Combining Offline and Online Classifiers for Life-long Learning

Lydia Fischer*†, Barbara Hammer* and Heiko Wersing†
*Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany
†HONDA Research Institute Europe, Carl-Legien-Str. 30, 63073 Offenbach am Main, Germany

***Abstract*—One of the greatest challenges of life-long learning architectures is how to efficiently and reliably cope with the stability-plasticity dilemma. We propose an extension of a flexible system combining a static offline classifier and an incremental online classifier that is well suited for life-long learning scenarios. The pre-trained offline classifier preserves ground knowledge that should be respected during training, while the online classifier enables learning of new or specific information encountered during use. The combination is realised by a dynamic classifier selection strategy based on confidences of both ingredients. We report exemplary results of this architecture for the case of learning vector quantization (LVQ) for several data sets, thereby including an extensive comparison to alternative state of the art algorithms for incremental learning such as incremental generalised LVQ and the support vector machine.**

I. Introduction

Several phenomena have caused a rapidly increasing interest in life-long learning technology [1]: digital data are more and more available in almost all areas of daily life, including distributed sensors in households, intelligent smartphones and wearable technologies, or advances as regards industry 4.0. At the same time, electronic systems get smart and there is an increasing trend for their personalisation such that the capability of systems to learn during their lifetime becomes an essential property. Application scenarios where quite some research effort concerns life-long learning include emerging areas like autonomous robotics/driving, or assistive systems [2], [3], [4].

Online learning crucially depends on the capability of a system of learning new concepts while preserving already known information over its lifetime [1]. This setting violates usual assumptions of classical learning scenarios as formalised e. g. in PAC learning [5]: here often i. i. d. data are assumed, further, many optimisation schemes depend on a formalisation of the problem using all training data. Thus, the most common setting in machine learning is still a batch learning scenario. In contrast, humans always learn in an online setting, while interacting with their environment. The humans keep relevant knowledge from the past and use it to support prospective learning. It is notable that humans embody powerful strategies for life-long learning which ensure that relevant basic concepts are stable while important changes can rapidly be integrated into their knowledge. These two goals, encountering stability for ground concepts and, at the same time, flexibility to deal with a changing environment or new concepts, constitute conflicting requirements which are difficult to realise in technical systems. Often the so called *catastrophic forgetting effect* (CFE) can be noticed, i. e. there is no guarantee that ground knowledge (e. g. security of a system) is respected unless it is explicitly hard-coded in the decisions of the system [6], [7].

Within machine learning, the family of incremental learning approaches recently caused a lot of attention, see e. g. [8]. A typical problem in this context is *concept-drift* which occurs in dynamically changing environments [9]. Here we consider the occurrence of new classes and changing data distributions within this concept. We do not assume data points that change their labels. There exist methods which directly try to inspect when concept-drift occurs [10], while we do this implicitly. There are at least two ways how to deal with concept drift: one can rely on a flexible model changing with its environment, facing the risk of forgetting known concepts [11], [12], [13] or one can rely on a combination of different architectures, enabling a more flexible control about which information to keep and which one to adapt [14], [15]. There exist several approaches using more than two classifiers, ensembles, to gather the diversity of the seen data in divers models enabling good generalisation performance and dealing with different types of concept-drift [10], [15], [16]. We address the question how to efficiently realise a combination of different architectures, and how their performance scales with respect to their ability to reliably deal with both, basic known concepts and newly faced events in comparison to alternative model designs.

A lately proposed hybrid architecture [17], [18] combines two complementary classifiers via a dynamic classifier selection [19], [20], [21], [22]. One classifier is completely static during the application time after it is pre-trained with offline available data (offline model). The second classifier starts from scratch and it learns incrementally during its whole lifetime. The online model of the architecture allows to deal with concept-drift: if the known data distribution changes or new classes occur the architecture follows this changes using the online model, while the static model conserves the already learned knowledge. This proposal is conceptually simple, however, it uses a heuristic method for classifier combination based on error counting.

This article presents an extension of this hybrid architecture, a detailed analysis, and comparisons to alternative designs. While keeping the basic structure, we substitute the single parts of the architecture by more fundamental choices which rely on the notion of the certainty of the two parts for efficient classifier selection. Combining a static and a highly flexible incremental classifier in such a way seems promising with respect to the system's stability and plasticity. On an abstract level one can interpret the static model as kind of a long-term memory and the online learning part serves as a kind of short-term memory. The advantage of this architecture is the higher robustness against noise and CFE than pure incremental approaches without a backup model, as we will show in several benchmarks.

We will demonstrate the capability of this architecture for life-long learning tasks based on learning vector quantisation (LVQ) models as classification schemes [23]. LVQ techniques have the advantage of combining a well understood theoretical foundation with an intuitive and interpretable classification scheme. Further, being a prototype method, it constitutes one of the rare techniques which inherently provides a sufficient statistics of the gathered information in terms of the prototype locations, a fact which has already been used to extend prototype-based models to efficient online schemes for massive data which are capable of dealing with non i. i. d. distributions in a natural way [24]. We will rely on powerful formalisations of LVQ learning in terms of cost functions, such as proposed in the approaches [25], [26], [27], and we will use its recent extensions to incremental learning like the incremental, online LVQ (ioLVQ) [28]. Notably, state of the art LVQ classifiers usually integrate the powerful concept of metric learning [29], [30], [31]. For a comparison we refer to state of the art incremental learners, such as incremental support vector machines (iSVM)[1] [12], [32]. Note that the latter, unlike LVQ classifiers, often require extensive storage space due to a growing number of support vectors. LVQ, depending on a prototypical data representation rather than a representation of class boundaries, usually relies on much smaller resources as we will demonstrate in a number of experiments.

At first we briefly describe the assumed scenario, followed by the introduction of the used LVQ scheme. Then we present the extended architecture combining an online and offline model. Later on we will explain its parts, especially specifying how decisions on how to combine and how to train can be based on certainty values. Finally, we compare the architecture to purely incremental LVQ schemes as well as iSVM as regards their accuracy and model complexity, using several benchmarks.

II. Description of the Scenario

We assume a scenario with two training phases. The first phase is offline and it assumes complete available training data such that multiple passes trough the data are possible. A second phase is assumed to be online, i. d. training data points are available one after another and they can be used for training only once. Further, we assume training data with original class labels for both phases. For test data the original class labels are used to evaluate the performance of the approaches only.

Such a scenario is interesting for systems delivered with a pre-trained classification model which can be personalised/adapted for the users needs, e. g. via direct user interaction. A user can specify new labelled training data, e. g. instances of already known classes or completely new ones.

III. Learning Vector Quantization

We use the learning technique proposed in [23] that has gained much attention recently in the context of big data and interpretable models due to its flexibility and intuitive classification scheme, see e. g. [33], [34], [35], [36], [37], [38], [39], [40], [41], [42]. Essentially, it offers an efficient way for prototype-based data classification. Assume υ training data points $\mathbf{x}_i$ with class label y_i such that $(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, \ldots, C\}$. A LVQ classifier consists of a set of prototypes $W = \{\mathbf{w}_j \in \mathbb{R}^n\}_{j=1}^k$ equipped with class labels $c(\mathbf{w}_j) \in \{1, \ldots, C\}$. A given point $\mathbf{x}_i$ is classified according to the label of the closest prototype, the best matching unit (BMU), as measured in the squared Euclidean distance $\|\mathbf{x} - \mathbf{w}\|^2$ or variants thereof.

[1]iSVM code: https://github.com/diehl/Incremental-SVM-Learning-in-MATLAB

Given training data, the generalised LVQ (GLVQ) [25] is a learning technique of LVQ that is based on a cost function. It optimises the location of prototypes with step size ε_w by means of a stochastic gradient descent on its cost function

$$E = \sum_{i=1}^{\upsilon} \Phi(\underbrace{(d^+(\mathbf{x}_i) - d^-(\mathbf{x}_i))/(d^+(\mathbf{x}_i) + d^-(\mathbf{x}_i))}_{=:\mu(\mathbf{x}_i)}) \quad (1)$$

while performing several training epochs through the data. The function Φ is monotonic increasing, e. g. the logistic function. $d^\pm$ is the distance of the data point $\mathbf{x}_i$ to the closest prototype $\mathbf{w}^\pm$ of the correct/incorrect class. This cost function strongly correlates to the classification error because a data point is classified correctly iff the nominator of the cost function is negative. Note that the relative similarity (RelSim)

$$\text{RelSim}(\mathbf{x}) := -\mu(\mathbf{x}) = (d^-(\mathbf{x}) - d^+(\mathbf{x}))/(d^-(\mathbf{x}) + d^+(\mathbf{x}))$$

relates to the summands of the GLVQ costs and can be interpreted as a certainty of the classification (for this reason multiplied by -1): The values $\text{RelSim}(\mathbf{x})$ are contained in the interval $(-1, 1)$ and values near 0 refer to high uncertainty, high values near 1 refer to high certainty, and negative ones refer to a wrong classification since $d^+ > d^-$. It has been analysed lately that this value serves as an efficient estimation of a confidence for classification with rejection showing similar performance as an explicit probabilistic modelling but at lower computational costs [43]. The calculation of the RelSim for unlabelled data points $\mathbf{x}$ is based on the estimated class label with respect to the model, i. e. d^+ is the distance of $\mathbf{x}$ to the BMU $\mathbf{w}_s$ (defines the label of $\mathbf{x}$) and hence d^- is the distance of $\mathbf{x}$ to any prototype with a different class label than $\mathbf{w}_s$.

LVQ can be extended by the powerful concept of metric learning, which got popular in distance-based classification lately [29], [30], [31]. Especially, there exists a generalisation of the GLVQ towards a general quadratic form $(\mathbf{x}-\mathbf{w}_j)^T\Lambda(\mathbf{x}-\mathbf{w}_j)$ with positive semi-definite matrix Λ which is proposed under the acronym GMLVQ [27]. A local version thereof is the local GMLVQ (LGMLVQ) [27] where each prototype $\mathbf{w}_j$ has its own local metric $d_j(\mathbf{x}, \mathbf{w}_j) = (\mathbf{x} - \mathbf{w}_j)^T\Lambda_j(\mathbf{x} - \mathbf{w}_j)$. The updates of the local metrics regulates the step size ε_m. Formal learning theoretical guarantees have been derived for GMLVQ networks in [27].

The ioLVQ [28] is usable in incremental online scenarios where training data points arrive one after another. It starts with no prototypes (no knowledge), i. e. $W = \emptyset$ and the formulas for prototype and metric updates are the same as for the batch LVQ versions but the set of prototypes is dynamically changed to account for data drift or new classes. Insertion and deletion of prototypes rely on the objective to decrease the costs (1):

1. New classes [44]: Each training data point $(\mathbf{x}, y)$ of a new class, i. e. $y \neq c(\mathbf{w}_j), \forall j$, is directly used as a new prototype with label y. The class label is available since we assume an interaction with the system, e. g. a user provides data points of a new class. Hence, we do not tackle an emergence of a new class from unsupervised samples like e. g. [45].

2. Prototype insertion: Reducing misclassification lowers the costs (1). Using the idea from [46], [47], a set S with maximum storage capacity $g_{\max}$ stores errors during training. Once $|S| = g_{\max}$, a prototype with label p is added for each class p for which S stores errors. Its location is chosen to minimise the costs of the LVQ model, i. e. the point $(\mathbf{x}_i, p)$ in S with lowest $\text{RelSim}(\mathbf{x}_i)$ identifies the prototype location. Note that this also relates to a high uncertainty, hence a potentially useful location. After this insertion, S is cleared, i. e. $S := \emptyset$.

3. Prototype deletion: Unlimited prototype insertion can result in prototypes with Voronoi cells where more wrong than correct classifications are made. We delete prototypes based on their contribution to the costs (1), using the idea of [48]. Each prototype $\mathbf{w}$ gets a parameter $\eta(\mathbf{w})$, initialised by zero, that sums up the certainty of the classifications of data points in its Voronoi cell. The iterative update

$$\eta(\mathbf{w}_l) := \eta(\mathbf{w}_l) + \text{RelSim}(\mathbf{x})$$

efficiently computes $\eta(\mathbf{w}_l)$ for the BMU $\mathbf{w}_l$ provided the point $(\mathbf{x}, y)$ is presented. The value $\eta(\mathbf{w}_l)$ is negative iff on average, the prototype accounts for more wrongly classified points than correct ones, weighted by their certainty. After every r_{num} training data points, prototypes with $\eta(\mathbf{w}) < 0$ are deleted.

Due to its representation of data in terms of prototypes, efficient incremental learning schemes have been proposed [37], [38], [39], [42], [46], [47], [49]. These schemes rely on the fact that prototypes can be used as a summary statistics to represent the already seen data, such that incremental online adaptation is easily possible based on the learned prototypes and new data. Interestingly, recent approaches [8], [11], [13] particularly address the case that data display a trend, i. e. they can deal with non i. i. e. data distributions. However, these schemes heavily depend on the choice of crucial system parameters and cannot provide easily adaptive models for life-long learning yet. In particular, it is not yet possible to easily combine these incremental schemes with basic models which securely represent known static knowledge, i. e. their capability of dealing with the stability/plasticity dilemma severely depends on a correct choice of learning parameters [28]. In the following, we will aim for a flexible life-long learning scheme that is based on a hybrid online/offline learning architecture combining stability and plasticity of the system in an optimal way, offering elegant schemes e. g. for a personalisation of basic models in daily use. Further we will analyse several incremental approaches, e. g. ioLVQ [28] and an iSVM [32] and we will give an overview about their properties and requirements.

IV. Combining Offline and Online Learning (OOL)

The main idea of the OOL architecture (Fig. 1) proposed in [17], [18] combines a pre-trained offline model that provides known knowledge of the desired task with an incremental online model that learns special characteristics or new classes during the life-long learning application in its (dynamic) environment. The model that is more reliable in its classification with respect to the certainty value for a given data point will define the class label. A schematic system architecture is displayed in Fig. 1. In the following, we elaborate the basic components, whereby a crucial point is how to define and train an online classifier in an optimal way, and how to combine the two models, integrating the new RelSim certainty.

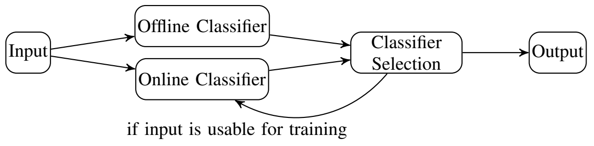

Fig. 1. OOL architecture combining an online and an offline classifier [18].

Input: The input of the system is a data point $\mathbf{x} \in \mathbb{R}^n$ with or without a label y denoting one of the C known classes or a new one. The offline and online classifier receive the input.

Offline Classifier: This container refers to any offline classifier that provides a certainty value in addition to its predicted class label. Here we choose a pre-trained LGMLVQ model (instead of GMLVQ [18]), i. e. a set of trained prototypes $\mathbf{w}_j$ for the C known classes with their local metrics d_j. The offline classifier is static, i. e. it preserves the gained knowledge during the whole application. The output of the offline classifier is a class label y^{off} and the related certainty value ($\text{RelSim}^{\text{off}}(\mathbf{x})$) for an input data point $\mathbf{x}$ that is passed to the classifier selection.

Online Classifier: This container refers to any online incremental classifier that provides a certainty value in addition to its predicted class label. It will be the focus of this article to investigate how to efficiently set up and integrate an online LVQ model to obtain an balance of stability and flexibility of the overall system. Here we choose the ioLVQ [28] based on an LGMLVQ model instead of an incremental GMLVQ which only allows prototype insertion [18]. Update rules of the online model are performed under some conditions (see classifier selection) only. While training, it is vital that suitable training instances are available for the online classifier in order to gain specific knowledge that is unknown in the offline classifier. The output of the online classifier for an input data point $\mathbf{x}$ is a class label y^{on} and the related certainty value ($\text{RelSim}^{\text{on}}(\mathbf{x})$) which is passed to the classifier selection.

Classifier Selection: The classifier selection of the system has two functions: 1) mediating between the online and offline classifier based on their certainty values and 2) deciding if the input is usable as training instance for the online classifier.

These two aspects work as follows: 1) we mimic the idea of [21] with a better suited certainty measure instead of a heuristic based on error counting [18]. We choose the more reliable classifier which decides the class label of the system

$$y^{\text{sys}} := \begin{cases} y^{\text{off}}, \text{if } \text{RelSim}^{\text{off}}(\mathbf{x}) \geq \text{RelSim}^{\text{on}}(\mathbf{x}) \\ y^{\text{on}}, \text{else} \end{cases} .$$

2) training of the online model is controlled as follows: Firstly the input of the system necessarily needs a class label, i. e. $(\mathbf{x}, y)$. Mainly the system uses no input data point for training if the offline classifier is reliable and provides the correct class label. If at least one of the three following conditions is valid, the online classifier uses the input for training.

1. The online classifier is more reliable than the offline one
$$\text{RelSim}^{\text{on}}(\mathbf{x}) > \text{RelSim}^{\text{off}}(\mathbf{x}) \qquad (2)$$
2. The offline classifier is more reliable but it provides a wrong class label.
$$\text{RelSim}^{\text{on}}(\mathbf{x}) < \text{RelSim}^{\text{off}}(\mathbf{x}) \land y^{\text{off}} \neq y \qquad (3)$$

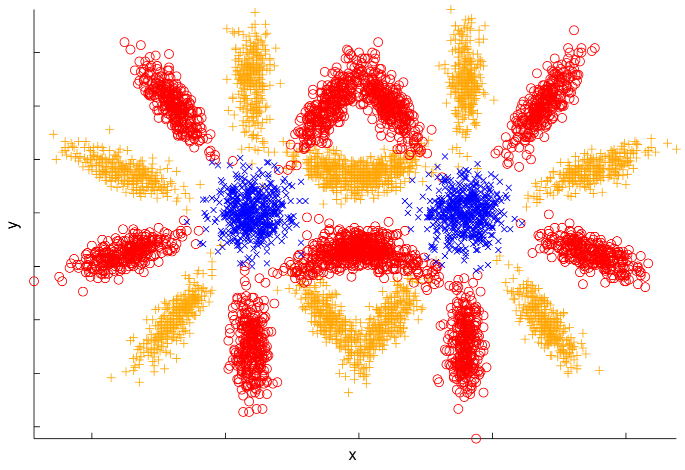

Fig. 2. Color encodes class. The left/right blossom forms a data partition.

3. Both models are unreliable (here: $\Gamma = 0.6$).

$$\min\{\mathrm{RelSim}^{\mathrm{on}}(\mathbf{x}), \mathrm{RelSim}^{\mathrm{off}}(\mathbf{x})\} < \Gamma \qquad (4)$$

Output: The system output is the estimated class label corresponding to the classifier which provided this label of the system y^{sys} for the given input with its certainty value.

The extended OOL architecture provides a particularly simple scheme promising an easy control of the model adaptation. It is also designed to perform better than pure incremental approaches with respect to the CFE because of its static offline model. We will show that the extended OOL architecture enables an effective learning in dynamic settings where the data distribution used for learning is changing, i. e. the crucial assumption of training data being i. i. d is violated.

V. Experiments

We consider experiments on three data sets: a 2D data set (Blossom) for visualisation and analysing the different architectures, the U.S. Postal Service (USPS[2]) Handwritten Digits data as a benchmark data set, and a real world outdoor data set (Outdoor) obtained from a robot during its application. Each data set is divided into two partitions. One partition of the data simulates a data distribution which is only accessible in the second training phase of the system while the other partition/distribution provides training instances for both phases.

- Blossom: Artificially created three class data set (Fig. 2). Each blossom forms one partition of the data.
- USPS: This data set contains 8-bit grayscale images of ”0” to ”9”; 1100 instances per class. The dimension of each instance is reduced to 30 with principal component analysis [50]. Half of the classes form the first respectively the second partition of the data.
- Outdoor: The outdoor obstacle data set includes 4000 instances consisting of the 21 values of a 6 bin rg chromaticity diagram. Each instance belongs to one of the 40 objects such as dog, red ball, book, etc. lying on a garden lawn (Fig. 3). Half of the classes form the first respectively the second partition of the data.

[2]Data is obtained from: http://www.cs.nyu.edu/~roweis/data.html

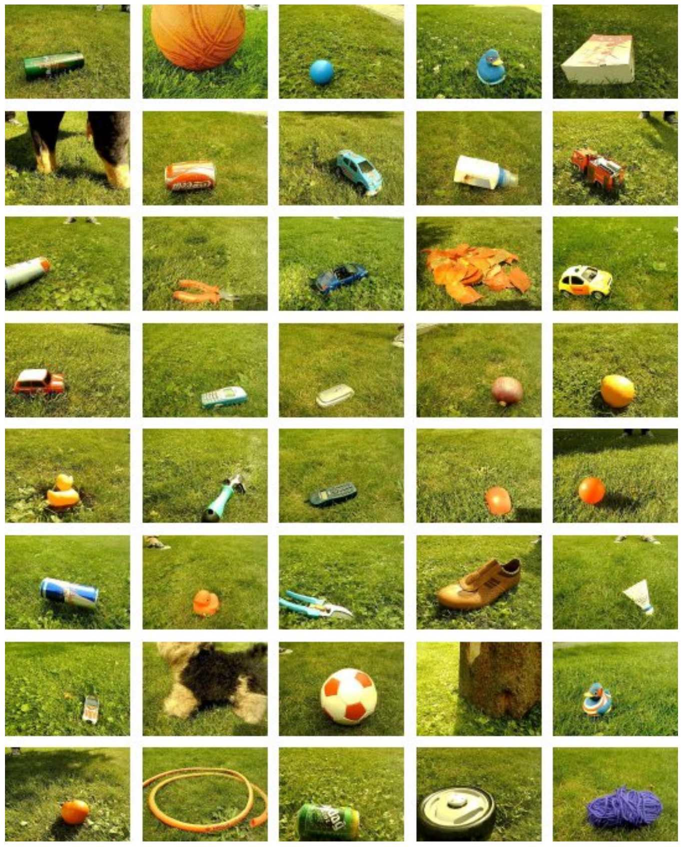

Fig. 3. The objects of the Outdoor data set.

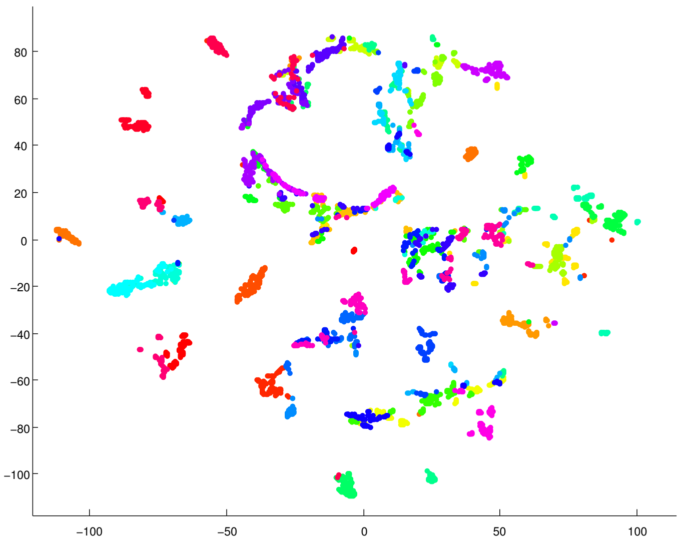

Fig. 4. A 2D visualisation of the Outdoor data set with the Fisher t-SNE [51]. The different colors denote the different classes.

Figure 4 shows a 2D visualisation of the data with the Fisher t-SNE[3] showing highly overlapping classes.

Unless stated otherwise, we consider randomly ordered data presented in a single pass, like in a streaming setting, using constant learning rates (except for the batch LGMLVQ). The evaluation for the Blossom and the USPS data set is based on

[3]Thanks to Alexander Schulz for providing the code.

TABLE I. EXPLANATION OF THE INCREMENTAL LVQ APPROACHES USED IN THE EXPERIMENTS.

	OOL	ioLVQ	Combi
model$^{\text{off}}$	LGMLVQ	ioLGMLVQ	LGMLVQ
init with model$^{\text{off}}$	no	yes	yes
model$^{\text{on}}$	ioLGMLVQ	ioLGMLVQ	ioLGMLVQ

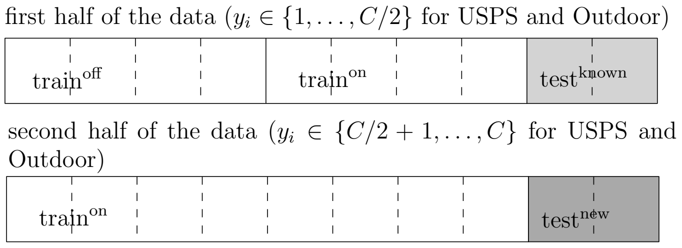
first half of the data ($y_i \in \{1, \dots, C/2\}$ for USPS and Outdoor)
train$^{\text{off}}$
train$^{\text{on}}$
test$^{\text{known}}$
second half of the data ($y_i \in \{C/2+1, \dots, C\}$ for USPS and Outdoor)
train$^{\text{on}}$
test$^{\text{new}}$

Fig. 5. Each data set is partitioned in two balanced sets: the first and the second half of the data. Two folds of each set form a test set: test$^{\text{known}}$ and test$^{\text{new}}$ respectively. For the Blossom data set holds that the left blossom forms the first half of the data and the right blossom forms the second half (Fig.2).

a 10-fold cross validation with ten repeats as follows: Each data set is decomposed in two balanced partitions as stated before. The first partition symbolises data which is available for offline training and data which can be encountered during use. The second partition symbolises data which can be only encountered during use and it contains new classes or drifted data of known classes (new data distribution). For a 10-fold cross validation, each partition is divided into 10 folds (see Fig. 5). The offline training set train$^{\text{off}}$ consists of four folds of the first half of the data and the online training data train$^{\text{on}}$ (encountered during use) consists of four folds of the first half and of eight folds of the second half of the data. To evaluate the different approaches in a proper way we define three different test sets : test$^{\text{known}}$, test$^{\text{new}}$ and test. The test set test$^{\text{known}}$ are two folds of the first half of the data and its error will show the performance on these data which samples the static ground knowledge (i. e. the performance tests the stability of a learning method). The second test set is test$^{\text{new}}$ that are two folds of the second half of the data and it will show the accuracy of newly gained knowledge in the online part of the system, i. e. it tests the flexibility of a method. The overall performance is checked with the last test set test $=$ test$^{\text{known}} \cup$ test$^{\text{new}}$, i. e. it tests the capability to deal with both challenges, stability and plasticity.

The Outdoor data set has the special characteristic that its data points represent sequences of objects where one sequence consists of serial frames of an object approached by a mobile robot. Especially for each object there exist ten different sequences, each with ten data points which is treated as a unit during the experiments (except in batch LGMLVQ). We use the same training and testing scheme as mentioned before with the difference that only 5-partitions are generated (since there are too few sequences for 10-folds) and that we choose randomly the sequences for the different train/test sets. The training data are presented in a single pass with randomly sequence order. This experiment is performed ten times.

We compare the OOL architecture with four alternatives: ioLVQ, Combi, Batch (Tab. I) and iSVM defined as follows:

OOL: This is the proposed extension of the architecture [17], [18] described in sec. IV. The static offline classifier (batch LGMLVQ) is trained offline with the train set train$^{\text{off}}$ and the train set train$^{\text{on}}$ is applied to the architecture and data points fulfilling one of the three conditions (2), (3), (4) are used to train the online classifier (ioLVQ). This scheme of combining an offline trained model with an online, incremental model is designed to avoid the CFE, especially to move in the direction of guaranteeing certain performances on defined data, e. g. for safety relevant applications. The static offline model is fine tuned with respect to the offline available data and preserves this knowledge during the whole application. This offline model is accompanied by a flexible online model which is capable of adapting to new data and classes.

ioLVQ: This model is a purely incremental model without a guaranteed static offline part as provided by OOL. For this purpose, we use the recently proposed ioLVQ [28]. The resulting model is obtained purely from incremental learning and serves as comparison. We expect that it more easily suffers from the CFE while potentially being more accurate for the online data. Here, we first train the incremental online LGMLVQ (ioLGMLVQ) [28] using train$^{\text{off}}$, before using train$^{\text{on}}$.

Combi: This model is similar to the previous one, but uses batch LGMLVQ [27] instead of ioLGMLVQ for the first training phase with data train$^{\text{off}}$. In this setting, we can expect to gather the best prototypes with respect to the available offline data. The obtained model provides an initialisation for the training of the ioLGMLVQ [28] with train$^{\text{on}}$. Combining these models has the advantage of gathering the best prototypes with respect to the offline available data and using this knowledge as meaningful initialisation for the online incremental learning part. This can boost the performance of the online model since it does not start from scratch. Because there is no static knowledge of the offline trained data, the CFE can have a bad impact on the performance of this model like in ioLVQ.

Batch: The batch LGMLVQ [27] trains on the union of both train sets. The LGMLVQ is a powerful model if the access to all training data is available at once. It is inept for tasks where training data is encountered during the application especially if there is a concept drift or new classes since the number of classes and the number of prototypes per class have to be set as parameters before training. This mode serves as comparison to the architecture, showing a baseline performance.

iSVM [32]: The used code for the iSVM is adapted to bound parts of the memory. Hence, we set the number of the so called reserve vectors of the implementation[1] to one corresponding to the full online case we consider here. The iSVM trains with the data from train$^{\text{off}}$ followed by the train set train$^{\text{on}}$. Required parameters are the soft-margin regularization constant C and the Gaussian kernel parameter γ. Note, that we use the binary classification implementation in a one vs. all mode enabling multi-class classification. This model serves as a comparison to a state of the art approach focusing on class borders rather than typical representatives like LVQ.

Table II shows the parameters of the experiments and Tab. III shows the results. It contains the accuracies of the three test sets and the number of prototypes, respectively the number of support vectors. We mark in boldface the results laying on the Pareto-front with respect to the two objectives: accuracy on test and the number of prototypes. A result is Pareto optimal if there is no other result which is better in both objectives. First

TABLE II. PARAMETERS OF THE OFFLINE AND THE ONLINE MODEL OF THE DIFFERENT APPROACHES

	LGMLVQ				ioLVQ				iSVM	
	training epochs	prototypes/class	ε_w	ε_m	g_{max}	r_{num}	ε_w	ε_m	C	γ
Blossom	100	5 or 1	0.3	0.07	20	320	0.1	0.01	2^5	$2^{3.5}$
USPS	100	1	0.4	0.01	20	400	0.4	0.01	2^5	$2^{3.5}$
Outdoor	250	5 or 6	$4 \cdot 10^{-5}$	10^{-5}	5	900	$4 \cdot 10^{-5}$	10^{-5}	2^5	0.02

TABLE III. WE REPORT THE AVERAGE ACCURACY ON DIFFERENT TEST SETS AND THE CORRESPONDING AVERAGE PROTOTYPE NUMBER $|W|$ OF THE APPROACHES.

The value $|W|$ denotes the average prototype number of the approaches, respectively the support vectors. For the OOL architecture it is the sum of the prototypes of the online and the offline model. Results that are in bold lay on the Pareto-front with respect to overall accuracy on the test set test and the model complexity $|W|$. The results marked with * are the best ones compared to all other approaches on test$^{\text{known}}$ disregarding batch LGMLVQ. The t-test for the result of OOL of Blossom is significantly different ($p < 10^{-3}$) to iSVM and batch LGMLVQ. For the USPS data set, every alternative to OOL provides significantly different ($p < 10^{-3}$) results. Since there are only 10 results averaged for Outdoor, no t-test is done.

Blossom	OOL	ioLVQ	Combi	iSVM	batch LGMLVQ
$\|W\|$	27.0	25.4	**25.2**	**76.8**	22
test$^{\text{known}}$	95.0	90.7	91.7	97.5*	98.2
test$^{\text{new}}$	89.0	92.3	93.1	98.6	98.2
test	92.0	91.6	**92.4**	**98.1**	98.2
USPS[4]	OOL	ioLVQ	Combi	iSVM	batch
$\|W\|$	25.6	17.5	**14.8**	**304.3**	11
test$^{\text{known}}$	92.6*	86.7	91.9	91.1	95.8
test$^{\text{new}}$	91.7	94.3	94.0	95.6	95.4
test	92.1	90.5	**93.1**	**93.3**	95.6
Outdoor	OOL	ioLVQ	Combi	iSVM	batch
$\|W\|$	**127.9**	128.3	**120.5**	**1035.3**	200
test$^{\text{known}}$	65.1*	44.8	42.8	52.8	63.1
test$^{\text{new}}$	38.9	58.2	60.9	65.2	58.0
test	**53.2**	52.1	**51.9**	**59.2**	61.3

we discuss the results of the single approaches and then we draw conclusions from their comparison to each other.

Evaluation of the different approaches

OOL: It reaches the highest accuracies on the known data for all three data sets. The good performance on test$^{\text{known}}$ highlights the robustness of the architecture against the CFE. It performs well on the online data, even if the accuracies obtained on test$^{\text{new}}$ are slightly lower than for the known data. In particular the accuracy on test$^{\text{new}}$ of the Outdoor data set performs worse compared to the other approaches. The analysis of this effect indicates that the classifier selections needs to be improved. The different scaling of the certainty measure in the offline and online model seem to cause this effect. Further analysis of this issue are beyond the scope of this paper and will be addressed in future work. The overall accuracy of the OOL is mainly the average of the test accuracies on test$^{\text{known}}$ and test$^{\text{new}}$ and competes with the other approaches. Hence there exists a lower bound of the performance (performance of model$^{\text{off}}$) of the architecture, even if the online model has few knowledge or if it is disturbed, e. g. by noise. A previous comparison between the OOL and the original architecture [18] has shown that the accuracy is higher, at least similar to the original architecture but OOL needs much less prototypes.

[4]Scaled to [0,1] for the iSVM.

ioLVQ: The ioLVQ reaches the highest accuracies on the test set test$^{\text{new}}$ for all three data sets. This shows that the model learns new data fast and that it provides high performances especially on data that correspond to the last seen training data. Table III shows clearly that the accuracies of test$^{\text{known}}$ are below test$^{\text{new}}$ meaning the model suffers from a moderate forgetting effect. The difference is significant for the Outdoor and the USPS data set. The overall accuracy of the test set test is comparable to the other approaches.

Combi: The Combi models get the highest accuracies on test$^{\text{new}}$. This behaviour equals those of the ioLVQ models. The Combi models suffer from a forgetting effect comparing the accuracies of test$^{\text{known}}$ and test$^{\text{new}}$, too. The offline trained prototypes (batch) used as initialisation for the ioLVQ model in the online training affect the needed number of prototypes (Combi needs less prototypes than ioLVQ) and the accuracies of the model in general. Combi shows higher accuracies on the Blossom and the USPS data set compared to ioLVQ. While the differences among the accuracies of Combi and ioLVQ are small (except test$^{\text{known}}$ of USPS), they exist. Hence it seems suitable to start training in batch mode for partly available data.

Batch: This mode provides the baseline accuracies for the data sets, i. d. reachable accuracies if the whole data is available at once with a defined number of prototypes.

iSVM: The iSVM reaches its highest accuracy values on test$^{\text{new}}$. It exhibits the same effect that test$^{\text{known}}$ has a lower accuracy than test$^{\text{new}}$ which means it suffers from a forgetting effect, too. Comparing the results to the other approaches one has to say, that the accuracy is marginally higher except for the test$^{\text{known}}$. Note that the models use more support vectors than the other approaches use prototypes.

Comparison: The results in Tab. III indicate that there is no approach that surpasses all the other approaches in terms of performance and model complexity. It turns out that the approaches have different properties and suit different needs which will be discussed subsequently. In general iSVM can often provide a slightly better accuracy at the costs of a greatly enhanced model complexity. This effect can be expected due to the representation of a model in terms of its class boundaries for iSVM in contrast to its class centers for LVQ. However, if model complexity is not an issue, a similar hybrid system could easily be built based on SVM as content for the two containers for an online/offline model. When comparing incremental versions which are based on prototypes, the following conclusions can be drawn: The OOL surpasses all the other approaches on the set test$^{\text{known}}$ which means it preserves offline gained knowledge during the whole application. The accuracy on the test set test$^{\text{new}}$ is comparable to the other approaches but slightly lower except for the Outdoor data set. In general the ioLVQ and Combi models are good alternatives if focusing on a high accuracy on previously seen data, especially if no offline data is available. If one is more focused on preserving knowledge and gathering new information during the application, then the OOL is a better choice. In particular for Outdoor which is a real-world data set with a substantial experiment setting, the OOL performs well since the overall performance on the test set surpasses ioLVQ and Combi and is only slightly lower than the performance of the iSVM while using a fraction of resources. This makes OOL more attractive for devices with restricted memory. Additionally the performance of the OOL on test$^{\text{known}}$ is higher than the performance of the competitors on the Outdoor data which highlights the robustness on known data/classes.The baseline performance provided by the batch LGMLVQ is nearly reached by the incremental approaches. The number of the prototypes for the OOL, ioLVQ and Combi is still in the same range as the number of prototypes of the LGMLVQ while the the number of the support vectors of the iSVM is significantly higher. This shows the effectiveness of the incremental LVQ approaches, i. e. they provide a high performance with a low number of prototypes.

Discussion of the properties of the different approaches

The analysed approaches have different properties and hence are suited for different life-long learning tasks. In the following we discuss the availability of offline data, the model size, the overall performance of the system and the performance on known data/classes. This might help to choose the best approach for the given circumstances of the desired task.

Offline data: If data is offline available two schemes are possible: i) using them as training set for the offline model of the OOL or of Combi or ii) simply present them in an online fashion to the other approaches (ioLVQ, iSVM) to train the related model. The results in Tab. III indicate that it is beneficial to choose the first scheme i). If the offline data is used in the OOL, one gains a high and robust performance on the offline data, respectively the known data/classes during the ongoing application. Using Combi instead provides a highly flexible model that is less robust against the CFE but needs less prototypes compared to the ioLVQ or the number of support vectors of the iSVM. If there is no offline data available one can only choose between the ioLVQ and the iSVM.

Model size: In the proposed experiments, the approaches OOL, ioLVQ and Combi use less prototypes compared to the batch LGMLVQ while the iSVM needs significantly more support vectors. With the parameters g_{max} and r_{num} one can regulate indirectly the tendency of the complexity of a model, for details see [28]. Basically both parameter influences directly the insertion respectively the deletion strategy of the algorithm. A hard coded upper bound of prototypes is possible [48], e. g. for applications with a restricted memory.

Overall performance: If one is mainly interested in the performance of the system the iSVM is an adequate choice paying the price of a highly complex model, i. e. it needs a high number of support vectors. Additionally the parameters C and γ have to be defined which is not intuitive but critical because they are important for the performance of the iSVM model. The other approaches reach performances comparable to iSVM using a small number of prototypes compared to the number of support vectors and the required parameters g_{max} and r_{num} are intuitively understandable and hence easier to choose.

Performance on known data/classes: In safety relevant applications it is often very important to guarantee performances on defined data during the whole application time. With this view, the experiments have shown that the OOL provides good performances on the test set test$^{\text{known}}$ highlighting the robustness against the CFE while especially ioLVQ suffers from this effect on the evaluated data sets. Combi and iSVM tend also to provide worse results on known data compared to the performance on new data. Therefore, the best choice would be the OOL.

VI. Conclusion

The proposed OOL extended architecture is well suited for life-long learning tasks or streaming data. The general architecture consists of three parts: a static, pre-trained offline model, a flexible online model and a dynamic classifier selection part. Combining an offline and an online model has the advantage of learning new knowledge with the online model while preserving the ground (offline) information with the offline model. We have investigated the performance of the OOL using the example of learning vector quantization: LGMLVQ in batch mode serves as the offline model of the architecture, the LGMLVQ in incremental mode offers a flexible online model of the architecture and a dynamic classifier selection can be based on a certainty measure that is well suited for LVQ approaches. We analysed the properties and performances of the OOL in comparison to several other incremental approaches (ioLVQ, Combi, iSVM). The experiments on artificial and real life data have shown that the OOL is more robust against the CFE compared to other state of the art approaches and that it is able to gather new knowledge during application like other incremental learning approaches. Additionally, it turned out that none of the analysed approaches surpasses the others with respect to performance and model complexity. We discussed the properties of the approaches in order to give hints which one to choose for a desired task. The analyses give an overview of some existing life-long learning approaches highlighting the the robustness of the OOL against the CFE. This makes it attractive for safety related application where performance guarantees for known data are required.

Acknowledgment

The authors would like to thank Jeffrey Queißer for his support in related questions to his master thesis and Viktor Losing for providing the Outdoor data set with the related image showing the included objects and for the adapted iSVM code. We would like to thank the reviewers for their helpful advices. BH gratefully acknowledges funding by the CITEC center of excellence. LF acknowledges funding by the CoR-Lab Research Institute for Cognition and Robotics and acknowledges the financial support from Honda Research Institute Europe.

References

[1] D. L. Silver, Q. Yang, and L. Li, "Lifelong Machine Learning Systems: Beyond Learning Algorithms," in *AAAI Spring Symposium: Lifelong Machine Learning*, ser. AAAI Technical Report, vol. SS-13-05, 2013.

[2] S. Thrun and T. M. Mitchell, "Lifelong Robot Learning," Robotics and Autonomous Systems, Tech. Rep., 1993.

[3] D. Sykes, D. Corapi, J. Magee, J. Kramer, A. Russo, and K. Inoue, "Learning Revised Models for Planning in Adaptive Systems," in *Proc. of Int. Conf. on Software Engineering*. IEEE Press, 2013, pp. 63–71.

[4] D. Stavens, G. Hoffmann, and S. Thrun, "Online Speed Adaptation Using Supervised Learning for High-speed, Off-road Autonomous Driving," in *Proc. Int. Joint Conf. on Artif. Intell.*, 2007, pp. 2218–2224.

[5] V. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[6] Y. Jin and B. Sendhoff, "Alleviating Catastrophic Forgetting via Multi-Objective Learning," in *Proc. Int. Joint Conf. on Neural Netw., part of the IEEE World Congress on Comp. Intellig.* IEEE, 2006, pp. 3335–3342.

[7] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An Empirical Investigation of Catastrophic Forgeting in Gradient-Based Neural Networks," *ArXiv e-prints*, Dec. 2013.

[8] R. Polikar and C. Alippi, "Guest Editorial Learning in Nonstationary and Evolving Environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 9–11, 2014.

[9] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, p. 44, 2014.

[10] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, 2012.

[11] H. He, S. Chen, K. Li, and X. Xu, "Incremental Learning From Stream Data," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, 2011.

[12] G. Cauwenberghs and T. Poggio, "Incremental and Decremental Support Vector Machine Learning," in *Advances in Neural Information Processing Systems. NIPS*, 2000, pp. 409–415.

[13] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive Regularization of Weight Vectors," *Machine Learning*, vol. 91, no. 2, pp. 155–187, 2013.

[14] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An Incremental Learning Algorithm for Supervised Neural Networks," *IEEE Trans. Sys., Man, and Cybern., C*, vol. 31, no. 4, pp. 497–508, 2001.

[15] M. J. Hosseini, Z. Ahmadi, and H. Beigy, "Using a classifier pool in accuracy based tracking of recurring concepts in data stream classification," *Evolving Systems*, vol. 4, no. 1, pp. 43–60, 2013.

[16] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[17] H. Wersing and J. F. Queißer, "System for controlling an automated device," Patent EP 2 690 582 A1, 2014.

[18] J. F. Queißer, "Using context for the combination of offline and online learning," Master Thesis, Bielefeld University, 2012. [Online]. Available: http://pub.uni-bielefeld.de/publication/2670657

[19] B. V. Dasarathy and B. V. Sheela, "A Composite Classifier System Design: Concepts and Methodology," *Proceedings of IEEE*, vol. 67, no. 5, pp. 708–713, May 1979.

[20] M. Sabourin, A. Mitiche, D. S. Thomas, and G. Nagy, "Classifier combination for hand-printed digit recognition," in *2nd Int. Conf. Document Analysis and Recognition, ICDAR.* IEEE, 1993, pp. 163–166.

[21] K. S. Woods, W. P. Kegelmeyer, and K. W. Bowyer, "Combination of Multiple Classifiers Using Local Accuracy Estimates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 405–410, 1997.

[22] A. S. Britto Jr., R. Sabourin, and L. E. S. de Oliveira, "Dynamic selection of classifiers - A comprehensive review," *Pattern Recognition*, vol. 47, no. 11, pp. 3665–3680, 2014.

[23] T. Kohonen, *Self-Organization and Associative Memory.* Springer Series in Information Sciences, Springer-Verlag, third edition, 1989.

[24] N. Alex, A. Hasenfuss, and B. Hammer, "Patch clustering for massive data sets," *Neurocomputing*, vol. 72, no. 7-9, pp. 1455–1469, 2009.

[25] A. Sato and K. Yamada, "Generalized Learning Vector Quantization," in *Adv. Neural Inf. Proces. Sys. NIPS*, vol. 7, 1995, pp. 423–429.

[26] S. Seo and K. Obermayer, "Soft Learning Vector Quantization." *Neural Computation*, vol. 15, no. 7, pp. 1589–1604, 2003.

[27] P. Schneider, M. Biehl, and B. Hammer, "Adaptive Relevance Matrices in Learning Vector Quantization," *Neural Computation*, vol. 21, no. 12, pp. 3532–3561, 2009.

[28] L. Fischer, B. Hammer, and H. Wersing, "Certainty-based Prototype Insertion/Deletion for Classification with Metric Adaptation," in *Proc. ESANN*, 2015, pp. 7–12.

[29] M. Biehl, B. Hammer, and T. Villmann, "Distance Measures for Prototype Based Classification," in *Brain-Inspired Computing - International Workshop, BrainComp*, 2013, pp. 100–116.

[30] A. Bellet, A. Habrard, and M. Sebban, "A Survey on Metric Learning for Feature Vectors and Structured Data," *ArXiv e-prints*, Jun. 2013.

[31] A. Bellet and A. Habrard, "Robustness and Generalization for Metric Learning," *Neurocomputing*, vol. 151, no. 1, pp. 259–267, 2015.

[32] C. Diehl and G. Cauwenberghs, "SVM Incremental Learning, Adaptation and Optimization," in *Int. Joint Conf. on Neural Netw.*, vol. 4, 2003, pp. 2685–2690.

[33] I. Giotis, K. Bunte, N. Petkov, and M. Biehl, "Adaptive Matrices and Filters for Color Texture Classification," *Journal of Mathematical Imaging and Vision*, vol. 47, pp. 79–92, 2013.

[34] M. Biehl, K. Bunte, and P. Schneider, "Analysis of Flow Cytometry Data by Matrix Relevance Learning Vector Quantization," *PLoS ONE*, vol. 8, no. 3, p. e59401, 2013.

[35] M. Biehl, P. Sadowski, E. B. G. Bhanot, A. Dayarian, P. Meyer, R. Norel, K. Rhrissorrakrai, M. D. Zeller, and S. Hormoz, "Inter-species prediction of protein phosphorylation in the sbv IMPROVER species translation challenge," *Bioinformatics*, 2014.

[36] J. G.-J. de Vries, S. C. Pauws, and M. Biehl, "Insightful stress detection from physiology modalities using learning vector quantization," *Neurocomputing*, vol. 151, Part 2, pp. 873 – 882, 2015.

[37] S. Kirstein, H. Wersing, and E. Körner, "A Biologically Motivated Visual Memory Architecture for Online Learning of Objects," *Neural Networks*, vol. 21, no. 1, pp. 65–77, 2008.

[38] S. Kirstein, H. Wersing, H.-M. Gross, and E. Körner, "A Life-Long Learning Vector Quantization Approach for Interactive Learning of Multiple Categories," *Neural Networks*, vol. 28, pp. 90–105, 2012.

[39] S. Kirstein, A. Denecke, S. Hasler, H. Wersing, H. Gross, and E. Körner, "A vision architecture for unconstrained and incremental learning of multiple categories," *Memetic Comp.*, vol. 1, no. 4, pp. 291–304, 2009.

[40] K. Bunte, P. Schneider, B. Hammer, F. Schleif, T. Villmann, and M. Biehl, "Limited Rank Matrix Learning, Discriminative Dimension Reduction and Visualization," *Neural Networks*, vol. 26, pp. 159–173, 2012.

[41] D. Nova and P. Estevez, "A review of learning vector quantization classifiers," *Neural Comp. and Appl.*, vol. 25, no. 3-4, pp. 511–524, 2014.

[42] X. Zhu, F. Schleif, and B. Hammer, "Adaptive conformal semi-supervised vector quantization for dissimilarity data," *Pattern Recognition Letters*, vol. 49, pp. 138–145, 2014.

[43] L. Fischer, B. Hammer, and H. Wersing, "Efficient rejection strategies for prototype-based classification," *Neurocomputing*, vol. 169, pp. 334 – 342, 2015.

[44] Y. Xu, F. Shen, and J. Zhao, "An incremental learning vector quantization algorithm for pattern classification," *Neural Computing and Applications*, vol. 21, no. 6, pp. 1205–1215, 2012.

[45] K. B. Dyer, R. Capo, and R. Polikar, "COMPOSE: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 1, pp. 12–26, 2014.

[46] S. Kirstein, H. Wersing, and E. Körner, "Rapid Online Learning of Objects in a Biologically Motivated Recognition Architecture," in *Pattern Recognition Symposium DAGM*, 2005, pp. 301–308.

[47] T. C. Kietzmann, S. Lange, and M. Riedmiller, "Incremental GRLVQ: Learning Relevant Features for 3D Object Recognition," *Neurocomputing*, vol. 71, no. 13-15, pp. 2868–2879, 2008.

[48] M. Grbovic and S. Vucetic, "Learning Vector Quantization with Adaptive Prototype Addition and Removal," in *International Joint Conference on Neural Networks IJCNN*, 2009, pp. 994–1001.

[49] Y. Xu, S. Furao, O. Hasegawa, and J. Zhao, "An Online Incremental Learning Vector Quantization," in *Adv. 13th Pacific-Asia Conf. Knowledge Discovery and Data Mining PAKDD*, 2009, pp. 1046–1053.

[50] L. J. P. van der Maaten, "Matlab Toolbox for Dimensionality Reduction," 2013, http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html.

[51] A. Gisbrecht, A. Schulz, and B. Hammer, "Parametric nonlinear dimensionality reduction using kernel t-SNE," *Neurocomputing*, vol. 147, pp. 71–82, 2015.