An Adaptive Ensemble of Fuzzy ARTMAP Neural Networks for Video-Based Face Classification

Jean-François Connolly, Eric Granger and Robert Sabourin

***Abstract*—A key feature in population based optimization algorithms is the ability to explore a search space and make a decision based on multiple solutions. In this paper, an incremental learning strategy based on a dynamic particle swarm optimization (DPSO) algorithm allows to produce heterogeneous ensembles of classifiers for video-based face recognition. This strategy is applied to an adaptive classification system (ACS) comprised of a swarm of fuzzy ARTMAP (FAM) neural network classifiers, a DPSO algorithm, and a long term memory (LTM). The performance of this ACS with an ensemble of FAM networks selected among local bests of the swarm, is compared to that of the ACS with the global best network under different incremental learning scenarios. Performance is assessed in terms of classification rate and resource requirements for incremental learning of new data blocks extracted from real-world video streams, and are given along with reference kNN and FAM classifier optimized for batch learning. Simulation results indicate that the learning strategy maintains diversity within the ensemble classifiers, providing a significantly higher classification rate than that of the best FAM network alone. However, classification with an ensemble requires more resources.**

I. Introduction

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 technology. Given that face images may be sampled discreetly, without cooperation, face recognition has been employed in video surveillance to detect the presence of individuals belonging to a watch list, e.g., screening for criminals or terrorists in an airport setting. The rapid and covert identification of individuals of interest from surveillance cameras remains a very challenging problem.

A critical function in these applications is the classification of face regions captured from video streams. Typically, face recognition systems employ statistical or neural pattern classifiers to map an $\mathbb{R}^I$ input features space to a set of K predefined class labels $\Omega = \{C_1, C_2, ..., C_K\}$, where each class k ($k = 1, ..., K$) 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 data.

Since the collection and analysis of such data is often expensive and time consuming, it may 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 samples, missing components of the input observations, missing class labels during learning, and unfamiliar classes (not present in the learning data set). Moreover, in video surveillance applications, samples acquired from video streams in *unconstrained* scenes are generally of poor quality with low resolution. They are subject to considerable variations due to limited control over operational conditions (e.g., illumination, pose, facial expression, orientation and occlusion). Inter- and intra-class variability thus make for very complex class distributions $p_k(\mathbf{a})$.

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. For instance, 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). In addition to previously mentioned challenges, an individual's physiology may change over time, either temporary (e.g., haircut, glasses, etc.) or permanently (e.g., aging). In the $\mathbb{R}^I$ features space, new information, such as input features and output classes, may suddenly emerge. Previously acquired data may eventually become obsolete in dynamically changing classification environments, where $p_k(\mathbf{a}, t)$ vary or drift in time [1], [2].

To avoid a growing divergence between the model of each individual and their underlying class distributions $p_k(\mathbf{a}, t)$ when new data and knowledge becomes available, a classifier should be able to efficiently adapt its face models by adjusting its parameters (e.g., synaptic weights for a neural network) and architecture through *supervised incremental learning*. In previous work, the authors have proposed an adaptive classification system (ACS) that features a neural network suitable for incremental learning, a dynamic particle swarm optimization (DPSO) module, and a long term memory (LTM) to store validation data [3]. They have shown that a classifier that allows for *supervised incremental learning* should

1) learn additional information from new data,
2) require access to some previous learning data for validation processes,
3) preserve previously acquired knowledge,

The authors are with the Laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA), École de technologie supérieure, Université du Québec, Montréal, Canada. Email addresses: jfconnolly@livia.etsmtl.ca, Eric.Granger@etsmtl.ca, and Robert.Sabourin@etsmtl.ca.

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada.

4) accommodate new classes that may be introduced with the new data, and
5) adapt its learning dynamics by adjusting its hyperparameters for accurate and timely recognition.

The authors have also shown that a FAM classifier trained through batch learning of all cumulative data generally outperforms the ones trained with incremental learning [4].

To improve performance, it is possible to consider several viewpoints of the same problem by using a ensemble of classifiers. It is well known that ensemble averaging is effective in reducing the variance of errors by the networks [5]. However, when the classifiers cooperate and exchange information, their design and training are *interdependent*, leading to better ensembles [6], [7]. For instance, if all classifiers are trained on the same data, but with different set of hyperparameters, and the information regarding their performance is used to maximize their accuracy and diversity, an *heterogeneous* ensemble will be created and the generalization capabilities of the system may improve [8], [9].

In this paper, a specialized DPSO-based strategy is proposed for supervised incremental learning of new data in video-based face recognition. It uses the adaptive classification system (ACS) developed in [3], but with an heterogeneous ensemble of fuzzy ARTMAP (FAM) neural networks [10] (instead of only one) that are guided by a dynamic particle swarm optimization (DPSO) algorithm capable of finding and tracking several changing local optima [11]. As each particle corresponds to a FAM network, the ACS cojointly optimizes all *system parameters* – user-defined hyperparameters, weights, and architecture – of a swarm of classifiers such as classification rate is maximized. It then selects the local best DPSO solutions to form an heterogeneous ensemble of classifiers as these solutions were naturally evolve to be diverse.

This study focuses on video-based face recognition applications in which two incremental learning scenarios may occur – enrollment and update of face models. 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 – Information Technology of the Canadian National Research Council (IIT-NRC) [12] and Motion of Body (MoBo) [13]. The performance of the ACS using an ensemble of local best networks is compared to that of the ACS using the global best network. For reference, the performance of FAM, trained with the PSO learning strategy, and kNN are given for batch learning.

In Section II, the ACS is described, along with the LTM used to store and manage validation data, the fuzzy ARTMAP neural network used for classification, the DPSO algorithm used to optimize its hyperparameters, and the implementation of the ensemble. Then, the data bases, incremental learning scenarios, performance measures and protocol used for proof-of-concept simulations are described in Section III. Finally, experimental results are presented and discussed in Section IV.

II. Adaptive Classification System

Figure 1 depicts the adaptive classification system (ACS) proposed in this paper for incremental learning of new data. This system is composed of a swarm of pattern classifiers suitable for supervised incremental learning, a dynamic optimization module that tunes the user-defined hyperparameters of each classifier, and a long term memory (LTM) that stores and manages incoming data.

When a new block of learning data D_t becomes available to the system at a discrete time t, it is employed to train the swarm of incremental classifiers and update the LTM. The classifiers then interacts with the dynamic optimization module using an original DPSO-based learning algorithm that cojointly optimizes the *system parameters* – the vector of user-defined hyperparameters,[1] synaptic weights, and architecture – such that classification rate is maximized. Once the optimization process is over, the fusion module selects and combines the classifiers that converged on the different local maxima in the optimization space, thus creating an heterogeneous ensemble. In this paper, the fuzzy ARTMAP neural network (FAM) is employed as an incremental learning classifier and a dynamic particle swarm optimization (DPSO) algorithm is used for optimization.

The rest of this section provides additional details on each part of the adaptive classification system: the LTM, the FAM neural network, and the DPSO-based learning strategy.

A. Long Term Memory

The objective of the LTM is to ensure that a representative set of all class distributions is present during the validation process, when training FAM over several training epochs[2] and fitness estimations of the objective function. The LTM functions according to two parameters: (1) the proportion of D_t used to fill and update the external data base, p_D, and (2) the maximal number of patterns per class k in the external data base $|\text{LTM}|_k$. Each time a new D_t is presented to the network, a proportion p_D of D_t is transferred to the LTM for either addition or update. When the LTM is full, it is managed as a FIFO (first in, first out) data structure, and the outdated data is discarded. For each class, if the number of patterns transferred exceeds $|\text{LTM}|_k$, the excess samples are randomly selected and assigned to training.

The rest of the data is used for training, validation, and fitness estimation. Training data from D_t is combined with the excess data not used to fill and/or update the LTM (due to size limitations) to create the training data set D_t^{t}. Data from D_t dedicated to fitness and validation is then combined to the data contained in the LTM. This latter combination is class-wise and is divided in two, to create the validation data set, D_t^{v}, and the fitness estimation data set, D_t^{f}.

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

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

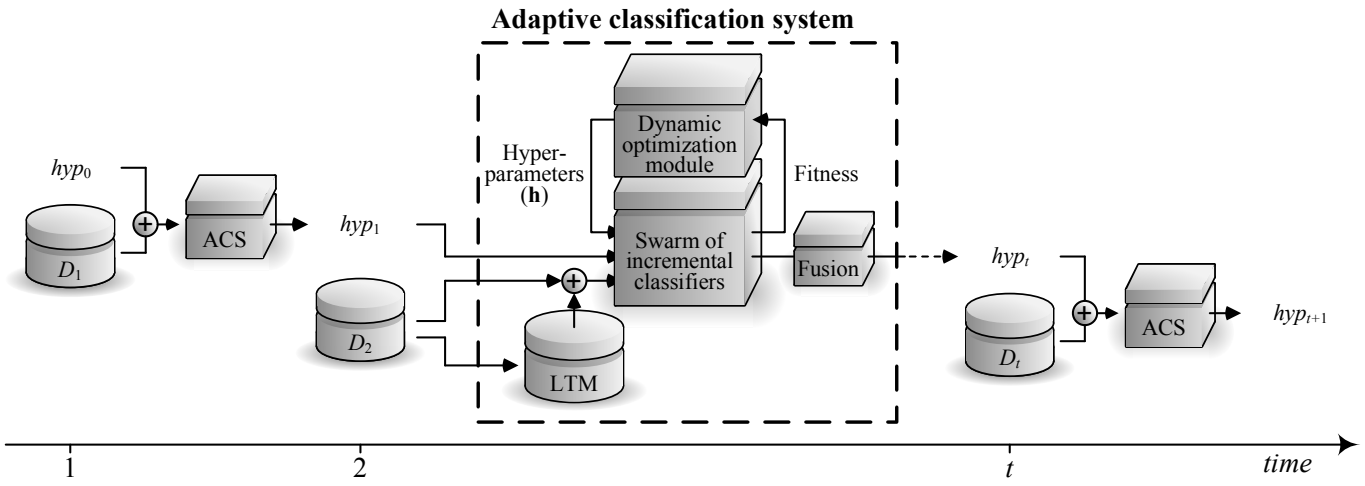


Fig. 1. The adaptive classification system (ACS) during a generic incremental learning scenario where new blocks of data are used to update a swarm of classifiers. Let D_1, D_2, ... be blocks of training data available at different instants in time. The ACS starts with an initial hypothesis hyp_0 which 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 blocks.

B. 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. The fuzzy ARTMAP (FAM) consists of three layers: (1) an input layer F_1 of $2I$ neurons (for a $\mathbb{R}^I$ input features space), (2) a competitive layer F_2, where each node is associated to a recognition category in the feature space, and (3) a map field F^{ab} of K neurons (the number of classes) [10].

In supervised training mode, ARTMAP classifiers learn an arbitrary mapping between training set patterns $\mathbf{a} = (a_1, a_2, ..., a_I)$ and their corresponding binary supervision patterns $\mathbf{c} = (c_1, c_2, ..., c_K)$. These patterns are coded to have the value $c_k = 1$ if k is the target class label for $\mathbf{a}$, and zero elsewhere. Components of the vector $\mathbf{a}$ are scaled so that each $a_i \in [0, 1]$, $(i = 1 \ldots I)$. Complement coding doubles the number of components in the input vector, which becomes $\mathbf{A} \equiv (a_1, a_2, ..., a_I, 1-a_1, 1-a_2, ..., 1-a_I)$. The $2I$-dimensional vector $\mathbf{w}_j$, linking F_1 to F_2, may be visualized as the hyperbox that just encloses all the vectors $\mathbf{a}$ that selected node j during training.

Activation of the coding field F_2 is determined by the Weber law choice function $T_j(\mathbf{A}) = |\mathbf{A} \wedge \mathbf{w}_j|/(\alpha + |\mathbf{w}_j|)$, where $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$ and $|\mathbf{p}| \equiv \sum_{i=1}^{2I} |p_i|$. With winner-take-all coding, the F_2 node j^* that receives the largest $F_1 \rightarrow F_2$ input $T_{j^*}(\mathbf{A})$ becomes active. Node j^* remains active if it satisfies the matching criterion: $|\mathbf{A} \wedge \mathbf{w}_{j^*}|/|\mathbf{A}| = |\mathbf{A} \wedge \mathbf{w}_{j^*}|/I > \rho$, where $\rho \in [0, 1]$ is the dimensionless *vigilance parameter.* Otherwise, the network resets the active F_2 node and searches until j^* satisfies the matching criterion. If node j^* then makes an *incorrect* class prediction, it is deactivated until another pattern is presented to the network. A *match tracking* signal then adjusts vigilance and induce a new search, which continues until either some F_2 node becomes active for the first time, in which case j^* learns the correct output class label $k^* = k(j^*)$; or a node j^* that has previously learned to predict k^* becomes active. During testing, a pattern $\mathbf{a}$ that activates node j^* is predicted to belong to the class $k^* = k(j^*)$.

During training and testing, FAM internal dynamics are governed by the vector of four user-defined hyperparameters $\mathbf{h} = (\alpha, \beta, \epsilon, \bar{\rho})$: the choice parameter α, the learning parameter β, the match tracking parameter ϵ, and the baseline vigilance parameter $\bar{\rho}$. These hyperparameters are inter-related and each has a distinct impact on network dynamics. While α and ϵ determine the depth of search attained before an uncommitted node is selected, $\bar{\rho}$ limits the region around the training sample in which the tested F_2 nodes pass the vigilance test and sets the maximal size of the recognition categories in the $\mathbb{R}^I$ features space. Low vigilance allows large hyperboxes and leads to broad generalization and abstract memories, while high vigilance yields small hyperboxes, leading to narrow generalization and detailed memories. During learning, β determines how fast 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 hyperbox is just large enough to enclose the cluster of training set patterns $\mathbf{a}$ to which it has been assigned. That is, an $2I$-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 $\mathbf{h}_{\text{std}} = (\alpha = 0.001, \beta = 1, \epsilon = 0.001, \bar{\rho} = 0)$ is commonly used to minimize network complexity [10].

C. 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. 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). Unlike evolutionary algorithms (such as genetic algorithms), each particle always keep in memory its best position and the best position of its surrounding.

During incremental learning of new data blocks D_t, the FAM hyperparameters must be adjusted to maximize the classification rate. It has been shown that this adaptation corresponds to a dynamic optimization problem such as

$$\text{maximize} \left\{ f(\mathbf{h}, t) \mid \mathbf{h} \in \mathbb{R}^4, t \in \mathbb{N}_1 \right\}, \tag{1}$$

where $f(\mathbf{h}, t)$ is the classification rate of FAM for a given vector of hyperparameters $\mathbf{h} = (\alpha, \beta, \epsilon, \bar{\rho})$, after learning data set D_t and at a discreet time t [3]. Hence, a dynamic version of the original PSO algorithm is needed to properly track optimal system parameters through time.

Originally developed for static optimization problems, the PSO algorithm has been adapted for dynamic optimization problems 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 particle swarm optimization algorithms developed to insure diversity in the swarm are presented in [11], [14], [15], [16]. Change detection and memory adjustment mechanisms for DPSO are presented in [17], [18], [19], [20].

In this paper, the adaptive classification system (ACS) uses a Dynamical Niching PSO (DNPSO) [11] to maximize FAM classification rate as a function of its hyperparameters vector $\mathbf{h} = (\alpha, \beta, \epsilon, \bar{\rho})$ when learning new data blocks D_t (Figure 1). The optimization space is defined by the four FAM hyperparameters ($\alpha \in [0.001, 100]$, $\beta \in [0, 1]$, $\epsilon \in [-1, 1]$, and $\bar{\rho} \in [0, 1]$. At each DNPSO iteration τ, the objective function of particle n, $f(\mathbf{h}_n(\tau), t)$, is defined by the classification rate of the FAM network corresponding to particle n, FAM_n, evaluated at that particle's position, $\mathbf{h}_n(\tau)$.

The DNPSO algorithm adapted to dynamic optimization has been shown to converge toward global maximum in a multimodal type III optimization environment, where both location and value of optima points change, with the moving peaks benchmark [11]. It maintains diversity (1) by using a local neighborhood topology, in which subswarms are dynamically created around master particles, that are their own best position amongst their neighborhood, and (2) by allowing free particles that do not belong to any any subswarms, to move independently. DNPSO was adapted for dynamic optimization problems by simply updating the fitness of their best position $f(\mathbf{h}_n^*, t)$ at each iteration. Normally, this would double the number objective function evaluations, leading to a very costly process. However, for an ACS, changes in the objective function only occur when a new data block D_t becomes available. Thus, the fitness of the best particle positions is only updated when a new D_t is presented to the system, *before* the iterative DNPSO process.

Algorithm 1 DPSO-based learning strategy for the ACS using a swarm of FAM networks.

Inputs: New data sets D_t for learning.

Outputs: The N best vectors $\mathbf{h}$ and networks FAM_n.

1: Initialization:
- initialize all N networks FAM_n and FAM_n^{start},
- set the swarm parameters,
- set PSO iteration counter at $\tau = 0$, and
- randomly initialize particles positions and velocities.

Upon reception of a new data block D_t, the following-incremental process is initiated:

2: **for** each particle n, where $1 \leq n \leq N$ **do**
3: Training of FAM_n with validation using D_t^{t} and D_t^{v}, and $f(\mathbf{h}_n^*, t)$ estimation using D_t^{f}.
4: **end for**
5: **while** PSO did not reach stopping condition **do**
6: Update particle positions according to the DNPSO algorithm.
7: **for** each particle n, where $1 \leq n \leq N$ **do**
8: $FAM_{\text{est}} \leftarrow FAM_n^{\text{start}}$
9: Training of FAM_{est} with validation using D_t^{t} and D_t^{v}, and $f(\mathbf{h}_n(\tau), t)$ estimation using D_t^{f}.
10: **if** $f(\mathbf{h}_n(\tau), t) > f(\mathbf{h}_n^*, t)$ **then**
11: $\{\mathbf{h}_n^*, FAM_n, f(\mathbf{h}_n^*, t)\} \leftarrow \{\mathbf{h}_n(\tau), FAM_{\text{est}}, f(\mathbf{h}_n(\tau), t)\}$
12: **end if**
13: **end for**
14: $\tau = \tau + 1$
15: **end while**
16: **for** each particle n, where $1 \leq n \leq N$ **do**
17: $FAM_n^{\text{start}} \leftarrow FAM_n$
18: **end for**

D. Heterogeneous Ensemble of FAM Networks

Algorithm 1 describes the DPSO learning strategy for ensembles of FAM networks in dynamic classification environments. This algorithm supposes that the incremental process starts without any a priori knowledge of the class distributions $p_k(\mathbf{a})$. Given a new learning data block D_t, it cojointly optimizes the system parameters using a swarm with N particles, and stores $2N + 1$ FAM networks. Indeed, for each particle, the system stores $n = 1...N$ networks FAM_n^{start} in a short term memory to preserve the model associated with the best position of each particle ($\mathbf{h}_n^*$) at time $t-1$, and FAM_n, the model associated with $\mathbf{h}_n^*$ during the optimization process at time t. It also stores FAM_{est}, a network employed for fitness estimation by the algorithm.

First, the initialization process takes place at line 1. All the neural networks are initialized, and the swarm's parameters are set. Each particle position is then randomly initialized within their allowed range.

When a new D_t becomes available, the optimization process begins. Fitness associated to each particle best position, $f(\mathbf{h}_n^*, t)$, is updated according to the new data along with each network *FAM*$_n$ (lines 2–4). Then, unless one of the stopping criteria is reached, the optimization process continues were it was previously halted (lines 5–15). The DNPSO algorithm defines the subswarms and free particles, and iteratively update each particle's new position along with their fitness (lines 6–13). If new personal best positions are found, the position, fitness, and network associated to the personal best (*FAM*$_n$) are updated (lines 10–12). In the cases of equality between $f(\mathbf{h}_n(\tau), t)$ and $f(\mathbf{h}_n^*, t)$, simpler models are preferred. The position with the lighter network (the one with the less F_2 nodes) then becomes $\mathbf{h}_n^*$. Finally, the iteration counter is incremented (line 14).

Once the DNPSO algorithm converges, the neural networks associated to each personal best (*FAM*$_n$) are stored as *FAM*$_n^{\text{start}}$ (lines 16–18). Those networks will serve as a short term memory of the swarm's state time t, and for learning data block D_{t+1}. Each time a particle's fitness (i.e., classification rate) is estimated on D_{t+1}, the best network previously obtained at time t, saved in *FAM*$_n^{\text{start}}$, serves as a starting state and is copied to *FAM*$_{\text{est}}$ prior training.[3] Moreover, to minimize the impact of pattern presentation order on FAM, overall fitness is defined using the average classification rate of FAM trained on D_t^{t} over five different random pattern presentation orders, and *FAM*$_{\text{est}}$ is the network that yielded the best classification rate.

This way, each particle is allowed to evolve according to its own knowledge of previous learned training data, and offers a different perspective of the problem. Thus, diversity is not only maintain for the particle positions, with the use of the DNPSO mechanism, but also for the models associated with each particle. By selecting a subset of networks amongst the swarm, the creation of an heterogeneous ensemble of classifiers [9] is possible. When used for class prediction, an ensemble of networks is directly selected from the swarm: one for each local best (including the global best), and completed with networks associated to the best free particles. A majority vote then decides the prediction. In the case of a tie, simpler models are again preferred, and the class predicted by the smallest networks (the ones that yielded the fewer F_2 nodes) is declared winner of the vote.

III. Experimental Methodology

A. 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. In both cases, preliminary face detection is performed using the Viola-Jones algorithm included in the OpenCV C/C++ computer vision library. It produced regions of interest (ROIs) between 29×18 and 132×119 pixels for each face detection in the video sequences. 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 $\mathbf{a} = \{a_1, a_2, ..., a_{576}\}$, where each feature $a_i \in [0, 1]$ represents a normalized grayscale value.

The first data base was collected by the Institute for Information Technology of the Canadian National Research Council (IIT-NRC) [12]. It is composed of 22 video sequences captured from eleven individuals positioned in front of a computer. For each individual, two color video sequences of about fifteen seconds are captured at a rate of 20 frames per seconds 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, almost the same background, and 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. The number of ROIs detected per class for the IIT-NRC data base is displayed in Table I.

The second video data base is called Motion of Body (MoBo), and was collected at Carnegie Mellon University under the HumanID project [13]. Each video 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. Only the video sequences with visible faces were kept: full frontal view and both sides with an angle of about $45°$ with the full frontal view. Moreover, this data base was reduced in order to be roughly the same size of the IIT-NRC 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 (the first 24 ROIs detected 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.

B. Incremental Learning Scenarios

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

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

[3]For the first learning block, the networks *FAM*$_n^{\text{start}}$ are in an initial state.

TABLE I
NUMBER OF LEARNING AND TEST PATTERNS PER INDIVIDUAL FOR THE IIT-NRC DATA BASE.

Type of data	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

ACS performance for K classes, the first learning block D_1 is composed of two classes, and each successive block D_t, where $2 \leq t \leq K-1$, 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.

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 of 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 updated 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 $2 \leq t \leq K+1$, 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.

C. Experimental Protocol

Learning is performed over ten trials using ten folds cross-validation. Within each replication, there are five different trials using different class presentation order, for a total of fifty replications. Out of the ten folds, eight are dedicated to training (D_t^{t}), one fold is combined with half of LTM to validate and determine the number of FAM 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 DPSO algorithm (D_t^{f}). In this experiment, initialization and update of the LTM is performed with $p_D = 1/6$ and $|LTM|_k = 20$.

Performance of supervised incremental learning with the ACS is assessed with system parameters that are adjusted on each learning block D_t using:

1) *EoFAM*$_t \leftarrow$ an ensemble of FAM networks, and
2) *FAM*$_{n^*,t} \leftarrow$ the FAM network corresponding to the global best solution (i.e., $\mathbf{h}_{n^*}^*$).

For reference, the performance of FAM trained with the canonical PSO learning strategy (*FAM*$^B_{n^*,t}$), and kNN^B are given for batch learning. At a given time t, the batch PSO learning strategy 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 $B_t = D_1 \cup ... \cup D_t$) [21]. During batch learning, data is also separated in folds for ten fold cross-validation particles fitness estimation system parameters are adjusted with canonical PSO.

TABLE II
PARAMETERS FOR THE DNPSO

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	0.2
Minimal velocities of free particles	0.0001

The DNPSO parameters for this experiment (Table II) yield a maximum of four subswarms. The size of the ensemble is then fixed at five to include all possible local best solutions (including the global best) completed with the network(s) amongst the swarm with the best accuracy. Since the distances between particles are measured during the DNPSO algorithm, the swarm evolves in a *normalized* $\mathbb{R}^4$ space to avoid any bias due to the domain of each hyperparameter. Each position is then denormalized to fit the hyperparameters domain before being applied to FAM. For each D_t, the DPSO optimization process is set to either stop after 10 iterations without improvement of the best FAM network (*FAM*$_{n^*,t}$) classification rate, or after 100 iterations (for the current D_t).

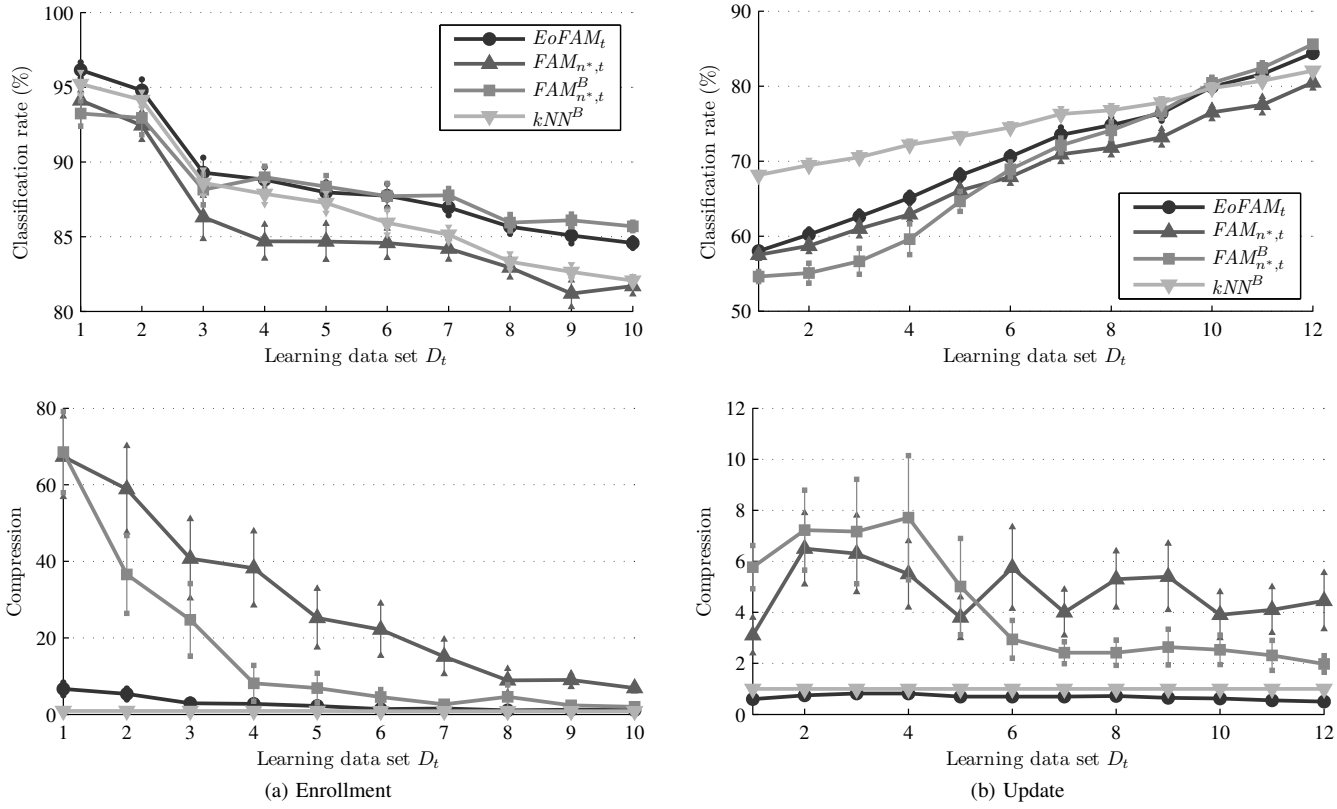
The average performance of FAM is assessed in terms of classification rate and resources requirements, measured by compression. *Classification rate* is estimated as the ratio of correctly classified test subset patterns over all test set patterns and *compression* refers to the average number of training patterns per category created in the F_2 layer.

IV. RESULTS AND DISCUSSION

Figures 2 presents the average classification rate and compression achieved by the ACS during supervised incremental learning of the IIT-NRC data base with an ensemble (*EoFAM*$_t$), and with the global best network alone (*FAM*$_{n^*,t}$). Performance is also shown during batch learning with the global best network alone (*FAM*$^B_{n^*,t}$) and kNN^B. For all cases, the classification rate and compression achieved after learning all data from the MoBo data base are also shown during both enrollment and update learning scenarios in Table III.

A. Enrollment Scenario

As depicted in Figure 2a, the best classification rate obtained during incremental learning is achieved by the ACS with *EoFAM*$_t$, followed by the global best *FAM*$_{n^*,t}$. As the amount of training data and complexity of the decision boundaries increase, the classification rate is greater for *FAM*$_{n^*,t}$ declines more severely. The difference between the two prediction



(a) Enrollment (b) Update

Fig. 2. Average classification rate and compression of the ACS versus learning block during both learning scenarios. Performance was evaluated during incremental learning with the ensemble (*EoFAM*$_t$) and the global best network alone (*FAM*$_{n^*,t}$). The performance of kNN^B and the global best network optimized during batch learning (*FAM*$^B_{n^*,t}$) are shown for reference. Error bars correspond to the 90% confidence interval.

methods start at 2% ($96 \pm 1\%$ for *EoFAM*$_t$, versus $94 \pm 1\%$ for *FAM*$_{n^*,t}$), grows to 4% at $t = 4$, and ends at 3% ($85 \pm 1\%$ and $82 \pm 1\%$).

These last results are similar to those obtained with batch learning. While classification rate obtained with the ACS during batch learning (*FAM*$^B_{n^*,t}$) starts lower than *EoFAM*$_t$ ($93 \pm 1\%$), the two are comparable for $3 \leq t \leq 6$, and becomes significantly higher by no more than 1% for $t = \{7, 9, 10\}$. Meanwhile, classification rate with kNN^B is comparable to that obtained with *EoFAM*$_t$ up to $t = 5$, and then decreases to 82 ± 1, a classification rate that is similar to that of the ACS with *FAM*$_{n^*,t}$.

In addition to performing the best with simple boundaries, the ACS with ensemble is defined according to the inherent diversity of the local best FAM nets (i.e., particles). When data is added to the ACS, the ensemble is able to compensate for not training with all the data at once by having multiple viewpoints of the problem. It can thus maintain a high level of performance with little difference to that obtained with *FAM*$^B_{n^*,t}$.

The improved classification rate of *EoFAM*$_t$ comes at a cost. As new data is added, the classification environment becomes more complex and compression decreases for all cases of the ACS, while always remaining at one for kNN^B. Since five neural networks are selected for class prediction, it is expected that compression of *EoFAM*$_t$ would be amongst the lowest (starting at 7 ± 2, and ending at 1.1 ± 0.1). Still, once all data has been presented to the ACS, the average network compression of the ensemble is below that of the global best network alone (7 ± 1). This suggests that criterion other than classification rate and diversity should be favored to minimize resources.

Results with the MoBo data base confirm those obtained with the IIT-NRC data. 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. Classification rates follow similar trends excepted that they are generally higher, and that batch learning methods performs better than that of incremental learning (see Table III). Moreover, compressions are much higher than with the IIT-NRC data base.

B. Update Scenario

As Figure 2b depicts, the same tendencies as with the enrollment scenario can be observe when all classes are defined in D_1. During incremental learning, *EoFAM*$_t$ performs better than the global best network alone with a classification rate starting at $58.0 \pm 0.4\%$ and ending at $84.4 \pm 0.5\%$ (compared to $57.5 \pm 5\%$ and $80.5 \pm 0.8\%$ with *FAM*$_{n^*,t}$). The classification rate obtained with the ACS using batch learning (*FAM*$^B_{n^*,t}$) still starts lower, with $54.6 \pm 0.7\%$, increases faster

TABLE III
AVERAGE CLASSIFICATION RATE AND COMPRESSION AFTER LEARNING THE MOBO DATA BASE DURING BOTH SCENARIOS.

Enrollment

Performance	$EoFAM_t$	$FAM_{n^*,t}$	$FAM^B_{n^*,t}$	kNN^B
Class. rate (%)	92 ± 1	79 ± 2	98 ± 1	96 ± 1
Compression	2.4 ± 0.4	46 ± 6	7 ± 3	1 ± 0

Update

Performance	$EoFAM_t$	$FAM_{n^*,t}$	$FAM^B_{n^*,t}$	kNN^B
Class. rate (%)	95 ± 1	93 ± 1	97 ± 1	96 ± 1
Compression	0.9 ± 0.2	8 ± 2	8 ± 3	1 ± 0

than with both cases of incremental learning, and ends the highest with 85.6 ± 0.3. As all classes are defined from the start, compressions are consistently lower than those obtained during the enrollment scenario.

However, when the batch learning methods are utilized, two differences are observed. Unlike with FAM, kNN^B computes L^2 distances (instead of L^1), relies on validation data only to fix the value of k, and does not use categories, whose creation and growth are sensitive to pattern and class order presentation. As such, it performs much better if only few samples are available to define all classes. When using the ACS with batch learning, the LTM is unnecessary, and all data are directly assigned to the training and validation data sets (D_t^{t}, D_t^{v}, and D_t^{f}). Therefore, fewer data are used for validation during network training (to find the number of training epochs) and fitness estimation on the objective function, leading to a lower classification rate than those obtained with LTM.

Results with the IIT-NRC data are once again confirmed those obtained with the MoBo data (Table III). As with the enrollment scenario, class distributions are more compact and both classification rate and compression are higher. However, updating classes through incremental learning yields a higher classification rate at the expense of lower compression.

V. CONCLUSION

In this paper, an incremental learning strategy, based on dynamic particle swarm optimization, that allows to produce heterogeneous ensembles of classifiers is considered for video-based face recognition. This strategy is applied to an adaptive classification system (ACS) architecture where a swarm of fuzzy ARTMAP (FAM) neural network classifiers are guided by a dynamic particle swarm optimization algorithm to co-jointly optimize all classifiers system parameters such as classification rate is maximized. It then selects an heterogeneous ensemble amongst the swarm based on accuracy and diversity.

Overall results indicate that when the ACS uses the DPSO learning strategy, it allows to effectively select an ensembles of heterogeneous classifiers. These ensembles provide a significantly higher classification rate than when the ACS uses only the global best FAM. However, classification with an ensemble requires more resources as predictions are provided by a set of networks instead of just one. Future work should then focus on limiting computing and memory resources by using adequate optimization techniques. Moreover, it should also be verified that maximizing particle diversity in the optimization environment is sufficient to maximize model diversity in the classification environment.

REFERENCES

[1] E. Granger, M. A. Rubin, S. Grossberg, and P. Lavoie, "A what-and-where fusion neural network for recognition and tracking of multiple radar emitters," *Neural Networks*, vol. 14, pp. 325–344, 2001.

[2] A. Tsymbla, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, vol. 9, no. 1, pp. 56–68, 2008.

[3] J.-F. Connolly, E. Granger, and R. Sabourin, "An adaptive classification system for video-based face recognition," *Information Sciences (in press)*, 2010.

[4] ——, "Supervised incremental learning with the fuzzy artmap neural network," in *Artificial Neural Networks in Pattern Recognition*, Paris, France, Jul. 2008, pp. 66–77.

[5] A. J. C. Sharkey, *Combining artificial neural nets: Ensemble and modular multi-net systems*. New York: Springer-Verlag, 1999, ch. Multi-net systems.

[6] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Evolving a cooperative population of neural networks by minimizing mutual information," in *Proceeding of the IEEE Congress on Evolutionary Computation (CEC)*, Seoul, Korea, May 2001, pp. 384–389.

[7] Z.-H. Zhou and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 293–253, 2002.

[8] L. I. Kuncheva, "Classifier ensembles for changing environments," in *Proc. 5th Int. Workshop on Multiple Classifier Systems (MSC)*, Cagliari, Italy, 2004, pp. 1–15.

[9] G. Valentini, "Ensemble methods based on bias-variance analysis," Ph.D. dissertation, PhD thesis, University of Genova, 2003.

[10] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, 1992.

[11] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "Evaluating the performance of DNPSO in dynamic environments," in *IEEE Int'l Conference on Systems, Man, and Cybernetics*, Singapore, Oct. 2008, pp. 12–15.

[12] 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, May 2005, pp. 325–344.

[13] R. Gross and J. Shi, "The cmu motion of body (mobo) database, 2001," Carnegie Mellon University, Tech. Rep. CMU-RI-TR-01-18, 2001.

[14] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information Science*, vol. 178, no. 15, pp. 3096–3109, 2008.

[15] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Genetic And Evolutionary Computation Conference*, Seattle, USA, Jul. 2006, pp. 51–58.

[16] E. Özcan and M. Yýlmaz, "Particle swarms for multimodal optimization," in *Adaptive and Natural Computing Algorithms*, Warsaw, Poland, Apr. 2007, pp. 366–375.

[17] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, Coimbra, Portugal, Apr. 2004, pp. 489–500.

[18] A. Carlisle and G. Dozier, "Tracking changing extrema with adaptive particle swarm optimizer," in *World Automation Congress*, Orlando, Florida USA, Jun. 2002, pp. 265–270.

[19] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," in *IEEE Congress on Evolutionary Computation*, vol. 2, Honolulu, USA, May 2002, pp. 1666–1670.

[20] H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, Valencia, Spain, Apr. 2007, pp. 637–646.

[21] E. Granger, P. Henniges, L. S. Oliveira, and R. Sabourin, "Supervised learning of fuzzy artmap neural networks through particle swarm optimization," *Journal of Pattern Recognition Research*, vol. 2, no. 1, pp. 27–60, 2007.