Information Sciences 192 (2012) 50–70

Contents lists available at ScienceDirect

Information Sciences

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

An adaptive classification system for video-based face recognition

Jean-François Connolly, Eric Granger [*], Robert Sabourin

Laboratoire d'imagerie, de vision et d'intelligence artificielle, École de technologie supérieure, Université du Québec, 1100, rue Notre-Dame Ouest, Montréal, Canada H3C 1K3

A R T I C L E   I N F O

Article history:
Received 16 August 2009
Received in revised form 2 January 2010
Accepted 21 February 2010
Available online 6 March 2010

Keywords:
Incremental learning
ARTMAP neural networks
Dynamic particle swarm optimization
Pattern recognition
Face recognition
Image/video processing

A B S T R A C T

In many practical applications, new information may emerge from the environment at different points in time after a classification system has originally been deployed. For instance, in biometric systems, new data may be acquired and used to enroll or to update knowledge of an individual. In this paper, an adaptive classification system (ACS) is proposed for video-based face recognition. It combines a fuzzy ARTMAP neural network classifier, dynamic particle swarm optimization (DPSO) algorithm, and a long term memory (LTM). A novel DPSO-based learning strategy is also presented for incremental learning of new data with this ACS. This strategy allows to cojointly optimize the classifier weights, architecture, and user-defined hyperparameters such as classification rate is maximized. Performance of this system is assessed in terms of classification rate and resource requirements for incremental learning of data blocks coming from real-world video data bases. The necessity of a LTM to store validation data is shown empirically for different enrollment and update scenarios. In addition, incremental learning is shown to constitute a dynamic optimization problem where the optimal hyperparameter values change in time. Simulation results indicate that the proposed system can provide a significant higher classification rate than that of fuzzy ARTMAP alone during incremental learning. However, optimization of ACS parameters requires more resources. The ACS needs several training sequences to produce the optimal solution, and adapting fuzzy ARTMAP parameters according to classification rate tends to require more category neurons and training epochs.

1. Introduction

Biometric systems seek to recognize individuals from their behavioral or physiological characteristics such as the face, finger print, iris, signature and voice [24]. Since these characteristics are unique for each individual, and cannot be lost, stolen or reproduced, as with current approaches (*e.g.*, passwords, access cards and identification numbers and cards), they can be used to prevent theft and fraud. There are three types of applications in biometric recognition – verification, identification, and surveillance [24]. In verification applications, an individual enrolled in the system identifies himself and provides a biometric sample. Then, the biometric system seeks to authenticate that the sample corresponds to the model of that specific individual. In contrast, in identification applications, an individual provides a biometric sample, and the system seeks to determine if the sample corresponds to the model of any of the individuals enrolled to the system. Surveillance applications differ slightly from identification in that the sampling process is performed discretely in an *unconstrained* scene, and it seeks to determine if a given biometric sample corresponds to the model of a restrained list of individuals under surveillance, *e.g.*, screening for criminals or terrorists in an airport setting.

[*] Corresponding author.
E-mail addresses: jfconnolly@livia.etsmtl.ca (J.-F. Connolly), Eric.Granger@etsmtl.ca (E. Granger), Robert.Sabourin@etsmtl.ca (R. Sabourin).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.02.026

Over the past decade, face recognition has received considerable attention in the area of biometrics due to the wide range of commercial and law enforcement applications, and to the availability of affordable technologies. *Video-based* face recognition has the advantage other very reliable characteristics for biometric recognition, such as iris and fingerprint scans, that it does not require the cooperation of individuals involved in the process [45]. It can thus be used for surveillance applications where control of the acquisition conditions are not possible. In addition, unlike applications of *image-based* face recognition, it is possible to recognize targeted subjects from a sequence of video frames, instead of only one image. As outlined in the following, video-based face recognition for surveillance applications remains a very challenging problem.

A critical function in face recognition systems is the classification of face regions captured in video streams. Typically, face recognition systems employ statistical or neural pattern classifiers to map an $\mathbb{R}^I$ input feature space to a set of K predefined class labels $\Omega = \{C_1, C_2, \ldots, C_K\}$, where each class k $(k = 1, \ldots, K)$ corresponds to the face model of an individual enrolled in the biometric system. From the classifier's perspective, an input pattern $\mathbf{a}$ associated with class k is sampled from an unknown probability distribution, $p_k(\mathbf{a})$, over the input feature space $\mathbb{R}^I$. In practical applications, the classifiers are designed a priori, using some prior knowledge of the underlying distributions $p_k(\mathbf{a})$, a set of user-defined *hyperparameters* (*e.g.*, learning parameter), and a limited amount of learning data.

Since the acquisition (collection and analysis) of such data is expensive and time consuming in many practical applications, it may therefore be incomplete in one of several ways. In *static classification environments*, where $p_k(\mathbf{a})$ remain fixed over time, these include a limited number of learning samples, missing components of the input observations, missing class labels during learning, and unfamiliar classes (not present in the learning data set) [20]. Moreover, in video-surveillance applications, learning samples acquired from video streams of *unconstrained* scenes are generally of poor quality with low resolution. They are also subject to considerable variations due to limited control over operational conditions (*e.g.*, illumination, pose, facial expression, orientation and occlusion). These challenges translate to very complex class distributions $p_k(\mathbf{a})$, mainly due to inter and intra-class variability. In addition to previously mentioned challenges, an individual's physiology may change over time, either temporarily (*e.g.*, haircut, glasses, etc.) or permanently (*e.g.*, ageing). In the $\mathbb{R}^I$ space, new informations, such as input features and output classes, may suddenly emerge, and previously acquired data may eventually become obsolete in *dynamic classification environments*, where class distributions $p_k(\mathbf{a}, t)$ vary or drift in time [20,40,43]. The overall result is a divergence between the biometric models learned by a classifier and the underlying distributions $p_k(\mathbf{a}, t)$ which may significantly degrade performance.

Although learning data is limited, it is common to acquire new data at some point in time after the classifier has originally been trained and deployed for operations. In particular, adaptation of video-based face recognition systems is required during enrollment (new classes are added to the system) and during update (pre-existing classes are refined using the new data). To avoid a growing divergence with the underlying class distributions $p_k(\mathbf{a}, t)$, the system should then efficiently adapt its face models as new learning data and knowledge becomes available.

The majority of statistical and neural pattern classifiers proposed in literature perform *supervised batch learning* of a finite data set, and assume a static classification environment. To account for new data, they must accumulate all cumulative data in memory and train from the start using all previously acquired learning data. Otherwise, new data may corrupt the classifier's previously acquired knowledge, and compromise its ability to achieve a high level of generalization during future operations. The memory and time complexity associated with storing and relearning from the start on all cumulative data is not feasible for several practical applications. Assuming that new learning data is available, a classifier that allows for *supervised incremental learning* should (1) allow learning of additional information from new data, (2) not require access to the previous learning data, (3) preserve previously acquired knowledge,[1] and (4) accommodate new classes that may be introduced with the new data [38]. Some classifiers proposed in literature are inherently able to perform supervised incremental learning: the Growing Self-Organizing Networks [15] and the ARTMAP Networks [7]. Other well known neural networks (MLP, SVM, and RBF) have also been modified to perform such learning [8,36,39]. In response to new learning data, these classifiers adapt their parameters (*e.g.*, synaptic weights for a neural network) and architecture according to these four incremental learning properties.

In order to mitigate corruption of previous knowledge when learning new data (3rd property), a 5th property should be considered for incremental learning – the classifier should (5) adapt its learning dynamics by adjusting its hyperparameters for accurate and timely recognition. In an unconstrained scene and dynamic classification environment, changes in the feature space are likely to occur over time, and re-adjustment of the classifier hyperparameters are needed. Incremental learning is then defined as a *dynamic optimization problem* in the hyperparameters space. Furthermore, the authors have shown in [10] that, unlike by the 2nd property stated, it is necessary to preserve some learning data for the validation process and fitness estimation. If not, adaptation is only performed according to new data, and the classifier is subject to the problem of catastrophic forgetting.

In this paper, an adaptive classification system (ACS) is proposed for video-based face recognition. It combines a fuzzy ARTMAP neural network classifier suitable for incremental learning [6], and a dynamic particle swarm optimization (DPSO) algorithm capable of finding and tracking several local optima in the optimization space [35]. This system also features a long term memory (LTM) used to store and manage a set of data for cross-validation and unbiased estimation of classification rate. A novel DPSO-based learning strategy is also proposed for incremental learning of new data with this ACS. When new data becomes available, this strategy allows to cojointly optimize the classifier weights, architecture, and user-defined hyperparameters such as classification rate is maximized.

[1] The problem of learning new information incrementally, yet preserving pre-existing knowledge is referred to as the *stability-plasticity dilemma* [5].

This study focuses on video-based face recognition applications in which two incremental learning scenarios may occur – enrollment and update. Performance of this system is assessed in terms of classification rate and resource requirements for incremental learning of new data blocks from two real-world video data sets – IIT-NRC [17] and Motion of Body (MoBo) [21]. First, the necessity of storing validation data in LTM is observed empirically by comparing the performances of fuzzy ARTMAP network trained (1) by using standard hyperparameter values, and (2) by optimizing hyperparameters on each new data block, in both cases, with and without LTM. Second, dynamic changes in the fuzzy ARTMAP hyperparameters space are shown to occur in both scenarios during incremental learning. Performance is compared for fuzzy ARTMAP networks trained by optimizing hyperparameters on all new data blocks with (1) dynamic optimization, (2) static optimization, (3) canonical particle swarm optimization, and (4) only on the first data block.

In the next section, a general biometric system for face recognition system is presented. Then, in Section 3, a description of the adaptive classification system is presented, along with the long term memory used to store and manage validation data, the fuzzy ARTMAP neural network used for classification, and the DPSO algorithm used to optimize its hyperparameters. Then, the data bases, incremental learning scenarios, performance measures and the protocol used for proof-of-concept simulations are described in Section 4. Finally, experimental results are presented and discussed in Section 5.

2. Biometrics and face recognition from video sequences

The adaptive classification system proposed in this paper is applied to the recognition of faces in video streams of a video-surveillance application and replaces the classification module and biometric data base of Fig. 1. However, it can also be employed to a wide range of real-world pattern recognition applications in which complex and changing environments are modelled using neural and statistical classifiers, but where learning data is limited. In face recognition applications, it is assumed that these systems capture a sequence of 2D images or video frames from the *real environment* (external 3D scene) via one fixed camera. Each frame provides the system with a particular view of individuals occupying the scene. First, the system performs segmentation on each frame to locate and isolate regions of interest (ROIs) corresponding to the faces in a frame. Invariant and discriminant features are then extracted from the ROIs and mapped to $\mathbb{R}^I$ feature space. Those feature patterns are employed for classification. That is, feature patterns are matched to the face model of individuals enrolled to the biometric system. Finally, classification scores are used to provide application-specific decisions. For verification applications, the decision module accepts or rejects the authenticity, and for identification and surveillance applications, it outputs a list of the most likely or of all possible matching identities, respectively.

A typical approach to recognizing faces in video consist in applying techniques developed for static 2D images on high quality ROIs produced through face segmentation. Several powerful techniques proposed to recognize faces in static 2D images are described in [44,45]. However, the performance of these techniques may degrade considerably when applied in unconstrained scenes.

More recently, some authors have combined spatial and temporal information contained in video sequences to provide a higher level of accuracy in unconstrained scenes [32]. These track-and-classify systems combine the responses of a classifier to kinematic information of individuals and faces in a scene. For instance, a distributed sensor network is proposed by Foresti and Snidaro [14] as a solution to the problem of partial occlusion that occurs in dynamics environments. Li and Chellappa [28] have introduced a face verification system which exploits the trajectories of Gabor facial features to identify individuals through hypothesis testing, using a posterior density characterized by the motion. A time series states space has been proposed by Zhou et al. [46] to fuse temporal information in video, which simultaneously characterizes the kinematics and identity of individuals in a probabilistic framework. Barry and Granger [1] have applied the What-and-Where Fusion neural network to the identification of individuals. This network simultaneously tracks multiple faces in an environment and accumulates their classifier predictions over time to improve classification. Matta and Dugelay [31] uses a multimodal system integrating the displacement signals of the head and physiological information with a probabilistic extension of the

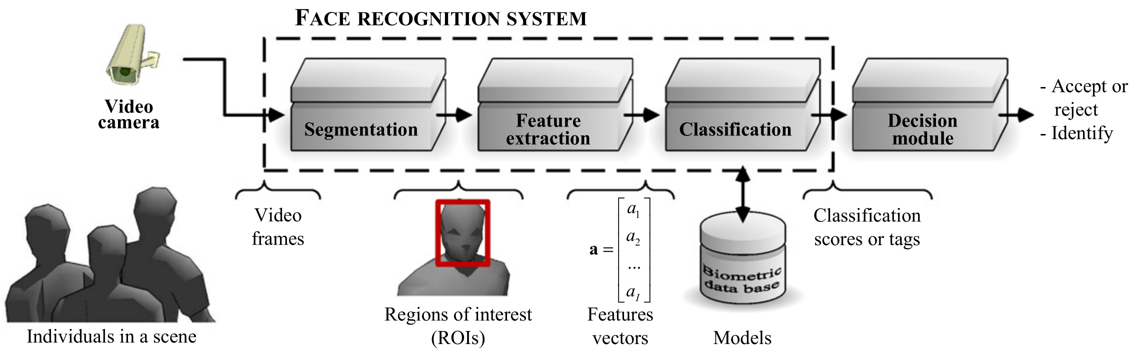

Fig. 1. A general biometric system for face recognition. In this paper, both classification module and biometric data base are replaced by the adaptive classification system.

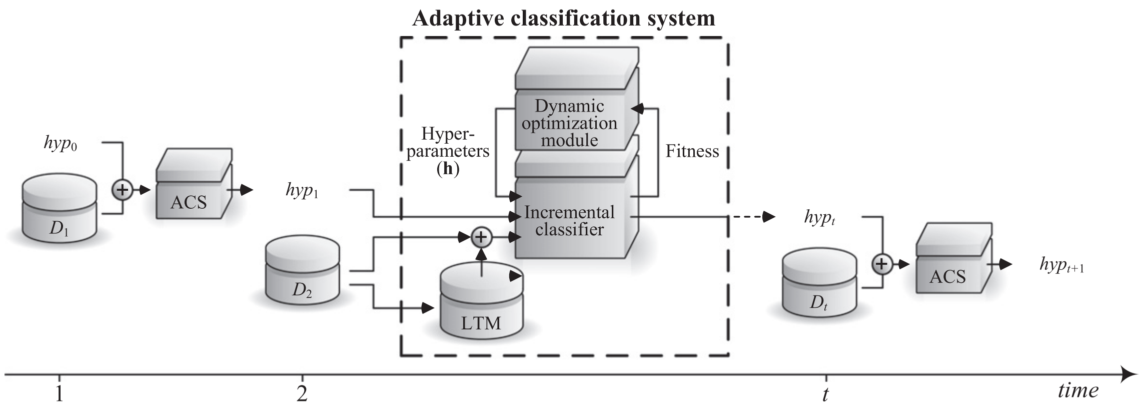

Fig. 2. The evolution of a new adaptive classification system (ACS) according to generic incremental learning scenario. New blocks of data are used by the ACS to update the classifier over time. Let $D_1, D_2, \ldots$ be blocks of learning data available at different instants in time. The ACS starts with an initial hypothesis hyp_0 which constitutes the prior knowledge of the domain. Each hypothesis hyp_{t-1} are updated to hyp_t by the ACS on the basis of the new data block D_t.

Eigenface approach. Majumdar and Nasiopoulos [30] proposed an image-to-image-based recognition approach that uses color information and a kernel classifier for face authentication. Finally, Mian [33] uses an unsupervised learning approach to determine the identity of an individual on the basis of best temporally cohesive matches between clusters of video sequence.

With these systems, the underlying data distribution $p(\mathbf{a})$ is considered static in nature and learning occurs only once, during a preliminary design phase. As discussed, once the face recognition system is deployed, temporary and permanent changes may occur in complex real-world environments and the initial learning data may no longer be representative nor sufficient to properly define the underlying class probability distributions $p_k(\mathbf{a}, t)$. This may lead to significant degradation in performance during operations. Assuming that new data becomes available, classifiers found in most face recognitions systems in literature would require relearning from the start using all previously acquired data through supervised batch learning. Performing incremental learning with only the new data would therefore be an undisputed asset as the memory and time complexity associated with storing and training is greatly reduced. In addition, it can maintain a high level of performance by reducing the divergence between class models and underlying distributions.

3. Adaptive classification system

Fig. 2 depicts the evolution of the adaptive classification system (ACS) proposed in this paper for supervised incremental learning of new data. This novel system is composed of a pattern classifier that is suitable for supervised incremental learning, a dynamic optimization module that tunes the user-defined hyperparameters of the classifier, and a long term memory (LTM) that manages and stores incoming learning data used for validation and fitness evaluation.

When a new block of learning data D_t becomes available to the system at a discrete time t, part of the data is employed to train the incremental classifier and update the LTM. The classifier then interacts with the dynamic optimization module using a DPSO-based algorithm that cojointly optimizes the vector of user-defined hyperparameters $\mathbf{h}$, parameters, and architecture such that classification rate maximized. In this paper, the fuzzy ARTMAP neural network [6] is employed as an incremental learning classifier and a dynamic version the particle swarm optimization (PSO) algorithm [26] is used for optimization.

Most techniques used to optimize fuzzy ARTMAP hyperparameters found in literature allow the optimization of only one or two hyperparameters, even though there are four inter-dependent parameters [3,12,16]. In previous work, the authors have introduced a PSO-based learning strategy for mono-objective optimization of all four hyperparameters [19]. It is based on the concept of neural network evolution in that it determines the optimal vector hyperparameters and network weights and architecture such that classification rate is maximized. The PSO strategy has been shown to provide a significantly higher classification rate on several synthetic and real-world data sets [1,19].

While a key feature of ARTMAP networks is their ability to learn new information incrementally, without catastrophic forgetting, those optimization methods have all been developed for batch supervised learning of a finite data set The adjustment of fuzzy ARTMAP hyperparameter vector[2] $\mathbf{h}$ is then defined as the static optimization problem such that:

$$\text{maximize}\{f(\mathbf{h});\ \mathbf{h} \in \mathbb{R}^4\}, \tag{1}$$

where the objective function $f(\mathbf{h})$ is the classification rate. In contrast, the ACS proposed in Fig. 2 performs incremental learning. As shown in Appendix A, incremental learning of new data from the class probability distributions $p_k(\mathbf{a}, k)$ translate to an

[2] Let $\mathbf{h}$ be a vector of user-defined hyperparameters that sets classifier dynamics. For fuzzy ARTMAP, it is composed of the four hyperparameters $\mathbf{h} = (\alpha, \beta, \epsilon, \rho)$ described in Section 3.2.

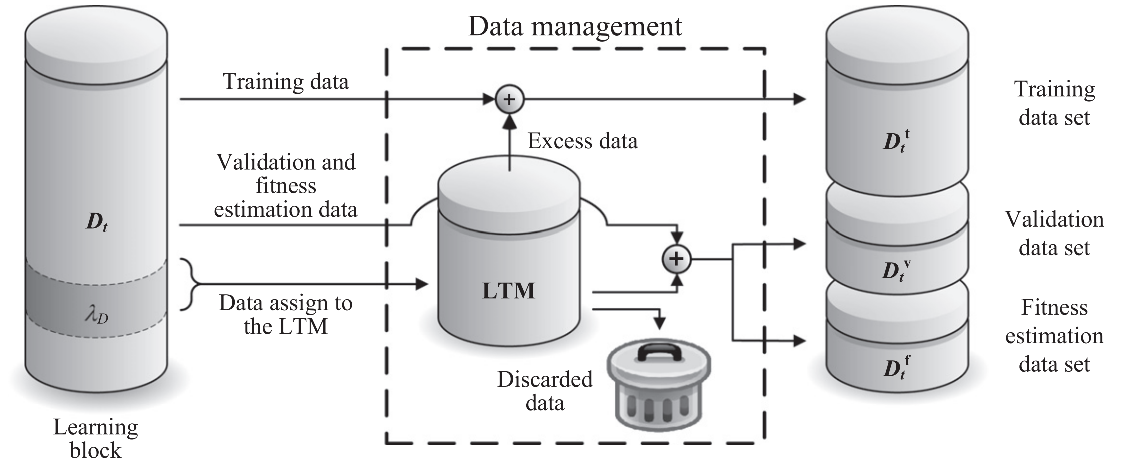

Fig. 3. Data management for the learning process using the long term memory. When a learning block D_t is available, a proportion λ_D of this data is assign to the long term memory, and the rest is used for training, validation, and performance estimation. When the LTM is updated, old data is discarded, while excess data not used to fill and/or update the LTM (dues to size limitations) is integrated to the training data from D_t to create the training data set D_t^{t}. Data contained in the LTM is then combined with data coming directly from D_t dedicated to validation and fitness estimation. This combination is class-wise divided in two, to create the validation data set D_t^{v} and the fitness estimation data set D_t^{f}.

objective function $f(\mathbf{h})$ that also changes in time. Adapting fuzzy ARTMAP hyperparameters vector **h** during incremental learning of data blocks D_t to maximize classification rate can thus be formulated as a dynamic optimization problem such as:

$$\text{maximize}\{f(\mathbf{h},t)\ \mathbf{h}\in\mathbb{R}^4,\ t\in\mathbb{N}_1\},\tag{2}$$

where $f(\mathbf{h},t)$ is the classification rate of fuzzy ARTMAP for a given vector of hyperparameters **h**, after learning data set D_t and at a discreet time t.

For an optimization space defined by fuzzy ARTMAP hyperparameters, three different types of dynamic optimization environment are then possible [13]:

- *type I environments* where the location of the optimum changes over time,
- *type II environments* where the location of the optimum remains fixed, but the value of the objective function at the position of the optimum changes, and
- *type III environments* where both the location and the value of optima points change.

As results presented in Appendix A suggest the presence of a type III optimization environment for fuzzy ARTMAP hyperparameters adjustment during incremental learning. The ACS employs a DPSO algorithm called Dynamic Niching PSO designed for such environments [35]. The rest of this section provides additional details on each part of the adaptive classification system: the long term memory, the fuzzy ARTMAP neural network, and the DPSO-based learning strategy.

3.1. Long term memory

During incremental learning, each new learning block of data D_t is divided into D_t^{t} and D_t^{v} for training with validation over several training epochs,[3] and into D_t^{f} for estimation of the fitness on the objective function $f(\mathbf{h},t)$.

It has been shown in [10], that the data sets used to guide the particles in the optimization space during a PSO-based incremental learning algorithm (D_t^{f}) should contain a representative set of samples from all classes $C_k\in\Omega$ to avoid a decline in fuzzy ARTMAP performance. As Fig. 3 depicts, some of the data of each learning block is used to create and maintain a long term memory (LTM). The LTM functions according to two parameters: (1) the proportion of D_t used to fill and update the external data base, λ_D, and (2) the maximal number of patterns per class in the external data base $|C_k|_{\mathrm{LTM}}$. Each time a new D_t is presented to the network a proportion λ_D of D_t is randomly selected and transferred to the LTM for either addition or update. The LTM is managed as a FIFO (first in, first out) data structure, and the outdated data that surpasses $|C_k|_{\mathrm{LTM}}$ is discarded. For each class, if the number of patterns transferred exceeds $|C_k|_{\mathrm{LTM}}$, the excess samples are randomly selected and integrated to D_t^{t}.

3.2. Fuzzy ARTMAP neural networks

ARTMAP refers to a family of self-organizing neural network architectures that is capable of fast, stable, on-line, unsupervised or supervised, incremental learning, classification, and prediction [7]. A key feature of ARTMAP networks is their un-

[3] An epoch is defined as one complete presentation of all the patterns of a finite training data set.

ique solution to the stability-plasticity dilemma. They can adjusts previously learned categories in response to familiar inputs, and creates new categories dynamically in response to inputs different enough from those previously seen.

Several ARTMAP networks have been proposed in order to improve the performance of these architectures. They can be broadly divided according to their internal matching process, which depends on either deterministic or probabilistic category activation. The deterministic type consists of networks such as fuzzy ARTMAP, ART-EMAP, ARTMAP-IC, default ARTMAP, simplified ARTMAP, distributed ARTMAP, etc., and represent each class using one or more category hyper-rectangles. In contrast, the probabilistic type consists of networks such as PROBART, PFAM, MLANS, Gaussian ARTMAP, ellipsoid ARTMAP, boosted ARTMAP, μARTMAP, etc., and represent each class using one or more probability density functions.

The fuzzy ARTMAP integrates the fuzzy ART to process both analog and binary-valued input patterns to the original ARTMAP architecture [6]. This simple and popular neural network has been designed with the ability to perform supervised incremental learning as defined in [38]. In supervised learning mode, the sequential learning process grows the number of recognition categories according to a problem's complexity. The vigilance and match tracking process provide the mechanisms to control the local impact of new data on the existing knowledge structure. Even if fuzzy ARTMAP is able to perform well with few training data [22], previous research by the authors has revealed that the average classification rate of an ARTMAP network trained through incremental learning is usually significantly lower than if trained on all the data through batch learning [9,18].

Fuzzy ARTMAP consists of three layers (Fig. 4): (1) an input layer F_1 of $2I$ neurons (for a $\mathbb{R}^I$ input feature space), (2) a competitive layer F_2 of J neurons, and (3) a map field F^{ab} of K neurons (the number of classes). The F_1 and F_2 layers are connected through a set of real-valued weights $\mathbf{W} = \{w_{ij} \in [0,1] : i = 1,2,\ldots,2I;\ j = 1,2,\ldots,J\}$ and the F_2 layer is connected, through learned associative binary weights $\mathbf{W}^{ab} = \left\{w^{ab}_{jk} \in \{0,1\} : j = 1,2,\ldots,J;\ k = 1,2,\ldots,K\right\}$, to a K nodes map field F^{ab}. Each F_2 node j represents a recognition category as an I-dimensional hyper-rectangle in the feature space, and is associated to one of the K output classes with the vector $\mathbf{W}^{ab}_j = \left(w^{ab}_{j1}, w^{ab}_{j2}, \ldots, w^{ab}_{jK}\right)$. The weights connected to each node correspond to a prototype vector $\mathbf{w}_j = (w_{1j}, w_{2j}, \ldots, w_{2Ij})$.

In supervised training mode, ARTMAP classifiers learn an arbitrary mapping between training set patterns $\mathbf{a} = (a_1, a_2, \ldots, a_I)$ and their corresponding binary supervision patterns $\mathbf{c} = (c_1, c_2, \ldots, c_K)$. These patterns are coded to have unit value $c_k = 1$ if k is the target class label for $\mathbf{a}$, and zero elsewhere. The following algorithm describes fuzzy ARTMAP learning:

1. *Initialization:* Initially, all the F_2 nodes are uncommitted, all weight values w_{ij} are initialized to 1, and all weight values w^{ab}_{jk} are set to 0. An F_2 node becomes committed when it is selected to code an input vector $\mathbf{a}$, and is then linked to an F^{ab} node. Values of the learning $\beta \in [0,1]$, choice $\alpha > 0$, match tracking $\epsilon = 0^+$, and baseline vigilance $\bar{\rho} \in [0,1]$ parameters are set.
2. *Input pattern coding:* When a training pair $(\mathbf{a}, \mathbf{c})$ is presented to the network, $\mathbf{a}$ undergoes a transformation called complement coding, which doubles its number of components. The complement-coded input pattern has $2I$ dimensions and is defined by $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) = \left(a_1, a_2, \ldots, a_I; a^c_1, a^c_2, \ldots, a^c_I\right)$, where $a^c_i = (1 - a_i)$, and $a_i \in [0,1]$. The vigilance parameter ρ is reset to its baseline value $\bar{\rho}$.
3. *Prototype selection:* Complement-coded pattern $\mathbf{A}$ activates layer F_1 and is propagated through weighted connections $\mathbf{W}$ to layer F_2. Activation of each node j in the F_2 layer is determined by the *Weber law choice function*:

$$T_j(\mathbf{a}) = \frac{|\mathbf{A} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{3}$$

where $|\cdot|$ is the L^1 norm operator defined by $|\mathbf{w}_j| \equiv \sum_{i=1}^{2I} |w_{ij}|$, $\wedge$ is the fuzzy AND operator, $(\mathbf{A} \wedge \mathbf{w}_j)_i \equiv \min(A_i, w_{ij})$, and α is the *choice parameter*. The F_2 layer produces a binary, winner-take-all pattern of activity $\mathbf{y} = (y_1, y_2, \ldots, y_J)$ such that only the node $j = j^*$ with the greatest activation value $j^* = \arg\max\{T_j : j = 1,2,\ldots,J\}$ remains active; thus $y_{j^*} = 1$ and $y_j = 0,\ j \neq j^*$. If more than one T_j is maximal, the node j with the smallest index is chosen. Node j^* propagates its top-

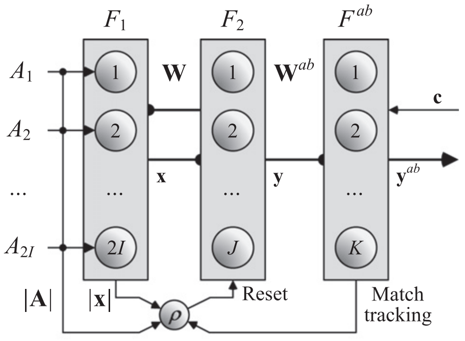

Fig. 4. Fuzzy ARTMAP neural network.

down expectation, or prototype vector $\mathbf{w}_{j^*}$, back onto F_1 and the *vigilance test* is performed. This test compares the degree of match between $\mathbf{w}_{j^*}$ and $\mathbf{A}$ against the dimensionless *vigilance parameter* $\rho \in [0,1]$:

$$\frac{|\mathbf{A} \wedge \mathbf{w}_J|}{|\mathbf{A}|} = \frac{|\mathbf{A} \wedge \mathbf{w}_J|}{I} \geqslant \rho. \tag{4}$$

If the test is passed, then node j^* remains active and resonance is said to occur. Otherwise, the network inhibits the active F_2 node (*i.e.*, T_{j^*} is set to 0) until Step 3 begins anew, and continues searching for another node j^* that passes the vigilance test. If such a node does not exist, an uncommitted F_2 node becomes active and undergoes learning (Step 5).

4. *Class prediction:* Pattern $\mathbf{c}$ is fed directly to the map field F^{ab}, while the F_2 category $\mathbf{y}$ learns to activate the map field via associative weights $\mathbf{W}^{ab}$. The F^{ab} layer produces a binary pattern of activity $\mathbf{y}^{ab} = (y_1^{ab}, y_2^{ab}, \ldots, y_K^{ab}) = \mathbf{c} \wedge \mathbf{w}_{j^*}^{ab}$ in which the most active F^{ab} node $k^* = \arg\max\{y_k^{ab} : k = 1, 2, \ldots, K\}$ yields the class prediction $(k^* = k(j^*))$. If node k^* constitutes an incorrect class prediction, then a *match tracking* signal adjust the vigilance parameter ρ according to:

$$\rho = \frac{|\mathbf{A} \wedge \mathbf{W}_{j^*}|}{I} + \epsilon, \tag{5}$$

the network deactivates node j^* until the network is presented with the next training pair $(\mathbf{a}, \mathbf{c})$, and another search is induced among F_2 nodes in Step 3. This search continues until either an uncommitted F_2 node becomes active (and learning directly ensues in Step 5), or a node j^* that has previously learned the correct class prediction k^* becomes active.

5. *Learning:* Learning input $\mathbf{a}$ involves updating prototype vector $\mathbf{w}_{j^*}$, and, if j^* corresponds to a newly-committed node, creating an associative link to F^{ab}. In the first case, the prototype vector of F_2 node j^* is updated according to:

$$\mathbf{w}'_{j^*} = \beta(\mathbf{A} \wedge \mathbf{w}_{j^*}) + (1 - \beta)\mathbf{w}_{j^*}, \tag{6}$$

where β is a fixed *learning rate parameter.* A new association between F_2 node j^* and F^{ab} node k^* $(k^* = k(J))$ is learned by setting $w_{j^*k}^{ab} = 1$ for $k = k^*$, where k^* is the target class label for $\mathbf{a}$, and 0 otherwise. The next training subset pair $(\mathbf{a}, \mathbf{c})$ is presented to the network in Step 2.

Once the weights $\mathbf{W}$ have been found through this process, ARTMAP can predict a class label for an input pattern by performing Steps 2–4 without any vigilance or match tests. During testing, a pattern $\mathbf{a}$ that activates node j^* is predicted to belong to class $k^* = k(j^*)$.

During training and testing fuzzy ARTMAP internal dynamic is governed by four user-defined hyperparameters: the choice parameter α, the learning parameter β, the match tracking parameter ϵ, and the baseline vigilance parameter $\bar{\rho}$. Each of these hyperparameters are inter-related, and has a distinct impact on network dynamics. While α and ϵ determine the depth of search attained before an uncommitted node is selected in the learning algorithm during Steps 3 (Eq. (3)) and 4 (Eq. (5)), $\bar{\rho}$ limits the maximum expansion of the recognition categories in the $\mathbb{R}^I$ feature space (Eq. (4)). Low vigilance allows large hyper-rectangles and leads to broad generalization and abstract memories, while high vigilance yields small hyper-rectangles, leading to narrow generalization and detailed memories. During Step 5, β determines the speed with which the recognition categories are expanded to fit $\mathbf{a}$. The algorithm can be set to slow learning with $0 < \beta < 1$, or to fast learning with $\beta = 1$. With fast learning, each hyper-rectangle is just large enough to enclose the cluster of training set patterns $\mathbf{a}$ to which it has been assigned. That is, an I-dimensional prototype vector $\mathbf{w}_j$ records the largest and smallest component values of training subset patterns $\mathbf{a}$ assigned to category j. A standard vector of hyperparameters $\mathbf{h}_{\text{std}} = (\alpha = 0.001, \beta = 1, \epsilon = 0.001, \bar{\rho} = 1)$ is commonly used to minimize network complexity [6].

3.3. Dynamic particle swarm optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique that was inspired by social behavior of bird flocking or fish schooling [25,26]. With PSO, each particle corresponds to a single solution in the optimization space, and the population of particles is called a swarm. Particles move through the optimization space and change their course under the guidance of a cognitive influence (*i.e.*, their own previous search experience) and a social influence (*i.e.*, their neighborhood previous search experience) and unlike evolutionary algorithms (such as genetic algorithms), each particle always keep in memory its best position and the best position of its surrounding.

Originally developed for static optimization problems, the PSO algorithm has been adapted for dynamic optimization problems by adding mechanisms to (1) modify the social influence to maintain diversity in the optimization space and detect several optima, (2) detect changes in the objective function by using the memory of each particle, and (3) adapt the memory of its population if change occur in the optimization environment. The latest PSO algorithms developed to insure diversity in the swarm are presented in [11,29,34,37], while change detection and memory adjustment mechanisms are presented in [2,4,23,42].

When the ACS learns a new data blocks D_t (Fig. 2), a Dynamical Niching PSO (DNPSO) algorithm adapted for dynamic optimization [35] is used to maximize fuzzy ARTMAP classification rate as function of its hyperparameters vector $\mathbf{h} = (\alpha, \beta, \epsilon, \bar{\rho})$. The optimization space is defined by the four fuzzy ARTMAP hyperparameters, and the performance of each particle's position is its value on the objective function $f(\mathbf{h}, t)$.

This PSO algorithm is simple to implement and has been shown to rapidly converge toward global maximum in a multimodal type III optimization environment with the moving peaks benchmark [35]. It maintains diversity with a local neighborhood topology and by dynamically creating subswarms around certain particles, called masters, that are their own best position amongst their neighborhood. Particles that are not part of any subswarms are called free particles and are allowed to move by themselves. Once the subswarms have been defined, position of particles that are members of a subswarm are updated using

$$\mathbf{h}(\tau+1) = \mathbf{h}(\tau) + w_1(\mathbf{h}(\tau) - \mathbf{h}(\tau-1)) + r_1(\tau)w_2/2\left(\mathbf{h}^*_{master} - \mathbf{h}(\tau)\right) + r_2(\tau)w_2/2(\mathbf{h}^*_n - \mathbf{h}(\tau)), \tag{7}$$

where $\mathbf{h}_n(\tau)$ is the position of particle n in the optimization space at iteration τ, w_1 and w_2 are inertia weights, $r_1(\tau)$ and $r_2(\tau)$ are random numbers generated at each iteration, $\mathbf{h}^*_{master}$ and $\mathbf{h}^*_n$ are respectively the current position of the subswarm master's personal best (social influence) and particle n personal best (cognitive influence). On the other hand, free particles move only according to their own cognitive influence using:

$$\mathbf{h}(\tau+1) = \mathbf{h}(\tau) + w_1(\mathbf{h}(\tau) - \mathbf{h}(\tau-1)) + r_3(\tau)w_2\left(\mathbf{h}^*_n - \mathbf{h}(\tau)\right), \tag{8}$$

where $r_3(\tau)$ is another random number generated at each iteration. The global best particle is referred to as *gbest*, and in case there is a tie for the global best position, the particle with the smallest index wins. If the maximal number of subswarms is set to one, its maximal size and the neighborhood size is equal to the swarm's total number of particles, the DNPSO is then equivalent to the canonical PSO described in [26]. All the particles will then converge toward the only master (*i.e.*, the global best) according to Eq. (7).

Initially developed in [34], DNPSO was adapted for dynamic optimization problem by simply updating the performance of their best position $f\left(\mathbf{h}^*_n, t\right)$ at each iteration. Normally, for DPSO algorithms, this would double the number of time values on the objective function are evaluated, leading to a very costly process. For our ACS, changes in the objective function only occur only when a new data block D_t becomes available. Thus, the performance of the particles best position is only updated when D_t is presented to the system, *before* the iterative DNPSO process.

Algorithm 1 describes the DPSO-based incremental learning strategy for co-optimization of hyperparameters, weight and architecture of the fuzzy ARTMAP neural network. Given new learning data block D_t, it produces the optimal set of hyperparameters and network using a particle swarm with N particles, and $N+2$ fuzzy ARTMAP neural networks – one network per particle FAM_n, used to preserve the model associated to the best position of that particle $\left(\mathbf{h}^*_n\right)$, one temporary neural network used for the fitness estimation during the algorithm $(FAM_{\text{estimation}})$, and one optimal network (FAM_{optimal}).

Algorithm 1. DPSO-based incremental learning strategy for the ACS using a fuzzy ARTMAP neural network classifier

```
Input: A particle swarm with DNPSO parameters, neural networks: $FAM_p$, where $1 \leqslant p \leqslant P$, $FAM_{\text{estimation}}$, and $FAM_{\text{optimal}}$,
  and new data sets $D_t$ for learning.
Output: (1) $FAM_{\text{optimal}}$ (weights and architecture obtained with the optimal $\mathbf{h}$) and (2) $FAM_n$ where $1 \leqslant n \leqslant N$ (set of
  fuzzy ARTMAP neural networks associated to the best position of each particles).
1:  Set the swarm parameters $(N, w_1, w_2)$.
2:  Randomly initialize particles positions for $t = 0$ and $t = -1$ within their range.
3:  Initialize $FAM_{\text{optimal}}$ and all $FAM_n$, where $1 \leqslant n \leqslant N$.
4:  Set PSO iteration counter at $\tau = 0$.
5:  for each new $D_t$ do
6:    for each particle $n$, where $1 \leqslant n \leqslant N$ do
7:      $FAM_n \leftarrow FAM_{\text{optimal}}$
8:      Training of $FAM_n$ with validation using $D^{\text{t}}_t$ and $D^{\text{v}}_t$, and $f\left(\mathbf{h}^*_n, t\right)$ estimation using $D^{\text{f}}_t$.
9:    end for
10:   while DNPSO did not reach stopping condition do
11:     Define the subswarms and update position of each particle with Eqs. (7) and (8).
12:     for each particle $n$, where $1 \leqslant n \leqslant N$ do
13:       $FAM_{\text{estimation}} \leftarrow FAM_{\text{optimal}}$
14:       Training of $FAM_{\text{estimation}}$ with validation using $D^{\text{t}}_t$ and $D^{\text{v}}_t$, and $f(\mathbf{h}_n, t)$ estimation using $D^{\text{f}}_t$.
15:       if $f(\mathbf{h}_n(\tau), t) > f\left(\mathbf{h}^*_n, t\right)$ then
16:         $\mathbf{h}^*_n \leftarrow \mathbf{h}_n(\tau)$
17:         $f\left(\mathbf{h}^*_n, t\right) \leftarrow f(\mathbf{h}_n(\tau), t)$
18:         $FAM_n \leftarrow FAM_{\text{estimation}}$
19:       end if
20:     end for
21:     $\tau = \tau + 1$
22:   end while
23:   $FAM_{\text{optimal}} \leftarrow FAM_{gbest}$
24: end for
```

First, at Line 1, the DPSO swarm's parameters are set according to the DNPSO algorithm. Each particle position is then randomly initialized within their allowed range (Line 2). All the neural networks (FAM_n, $FAM_{\text{estimation}}$, and FAM_{optimal}) are initialized as described in Step 1 of the fuzzy ARTMAP learning algorithm (Line 3). To comply with Eqs. (7) and (8), a position at $t = -1$ is set in order to have an initial velocity. When a new block D_t becomes available, the optimization process continues where it stopped with D_{t-1} and the DNPSO algorithm updates the swarm's memory (Lines 6–9).[4] The network FAM_{optimal} found with D_{t-1} is then copied to each FAM_n, and thus serves as the initial condition for learning of D_t. For the first learning block, FAM_{optimal} will be in an initial state. FAM_n is then trained with validation using D_t^{t} and D_t^{v}, its classification rate is estimated using D_t^{f} and defined as the particle personal best fitness, $f(\mathbf{h}_n^*, t)$. Since the fitness is defined by the classification rate obtained with D_t^{f}, if there is a tie for the personal best position, the particle n with the smaller number of recognition categories is the personal best. The same procedure is also used to find the swarm's global best.

Unless the stopping criteria are reached (defined in Section 4), the DNPSO algorithm will iteratively evaluate each particle's fitness and update their position. The DNPSO algorithm first defines the subswarms and free particles, and computes the new particle positions using Eqs. (7) and (8) (Line 11). For each particle, the FAM_{optimal} found from D_{t-1} is copied to $FAM_{\text{estimation}}$ prior training (Line 13). $FAM_{\text{estimation}}$ is then trained using D_t^{t} and D_t^{v}, and its fitness is estimated on the basis of D_t^{f} (Line 14). Personal best position, fitness, and neural networks associated to the personal best FAM_n are then updated accordingly (Lines 16–18). Once the optimization process is completed for D_t, the FAM_{gbest} network, associated to the best vector of hyperparameters $\mathbf{h}_{gbest}$ is stored as FAM_{optimal} to preserve an optimal set of hyperparameters and network throughout the learning process (Line 23).

To minimize the impact of pattern presentation order on fuzzy ARTMAP performance in the DPSO-based strategy, Lines 7–8 and Lines 13–14 of Algorithm 1 are replaced with Algorithm 2. When a network FAM_{temp} is input, the classification rate is assessed on D_t^{f} for fuzzy ARTMAP trained on D_t^{t} over five different random pattern presentation orders. Fitness estimation is defined by the mean classification rate of those five replications, and the neural network trained with the best pattern presentation order. For each random patterns presentation order of D_t^{t}, FAM_{optimal} is copied in FAM_{temp}, FAM_{temp} is trained using D_t^{t} and D_t^{v}, and classification rate over D_t^{f} is evaluated. The FAM_{temp} network that provides the best classification rate is copied to $FAM_{\text{estimation}}$, and $f(\mathbf{h}, t)$ is defined as the mean classification rate over the five replications (Lines 9–10).

Algorithm 2. Evaluation of particle fitness for the DPSO incremental learning strategy

```
Input: Best temporary network, $FAM_{\text{temp}}$.
Output: A particle's performance and the best neural network to obtained that performance.
1:   Initialize $FAM_{\text{temp}}$
2:   for 5 patterns presentation order do
3:     $FAM_{\text{temp}} \leftarrow FAM_{\text{optimal}}$
4:     Training of $FAM_{\text{temp}}$ with validation using $D_t^{\text{t}}$ and $D_t^{\text{v}}$, and $f(\mathbf{h}, t)$ estimation using $D_t^{\text{f}}$
5:     if classification is the best so far then
6:       $FAM_{\text{temp}} \leftarrow FAM_{\text{estimation}}$
7:     end if
8:   end for
9:   $FAM_{\text{estimation}} \leftarrow FAM_{\text{temp}}$
10:  $f(\mathbf{h}(\tau), t) \leftarrow$ mean classification rate of the 5 replications
```

The computational time of Algorithm 1 at a time t depends on the number of: training patterns ($|D_t^{\text{t}}|$), training epochs, FAM_nF_2 layer nodes (J_n), input features (I), DNPSO particles (N), DNPSO iterations before the optimization stopping conditions are met (τ^*), and replications in Algorithm 2. Amongst those variables, N and the number of replications during Algorithm 2 are constant values, and the number of training epochs and τ^* are limited to maximal values after which training and optimization are forced to stop. During incremental learning, the time complexity of the ACS is defined as the worst-case execution time required learn a new D_t. In the worse case scenario, the hyperparameters are set to build large neural network such as $J_n = |D_1^{\text{t}} \cup \ldots \cup D_t^{\text{t}}|$ and complexity of Algorithm 1 is then $O(J_n|D_t^{\text{t}}|I)$. During operation, the time complexity to process one input pattern is $O(J_nI)$. This is comparable to that of a fuzzy ARTYMAP neural network alone.

The complexity of the DPSO strategy also depends on the time create subswarms and manage particle positions during DNPSO optimization process. However, in all cases, the complexity for the fitness evaluation of one particle is always greater than that of the DNPSO algorithm when it creates the subswarms and manages the particles in the optimization space. Thus, the complexity of the DNPSO algorithm itself should not be taken into consideration when using Algorithm 1.

[4] Dynamic PSO algorithms usually involve change detection at this point, while for static optimization, no detection is done and the swarm's memory remains intact.

4. Experimental methodology

4.1. Video data bases

In order to observe the impact on system performance of supervised incremental learning, proof-of-concept simulations are performed with two real-world video data bases for face recognition. The first data base was collected by the Institute for Information Technology of the Canadian National Research Council (IIT-NRC) [17]. It is composed of 22 video sequences captured from 11 individuals positioned in front of a computer. For each individual, two color video sequences of about 15 s are captured at a rate of 20 frames/s with an Intel webcam of a 160×120 resolution that was mounted on a computer monitor. Of the two video sequences, one is dedicated to training and the other to testing. They are taken under approximately the same illumination conditions (no sunlight, only ceiling light evenly distributed over the room), the same setup, and almost the same background. For all persons in the data base, each face occupies between 1/4 to 1/8 of the image. This data base contains a variety of challenging operational conditions such as motion blur, out of focus factor, facial orientation, facial expression, occlusion, and low resolution.

Face detection is performed using the Viola–Jones algorithm included in the OpenCV C/C++ computer vision library [41]. It produced regions of interest (ROIs) between 29×18 and 132×119 pixels for each face detection in a video sequence. The number of ROIs detected per class for the IIT-NRC database is displayed in Table 1. The features presented to the classifier are independent of camera resolution and color since the ROIs are converted in grayscale and normalized to 24×24 images where the eyes are aligned horizontally, with a distance of 12 pixels between them. Each ROI is vectorized into $\mathbf{a} = \{a_1, a_2, \ldots, a_{576}\}$, where each feature $a_i \in [0, 1]$ represents a normalized grayscale value.

The second video data base is the Motion of Body (MoBo), and was collected at Carnegie Mellon University under the HumanID project [21]. Each video sequences show one of 25 different individuals walking on a tread-mill so that they move their heads naturally to four different motion types when walking: slowly, fast, on an inclined surface, and while carrying an object. Six cameras are positioned at different locations around the individuals, and for each angle, individuals are filmed with a Sony DXC 9000 camera with a resolution of a 640×480 pixels.

The video sequences with visible faces (full frontal view and both sides with an angle of about 45° with the full frontal view) where processed with the Viola–Jones algorithm for all four types of walk. This data base was reduced in order to use roughly the same size of the IIT-NCR data base, while having, for each individual, the same number of ROIs from each motion types and camera angle. Data from 10 individuals was employed, with 288 ROIs per class (24 ROIs for each type of walk and camera angle) for a total of 2880 patterns. The data base was divided into a learning and test data sets of 1440 patterns each. For each type of walk and camera angle, the first 12 of the 24 RIOs sequence were assign to the learning data set, while the last 12 were assign to the test data set. The ROIs were scaled and vectorized into $\mathbf{a} = \{a_1, a_2, \ldots, a_{576}\}$ as with the IIT-NRC data base.

4.2. Incremental learning scenarios

Prior to computer simulations, each video data set is divided in blocks of data D_t, where $1 \leqslant t \leqslant T$, to emulate the availability of T successive blocks of training data to the ACS. Supervised incremental learning is performed according two different scenario.

4.2.1. Enrollment

In this scenario, each block contains ROIs of individuals that are not enrolled to the system. Classes are added incrementally to the system, one at a time. To assess ACS performance, the first learning bloc D_1 is composed of two classes, and each successive block D_t, where $2 \leqslant t \leqslant K - 1$, contains the ROIs captured in a video sequence corresponding to an individual that has not previously been enrolled to the system. For each D_t, performance is only evaluated for existing classes. To insure the invariance of results to class presentation order, this experiment is performed using five different random *class* presentation orders.

4.2.2. Update

In this scenario, each block contains ROIs of individuals that have previously been enrolled to the system. It is assumed that at a given time, the ROIs an individual is captured in a video sequence, and then learned by the system to refined its internal models. To assess ACS performance, all classes are initially learned with the first data block D_1 and are refined

Table 1
Number of learning and test patterns per individual ($C_k \in \Omega$) for the IIT-NRC data base.

Number of ROIs	Individuals											Total
	1	2	3	4	5	6	7	8	9	10	11	
Learning data	140	39	160	130	175	128	180	97	178	160	140	1527
Test data	142	40	159	131	186	134	190	100	188	168	147	1585

Table 2
DNPSO parameters.

Parameter	Value
Swarm's size N	20
Weights $\{w_1, w_2\}$	$\{0.73, 2.9\}$
Maximal number of subswarms	4
Maximal size of each subswarm	4
Neighborhood size	5
Minimal distance between two masters (in a normalized $\mathbb{R}^4$ space)	0.2
Minimal velocities of free particles (in a normalized $\mathbb{R}^4$ space)	0.0001

one class at a time with blocks D_2 through D_{K+1}. In order to better observe cases where classes are not initially well defined, block D_1 is composed of 10% of the data for each class, and each subsequent block D_t, where $2 \leqslant t \leqslant K+1$, is composed of the remaining 90% of one specific class. Here again, invariance to class order presentation is insured by repeating this experimentation with five different *class* presentation orders.

4.3. Experimental protocol

Learning is performed over 10 independent trials using 10-folds cross-validation.[5] Out of the 10-folds, eight are dedicated to training (D_t^{t}), one-fold is combined with half of LTM to validate and determine the number of fuzzy ARTMAP training epochs (D_t^{v}), and the remaining fold is combined with the other half of LTM to estimate the fitness of each particle during the PSO algorithm $\left(D_t^{\mathrm{f}}\right)$. In this paper, initialization and update of the LTM is performed with $\lambda_D = 1/6$ and $|C_k|_{\mathrm{LTM}} = 20$. For reference, the performance of fuzzy ARTMAP trained with the canonical PSO learning strategy, and kNN are given for batch learning. At a given time t, the batch learning methods consist of initializing the system, and learning all the data obtained thus far by incremental learning in one of block of data (*i.e.*, a batch learning block is defined by $B_t = D_1 \cup \cdots \cup D_t$). During batch learning, data is also separated in folds for 10-fold cross-validation particles fitness estimation. Two experiments are presented with both enrollment and update incremental learning scenarios using the proposed ACS. In experiment (A), the impact of storing validation data in a LTM for fitness estimation is assessed. The performance of an ACS based on fuzzy ARTMAP when it is train using:

1. $\mathbf{h}_{\mathrm{ro}}(t) \leftarrow$ system parameters[6] that are re-optimized on each learning block D_t using canonical PSO, and
2. $\mathbf{h}_{\mathrm{std}} \leftarrow$ hyperparameters that are set to standard values.

In all cases, training is performed with and without the use of the LTM. When training without LTM, all the data contained in D_t is divided in 10-folds for the 10-fold cross-validation. For re-optimized system parameters, the swarm is re-initialized and a new PSO optimization process is triggered every time a new D_t is presented to the system (unlike Algorithm 1).

Experiment (B) seeks to show the impact on performance that may be achieved using the proposed ACS for supervised incremental learning under the hypothesis that the objective function $f(\mathbf{h}, t)$ is indeed a type III dynamic optimization environment and that a specialized dynamic version of PSO is required to achieved high level of performance. Performance is assessed when fuzzy ARTMAP is trained using the external data base and:

1. $\mathbf{h}_{\mathrm{dnc}}(t) \leftarrow$ system parameters that are optimized on each learning block D_t using dynamic optimization with DNPSO (the swarm's memory is updated each time with each new D_t, *prior to* starting the optimization algorithm),
2. $\mathbf{h}_{\mathrm{dnc}}(1) \leftarrow$ system parameters that are optimized on only D_1 using DNPSO and are then fixed.
3. $\mathbf{h}_{\mathrm{stc}}(t) \leftarrow$ system parameters that are optimized using static optimization with DNPSO (the swarm's memory is *not* updated with each new D_t), and
4. $\mathbf{h}_{\mathrm{cnl}}(t) \leftarrow$ system parameters that are optimized using static optimization with canonical PSO (again, the swarm's memory is *not* updated with each new D_t).

As it is mentioned, a particle's personal best position are not updated with a static optimization method. When using Algorithm 1 with $\mathbf{h}_{\mathrm{stc}}(t)$ and $\mathbf{h}_{\mathrm{cnl}}(t)$, this simply means that Lines 6–9 are never executed. Unlike with experiment (A), system parameters are continuously optimized: the swarm's position at the beginning of the optimization process for D_t is the same as at the end of D_{t-1}.

The DNPSO parameters are set as specified by Table 2. Since the distances between particles are measures in the DNPSO algorithm, a swarm evolves in a *normalized* $\mathbb{R}^4 \in [0,1]$ space to avoid each hyperparameter's domain. The positions are then denormalized to fit the hyperparameters domain ($\alpha \in [0.001, 100]$, $\beta \in [0,1]$, $\epsilon \in [-1,1]$, and $\bar{\rho} \in [0,1]$) before being applied

[5] Within each replication, there are five different trials using different class presentation order, for a total of 50 replications.
[6] System parameters are the hyperparameters, weights and architecture of each fuzzy ARTMAP neural network.

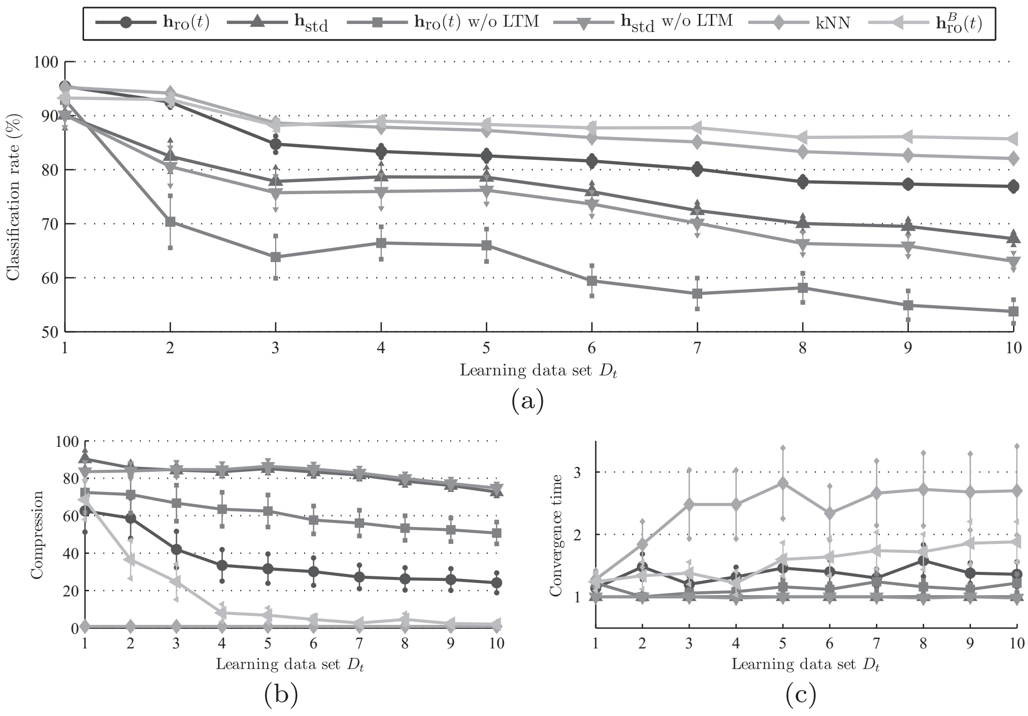

Fig. 5. Average classification rate, compression, and convergence time of the ACS versus learning block during the enrollment scenario. Performance was evaluated with and without LTM for $\mathbf{h}_{ro}(t)$ and $\mathbf{h}_{std}$. Error bars correspond to the 90% confidence interval. The performance for fuzzy ARTMAP with $\mathbf{h}^B_{ro}(t)$ and kNN during batch learning are shown for reference.

to fuzzy ARTMAP. For each D_t, the DNPSO optimization process is set to either stop after 10 iterations without improvement to the *gbest* classification rate, or after 100 iterations (for the current D_t).

Although the hyperparameters are not necessarily optimized, Algorithm 2 is *always* applied to minimize the impact of patterns presentation order. In other words, even when **h** does not change ($\mathbf{h}_{std}$ and $\mathbf{h}_{dnc}(1)$), the particles performance evaluation data set D^f_t is still used to find the best network out of the five replications.

The average performance of fuzzy ARTMAP is assessed in terms of classification rate and resources requirements. The amount of resources is measured by compression and convergence time. *Classification rate* is estimated as the ratio of correctly classified test subset patterns over all test set patterns, *compression* refers to the average number of training patterns per category prototype created in the F_2 layer, and *convergence time* is the number of training epochs required to complete learning. It does not include presentations of validation subsets used to perform cross-validation.

5. Results and discussion

5.1. Experiment (A) – Impact of the LTM for validation data

Figs. 5 and 6 present the average classification rate, compression, and convergence time achieved by the ACS with and without LTM data, and for hyperparameters that are re-optimized ($\mathbf{h}_{ro}(t)$) and standard hyperparameters ($\mathbf{h}_{std}$), during both incremental learning scenarios. For reference, performance is also shown for hyperparameters re-optimized during batch learning $\mathbf{h}^B_{ro}(t)$ and kNN. Tables 3 and 5 show an example of the average confusion matrix for only one of the five class presentation orders (*i.e.*, 10 replications out of 50). That is, the classification rate of each learned class at a time t ($C_{k'}(t)$) in the set of predefined classes Ω, versus all other classes defined at that time ($\{C_k \in \Omega(t)|k \neq k'\}$). For the update scenario, all classes are defined from the start and $\Omega(t) = \Omega$. Finally, the classification rate of each specific classes for the enrollment and update scenarios is given in Tables 4 and 6, respectively, after learning all blocks of data, for the same class presentation order.

5.1.1. Enrollment scenario

As Fig. 5 depicts, best classification rate during incremental learning is achieved by the ACS with LTM and $\mathbf{h}_{ro}(t)$. When using the LTM, 1/3 of the data available in D_1 is used for validation and fitness estimation, compared to 1/5 when no LTM is employed. Since only two classes are present in D_1, data distribution for the first blocks is relatively simple. Results obtained after D_1 then indicate that if the same learning strategy is applied, classification rate obtained with larger validation and fit-

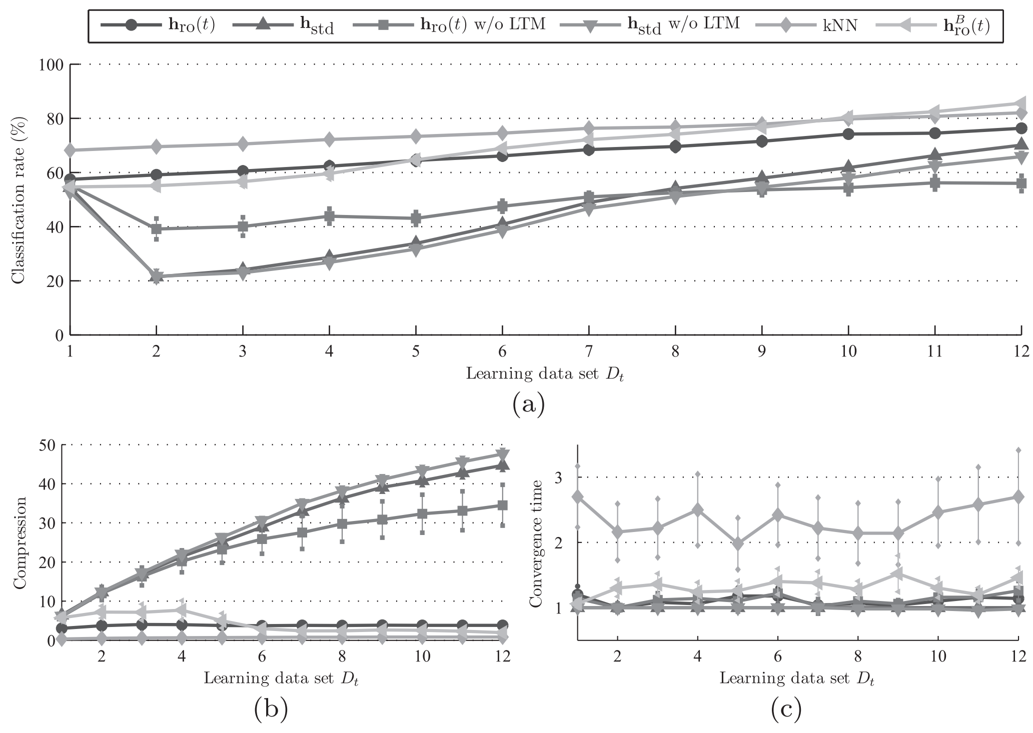

Fig. 6. Average classification rate, compression, and convergence time of the ACS versus learning block during the update scenario. Performance was evaluated with and without LTM for $\mathbf{h}_{\mathrm{ro}}(t)$ and $\mathbf{h}_{\mathrm{std}}$. Error bars correspond to the 90% confidence interval. The performance for fuzzy ARTMAP with $\mathbf{h}_{\mathrm{ro}}^{B}(t)$ and kNN during batch learning are shown for reference.

Table 3
Average classification rate achieved by the ACS for the added classes with each learning block D_t for one class presentation order during the enrollment scenario. The classification rate of the new class added with D_t ($C_{k'}(t)$) is presented with that of the remaining classes present at that time ($\{C_k \in \Omega(t) | k \neq k'\}$). Each cell is presented in percentage and with the 90% confidence interval.

Training strategy	D_t $C_{k'}(t)$	D_2 11	D_3 5	D_4 4	D_5 8	D_6 9	D_7 7	D_8 3	D_9 6	D_{10} 2
$\mathbf{h}_{\mathrm{ro}}(t)$ w/LTM	Class. rate for $C_{k'}(t)$	90 ± 6	84 ± 5	95 ± 4	99 ± 1	71 ± 8	75 ± 5	68 ± 7	89 ± 4	49 ± 5
	Class. rate for $\{C_k \in \Omega(t) \| k \neq k'\}$	94 ± 1	76 ± 4	79 ± 3	80 ± 2	82 ± 2	79 ± 2	78 ± 2	76 ± 2	77 ± 2
$\mathbf{h}_{\mathrm{ro}}(t)$ w/o LTM	Class. rate for $C_{k'}(t)$	100 ± 1	95 ± 2	96 ± 2	98 ± 1	87 ± 4	88 ± 4	77 ± 13	91 ± 7	72 ± 11
	Class. rate for $\{C_k \in \Omega(t) \| k \neq k'\}$	31 ± 22	36 ± 16	54 ± 10	61 ± 8	48 ± 6	51 ± 9	56 ± 9	47 ± 6	52 ± 6

Table 4
Average classification rate per class for one class order presentation of the enrollment incremental learning scenario for $\mathbf{h}_{\mathrm{ro}}(t)$ and $\mathbf{h}_{\mathrm{std}}$, with and without the LTM. Results are obtained after enrollment of all classes $C_k \in \Omega$. Each cell is presents the classification rate in percentage along with the 90% confidence interval.

Training strategy	$C_k \in \Omega$											Ω
	1	2	3	4	5	6	7	8	9	10	11	
$\mathbf{h}_{\mathrm{ro}}(t)$ w/LTM	85 ± 5	49 ± 5	91 ± 5	73 ± 4	49 ± 9	88 ± 5	81 ± 4	70 ± 7	82 ± 4	61 ± 7	91 ± 3	77 ± 1
$\mathbf{h}_{\mathrm{std}}$ w/LTM	62 ± 24	5 ± 4	84 ± 5	56 ± 6	93 ± 6	78 ± 4	67 ± 5	32 ± 7	83 ± 2	37 ± 7	77 ± 4	67 ± 1
$\mathbf{h}_{\mathrm{ro}}(t)$ w/o LTM	62 ± 24	72 ± 11	45 ± 25	66 ± 19	21 ± 19	73 ± 21	61 ± 17	37 ± 25	67 ± 12	47 ± 17	39 ± 25	54 ± 2
$\mathbf{h}_{\mathrm{std}}$ w/o LTM	36 ± 8	7 ± 5	81 ± 8	47 ± 10	87 ± 7	78 ± 5	68 ± 7	43 ± 11	81 ± 4	40 ± 7	75 ± 6	63 ± 2

ness estimation data sets (D_t^{v} and D_t^{f}) yields higher classification rates. For example, when $\mathbf{h}_{\mathrm{ro}}(t)$ is used, the classification rate with LTM is 95.4 ± 0.6%, versus 93 ± 1% without LTM.

As the amount of training data and the complexity of the decision boundaries increase, all hyperparameters settings follow the same degradation in classification rate. After learning all data, the highest performance is obtained with batch learning (85.6 ± 0.3% for $\mathbf{h}_{\mathrm{ro}}^{B}(t)$ and 82.3 ± 0.1% for kNN), followed by incremental learning with $\mathbf{h}_{\mathrm{ro}}(t)$ and the LTM ($77 \pm 1\%$), $\mathbf{h}_{\mathrm{std}}$ with and without the LTM (67 ± 1% and 63 ± 2%), and finally $\mathbf{h}_{\mathrm{ro}}(t)$ without the LTM (54 ± 2%).

However, higher classification rate comes with a cost. Fig. 5(b) shows that compression, when using $\mathbf{h}_{\mathrm{ro}}(t)$ with the LTM starts lower than that obtained without LTM (63 ± 11 versus 69 ± 11), decreases to 32 ± 8 at D_5 (compared to 63 ± 9 without the LTM), and does not change significantly afterwards. Moreover, the average number of training epochs needed when using LTM (1.4 ± 0.1) is higher than that of $\mathbf{h}_{\mathrm{ro}}(t)$ without LTM (1.1 ± 0.1), confirming that using a LTM with larger data sets for validation leads to a greater number of training epochs (Fig. 5(c)).

Fig. 5 also underline the necessity of storing validation data from all classes when fuzzy ARTMAP is trained with $\mathbf{h}_{\mathrm{std}}$. The networks selected when using LTM are more accurate, yet only more complex on D_1, compared to networks selected without the LTM. After incremental learning of 10 blocks with $\mathbf{h}_{\mathrm{std}}$ and LTM, classification rate is 5 ± 4% higher and compression is comparable to that obtained with $\mathbf{h}_{\mathrm{std}}$ without LTM. In both cases, convergence time with $\mathbf{h}_{\mathrm{std}}$ is one. For fuzzy ARTMAP trained with $\mathbf{h}_{\mathrm{ro}}(t)$ and without LTM, Table 3 shows that since D_t^{f}, where $2 \leqslant t \leqslant 10$, is only composed of one class $(C_{k'}(t))$, optimization is performed according to that class at the expense of all others $(\{C_k \in \Omega(t)|k \neq k'\})$. While the classification rate for the class learned at a time $t,\ C_{k'}(t)$, is typically high (above 80%, except for classes C_2 and C_5), the average overall classification rate for $\{C_k \in \Omega(t)|k \neq k'\}$ degrades considerably (ends at about 54% after D_{10}). In contrast, by estimating the fitness with LTM, PSO optimization is performed according to all classes and, although classification rate for $C_{k'}(t)$ is lower than without the LTM for all learning blocks, it is always significantly higher for $\{C_k \in \Omega(t)|k \neq k'\}$.

As mentioned, when the decision boundaries between class distributions are complex, it is difficult for the ACS to maintain a high classification rate with compact fuzzy ARTMAP networks. As classes are added to the ACS using $\mathbf{h}_{\mathrm{std}}$, the recognition categories created for new classes tend to expand into the boundaries of the older class distributions. The order in which classes are learned may then prove crucial for optimal performance. With the class presentation order given in Tables 3 and 4, excepted for class C_2, which contains only 39 learning patterns, classification rates obtained for the latest classes added to the system using $\mathbf{h}_{\mathrm{ro}}(t)$ with LTM (classes $C_3,\ C_6,\ C_7$, and C_9) is significantly higher than those obtained using without LTM. The performance of these later classes are obtained at the expense of those present in previous blocks. However, no matter the choice of hyperparameters, classes C_1 and C_3 are not greatly affected by the addition of latter classes, suggesting that data distributions of classes C_1 and C_3 do not overlap those of classes $C_3,\ C_6,\ C_7$, and C_9.

5.1.2. Update scenario

In the update scenario, all classes are defined from the start in D_1 with only 10% of the available learning data. While kNN yields a classification rate of 68.1 ± 0.4 after learning D_1, the classification rate of the ACS for all system parameters settings starts below 60%. This indicates that decision boundaries are very complex, and learning D_1 with limited data from each class is a difficult task for the ACS. As it was presented in [9,18], this shows the importance of D_1 when fuzzy ARTMAP undergoes incremental learning, as it forms the basis for future updates with video data. An ACS should then be initiated with enough representative data from the environment.

At the beginning of the update process $(t \leqslant 4)$, using more validation data with the LTM results in an increase in classification rates of the ACS when $\mathbf{h}_{\mathrm{ro}}(t)$ is used during incremental learning. Moreover, the ACS with $\mathbf{h}_{\mathrm{ro}}(t)$ and LTM gives a similar classification rate as the reference systems, minus the effects of knowledge corruption. While the classification rate with $\mathbf{h}_{\mathrm{ro}}(t)$ starts at 57.4 ± 0.5% at $t = 1$ and steadily increases up to 76 ± 1% at $t = 10$, classification rate with $\mathbf{h}_{\mathrm{ro}}^{B}(t)$ starts at 55 ± 1%, reaches 60 ± 2% at $t = 4$, and increases faster than that of $\mathbf{h}_{\mathrm{ro}}(t)$ to end at 85.6 ± 0.3%. This sudden increase in performance also correspond to a decrease in compression. While compression of $\mathbf{h}_{\mathrm{ro}}(t)$ starts at 3 ± 1 at $t = 1$ and remains steady at 4 ± 1 for $2 \leqslant t \leqslant 12$, compression of $\mathbf{h}_{\mathrm{ro}}^{B}(t)$ increases from 6 ± 1 $(t = 1)$ to 8 ± 2 $(t = 4)$, and suddenly drops to 2.4 ± 0.4 $(t = 7)$ without changing significantly afterwards. Overall, $\mathbf{h}_{\mathrm{ro}}^{B}(t)$ needed about 1.7 more nodes than $\mathbf{h}_{\mathrm{ro}}(t)$ to obtained a classification rate 10% higher. It outperforms kNN classification rate, compression, and convergence time.

Meanwhile, classification rates obtained with $\mathbf{h}_{\mathrm{std}}$ and $\mathbf{h}_{\mathrm{ro}}(t)$ (without LTM) decrease considerably $t = 2$ (22 ± 2% for both cases of $\mathbf{h}_{\mathrm{std}}$ and to 39 ± 4% for $\mathbf{h}_{\mathrm{ro}}(t)$ without LTM). However, updating all classes using $\mathbf{h}_{\mathrm{std}}$ increases overall performances (classification rate increases by about 15% with LTM, 13% without LTM, and with a higher compression in both cases), while using $\mathbf{h}_{\mathrm{ro}}(t)$ without LTM only results in a gain in compression (classification rates after learning D_1 and D_{12} that are both 59 ± 2%).

Table 5
Average classification rate achieved by the ACS for the updated classes with each learning block D_t for one class presentation order during the update scenario. The classification rate of the updated class with D_t $(C_{k'}(t))$ is presented with that of the remaining classes $(\{C_k \in \Omega|k \neq k'\})$. Each cell is presented in percentage and with the 90% confidence interval.

Training strategy	D_t $C_{k'}(t)$	D_2 11	D_3 5	D_4 1	D_5 7	D_6 9	D_7 3	D_8 10	D_9 4	D_{10} 8	D_{11} 6	D_{12} 2
$\mathbf{h}_{\mathrm{ro}}(t)$ w/LTM	Class. rate for $C_{k'}(t)$	90 ± 4	77 ± 6	98 ± 2	88 ± 3	87 ± 5	98 ± 1	78 ± 7	87 ± 6	70 ± 12	92 ± 4	92 ± 4
	Class. rate for $\{C_k \in \Omega\|k \neq k'\}$	55 ± 1	62 ± 1	60 ± 1	64 ± 2	68 ± 2	65 ± 3	70 ± 2	74 ± 3	76 ± 3	74 ± 3	73 ± 3
$\mathbf{h}_{\mathrm{ro}}(t)$ w/o LTM	Class. rate for $C_{k'}(t)$	98 ± 2	86 ± 7	95 ± 3	82 ± 8	91 ± 2	99 ± 1	80 ± 7	85 ± 7	72 ± 13	90 ± 5	74 ± 13
	Class. rate for $\{C_k \in \Omega\|k \neq k'\}$	37 ± 8	32 ± 10	36 ± 6	39 ± 6	43 ± 6	46 ± 4	50 ± 5	51 ± 7	51 ± 8	59 ± 7	51 ± 10

Table 6
Average classification rate per class for one class order presentation of the update incremental learning scenario for $\mathbf{h}_{ro}(t)$ and $\mathbf{h}_{std}$, with and without the LTM. Results are obtained after update of all classes $C_k \in \Omega$. Each cell is presents the classification rate in percentage along with the 90% confidence interval.

$\mathbf{h}$	$C_k \in \Omega$											Ω
	1	2	3	4	5	6	7	8	9	10	11	
$\mathbf{h}_{ro}(t)$ w/LTM	83 ± 8	92 ± 5	92 ± 5	82 ± 8	53 ± 8	92 ± 4	82 ± 2	69 ± 12	70 ± 14	59 ± 7	60 ± 11	76 ± 1
$\mathbf{h}_{std}$ w/LTM	57 ± 28	11 ± 5	94 ± 3	38 ± 9	83 ± 5	85 ± 5	78 ± 6	34 ± 6	84 ± 2	52 ± 5	78 ± 4	70 ± 1
$\mathbf{h}_{ro}(t)$ w/o LTM	57 ± 28	74 ± 14	54 ± 27	59 ± 25	18 ± 15	56 ± 25	52 ± 20	69 ± 18	72 ± 12	41 ± 19	44 ± 25	56 ± 3
$\mathbf{h}_{std}$ w/o LTM	38 ± 8	3 ± 3	84 ± 6	38 ± 9	89 ± 8	83 ± 5	70 ± 7	44 ± 7	81 ± 3	38 ± 7	71 ± 6	66 ± 2

As with the enrollment learning scenario, Table 5 shows that the ACS without the LTM is only optimized for classes updated with each D_t. Classification rates for $C_{k'}(t)$ are either higher than or comparable to those of $\mathbf{h}_{ro}(t)$ with the LTM, while the classification rates for $\{C_k(t) \in \Omega | k \neq k'\}$ are degraded compared to those obtained when using the LTM. Coarse decision boundaries created at the beginning of the update process (D_2) are refined when new data becomes available. As the overall classification rate increases, the difference between $C_{k'}(t)$ and $\{C_k \in \Omega | k \neq k'\}$ decreases (from 45 ± 5% at $t = 2$, to 19 ± 7% at $t = 12$).

Unlike with the enrollment scenario, Tables 5 and 6 show that individual classification rates obtained after learning all data are less sensitive to class order presentation when all classes defined in D_1. Classification rates of $C_{k'}(t)$ obtained with $\mathbf{h}_{ro}(t)$ and LTM in Table 5 are no longer systematically below those of $\mathbf{h}_{ro}(t)$ without LTM. Moreover, individual classification rates from Table 6 differ less from the overall classification rate and their dispersion is lower. As an example, $\mathbf{h}_{ro}(t)$ with LTM yield a standard deviation of 16 for the individual classification rates during the enrollment scenario, and a standard deviation of 14 for the update scenario. This lead to higher global classification rates over all classes.

5.2. Experiment (B) – Impact of dynamic optimization

Figs. 7 and 10 present the average classification rate, compression, and convergence time achieved by the ACS with a LTM and with system parameters that are optimized using: dynamic optimization with DNPSO ($\mathbf{h}_{dnc}(t)$), DNPSO on only D_1 and are then fixed ($\mathbf{h}_{dnc}(1)$), static optimization with DNPSO ($\mathbf{h}_{stc}(t)$), static optimization with canonical PSO ($\mathbf{h}_{cnl}(t)$), and the

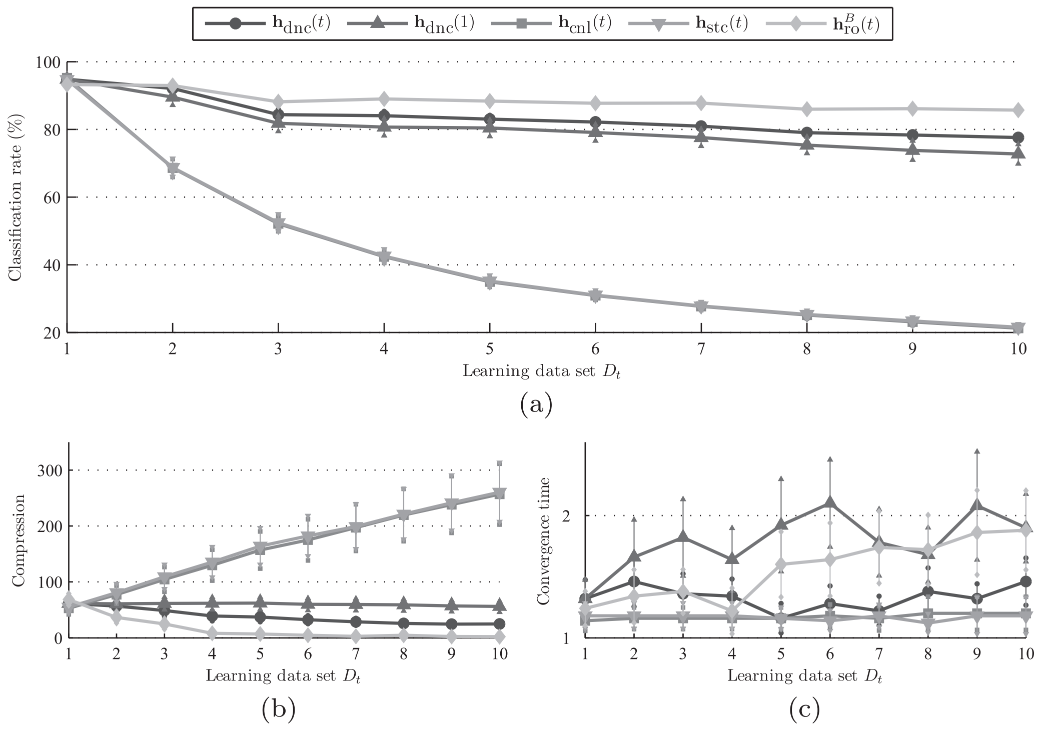

Fig. 7. Average classification rate, compression, and convergence time of the ACS versus learning block during the enrollment scenario. Performance was evaluated with the LTM for $\mathbf{h}_{dnc}(t)$, $\mathbf{h}_{dnc}(1)$, $\mathbf{h}_{stc}(t)$, and $\mathbf{h}_{cnl}(t)$. Error bars correspond to the 90% confidence interval. The performance for fuzzy ARTMAP with $\mathbf{h}^B_{ro}(t)$ during batch learning is shown for reference.

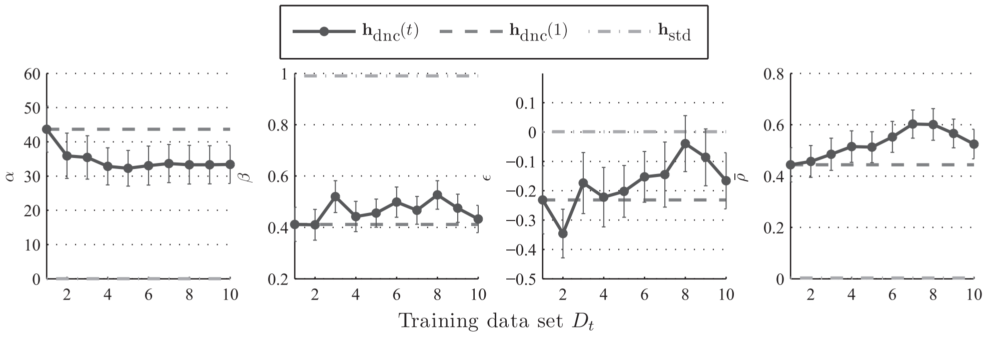

Fig. 8. Evolution of hyperparameter values obtained with the ACS using $\mathbf{h}_{\rm dnc}(t)$ compared to the ACS based on $\mathbf{h}_{\rm dnc}(t)$ and $\mathbf{h}_{\rm std}(t)$ during the enrollment scenario. The mean of each hyperparameter is shown with its 90% confidence interval.

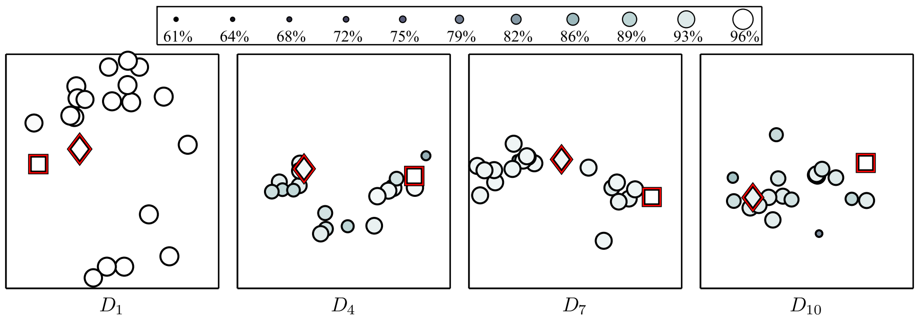

Fig. 9. A two-dimensional projection using Sammon's mapping that illustrates the evolution of each particle's personal best, and the swarm's global best positions when the proposed ACS performs incremental learning with $\mathbf{h}_{\rm dnc}(t)$ (diamond) for the enrollment scenario. The global best particle position obtained for batch learning with $\mathbf{h}^B_{\rm ro}(t)$ (square) is also shown for reference. Positions are shown along the estimation of $f(\mathbf{h}, t)$ (see legend) when the optimization stopping conditions have been reached for different points in time ($t \in \{1, 4, 7, 10\}$) during the update scenario for one replication and the same class presentation order of the previous sections.

reference batch learning method $\left(\mathbf{h}^B_{\rm ro}(t)\right)$. The evolution of hyperparameters found via $\mathbf{h}_{\rm dnc}(t)$ is shown for all class presentation order in Figs. 8 and 11 after incremental learning of different blocks for both incremental learning scenario. Figs. 9 and 12 shows the position of the swarms at different moments in time for the same class presentation order used in Tables 3–6.

5.2.1. Enrollment scenario

Fig. 7(a) illustrates that, when the proposed ACS is used with a static optimization algorithm ($\mathbf{h}_{\rm stc}(t)$ and $\mathbf{h}_{\rm cnl}(t)$), classification rate declines significantly during the enrollment learning scenario. Unlike with dynamic optimization ($\mathbf{h}_{\rm dnc}(t)$), static optimization algorithms does not automatically update the fitness corresponding to the position of each particle's personal best when a new D_t becomes available, requiring the FAM_n networks to be trained on $D^{\rm t}_t$ (Line 8). As classes are added to the ACS, decision boundaries become more complex and fitness values estimated on $D^{\rm f}_t$, initially 100% after learning D_1, decline in time. The particle's personal best positions of static PSO algorithms ($\mathbf{h}_{\rm stc}(t)$ and $\mathbf{h}_{\rm cnl}(t)$) are thus never redefined, and the FAM_n networks, which learn two classes on D_1, are never updated afterwards. The rest of the learning process is then always based on a FAM_n neural network that learned only two classes.

When using a dynamic PSO algorithm, Fig. 8 shows that $\mathbf{h}_{\rm dnc}(t)$ changes such that fuzzy ARTMAP can maintain a higher classification rate with low confidence interval. Although the confidence interval for all hyperparameters tends to be large, Fig. 8 still indicates that they vary according to some pattern no matter class presentation order. Moreover, the impact of new data on fuzzy ARTMAP hyperparameters does not appear to diminished as more classes are presented to ACS with $\mathbf{h}_{\rm dnc}(t)$.

While α changes significantly only once from 44 ± 7 at $t = 1$ to 36 ± 7 at $t = 2$, β starts at 0.40 ± 0.07 and changes significantly four times at $t \in 3, 4, 8, 9$ (to 0.52 ± 0.06, 0.44 ± 0.06, 0.53 ± 0.06, and 0.43 ± 0.06). Hyperparameter ϵ starts at −0.23 ± 0.11 and changes four times at $t \in 2, 3, 8, 10$ (to −0.35 ± 0.08, −0.17 ± 0.10, −0.04 ± 0.09, and −0.17 ± 0.09). Finally, $\bar{\rho}$ starts at 0.44 ± 0.07, increases to 0.60 ± 0.05 at $t = 7$ and decreases to 0.52 ± 0.06. Fig. 9 shows the evolution of particles in the DNPSO swarm mapped in two dimensions space using Sammon's mapping [27]. As expected, the classification rates estimated for most of the networks after D_1 are 100%. As classes are added to the system, DNPSO subswarms moves in the hyperparameters space as new peaks appear and disappear in the objective function. Even if the global best solution obtained during incremental learning is not always near the global best obtained with $\mathbf{h}^B_{\rm ro}(t)$, the latter is always found by one of

Table 7
Average classification rate (in percentage) and compression after incremental learning of all the MoBo data base for the enrollment scenario. Each cell is presented with the 90% confidence interval.

Performance indicator	$\mathbf{h}_{\text{dnc}}(t)$	$\mathbf{h}_{\text{dnc}}(1)$	$\mathbf{h}_{\text{stc}}(t)$	$\mathbf{h}_{\text{cnl}}(t)$
Classification rate (%)	79 ± 2	78 ± 4	20 ± 1	20 ± 1
Compression	45 ± 3	97 ± 80	480 ± 10	480 ± 10

the DNPSO subswarms. This indicates that the optimization space defined by fuzzy ARTMAP hyperparameters adjustment for the enrollment scenario does in fact correspond to a type III optimization environment (see Section 3).

Beside the reference $\mathbf{h}^{B}_{\text{ro}}(t)$, the ACS based on $\mathbf{h}_{\text{dnc}}(t)$ achieves the highest (78 ± 1%), followed by $\mathbf{h}_{\text{dnc}}(1)$ (72 ± 3%), $\mathbf{h}_{\text{stc}}(t)$, and $\mathbf{h}_{\text{cnl}}(t)$ (both at 21 ± 1%). Compared to $\mathbf{h}_{\text{dnc}}(1)$, classification rates with $\mathbf{h}_{\text{dnc}}(t)$ starts 94.8 ± 0.6% and become significantly different to that of $\mathbf{h}_{\text{dnc}}(1)$ as of $t = 2$. As in previous results, a lower compression (25 ± 1 at $t = 10$) is necessary to maintain higher classification rates.

Results with the MoBo data base confirms the results obtained with the IIT-NRC data base. However, since the acquisition of the MoBo data is more constrained than that of the IIT-NRC data, class distributions $p_k(\mathbf{a})$ are more compact and are less likely to vary significantly from one block to the next. As Table 7 shows, classification rates are comparable for $\mathbf{h}_{\text{dnc}}(t)$ and $\mathbf{h}_{\text{dnc}}(1)$, and compressions are twice as high as those obtained with the IIT-NRC data base.

5.2.2. Update scenario

Fig. 10 also shows that using an ACS based on static optimization algorithms ($\mathbf{h}_{\text{stc}}(t)$ and $\mathbf{h}_{\text{cnl}}(t)$) results in poor incremental learning capabilities. In some cases, both DNPSO, applied without updating the personal best when new data is available ($\mathbf{h}_{\text{stc}}(t)$), and canonical PSO ($\mathbf{h}_{\text{cnl}}(t)$) algorithms find an hyperparameter vector that remains an optimum through during the entire learning process. But in other cases, the swarm stays in a region of the optimization space where classification rate does not improve and, as with the enrollment scenario, fitness corresponding to personal best positions does not improve and the *FAM*$_n$ neural networks are never updated. Since the DNPSO algorithm, used without updating the personal best, is able to maintain diversity in the optimization space, it tends to provide a higher level of performance after learning all data but shows no significant differences with canonical PSO (68 ± 2% for $\mathbf{h}_{\text{stc}}(t)$ versus 67 ± 2% $\mathbf{h}_{\text{cnl}}(t)$). In both cases the average classification rate remains below 70%.

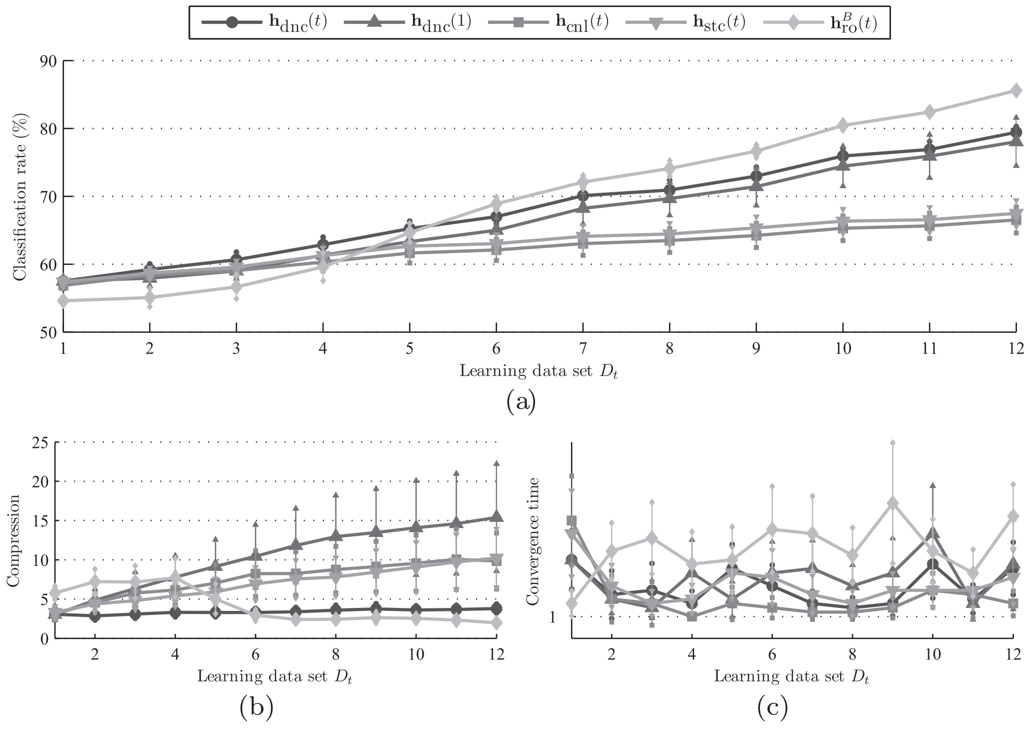

Fig. 10. Average classification rate, compression, and convergence time of the ACS versus learning block during the update scenario. Performance was evaluated with the LTM for $\mathbf{h}_{\text{dnc}}(t)$, $\mathbf{h}_{\text{dnc}}(1)$, $\mathbf{h}_{\text{stc}}(t)$, and $\mathbf{h}_{\text{cnl}}(t)$. Error bars correspond to the 90% confidence interval. The performance for fuzzy ARTMAP with $\mathbf{h}^{B}_{\text{ro}}(t)$ during batch learning is shown for reference.

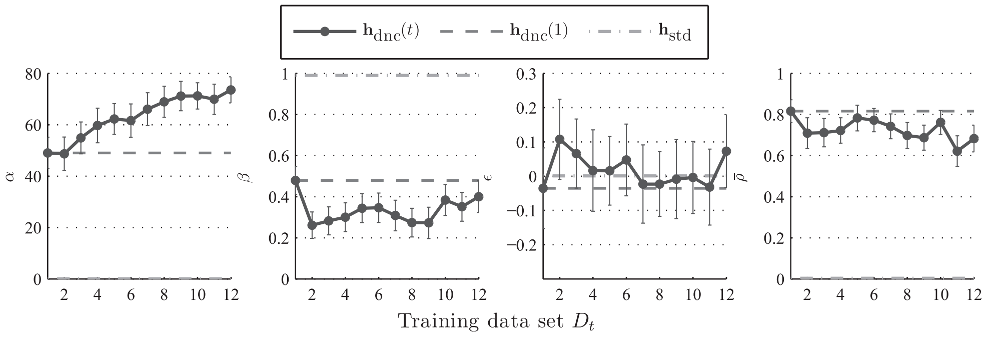

Fig. 11. Evolution of hyperparameter values obtained with the ACS using $\mathbf{h}_{\mathrm{dnc}}(t)$ compared to the ACS based on $\mathbf{h}_{\mathrm{dnc}}(t)$ and $\mathbf{h}_{\mathrm{std}}(t)$ during the update scenario. The mean of each hyperparameter is shown with its 90% confidence interval.

As blocks of data are presented to the ACS during the update scenario, Fig. 11 shows that all four hyperparameters are also adjusted during the update scenario. While, α steadily increases from 49 ± 7 to 74 ± 5, β significantly changes five times ($t = 2, 5, 9, 10$) with values ranging from 0.26 ± 0.06 to 0.48 ± 0.07, ϵ changes two times ($t = 2, 7$) with values ranging from -0.04 ± 0.12 to 0.11 ± 0.12 and very high confidence intervals, and $\bar{\rho}$ changes five times ($t = 2, 5, 9, 10, 11$) with values ranging from 0.62 ± 0.08 to 0.82 ± 0.06.

Like with the enrollment scenario, when observing the evolution of particles in the DNPSO swarm mapped in two dimensions space using Sammon's mapping, Fig. 12 indicates the presence of a type III dynamic optimization environment (Section 3). The personal best position of each particle are adjusted in response to peaks in the objective function $f(\mathbf{h}, t)$ that change position and values in time. However, since all classes are present in D_1, most of the feature space is define at the outset of the incremental learning process. Apart from the peak appearing in the middle of the optimization space at $t = 4$, most of the changes in $f(\mathbf{h}, t)$ happen in the interval $4 \leqslant t \leqslant 8$.

For the update scenario, the highest and more stable classification rate is achieved by dynamic optimization with $\mathbf{h}_{\mathrm{dnc}}(t)$ (Fig. 10(a)). Classification rates starts at 57.5 ± 0.4% and end at 79.4 ± 0.9%. It is almost always above classification rate obtained with $\mathbf{h}_{\mathrm{dnc}}(1)$ by at least 1%. As shown in Fig. 10(b), solutions obtained with D_1 are mostly heavy solutions that accommodate a complex input features space containing all classes. For $t > 1$, those solutions yield large fuzzy ARTMAP neural networks, similar to those obtain in batch learning, and provide high classification rates. Moreover, the global best positions found at $t = 1$ tend to remain in the vicinity of, at least, one subsequent local best position found during incremental learning, and of the global best positions found during batch learning (Fig. 12).

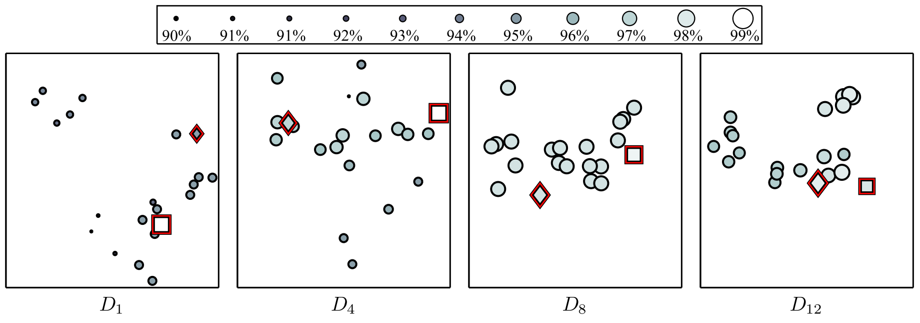

Fig. 12. A two-dimensional projection using Sammon's mapping that illustrates the evolution of each particle's personal best, and the swarm's global best positions when the proposed ACS performs incremental learning with $\mathbf{h}_{\mathrm{dnc}}(t)$ (diamond) for the update scenario. The global best particle position obtained for batch learning with $\mathbf{h}^{B}_{\mathrm{ro}}(t)$ (square) is also shown for reference. Positions are shown along the estimation of $f(\mathbf{h}, t)$ (see legend) when the optimization stopping conditions have been reached for different points in time ($t \in \{1, 4, 8, 12\}$) during the update scenario for one replication and the same class presentation order of the previous sections.

Table 8
Average classification rate (in percentage) and compression after incremental learning of all the MoBo data base for the update scenario. Each cell is presented with the 90% confidence interval.

Performance indicator	$\mathbf{h}_{\mathrm{dnc}}(t)$	$\mathbf{h}_{\mathrm{dnc}}(1)$	$\mathbf{h}_{\mathrm{stc}}(t)$	$\mathbf{h}_{\mathrm{cnl}}(t)$
Classification rate (%)	85 ± 2	88 ± 3	51 ± 1	52 ± 1
Compression	11 ± 2	20 ± 8	36 ± 9	33 ± 9

On the other hand, the ACS with $\mathbf{h}_{\text{dnc}}(1)$ also find lighter solutions that performs also well on D_1, but then gives lower classification rate than the ACS with $\mathbf{h}_{\text{dnc}}(t)$ when classes are updated. For $t \geqslant 6$, when batch learning surpass incremental learning, those solutions do not perform as well as the larger ones, and the confidence interval for the classification rate grows from 1.4 at $t = 4$ to 3.5 at $t = 12$ while the latter for compression eventually grows to 7.

Once again, results with the MoBo data base confirms the results obtained with the IIT-NRC data (Table 8). Classes in the MoBo data base are found to be more easily updated for $\mathbf{h}_{\text{dnc}}(t)$ and $\mathbf{h}_{\text{dnc}}(1)$. Both classification rates obtained with MoBo are over 5% higher than those obtain on the IIT-NRC data. Since acquisition conditions are more constrained, D_1 data structure is now more representative of the entire learning data set and solutions found with $\mathbf{h}_{\text{dnc}}(1)$ remain comparable to $\mathbf{h}_{\text{dnc}}(t)$ in terms of classification rate. The average compression is also higher for all hyperparameters settings, but as with IIT-NRC, it is higher with $\mathbf{h}_{\text{dnc}}(1)$. ACS using static optimization with MoBo also results in lighter solutions, those solutions yield lower classification rates.

6. Conclusion

In this paper, an adaptive classification system (ACS) is proposed for video-based face recognition. It combines a fuzzy ARTMAP neural network classifier, dynamic particle swarm optimization (DPSO) algorithm, and a long term memory (LTM). This ACS uses a novel DPSO-based learning strategy to cojointly optimize the classifier weights, architecture, and user-defined hyperparameters such as classification rate is maximized during incremental learning of new data. This DPSO-based learning strategy reconsiders the four properties of a classification system capable of supervised incremental learning (as defined by Polikar et al. [38]) in two ways. The 2nd property is modified to include the storage and management of previously acquired learning data for unbiased validation and fitness estimation. To avoid knowledge corruption, and thereby maintain a high level of performance, a 5th property is added to the others: a classifier must adapt its learning dynamics by adjusting its hyperparameters.

Using real-world video data bases, performance of this system is assessed in terms of classification rate and resource requirements, for different hyperparameter settings, with and without LTM. Overall results of experiments (A) and (B) demonstrate that optimizing fuzzy ARTMAP hyperparameters during incremental learning gives higher classification rates than when using standard or fixed hyperparameters ($\mathbf{h}_{\text{std}}$ and $\mathbf{h}_{\text{dnc}}(1)$). When property (5) of an incremental learning algorithm is applied, results indicate that, during incremental learning, fuzzy ARTMAP performance degrades unless some validation data are stored and updated in a LTM. Moreover, experiment (B) shows that, as more samples are learned by fuzzy ARTMAP with the LTM, peaks of the objective function (in the hyperparameters space) changes in time. Adjusting hyperparameters during incremental learning thus corresponds to a type III dynamic optimization problem and if a dynamic optimization algorithm is not employed to adjust classifier hyperparameters, then classification rate of fuzzy ARTMAP declines.

Results show that the proposed ACS requires more resources. Since the new DPSO-based learning strategy used by the ACS optimizes according to classification rate, it tends to produce fuzzy ARTMAP networks with a large number of F_2 layer nodes, and trains over longer convergence time. In order to keep the neural networks size and computational time to a minimum, future work would include designing an ACS that performs multi-objective optimization of fuzzy ARTMAP hyperparameters during supervised incremental learning. Moreover, results for both enrollment and update scenarios suggest that it may not be necessary to optimize fuzzy ARTMAP hyperparameters, weights, and architecture each time a new block of data becomes available. Since several training sequences are needed each time a fitness value is estimated, optimization is a costly process, and it would also be necessary to devise fitness-based detection measures that determines situations under which the ACS can benefit from incremental learning of blocks of data. Finally, devising a strategy to update the LTM with the most relevant data may improve performance and limit memory consumption.

Acknowledgements

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada. We also wish to thank the reviewers of this paper for their constructive comments.

Appendix A. Incremental learning as a dynamic optimization problem

Fig. 13 shows the evolution of the ACS classification rate for a type III dynamic optimization environment during a class enrollment learning scenario where only two fuzzy ARTMAP hyperparameters are adjusted, $\mathbf{h} = (\beta, \epsilon)$, while $\alpha = 0.001$ and $\bar{\rho} = 0$ (standard values). Results are shown for an algorithm similar to Algorithm 1 (Section 3.3) and the IIT-NRC data base (Section 4.1). The grid optimization method was applied with a 100×100 grid, instead of PSO, and for each point on the grid, $f(\mathbf{h}, t)$ was estimated by the average classification rate of fuzzy ARTMAP on the IIT-NRC test data when trained using 10-fold cross-validation with the learning data. Unlike the class enrollment learning scenario presented in Section 4.2, several classes are added to the system with each D_t: classes $\{C_k | k \in 1,2,3\}$ are learned with D_1, $\{C_k | k \in 4,5,6\}$ with D_2, $\{C_k | k \in 7,8,9\}$ with D_3, and $\{C_k | k \in 10,11\}$ with D_4.

The plateau on the objective function $f(\mathbf{h}, t)$ showed in Fig. 13(a) is actually a gentle slop getting higher with β. As the objective function changes during incremental learning, the global maximum moves in the hyperparameters space.

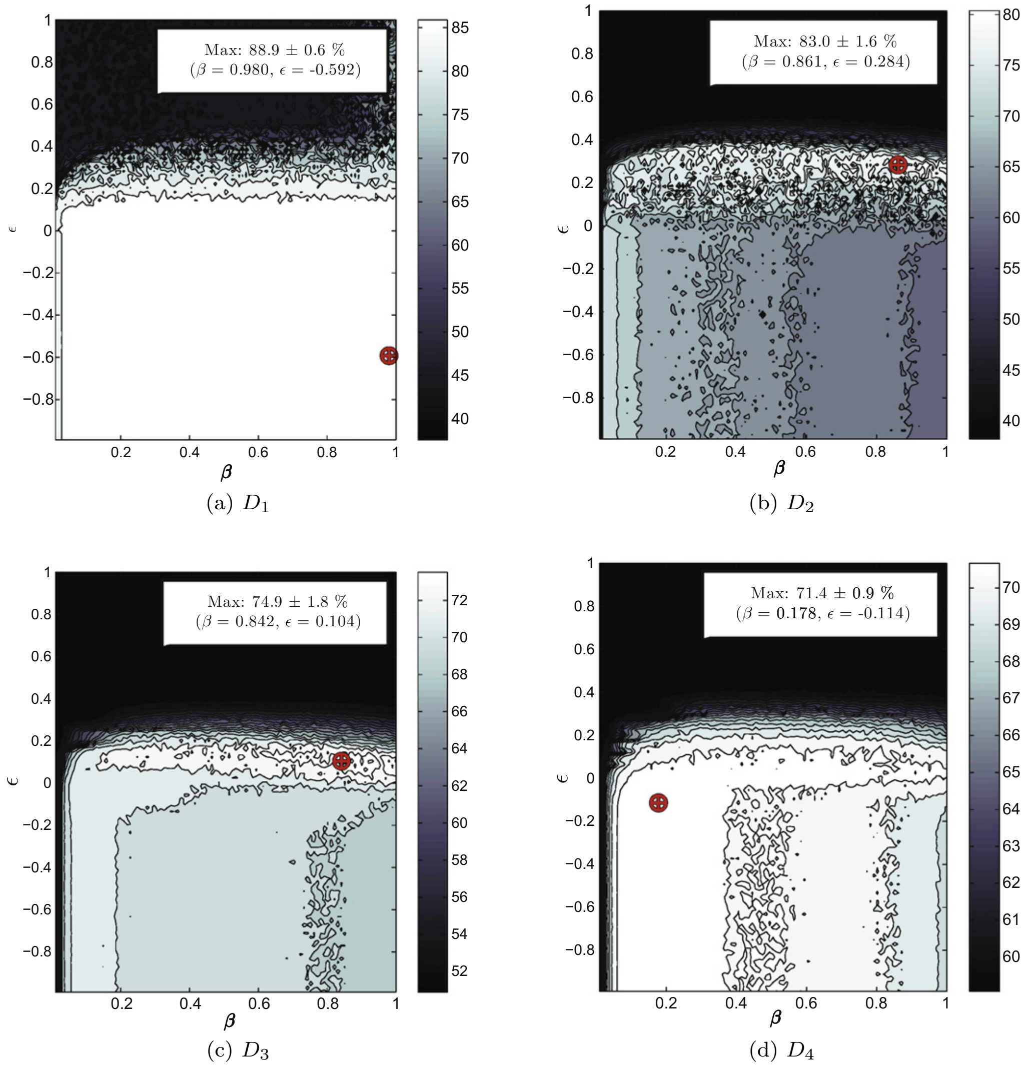

Fig. 13. Evolution of the objective function $f(\mathbf{h}, t)$, where $\mathbf{h} = (\beta, \epsilon)$, during an enrollment learning scenario of four learning data blocks D_t. The global maximum is shown along with its classification rate and its 90% confidence interval.

References

[1] M. Barry, E. Granger, Comparison of ARTMAP neural networks for classification for face recognition from video, in: IEEE Int'l Joint Conf. on Neural Networks, Orlando, USA, 2007.
[2] T. Blackwell, J. Branke, Multi-swarm optimization in dynamic environments, in: Applications of Evolutionary Computing, Coimbra, Portugal, 2004.
[3] A. Canuto, G. Howells, M. Fairhurst, An investigation of the effects of variable vigilance within the RePART neuro-fuzzy network, Journal of Intelligent and Robotic Systems: Theory and Applications 29 (4) (2000) 317–334.
[4] A. Carlisle, G. Dozier, Tracking changing extrema with adaptive particle swarm optimizer, in: World Automation Congress, Orlando, FL, USA, 2002.
[5] G.A. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Computer, Vision, Graphics and Image Processing (37) (1987) 54–115.
[6] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H.R.D.B. Rosen, Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, IEEE Transactions on Neural Networks 3 (5) (1992) 698–713.
[7] G.A. Carpenter, S. Grossberg, J.H. Reynolds, ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network, Neural Networks 4 (1991) 565–588.
[8] D. Chakraborty, N.R. Pal, A novel training scheme for MLPs to realize proper generalization and incremental learning, IEEE Transactions on Neural Networks 14 (1) (2003) 1–4.
[9] J.-F. Connolly, E. Granger, R. Sabourin, Supervised incremental learning with the fuzzy ARTMAP neural network, in: Artificial Neural Networks in Pattern Recognition, Paris, France, 2008.
[10] J.-F. Connolly, E. Granger, R. Sabourin, Incremental adaptation of fuzzy ARTMAP neural networks for video-based face classification, in: IEEE Symposium on Computational Intelligence for Security and Defence Applications (CISDA), Ottawa, Canada, 2009.
[11] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Science 178 (15) (2008) 3096–3109.
[12] A. Dubrawski, Stochastic validation for automated tuning of neural network's hyper-parameters, Robotics and Autonomous Systems (21) (1997) 83–93.

[13] A.P. Engelbrecht, Fundamental of Computational Swarm Intelligence, John Wiley & Sons, 2005.
[14] G.L. Foresti, L. Snidaro, A distributed sensor network for video surveillance of outdoor environments, in: IEEE Proc. on the Int'l Conf. on Image Processing, Rochester, USA, 2002.
[15] B. Fritzke, Growing self-organizing networks – why? in: Proc. of the European Symposium on Artificial Intelligence, Brugge, Belgium, 1996.
[16] W.-K. Fung, Y.-H. Liu, Adaptive categorization of ART networks in robot behavior learning using game-theoretic formulation, Neural Networks 16 (10) (2003) 1403–1420.
[17] D.O. Gorodnichy, Video-based framework for face recognition in video, in: Second Workshop on Face Processing in Video in Proc. on Conf. on Computer and Robot Vision, Victoria, Canada, 2005.
[18] E. Granger, J.-F. Connolly, R. Sabourin, A comparison of fuzzy ARTMAP and gaussian ARTMAP neural networks for incremental learning, in: IEEE Int'l Joint Conf. on Neural Networks, Hong Kong, China, 2008.
[19] E. Granger, P. Henniges, L.S. Oliveira, R. Sabourin, Supervised learning of fuzzy ARTMAP neural networks through particle swarm optimization, Journal of Pattern Recognition Research 2 (1) (2007) 27–60.
[20] E. Granger, M.A. Rubin, S. Grossberg, P. Lavoie, A what-and-where fusion neural network for recognition and tracking of multiple radar emitters, Neural Networks 14 (2001) 325–344.
[21] R. Gross, J. Shi, The CMU motion of body (MoBo) database, 2001, Tech. Rep. CMU-RI-TR-01-18, Carnegie Mellon University (2001).
[22] P. Henniges, E. Granger, R. Sabourin, Impact of FAM MT strategies on the recognition of handwritten digits, in: Artificial Neural Networks in Engineering, St. Louis, USA, 2006.
[23] X. Hu, R.C. Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, in: IEEE Congress on Evolutionary Computation, vol. 2, Honolulu, USA, 2002.
[24] A.K. Jain, A. Ross, S. Pankanti, Biometrics: a tool for information security, IEEE Transactions on Information Forensics and Security 1 (2) (2006) 125–143.
[25] J. Kennedy, Some issues and practices for particle swarms, in: IEEE Swarm Intelligence, Honolulu, USA, 2007.
[26] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE Int'l Joint Conf. on Neural Networks, Perth, Australia, 1995.
[27] Y.-H. Kim, K.H. Lee, Y. Yoon, Visualizing the search process of particle swarm optimization, in: Genetic and Evolutionary Computation Conference (GECCO) 2009, Montréal, Canada, 2009.
[28] B. Li, R. Chellappa, Gabor attributes tracking for face verification, in: IEEE Proc. on the Int'l Conf. on Image Processing, 2001.
[29] X. Li, J. Branke, T. Blackwell, Particle swarm with speciation and adaptation in a dynamic environment, in: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, 2006.
[30] A. Majumdar, P. Nasiopoulos, Frontal face recognition from video, in: Advances in Visual Computing, Las Vegas, USA, 2008.
[31] F. Matta, J.-L. Dugelay, Video face recognition: a physiological and behavioural multimodal approach, in: IEEE International Conference Image Processing, vol. 4, San Antonio, USA, 2007, pp. 497–500.
[32] F. Matta, J.-L. Dugelay, Person recognition using facial video information: a state of the art, Journal of Visual Language and Computing 20 (3) (2009) 180–187.
[33] A. Mian, Unsupervised learning from local features for video-based face recognition, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2008.
[34] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, DNPSO: a dynamic niching particle swarm optimizer for multi-modal optimization, in: IEEE Congress on Evolutionary Computation, Hong Kong, China, 2008.
[35] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, Evaluating the performance of DNPSO in dynamic environments, in: IEEE Int'l Conference on Systems, Man, and Cybernetics, Singapore, 2008.
[36] K. Okamoto, S. Ozawa, S. Abe, A fast incremental learning algorithm with long-term memory, in: IEEE Int'l Joint Conf. on Neural Networks, Portland, USA, 2003.
[37] E. Özcan, M. Yýlmaz, Particle swarms for multimodal optimization, in: Adaptive and Natural Computing Algorithms, Warsaw, Poland, 2007.
[38] R. Polikar, L. Udpa, S.S. Udpa, V. Honavar, Learn++ : an incremental learning algorithm for supervised neural networks, IEEE Transactions on Systems, Man, and Cybernetics 31 (4) (2001) 497–508.
[39] S. Ruping, Incremental learning with support vector machines, in: IEEE Int'l Conf. on Data Mining, San Jose, USA, 2001.
[40] A. Tsymbla, M. Pechenizkiy, P. Cunningham, S. Puuronen, Dynamic integration of classifiers for handling concept drift, Information Fusion 9 (1) (2008) 56–68.
[41] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Proc. Conf. Computer Vision and Pattern Recognition, Kauai, USA, 2001.
[42] H. Wang, D. Wang, S. Yang, Triggered memory-based swarm optimization in dynamic environments, in: Applications of Evolutionary Computing, Valencia, Spain, 2007.
[43] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101.
[44] X. Zhang, Y. Gaoa, Face recognition across pose: a review, Pattern Recognition 42 (11) (2009) 2876–2896.
[45] W. Zhao, R. Chellappa, P. Phillips, A. Rosenfeld, Face recognition: a literature survey, ACM Computing Surveys 35 (2003) 399–458.
[46] S. Zhou, V. Krueger, R. Chellappa, Probabilistic recognition of human faces from video, Computer Vision and Image Understanding 91 (2003) 214–245.