two new scenarios of incremental learning for class inclusion and separation that evaluate the handling of the concept drift. We demonstrate the proposed methodology on the evaluation of three different incremental classifiers, where we show that the proposed methodology provides a more complete and finer evaluation.

Keywords: incremental learning, classification, catastrophic forgetting, concept drift, methodology

1 Introduction

Evaluation of incremental learning algorithms is a complex task since there are many possible evaluation scenarios. Each scenario can evaluate multiple aspects of the incremental algorithm such as accuracy convergence, robustness against catastrophic forgetting (CF), or concept drift (CD) handling. Testing multiple aspects at once, however, does not usually help in the identification of the particular issues of the examined learning algorithm. Therefore, we need some basic evaluating scenarios, each addressing a specific aspect of the incremental learning algorithm, to tackle one issue at the time.

An incrementally trained classifier is a classifier that is being trained on consecutive tasks. Each task, the classifier is fed with labeled samples which cannot be stored but must be integrated into the classifier during the training. Such multiple training over the long period has two main challenges that are called *concept drift* and the *catastrophic forgetting* [2]. The symptom of catastrophic

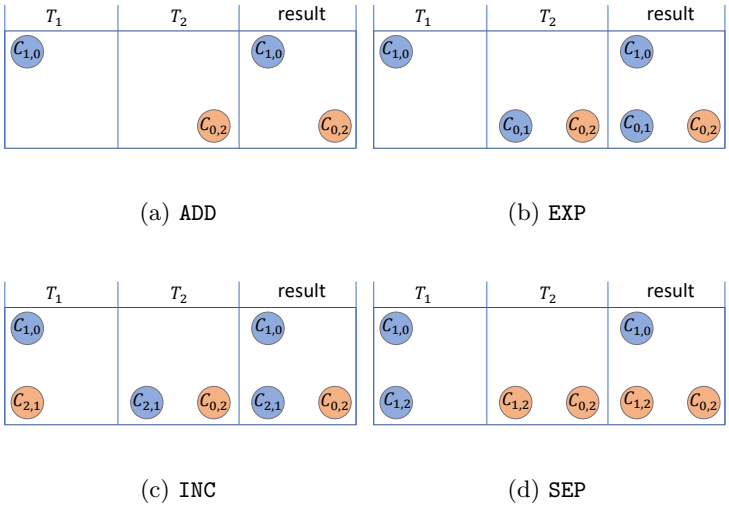
forgetting is a decrease of classifier performance, where the performance is measured on the previously trained tasks [4]. Such evaluation of the performance decrease is used as a metric for many proposed incremental algorithms [3, 6, 9], where authors introduce *incremental class learning* and *data permutation* scenarios [4].

In the incremental class learning scenario, the classifier is learning a different class each task, while in the data permutation scenario, the classifier learns on the same classes but with shuffled feature-vector components. Both scenarios examine how well is the classifier able to aggregate the new data without forgetting the learned class distributions, but it does not examine the algorithm adaptability to the concept drift. The concept drift is a consequence of a non-stationary environment where the class distribution changes in time [10]. The distribution change detection is still an open problem, and its solutions are dependent on the type of the concept drift [1,13]. A scenario where the concept drift is evident has to be designed to evaluate the concept drift handling on different incremental algorithms. An example of such evident concept drift is when the previously presented sample is presented again but with a different label [7,13]. The change of label requires the classifier to un-train the old label on the sample, and then train the new label. Such an operation should be tested during the evaluation of an incremental algorithm.

The evaluation methodology for incrementally trained classifiers observed from various papers [3, 6, 7, 12] can be generally divided into three steps:

1. Test the basic properties of the classifier (e.g., accuracy, generalization, how fast it converges) within just one task.
2. Test the behavior of the classifier in the *minimal incremental classification problem*, where we have two tasks during which we train the classifier on given samples of two classes.
3. Test scalability by adding more classes and by increasing the number of tasks.

The main contribution of this paper relates to the second step for which we introduce basic evaluation scenarios. We show that there are 2^9 possible scenarios that can be inferred from the basic presuppositions for the minimal incremental classification problem. In the context of the incremental algorithm evaluation, we filter symmetric and redundant scenarios to get four basic evaluation scenarios. We propose the following basic evaluation scenarios (depicted in Fig. 1): class addition (`ADD`), expansion (`EXP`), inclusion (`INC`), and separation (`SEP`). Scenarios `ADD` and `EXP` correspond to *incremental class learning* and *data permutation* [4], respectively, which are used to evaluate how the algorithm handles the catastrophic forgetting. The new scenarios `INC` and `SEP` introduce the label change described in various concept drift cases [7,13]. The benefit of using these basic scenarios is that they are easy to construct with existing datasets (e.g., MNIST [8]) and the classifier can be considered as a black box. Furthermore, by evaluating the classifier with each basic scenario, we can analyze its properties separately. The proposed evaluation is demonstrated on multiple incremental classifiers.



(a) `ADD` (b) `EXP`

(c) `INC` (d) `SEP`

Fig. 1. Illustration of four basic evaluation scenarios for evaluation of the catastrophic forgetting and concept drift handling. Each scenario has two consecutive tasks T_1, T_2 during which the classifier trains on the presented samples. The third right-most part of each sub-figure depicts the target state of the classes after the classifier is trained on T_1 and T_2. The blue and orange disks represent sample clusters labeled with the first l_1 and the second l_2 label, respectively. (a) All scenarios extends the `ADD` scenario which starts at T_1 with training on the cluster of the samples labeled with l_1 (blue) and ends with training on the cluster labeled with l_2 (orange). (b) In the `EXP` at T_2, we expand the class l_1 with new samples. (c) In the `INC`, the samples labeled as l_2 at T_1 are relabeled as l_1 at T_2. (d) In the `SEP`, a part of the class l_1 is relabeled as l_2 at T_2. Each disk contains the base set name defined in Sec. 2.

The rest of the paper is organized as follows. The formal definition and inference of scenarios are provided in Sec. 2. The evaluated incremental algorithms are introduced in Sec. 3 and the evaluation results are reported in Sec. 4 with a detailed discussion and interpretation of the evaluation results in Sec. 4.1. The paper is concluded in Sec. 5.

2 Basic Scenarios of Incremental Classification

The incrementally trained classifier is being trained during consecutive tasks T_1, T_2, T_3, ..., where for each consecutive task T_i, the classifier F^{T_i} is trained on batch $D^{T_i} = \{(\boldsymbol{x}^j, l^j)\}_{1 \le j \le m}$ of m labeled samples. Samples $\boldsymbol{x} \in X$ are labeled by one of n labels $l \in L = \{L_1, \ldots, L_n\}$. We are interested in the *minimal incremental classification problem* where we have just two tasks T_1, T_2 and two labels L_1, L_2. Having just two labels, during each task T_i, each sample $\boldsymbol{x} \in X$ is in one of three *states* $S = \{S_1, S_2, S_0\}$; the sample $\boldsymbol{x}$ is either

- S_1: presented with the first label,
- S_2: or the second label,
- S_0: or not presented.

Having just two tasks, each sample $\boldsymbol{x} \in X$ has *tuple of states* $(s, s') \in S^2$, where s and s' are states of $\boldsymbol{x}$ during T_1 and T_2, respectively. Let a *base set* $C_{s,s'} \subset X$ be a set of samples with the state tuple (s, s'). All nine base sets are pairwise disjoint, and their union gives X (see Fig. 2).

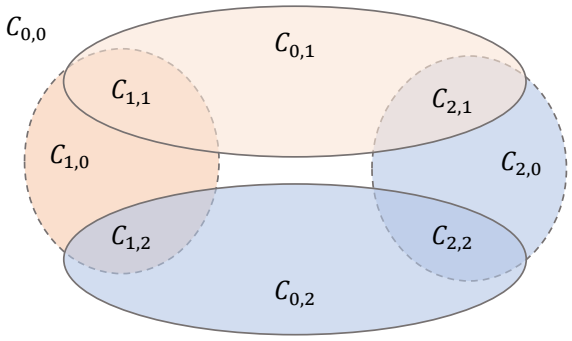


Fig. 2. Illustration of possible intersections of labeled sets that are presented during tasks T_1 (dashed border) and task T_2 (full line border). Sets labeled by the first and second labels l_1, l_2 are distinguished by the blue and orange color, respectively. For example, we can see that the base sets $C_{1,1}$ and $C_{2,2}$ are presented to the classifier at T_1 and T_2 while the base sets $C_{1,0}$ and $C_{2,0}$ are presented only at T_1. Note that this is a scenario where all base sets are non-empty.

Each base set $C_{s,s'}$ is either empty or non-empty. The base sets $C_{1,1}$, $C_{1,2}$, $C_{2,1}$, and $C_{2,2}$ are sets that contain samples that are sampled twice (in T_1 and then in T_2). If samples are taken from a continuous probability distribution, the base sets with samples presented twice are always empty (the probability of sampling the same point twice is zero). However, in the context of the classifier evaluation, we present to a classifier the same sample $\boldsymbol{x}$ with different labels l, l' to examine whether the classifier can change its model in such a way, that after the task T_2, the classifier labels $\boldsymbol{x}$ as l'. Thus, in the evaluation scenarios, the base sets $C_{1,1}$, $C_{1,2}$, $C_{2,1}$, and $C_{2,2}$ can be non-empty.

Let *scenario* be an assignment function $r : S^2 \to \{0, 1\}$, where $r(s, s') = 1$ if $C_{s,s'}$ is non-empty else $r(s, s') = 0$. There are 2^9 scenarios, which we prune with the following constraints. First, we consider that labels are symmetric and assume that the base set $C_{0,0}$ is always non-empty. Second, in the context of the incremental algorithm evaluation, we want to measure how well the classifier can classify samples trained only in T_1 despite changes in T_2. The base set $C_{1,0}$ (or $C_{2,0}$) is a set of samples that are trained only in T_1, and the base sets with samples that change labels in T_2 are $C_{0,2}, C_{0,1}, C_{2,1}$, and $C_{1,2}$. The combination

of non-empty $C_{1,0}$ with each of the above mentioned four base sets gives us four basic evaluation scenarios. Additionally, $C_{0,2}$ is non-empty in all four basic scenarios to ensure that after the task T_2, there are always two classes to classify. The basic evaluation scenarios are listed in Tab. 1 and illustrated in Fig. 1.

Table 1. Basic evaluation scenarios: class addition (`ADD`), expansion (`EXP`), inclusion (`INC`), and separation (`SEP`). Each presented scenario evaluates certain feature of incremental learning algorithm which handles catastrophic forgetting (CF) or concept drift (CD).

Scenario	Non-empty sets	Tested feature	Possible problem
`ADD`	$C_{1,0}, C_{0,2}$	Adding a new class	CF
`EXP`	$C_{1,0}, C_{0,2}, C_{0,1}$	Expanding a class	CF
`INC`	$C_{1,0}, C_{0,2}, C_{2,1}$	Untraining a class	CF, CD
`SEP`	$C_{1,0}, C_{0,2}, C_{1,2}$	Untraining part of a class	CF, CD

Thus the evaluation of a binary classifier $F^{T_i} : X \to \{1, 2\}$ is the examination of its performance for T_1 and T_2 in all basic scenarios, i.e.,

$$F^{T_1}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } x \in \{C_{1,0} \cup C_{1,1} \cup C_{1,2}\} \\ 2 & \text{if } x \in \{C_{2,0} \cup C_{2,2} \cup C_{2,1}\} \end{cases}, \tag{1}$$

$$F^{T_2}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } x \in \{C_{1,0} \cup C_{1,1} \cup C_{2,1} \cup C_{0,1}\} \\ 2 & \text{if } x \in \{C_{2,0} \cup C_{2,2} \cup C_{1,2} \cup C_{0,2}\} \end{cases}. \tag{2}$$

3 Incremental Classifiers

We introduce three incremental classifiers: `ENS`, `ENSGEN`, and `ENSGENDEL`, to present the proposed evaluation using the minimal scenarios listed in Tab. 1. The `ENS` is an ensemble of two multilayer perceptrons (MLPs), where each MLP is trained to classify its respective class. Each MLP is trained independently in the `ADD` scenario, and thus it should be robust to catastrophic forgetting. The `ENS` is trained each task with Alg. 1 and the label prediction is made by $F(\boldsymbol{x}) = \arg\max_{l \in L} f_l(\boldsymbol{x})$.

The `ENSGEN` is an extension of `ENS` where we generate (replay) samples from autoencoder. The technique where we generate samples to prevent catastrophic forgetting is called the *memory replay* [11]. Many implementations of the memory replay (i.e., sample generation) use autoencoders, e.g., [14]. For each label $l \in \{L_1, L_2\}$, we have an autoencoder that is composed of the encoder $e_l : X \to Z$ and decoder $d_l : Z \to X$, where we call $Z = [0, 1]^N$ the *latent space*. The autoencoder $d_l \circ e_l$ is trained on samples labeled with l (along with the classifier f_l), and during the next task, we let the autoencoder to generate samples that resemble the samples from the previous task. The sample generation method

GENERATE(d, f) is defined as

$$\text{GENERATE}(d, f) := \{d(\boldsymbol{z}) | \boldsymbol{z} \in \text{SAMPLE\_UNIFORMLY}(Z); f(d(\boldsymbol{z})) > 0.9\}, \quad (3)$$

where SAMPLE_UNIFORMLY(Z) gets M random samples of the latent space, which are first decoded by decoder d and then filtered by the classifier f. The update method of the `ENS` can be used also for the `ENSGEN` for which Alg. 1 is modified as follows. Line 4 and Line 5 of Alg. 1 are modified to

$$A_{L_1} \leftarrow A^o_{L_1} \cup \text{GENERATE}(d_{L_1}, f_{L_1}), \quad (4)$$
$$A_{L_2} \leftarrow A^o_{L_2} \cup \text{GENERATE}(d_{L_2}, f_{L_2}). \quad (5)$$

The condition in the minimization iterator (Line 7, Alg. 1) is changed to

$$\exists \boldsymbol{x} \in A_l : ||\boldsymbol{x} - d_l(e_l(\boldsymbol{x}))|| > \theta, \quad (6)$$

where we prioritize training of the autoencoder $d_l \circ e_l$ over the classifier f_l for two reasons. First, preliminary experiments showed that the classifier f_l is easier to train. Second, if the autoencoder is not well trained, then the GENERATE(d, f) method returns a small amount of samples in (4) and (5). The optimization of the autoencoder $d_l \circ e_l$ is implemented by adding

$$\mathcal{J}'_l \leftarrow \frac{1}{|A_l|} \sum_{\boldsymbol{x} \in A_l} ||\boldsymbol{x} - d_l(e_l(\boldsymbol{x}))||, \quad (7)$$
$$d_l, e_l \leftarrow \text{MINIMIZE}(\mathcal{J}_l, d_l, e_l). \quad (8)$$

after Line 9 of Alg. 1.

On the other hand, the `ENSGEN` classifier can fail to relabel some of the samples from task T_1 in the `INC` and `SEP` scenarios. The samples that need to be relabeled ($C_{2,1}$ and $C_{1,2}$) can be within the cluster of the generated samples (see (4) and (5)). The cost functions of the autoencoder $e \circ d$ and classifier f are both smooth (differentiable), thus by minimizing the cost function over the set of samples A_l (see Line 8, in Alg. 1 and (7)), we also minimize the cost in the *close neighborhood*[1] of samples A_l. Therefore, if two sets of samples share the close neighborhood but have conflicting minimization objectives (e.g., the two sets have different labels), the minimization process will slow down. We propose an extension of `ENSGEN`: the `ENSGENDEL` classifier, that "subtracts" the close neighborhood of the new samples $A^o_{l'}$ from the generated samples of label l. Hence, it cannot happen the classifier f_l will be trained to label $A^o_{l'}$ as l. The modification of `ENSGEN` is to replace Line 4 and Line 5 of Alg. 1 to

$$A_{L_1} \leftarrow A^o_{L_1} \cup \{\boldsymbol{x} \in \text{GENERATE}(d_{L_1}, f_{L_1}) | \forall \boldsymbol{a} \in A^o_{L_2} : ||\boldsymbol{x} - \boldsymbol{a}|| > \varepsilon\}, \quad (9)$$
$$A_{L_2} \leftarrow A^o_{L_2} \cup \{\boldsymbol{x} \in \text{GENERATE}(d_{L_2}, f_{L_2}) | \forall \boldsymbol{a} \in A^o_{L_1} : ||\boldsymbol{x} - \boldsymbol{a}|| > \varepsilon\}, \quad (10)$$

where ε is the minimum distance between the generated and new samples.

[1] In a metric space X, a neighborhood of the point $\boldsymbol{x}$ is defined as a ball of the radius r with $\boldsymbol{x}$ in the center: $\mathcal{B}_d(\boldsymbol{x}, r) = \{\boldsymbol{y} | d(\boldsymbol{x}, \boldsymbol{y}) < r; \boldsymbol{y} \in X\}$, where d is a metric function. A close neighborhood is a neighborhood with a very small radius r.

Algorithm 1 Update method for ensemble classifier `ENS`. The algorithm can be used as an update method for the `ENSGEN` or `ENSGENDEL` using the modifications described in Sec. 3.

Variables During one task, the algorithm gets a dataset batch $D = \{(\boldsymbol{x}^1, l^1), (\boldsymbol{x}^2, l^2), \dots\}$, where $\boldsymbol{x} \in X$ and $l \in \{L_1, L_2\}$. The dataset is used to train two binary classifiers f_{L_1}, f_{L_2}, where $f_l : X \to [0, 1]$. Parameters M and θ are the maximum epoch and threshold, respectively.

Result f_{L_1}, f_{L_2}: updated binary classifiers

function UPDATE(D, θ, f_{L_1}, f_{L_2})
  $A_{L_1}^o \leftarrow \{\boldsymbol{x} | l = L_1; (\boldsymbol{x}, l) \in D\}$ ▷ Separate samples by the label into two sets
  $A_{L_2}^o \leftarrow \{\boldsymbol{x} | l = L_2; (\boldsymbol{x}, l) \in D\}$
  $A_{L_1} \leftarrow A_{L_1}^o$ ▷ See Eqs. 4, 5 and Eqs. 9, 10
  $A_{L_2} \leftarrow A_{L_2}^o$
  for (l, l') **in** $\{(L_1, L_2), (L_2, L_1)\}$ **do**
    for M **times if** $\exists \boldsymbol{x} \in A_l : (1 - f_l(s)) > \theta$ **do**
      $$\mathcal{J}_l \leftarrow -\frac{1}{|A_{L_1}|+|A_{L_2}|}\left(\sum_{\boldsymbol{x}\in A_l} \ln(f_l(\boldsymbol{x})) + \sum_{\boldsymbol{x}\in A_{l'}} \ln(1 - f_l(\boldsymbol{x}))\right)$$
      $f_l \leftarrow$ MINIMIZE($\mathcal{J}_l, f_l$)
    end for
  end for
end function

4 Results

In this section, we report how the proposed evaluation scenarios (see Sec. 2) can improve the analysis of incremental classifiers that is demonstrated on the classifiers described in Sec. 3. Moreover, to set a baseline, we also train a single MLP classifier `SNG` with the layer sizes 728-500-500-2 with the softmax layer and cross-entropy loss function. The `ENS` classifier has $\theta = 0.1$, $M = 1000$, and two binary MLPs, each has the layer sizes 728-500-250-125-1. The `ENSGEN` and `ENSGENDEL` classifiers have $\theta = 7$, $M = 10$, $\varepsilon = 0.1$, and two autoencoders, each composed of encoder and decoder with the layer sizes 784-500-200-8 and 8-200-500-784, respectively. A rectifier is used as the activation function for all hidden layers. The output layers of the encoder and MLP in `ENS` have a sigmoid activation function. All neural networks are trained with Adam [5] with the learning rate set to 0.0001. All the hyperparameters were found empirically.

Different scenarios are created by using the MNIST [8] dataset which has roughly 7000 samples per MNIST class (zero, one,..., nine), where each MNIST sample is a 28×28 image of a digit. The dataset is divided into a training and testing set with the ratio 6 to 1. We construct scenarios by assigning some of the MNIST classes to the base sets $C_{i,j}$. Two assignment configurations are described in Tab. 2: the 021 assignment which is made from easily distinguishable digits (zeroes, twos, and ones), and the 197 assignment which contains digits that are harder to distinguish (ones, nines, and sevens). The classifiers are trained on scenarios created from the training set, and the evaluation is calculated on the

scenarios created from the testing set. The results are shown in Tab. 3 and Tab. 4.

Table 2. Configurations of the assignment of the MNIST classes to base sets that are used in the basic evaluation scenarios. The MNIST dataset has ten classes represented by the number 0 to 9. Since base sets $C_{0,1}$, $C_{2,1}$, and $C_{1,2}$ never appear together in the same scenario, the same MNIST class can be assigned to them.

Assignment	$C_{1,0}$	$C_{0,2}$	$C_{0,1}$	$C_{2,1}$	$C_{1,2}$
021	0	2	1	1	1
197	1	9	7	7	7

Table 3. Accuracy of classifiers after being trained on both tasks. The classifier accuracy is evaluated with respect to testing datasets for tasks T_1 and T_2 (see first and third columns in Fig. 1). The column T_2 shows the overall performance of the classifier, where the higher accuracy is always better. The low values in the T_1 column indicate the catastrophic forgetting since the classifier performs worse on the previous task. However, for the `INC` and `SEP` scenarios, values lower than one are expected because a classifier needs to forget (relabel) some of the previously presented samples.

Assignment	Classifier	`ADD`		`EXP`		`INC`		`SEP`	
		T_1	T_2	T_1	T_2	T_1	T_2	T_1	T_2
021	`SNG`	0.00	0.51	0.00	0.68	0.01	0.68	0.00	0.68
	`ENS`	0.69	0.84	0.00	0.68	0.00	0.68	0.22	0.48
	`ENSGEN`	0.99	0.98	0.96	0.98	0.46	0.98	0.58	0.90
	`ENSGENDEL`	0.99	0.98	0.95	0.97	0.45	0.98	0.47	0.98
197	`SNG`	0.00	0.47	0.07	0.62	0.54	0.97	0.00	0.64
	`ENS`	1.00	0.88	0.99	0.99	0.91	0.69	0.99	0.53
	`ENSGEN`	1.00	0.99	0.99	0.97	0.54	0.97	0.72	0.85
	`ENSGENDEL`	1.00	0.99	0.99	0.97	0.54	0.97	0.70	0.87

4.1 Discussion

The overall accuracy of the classifiers can be compared from the results in Tab. 3, where the regular evaluation on the `ADD` and `EXP` scenarios [3,6,7,12] is extended with the proposed `INC` and `SEP` scenarios. In the 197 assignment of the `INC` scenario, we can see that the `ENS` classifier is unable to relabel some of the previously presented samples (the accuracy in the T_1 column should be at most roughly 0.5, but it is 0.91 in the case of the `ENS` classifier). Such a low performance

Table 4. The accuracy of each evaluated classifier calculated after the task T_2 for each respective base set $C_{i,j}$. The intuitive interpretation of the table values is as follows: the $C_{1,0}$ column represents the ratio of $C_{1,0}$ that the classifier is able to "remember" after T_2, the $C_{2,1}$ and $C_{1,2}$ columns represent the ratio of the respective base set that the classifier was able to relabel during T_2, the $C_{0,1}$ and $C_{0,2}$ columns are just accuracies evaluated on the respective base set.

Assignment	Classifier	ADD		EXP			INC			SEP		
		$C_{1,0}$	$C_{0,2}$	$C_{1,0}$	$C_{0,2}$	$C_{0,1}$	$C_{1,0}$	$C_{0,2}$	$C_{2,1}$	$C_{1,0}$	$C_{0,2}$	$C_{1,2}$
021	SNG	0.00	1.00	0.01	0.99	0.99	0.01	0.99	0.98	0.00	1.00	1.00
	ENS	0.70	0.98	0.00	0.99	0.99	0.00	0.99	1.00	0.03	0.78	0.60
	ENSGEN	1.00	0.98	0.96	1.00	0.99	0.99	1.00	0.98	1.00	0.97	0.78
	ENSGENDEL	1.00	0.98	0.99	0.99	0.98	0.98	0.99	0.99	1.00	0.98	0.98
197	SNG	0.00	1.00	0.07	0.97	0.91	1.00	0.96	0.95	0.00	1.00	1.00
	ENS	1.00	0.76	1.00	0.99	0.98	0.99	0.98	0.09	1.00	0.55	0.00
	ENSGEN	1.00	0.99	1.00	0.97	0.97	1.00	0.97	0.96	1.00	0.97	0.57
	ENSGENDEL	1.00	0.99	0.99	0.99	0.96	0.99	0.98	0.94	1.00	0.98	0.63

at relabeling is most likely caused by the similarity of the digits used in the 197 assignment (ones, nines, and sevens) because in the 021 assignment, the ENS classifier can relabel the previously presented samples (the ENS has 0 accuracy in the T_1 column of the INC scenario). With the SEP scenario, we can distinguish the performance of the ENSGEN and ENSGENDEL classifiers, which have almost identical results in all other scenarios. Thus we gain more information about the evaluated classifiers by evaluation with the proposed scenarios SEP and INC.

The regular evaluation listed in Tab. 3 is good for a comparison of multiple classifiers. However, for a finer analysis of the classifiers, we propose to evaluate the accuracy on each base set, like it is shown in Tab. 4, where the column $C_{1,0}$ shows how well the classifier "remembers" the base set $C_{1,0}$ after the task T_2. The accuracies in the column $C_{1,0}$ show that the classifiers ENSGEN and ENSGENDEL remember the previously learned samples almost perfectly. Other interesting columns are $C_{2,1}$ and $C_{1,2}$, which show how well the classifier relabel the previously trained samples. In assignment 021 of the SEP scenario, the ENSGEN classifier has been able to relabel only 0.78 of samples, while ENSGENDEL has been able to relabel almost all of them. Such explicit information is lost in the regular overall evaluation (see Tab. 3) because the regular evaluation is evaluated over multiple base sets.

The results in assignment 197 are worse than results in assignment 021 in most of the cases. From this difference, we can draw a lesson that it is important to try more assignments, as it is pointed out in [12] because each MNIST class (or any other class of different dataset) has different qualities. The quantity is another aspect to consider: in this paper, the base sets are of equal cardinality (roughly). Scenarios with the base sets that have different cardinalities could evaluate the classifier robustness against unbalanced data. Thus, it is good practice to use basic evaluation scenarios with multiple different assignments for a thorough examination of the incremental classifier.

5 Conclusion

In this paper, we propose a generalization of the current methodology for incremental classifier evaluation by proposing four basic evaluation scenarios: class addition, expansion, inclusion, and separation. Three incremental classifiers are presented to demonstrate the methodology within the proposed evaluation scenarios. Each classifier has been evaluated with the proposed methodology, and we assess how well the classifier handles the catastrophic forgetting and the concept drift issues. Moreover, the proposed generalization allows us to design a finer evaluation that can test particular aspects of incremental learning; such are remembering the previously trained samples or selective relabeling of the previously learned samples. Such a detailed methodology for incremental learning evaluation should improve the development of incremental classifiers, and therefore, researchers are encouraged to consider it in their developments.

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

References

1. Freund, Y., Mansour, Y.: Learning under persistent drift. In: Ben-David, S. (ed.) Computational Learning Theory. pp. 109–118. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
2. Gepperth, A., Hammer, B.: Incremental learning algorithms and applications. In: European Symposium on Artificial Neural Networks (ESANN). pp. 357–368 (2016)
3. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. arXiv e-prints arXiv:1312.6211 (2013)
4. Kemker, R., Abitino, A., McClure, M., Kanan, C.: Measuring catastrophic forgetting in neural networks. CoRR **abs/1708.02072** (2017)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)
6. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017)
7. Lane, T., Brodley, C.E.: Approaches to online learning and concept drift for user identification in computer security. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. pp. 259–263. KDD'98, AAAI Press (1998)
8. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), `http://yann.lecun.com/exdb/mnist/`, cited on 2019-29-01
9. Lee, S., Kim, J., Ha, J., Zhang, B.: Overcoming catastrophic forgetting by incremental moment matching. CoRR **abs/1703.08475** (2017)
10. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodrguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521 – 530 (2012)
11. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019)

12. Pfülb, B., Gepperth, A., Abdullah, S., Kilian, A.: Catastrophic forgetting: Still a problem for dnns. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) Artificial Neural Networks and Machine Learning – ICANN 2018. pp. 487–497. Springer International Publishing, Cham (2018)
13. Wang, K., Zhou, S., Fu, C.A., Yu, J.X.: Mining changes of classification by correspondence tracing. In: Proceedings of the SIAM International Conference on Data Mining. pp. 95–106. SIAM (2003)
14. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Zhang, Z., Fu, Y.: Incremental classifier learning with generative adversarial networks. CoRR **abs/1802.00853** (2018)