Applied Soft Computing 13 (2013) 3149–3166

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Dynamic multi-objective evolution of classifier ensembles for video 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

ARTICLE INFO

Article history:
Received 1 January 2012
Received in revised form 6 June 2012
Accepted 20 August 2012
Available online 1 September 2012

Keywords:
Dynamic multi-objective optimization
Multi-classifier systems
Ensemble methods
Incremental learning
ARTMAP neural networks
Face recognition in video
Adaptive biometrics

ABSTRACT

Due to a limited control over changing operational conditions and personal physiology, systems used for video-based face recognition are confronted with complex and changing pattern recognition environments. Although a limited amount of reference data is initially available during enrollment, new samples often become available over time, through re-enrollment, post analysis and labeling of operational data, etc. Adaptive multi-classifier systems (AMCSs) are therefore desirable for the design and incremental update of facial models. For real time recognition of individuals appearing in video sequences, facial regions are captured with one or more cameras, and an AMCS must perform fast and efficient matching against the facial model of individual enrolled to the system. In this paper, an incremental learning strategy based on particle swarm optimization (PSO) is proposed to efficiently evolve heterogeneous classifier ensembles in response to new reference data. This strategy is applied to an AMCS where all parameters of a pool of fuzzy ARTMAP (FAM) neural network classifiers (i.e., a swarm of classifiers), each one corresponding to a particle, are co-optimized such that both error rate and network size are minimized. To provide a high level of accuracy over time while minimizing the computational complexity, the AMCS integrates information from multiple diverse classifiers, where learning is guided by an aggregated dynamical niching PSO (ADNPSO) algorithm that optimizes networks according both these objectives. Moreover, pools of FAM networks are evolved to maintain (1) genotype diversity of solutions around local optima in the optimization search space and (2) phenotype diversity in the objective space. Accurate and low cost ensembles are thereby designed by selecting classifiers on the basis of accuracy, and both genotype and phenotype diversity. For proof-of-concept validation, the proposed strategy is compared to AMCSs where incremental learning of FAM networks is guided through mono- and multi-objective optimization. Performance is assessed in terms of video-based error rate and resource requirements under different incremental learning scenarios, where new data is extracted from real-world video streams (IIT-NRC and MoBo). Simulation results indicate that the proposed strategy provides a level of accuracy that is comparable to that of using mono-objective optimization and reference face recognition systems, yet requires a fraction of the computational cost (between 16% and 20% of a mono-objective strategy depending on the data base and scenario).

1. Introduction

In biometric applications, matching is typically performed by comparing query samples captured with some sensors against biometric models designed with reference samples previously obtained during an enrollment process. In its most basic form, template matching is performed with biometric models consisting of a set of one or more templates (reference samples) stored in a gallery. To improve robustness and reduce resources, it may also consist of a statistical representation estimated by training a classifier on reference data. Neural or statistical classifiers then implicitly define a model of some individual's physiological or behavioral trait by mapping the finite set of reference samples, defined in an input feature space, to an output score or decision space. Still, the collection and analysis of reference data from individuals is often expensive and time consuming. Therefore, classifiers are often designed using some prior knowledge of the underlying data distributions, a set of user-defined hyperparameters (e.g., learning rate), and a limited number of reference samples.

In many biometric applications however, it is possible to acquire new reference samples at some point in time after a classifier has originally been trained and deployed for operations. Labeled and unlabeled samples can be acquired through re-enrollment sessions, post-analysis of operational data, or enrollment of new individuals in the system, allowing for incremental learning of labeled data

* Corresponding author. Tel.: +1 514 396 8650.
E-mail addresses: jfconnolly@livia.etsmtl.ca (J.-F. Connolly), Eric.Granger@etsmtl.ca (E. Granger), Robert.Sabourin@etsmtl.ca (R. Sabourin).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.08.039

and semi-supervised learning of reliable unlabeled data [29,51]. In video-based face recognition, facial images may also be tracked and captured discreetly and without cooperation over a network of IP cameras [29]. Face acquisition is subject to considerable variations (e.g., illumination, pose, facial expression, orientation and occlusion) due to limited control over unconstrained operational conditions. In addition, new information, such as input features and new individuals, may suddenly emerge, and underlying data distributions may change dynamically in the classification environment. The physiology of individuals (e.g., aging) and operational condition may therefore also change gradually, incrementally, periodically and abruptly over time [67]. Performance may therefore decline over time as facial models deviate from the actual data distribution [25,45,55].

Beyond the need for accurate face recognition techniques in video, efficient classification systems for various real-time applications constitute a challenging problem. For instance, video surveillance systems use a growing numbers of IP cameras, and must simultaneously process many video feeds. The computational burden increases with the number of matching operations, and thus the number of individuals and cameras, frame rate, etc.

This paper seeks to address challenges related to the design of robust adaptive multi-classifier systems (AMCSs) for video face recognition, where facial models may be created and updated over time, as new reference data becomes available. An incremental learning strategy driven by a dynamic particle swarm optimization (DPSO) and AMCS architecture were previously developed by the authors in [14]. In this DPSO-based strategy, each particle corresponds to a fuzzy ARTMAP (FAM) network, and a DPSO algorithm optimizes all classifier parameters (hyperparameters, weights, and architecture) of a swarm of base classifiers such that the error rate. While adaptation was originally performed only according to accuracy with mono-objective optimization, the new strategy and AMCS proposed in this paper is driven by a new multi-objective aggregated dynamic niching PSO (ADNPSO) algorithm that also considers the structural complexity of FAM networks during adaptation, allowing to design efficient heterogeneous ensembles of classifiers.

This approach also differs from previous work by the authors [14] in that a specialized archive is used to capture base classifiers from the swarm and maintain a pool. To further reduce the computational cost, this archive is constantly modified through time by adding non-dominated classifiers and removing dominated ones with locally Pareto-optimal criteria. These locally Pareto-optimal criteria are again used within that pool to select ensembles that are both accurate and with low complexity.

Most techniques in literature are suitable for designing classification systems with an adequate number of samples acquired from ideal environments, where class distributions remain unchanged over time. However, classifier ensembles are well suited for adaptation in changing environments. Adaptive ensemble-based techniques like Learn++[46] and other Boosting variants, where a new classifier is trained independently for new samples, and classifiers are weighted such that one criterion is maximized (classification accuracy on recent data), may provide a robust approach [38]. Other approaches discard classifiers when they become inaccurate or concept change is detected [41], although maintaining a pool with these classifiers allows to handle recurrent change. Moreover, methods that rely exclusively on adding new ensemble members become problematic if all classes are not represented within the new data. With the current face recognition application, for instance, when new data becomes available after a classifier is designed and deployed in the field, it will most likely belong to one or few individuals at a time. Previously trained classifiers will not recognize new classes, classifiers trained with the new data will not recognize older classes.

The proposed ADNPSO strategy evolves a pool of incremental learning FAM classifiers, and may refine and add classes on the fly. To increase the performance of heterogeneous ensembles, this strategy seeks to maintain diversity among the base classifiers during generation and evolution of pools, and during ensemble selection, according to several criteria. This paper focuses on video-based face recognition applications in which two incremental learning scenarios may occur – enrollment (initial design) and update of facial models. Performance of AMCSs is assessed in terms of classification rate and resource requirements for incremental learning of new data blocks from two real-world video data sets – Institute of Information Technology of the Canadian National Research Council (IIT-NRC) [22] and Motion of Body (MoBo) [26]. In experiments, the AMCS performs biometric identification of facial regions against the model of individuals in closed-set (1-against-K) identification, as found in access control applications.

The next section provides an overview of the state-of-the-art in adaptive biometrics and a general biometric system for video-based face recognition system. In Section 3, the AMCS framework considered in this paper is described, focusing on the relationship between the classification environment (where the FAM networks learn reference data), and the optimization environment (where particles evolve). The new incremental learning strategy used (including ADNPSO algorithm and specialized archive) to evolve the AMCS is presented in Section 4. The data bases, incremental learning scenarios, protocol, and performance measures used for proof-of-concept simulations are described in Section 5. Finally, experimental results are presented and discussed in Section 6.

2. Adaptive biometrics and video face recognition

The main problem addressed in this paper is the design of accurate and efficient adaptive systems to perform video-to-video face recognition, where video sequences are used for building the facial model of each individual during the learning phase. Adaptive systems have been proposed in the literature to refine biometric models for different traits (e.g., face and fingerprints) according to the intra-class variations in input samples [51]. With self-adaptive or semi-supervised learning strategies, biometric models are initially designed during enrollment using labeled training data, and then updated with highly confident unlabeled data obtained during operations [45,49]. These strategies are however vulnerable to outliers, dispersion and overlap in class distributions. Stringent criteria are required for selection of highly confident data, to minimize the probability of introducing impostor data into updated biometric models.

On the other hand, systems have used newly acquired labeled reference samples to update the selection of user template from a gallery via clustering and editing techniques [57], and have performed on-line learning of genuine samples over time to update each user's single super template [30]. It is however difficult to represent intra-class variations with a single template [51]. In either case, the biometric model of an individual tends to diverge from its underlying class distribution due to the limited reference data, complexity, and changes in the classification environment. In their efforts to avoid model corruption and to maintain a high level of accuracy, classifiers adapted incrementally over time tend to become complex [14].

Biometric systems specifically designed for the recognition of faces in video streams are relevant in different scenarios and applications. Applications of video-based face recognition range from open-set video surveillance, where individuals enrolled to a watch list are recognized among other unknown people in dense and moving crowds [18], to closed-set identification or verification for

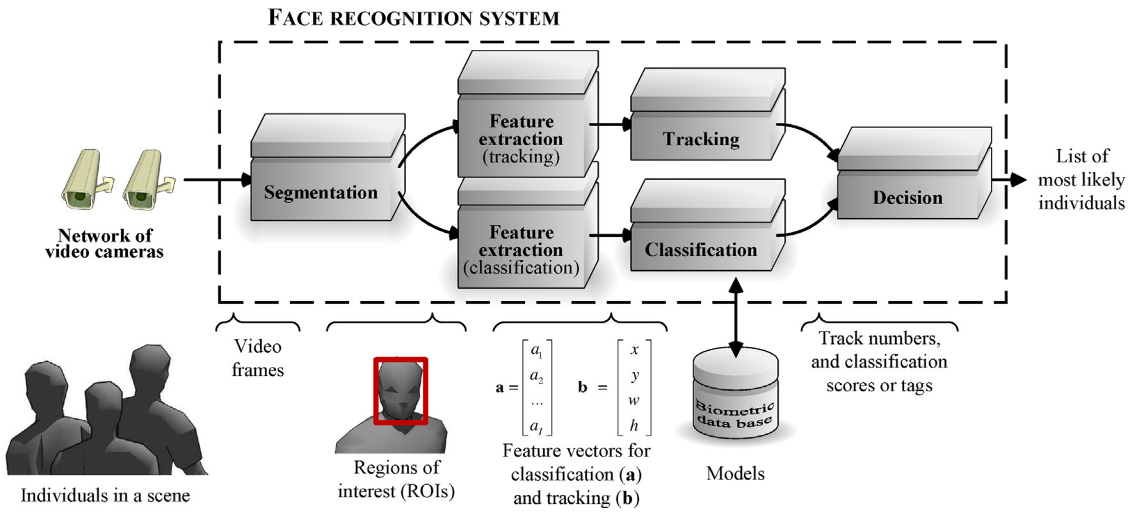
FACE RECOGNITION SYSTEM
Network of video cameras
Segmentation
Feature extraction (tracking)
Feature extraction (classification)
Tracking
Classification
Decision
List of most likely individuals
Individuals in a scene
Video frames
Regions of interest (ROIs)
$\mathbf{a} = [a_1, a_2, \ldots, a_I]^T$ $\mathbf{b} = [x, y, w, h]^T$
Feature vectors for classification (a) and tracking (b)
Biometric data base
Models
Track numbers, and classification scores or tags

Fig. 1. A generic track-and-classify biometric system for video-based face recognition.

access control applications, where individuals enrolled to a system are authenticated prior to accessing secured resources, possibly in conjunction with a password, access card, etc. [53]. In this paper, video-based face recognition is considered for closed-set identification applications.

In addition to difficulties mentioned earlier, video-based face recognition remains a very challenging problem since faces captured in video frames are typically low quality and generally small. The design of efficient systems for facial matching involves a trade-off between classification speed, accuracy and resources for storage of facial models. In video-based face recognition, fast classification is often required to process facial regions at near real-time processing (captured at 30 frames/s in each video feed). It is well-known that state-of-the-art systems are confronted with complex environments that change during operations, and their facial models are designed during a preliminary enrollment process, using limited data and knowledge of individuals. The need to design and store representative facial models for recognition – be it with more user templates or a statistical representation – increases the resource requirements of the system.

A typical approach used to recognize faces in video streams consists in exploiting only spatial appearance information, and applying extensions of still image techniques on high quality facial regions captured through segmentation [37]. Several powerful techniques proposed to recognize faces in static 2D images are described in [63,64]. The predominant techniques are the same used to represent faces in static 2D images: appearance-based methods like Eigenfaces, and feature-based methods like Elastic Bunch Graph Matching [63,64]. However, the performance of these techniques may degrade considerably when applied for video-based face recognition in unconstrained scenes. To reduce matching ambiguity and provide a higher level of accuracy, face recognition applications specifically designed toward video sequences combine spatial and temporal information contained in video streams [17].

In this paper, it is assumed that a track-and-classify system is used to accumulate the responses of a classifier using kinematic information of faces in a scene [37]. Fig. 1 depicts a general track-and-classify for spatio-temporal recognition of faces in video. It is assumed that 2D images in the video streams of an external 3D scene are captured using one or more IP or network cameras.

First, the system performs segmentation to locate and isolate regions of interest (ROIs) corresponding to the faces in a frame. From the ROIs, features are extracted for tracking and classification. The tracking features can be the position, speed, acceleration, and track number assigned to each ROI on the scene so that the tracker may follow the movement or expression of faces across video frames [25]. On the other hand, classifiers will require invariant and discriminant classification features extracted from the ROIs so that the classification module may match input feature patterns, mapped in an $\mathbb{R}^I$ input feature space, to the face models of individuals enrolled to the system. Facial matching may be implemented with templates, statistical, or neural pattern classifiers. With neural network classifiers, for instance, the facial model of individuals by the hyperparameters, synaptic weights, and architecture estimated during training.

Finally, the decision module may integrate the responses from the tracking and classification modules over several video frames. If the decision module employs a track-and-classify approach, the facial regions are presented to the face recognition system and predictions for each ROI are accumulated over time according to the facial trajectories defined by the tracker. With identification and surveillance applications for instance, ambiguity is reduced by accumulating responses (classification scores) over several frames over the trajectory of each individual in the scene, thus improving accuracy and robustness of face recognition in video [3].

Although the current paper uses a track-and-classify architecture other methods exist for spatio-temporal recognition in video sequences. Head and facial motion during the sequence can be exploited by either estimating the optical flow or tracking a few facial landmarks over time with a template matching strategy. Temporal dynamics and statistics of training video sequences can also be modeled using Hidden Markov Models, particle filters, or time series state space models. A probabilistic appearance manifold approach can also be used to exploit temporal information of each successive frame in a video sequence. Bayesian inference then allows to include temporal coherence in distance calculation during recognition. A review of recent techniques for spatio-temporal face recognition for video sequences can be found in [37].

With most systems for video face recognition, conditions for data acquisition are typically considered to be constrained, and the physiology of individuals and operational condition do not change over time. Systems are designed a priori, during a preliminary enrollment phase, but the number of reference samples and knowledge of class distributions are limited. Adapting the system in response to new reference data may allow to maintain a high level of performance by reducing the divergence over time between facial models and underlying data distributions in the real-world environments. However, most classification techniques used for face matching would require training from the start using all previously acquired data through supervised batch learning.

In the next section an ADNPSO strategy is proposed for supervised incremental learning which allows to enroll and update the facial model of individuals from video streams after the face recognition system has been deployed for operations. Efficient incremental learning is an undisputed asset as the memory and time complexity associated with storing and training is greatly reduced. The objective of this new ADNPSO strategy is to evolve classifiers according to both accuracy and network size, leading to more accurate and reliable systems that perform efficient matching of captured facial regions.

3. Adaptive classifier ensembles

Adapting facial models in changing classification environments, such as required for enrollment or update in video face recognition, raises the so-called stability–plasticity dilemma, where stability refers to retaining existing and relevant knowledge while plasticity enables learning new knowledge [27]. Since ensemble based methods allow to exploit multiple and diverse views of a problem, they have been shown to be efficient in such cases, where concepts (i.e., underlying data distributions) change in time [38].

For a wide range of applications, where adaptation is not necessarily required, classifier ensembles allow to exploit several views of a same problem to improve the overall accuracy and reliability. Recently, various methods employing adaptive ensembles of classifiers have been proposed to perform incremental learning [31,46]. With the use of a combination function, they also offer flexibility over single classifiers in how class models can be managed and adapted.

These methods can be divided in three general categories [34]. Dynamic combination, or “horse racing”, methods where individual base classifiers are trained in advance to form a fixed ensemble where only the combination rules are changed dynamically [6,61,62]. Second, methods that rely on new data to update the parameters of ensemble base classifiers an online learner [20]. If blocks of data are available, training can also be performed in batch mode while changing or not the combination rule at the same time [7,21,43,59]. The last main category consists of methods that grow ensembles by adding new base classifiers and replacing old or underperforming ones when new data is available [10,33,54,55]. Finally there are adaptive ensembles that use hybrid approaches that combine adding new base classifiers and adjusting the combination rule to update class models. The most notable are streaming random forests with entropy [1], Hoeffding tree with Kalman filter-based active change detection using adaptive sliding window [4], maintaining and choosing the better of two ensembles trained with current and old data [52], and the AdaBoost-like Learn++[46].

Among these methods, horse racing approaches cannot accommodate new knowledge since base classifiers in the ensemble are never updated with new data. On the other hand, while online learners and growing ensembles can be used to explore unknown regions of the feature space, and focus on the issue of concept drift, where underlying class distributions change in time. They often train and combine new classifiers to a pool without updating pre-existing classifiers at the risk of corrupting older knowledge. While these classifiers are trained with new data, their plasticity (or learning dynamics) tends to remain fixed throughout the learning process, without being adjusted to accommodate new knowledge. Video face recognition systems in unconstrained scenes are often faced with recurring changes regarding the environment (e.g., light effect over the course of a day) and the individuals to recognize (e.g., glasses). Since few reference samples are available, hidden concepts are often revealed (different known view points from a sensor or of a trait).

In practice, when new reference data becomes available during operations, it will most likely incorporate samples captured from one or few individuals at a time. With growing ensembles, previously trained classifiers will not be able to integrate new classes, and the new ones (trained with the new reference data) will represent only facial models of the latest individuals registered to the system.

In previous work, the authors have proposed an adaptive multiclassifier system (AMCS) that is driven by a strategy based on dynamic particle swarm optimization (DPSO) for supervised incremental learning for the design and update of facial biometric models [14]. Given its capabilities to perform supervised incremental learning of limited data, and to efficiently match the query samples to facial models in the system, the fuzzy ARTMAP (FAM) neural networks are used as the ensemble's base classifier. Using DPSO and a cooperative neural network co-evolution paradigm [47], the incremental learning strategy is applied to the optimization of a swarm of FAM networks in the hyperparameter search space. As illustrated in Fig. 2, DPSO explores the hyperparameter search spaces and guides a swarm of different FAM classifiers. They are trained on the same data, but using different learning dynamics, i.e., different hyperparameter settings. This process yields an *heterogeneous*[1] pool of classifiers that is diversified in both feature and decision spaces [58]. When new labeled reference data becomes available from the operational environment, classifier ensembles evolve to design new facial models or update existing ones, since this approach does not directly optimize FAM parameters (i.e., synaptic weights for neural networks), and can be applied other classifiers. Other examples showing how particle swarm optimization algorithms are applied in this manner are summarized in [24,31].

By applying the DPSO strategy within an adaptive classification system (ACS), the authors have previously shown that to perform incremental learning with constructive classifiers such as FAM networks, some of the older data must be stored in memory so that old and still valid knowledge is not overshadowed by newer concepts corresponding to incoming reference data [13]. They have also shown that optimizing FAM learning dynamics according to accuracy during supervised incremental learning corresponds to a dynamic mono-objective optimization problem [13]. Within the AMCS, the authors have then verified that with FAM networks, *genotype* (i.e., hyperparameter) diversity among solutions in the search space leads to ensemble diversity in the feature and decision spaces [14]. Although these AMCSs provide a high level of accuracy and robustness when only limited data is available, FAM networks are generated through mono-objective optimization of accuracy, and become structurally complex over time, as new data is learned.

3.1. An adaptive multiclassifier system

Fig. 3 depicts the evolution of an AMCS performing incremental learning of new data. It is composed of (1) a long term memory (LTM) that stores and manages incoming data for validation, (2) a population of base classifiers, each one suitable for supervised incremental learning, (3) a dynamic population-based optimization module that tunes the user-defined hyperparameters of each classifier, (4) a specialized archive to keep a pool of classifiers for ensemble selection, and (5) an ensemble selection and fusion module. This system differs from the AMCS presented in [14] in that

[1] This definition of heterogeneous ensembles differs with respect to certain others found in literature. In this paper, they are defined as similar classifiers that learn different data sets, or classifiers of different types train on the same data [42,48].

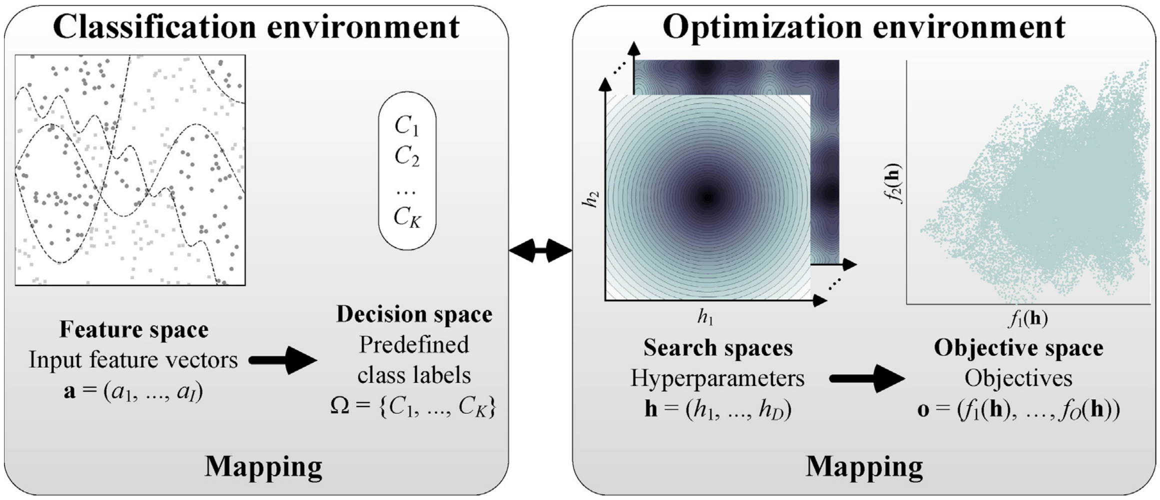

Fig. 2. Pattern classification systems may be defined according to two environments. A *classification environment* that maps a $\mathbb{R}^I$ input feature space to a decision space, respectively defined by feature vectors **a**, and a set of class labels C_k. As classifier learning dynamics is governed by a vector **h** of hyperparameters, the latter interacts with an *optimization environment*, where each value of **h** indicates a position in several search spaces, each one defined by an objective considered during the learning process. For several objective functions (each corresponding to a search space), solutions (trained FAM networks) can be projected in an objective space.

the optimization module now performs multi-objective (rather than mono-objective) optimization, and the pool of classifiers, from which ensemble selection is performed, is now an archive that is filled during the multi-objective optimization (MOO) process.

When a new block of learning data D_t becomes available to the system at a discrete time t, it is employed to update the LTM, and evolve the swarm of incremental classifiers (see Fig. 3). Each FAM network is associated to a particle in an hyperparameter search space, and a DPSO module, through a DPSO-based learning strategy, conjointly determines the classifier hyperparameters, architecture, and parameters such that FAM network's error rate *and* size are minimized. A specialized archive stores a pool of classifiers, corresponding to locally non-dominated solutions (of different structural complexity) found during the optimization process. Once the optimization process is complete, the selection and fusion module produces a heterogeneous ensemble by selecting classifiers from the archive (or pool), based on their accuracy, genotype, and phenotype diversity. It then combines them with a simple majority vote. The LTM stores reference samples from each individual for cross-validation during incremental learning and fitness estimation of particles on the objective function [13]. Data from D_t is partitioned and combined with that of the LTM to create three subsets: a training data set D_t^{t}, a validation data set D_t^{v}, and a fitness estimation data set D_t^{f}.

In this paper, a particular realization of this AMCS is considered. FAM neural networks [9] are employed to implement the swarm of incremental learning classifiers and a new ADNPSO algorithm is used for optimization according to multiple objectives. The rest of this section provides additional details on the FAM and on the optimization module. The ADNPSO algorithm, specialized archive, and selection and fusion modules are discussed in Section 4 along with the ADNPSO incremental learning strategy.

3.2. Fuzzy ARTMAP neural network classifiers

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. A key feature of these networks is their unique solution to the stability–plasticity dilemma. The fuzzy ARTMAP (FAM) integrates the unsupervised fuzzy ART neural network to process both analog and binary-valued input patterns into the original ARTMAP architecture [9]. Matching ROIs (represented with appearance pattern **a**) against the facial model of individuals enrolled to a face recognition system is typically the bottleneck, especially as the number of individuals grows, and the FAM classifier is used because it can perform supervised incremental learning of limited data for fast and efficient matching. The facial models are learned a priori (during training) by estimating the FAM weights, architecture and hyperparameters of each individual (i.e., output class) enrolled to the system.

The fuzzy ART neural network consists of two fully connected layers of nodes: a $2I$ node input layer F_1 to accommodate complement-coded input patterns, and a J node competitive

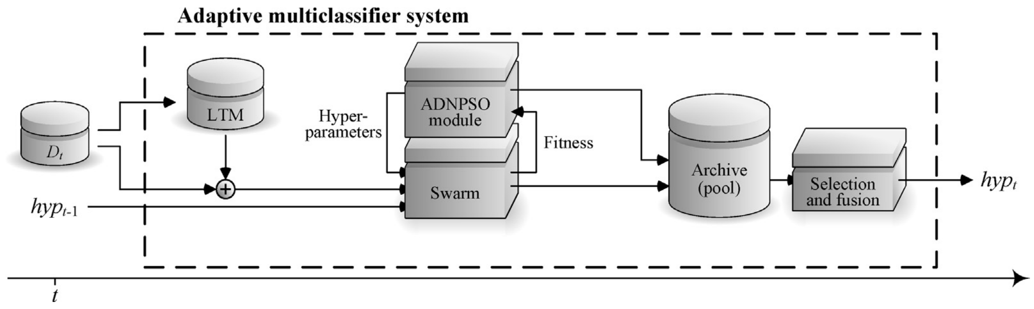

Fig. 3. Evolution over time of the adaptive multiclassifier system (AMCS) in a generic incremental learning scenario, where new blocks of data are used to update a swarm of classifiers. Let $D_1, D_2, \ldots$ be blocks of training data that become available at different instants in time $t = 1, 2, \ldots$. The AMCS starts with an initial hypothesis hyp_0 according to the prior knowledge of the classification environment. On the basis of new data blocks D_t, each hypothesis hyp_{t-1} is updated to hyp_t by the AMCS.

layer, F_2. A set of real-valued weights $\mathbf{W} = \{w_{ij} \in [0, 1] : i = 1, 2, \ldots, 2I; j = 1, 2, \ldots, J\}$ is associated with the F_1-to-F_2 layer connections. The F_2 layer is connected, through learned associative links, to an output K node map field F_{ab}, where K is the number of classes in the decision space. With FAM, a set of binary weights $\mathbf{W}^{ab} = \{w^{ab}_{jk} \in \{0, 1\} : j = 1, 2, \ldots, J; k = 1, 2, \ldots, K\}$ is associated with the F_2-to-F_{ab} connections. Each F_2 node $j = 1, \ldots, J$ corresponds to a category that learns a prototype vector $\mathbf{w}_j = (w_{1j}, w_{2j}, \ldots, w_{2Ij})$, and is associated with one of the output classes $k = 1, \ldots, K$. During the training phase, FAM dynamics is governed by four hyper-parameters: the choice parameter $\alpha > 0$, the learning parameter $\beta \in [0, 1]$, the baseline vigilance parameter $\overline{\rho} \in [0, 1]$, and the match-tracking parameter $\epsilon \in [-1, 1]$. For incremental learning, FAM is able to adjust previously learned categories, in response to familiar inputs, and to create new categories dynamically in response to inputs different enough from those already seen.

The following describes fuzzy ARTMAP during supervised learning of a finite data set. When an input pattern $\mathbf{a} = (a_1, \ldots, a_I)$ is presented to the network and the vigilance parameter $\rho \in [0, 1]$ is set to its baseline value $\overline{\rho}$. The input pattern $\mathbf{a}$ is complement-coded to make a $2I$ dimensions network's input pattern: $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2, \ldots, a_I; a^c_1, a^c_2, \ldots, a^c_I)$, where $a^c_i = (1 - a_i)$, and $a_i \in [0, 1]$. Each F_2 node is activated according to the *Weber law choice function*:

$$T_j(\mathbf{A}) = \frac{|\mathbf{A} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{1}$$

and the node with the strongest activation $j^* = \operatorname{argmax}\{T_j : j = 1, \ldots, J\}$ is chosen. The algorithm then verifies if $\mathbf{w}_{j^*}$ is similar enough to $\mathbf{A}$ using the vigilance test:

$$\frac{|\mathbf{A} \wedge \mathbf{w}_{j^*}|}{2I} \geq \rho. \tag{2}$$

If node j^* fails the vigilance test, it is deactivated and the network searches for the next best node on the F_2 layer. If the vigilance test is passed, then the map field F^{ab} is activated through the category j^* and FAM makes a class prediction $k^* = k(j^*)$. In the case of an incorrect class prediction $k^* = k(j^*)$, a match tracking signal adjusts $\rho = (|\mathbf{A} \wedge \mathbf{w}_{j^*}|/2I) + \epsilon$. Node j^* is deactivated, and the search among F_2 nodes begins anew. If node j^* passes the vigilance test, and makes the correct prediction, its category is updated by adjusting its prototype vector $\mathbf{w}_{j^*}$ to:

$$\mathbf{w}'_{j^*} = \beta(\mathbf{A} \wedge \mathbf{w}_{j^*}) + (1 - \beta)\mathbf{w}_{j^*}. \tag{3}$$

On the other hand, if none of the nodes can satisfy both conditions (vigilance test and correct prediction), then a new F_2 node is initialed. This new node is assigned to class K by setting $w^{ab}_{j^*k}$ to 1 if $k = k^*$ and 0 otherwise, and $\mathbf{w}'_{j^*} = \mathbf{A}$.

Once the weights $\mathbf{W}$ and $\mathbf{W}^{ab}$ have been found through this process, the fuzzy ARTMAP can predict a class label from an input pattern by activating the best F_2 node j^*, which activates a class $k^* = k(j^*)$ on the F_{ab} layer. Predictions are obtained without vigilance and match tests. During operation, time and memory complexity of FAM is proportional to its structural complexity and depends heavily on the number of F_1 and F_2 layer nodes. To perform predictions given an input pattern of I features F_1 layer and an F_2 layer of J nodes, FAM networks complement code the I features, compute the choice function for the J category prototypes, leading to a worst-case time and memory complexity per input sample of $O(IJ)$. During incremental learning, the F_2 layer tends to grow depending on the hyperparameter values, the number of reference samples, and the geometry of the underlying data distributions. In the worst case, FAM will memorize the training data set, and create one F_2 category node per reference sample. In this paper, it is assumed that incremental learning of new data is not performed on-line, but in a relatively short time frame.

A standard vector of hyperparameters $\mathbf{h}_{\text{std}} = (\alpha = 0.001, \beta = 1, \epsilon = 0.001, \overline{\rho} = 0)$ is commonly fixed to minimize network structural complexity [9]. The authors have shown that by adjusting these hyperparameters, it is possible to adapt FAM learning dynamics with regard to currently available training data [13,14,23]. It is possible to generate heterogeneous pools of classifiers [12]. Moreover, they have also verified the amount of diversity among hyperparameter vectors $\mathbf{h}$ of each classifier is correlated with the amount of diversity within a pool of classifier [14]. Using these results, the authors generate a pool of diversified FAM networks, and select ensembles among that pool for improved generalization capabilities. However, since these ensembles were created through a mono-objective optimization process focused only on accuracy, each network of the pool tends to create several prototype categories when learning new data, leading to a considerable computational cost.

3.3. Adaptation as a dynamic MOO problem

In this paper, the AMCS optimization module will seek to find the hyperparameter vector $\mathbf{h} = (\alpha, \beta, \epsilon, \overline{\rho})$ that seeks to maximize FAM accuracy while minimizing network structural complexity, that is:

$$\text{minimize}\{\mathbf{f}(\mathbf{h}, t) := [f_e(\mathbf{h}, t), f_s(\mathbf{h}, t)] | \mathbf{h} \in \mathbb{R}^4, t \in \mathbb{N}_1\}, \tag{4}$$

where $f_e(\mathbf{h}, t)$ is the generalization error rate and $f_s(\mathbf{h}, t)$ is the size of the F_2 layer (i.e., number of F_2 nodes) of the FAM network for a given hyperparameter vector $\mathbf{h}$, and after learning data set D_t incrementally at a discreet time t [13]. In this context, it has been shown that adapting the FAM classifier's hyperparameter vector $\mathbf{h} = (\alpha, \beta, \epsilon, \overline{\rho})$ according to $f_e(\mathbf{h}, t)$ corresponds to a dynamic mono-objective optimization problem [13]. More precisely, it constitutes a type III optimization environment, where both the location and value of optima positions change in time [19]. Although it was not explicitly verified, it is assumed that training FAM with different values of $\mathbf{h}$ leads to different numbers of FAM F_2 nodes and that the objective function $f_s(\mathbf{h}, t)$ also corresponds to a type III optimization environment. Still it is sufficient that only one of the objectives does so for the entire optimization problem to be considered dynamic.

As a MOO problem, the first goal of the optimization module is to find the Pareto front of non-dominated solutions according to several objectives (see Fig. 4). Given the set of objectives $\mathbf{o}$ to minimize, a vector $\mathbf{h}^d$ in the hyperparameter space is said to *dominate* another vector $\mathbf{h}$ if (see Fig. 4a):

$$\forall o \in \mathbf{o} : f_o(\mathbf{h}^d) \leq f_o(\mathbf{h}) \quad \text{and} \quad \exists o \in \mathbf{o} : f_o(\mathbf{h}^d) < f_o(\mathbf{h}). \tag{5}$$

The Pareto optimal set, defining a Pareto front, is the set of non-dominated solutions (Fig. 4b).

When adapting classifiers during incremental learning, another goal of the optimization algorithm is to seek hyperparameter values that generate a diversified pool of FAM networks among which ensembles can be selected. As illustrated in Fig. 5 with a simple MOO problem, the optimization process should provide accurate solutions with different network structural complexities. This results in ensembles with good generalization capabilities, but with a possibility of limiting overall computational cost.

In this particular case, the optimization algorithm also tackles a dynamic optimization problem by considering several objectives, and yield classifiers that correspond to vectors $\mathbf{h}$ that are not necessarily Pareto optimal (see Fig. 5). Classical DPSO algorithms as well suited as MOO algorithms, such as non-sorted genetic algorithm (NSGA) [15], strength Pareto evolutionary algorithm (SPEA) [66], and multi-objective PSO (MOPSO) [11]. The only other approaches in literature aimed at generating and evolving a diverse population of FAM networks in terms of structural complexity, yet contained non-dominated alternatives are presented in [24,35]. In [24], a

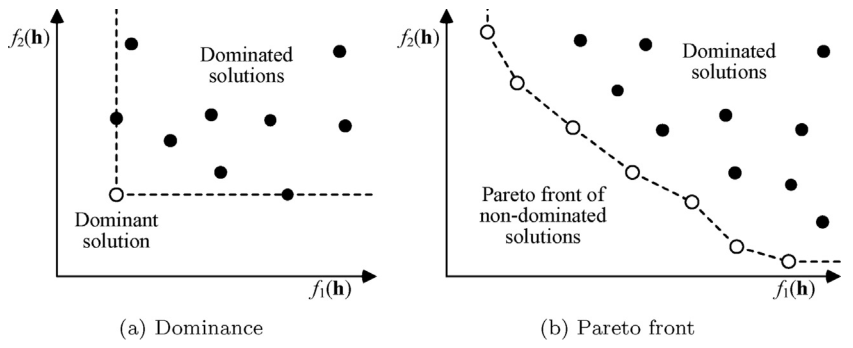

Fig. 4. Notion of dominance (a) and Pareto optimal front (b) for a MOO (minimization) problem in the objective space defined by two objectives $f_1(\mathbf{h})$ and $f_2(\mathbf{h})$.

MOPSO learning strategy is used to train FAM networks according to both error rate and network size. Although this strategy seeks to maintain phenotype diversity in the objective space, results showed that using MOPSO and a global Pareto-optimality criteria limits the number of non-dominated classifiers stored in the archive. To circumvent this issue, a mimetic archive was instead used in [35] to prune F_2 nodes and categorize FAM networks in subpopulations that are independently evolved with a genetic algorithm. With this method, FAM networks need to be pruned to maintain phenotype diversity, which is not the case in this paper.

4. Evolution of incremental learning ensembles

This paper seeks to address challenges related to the design of robust AMCSs for video face recognition applications, where

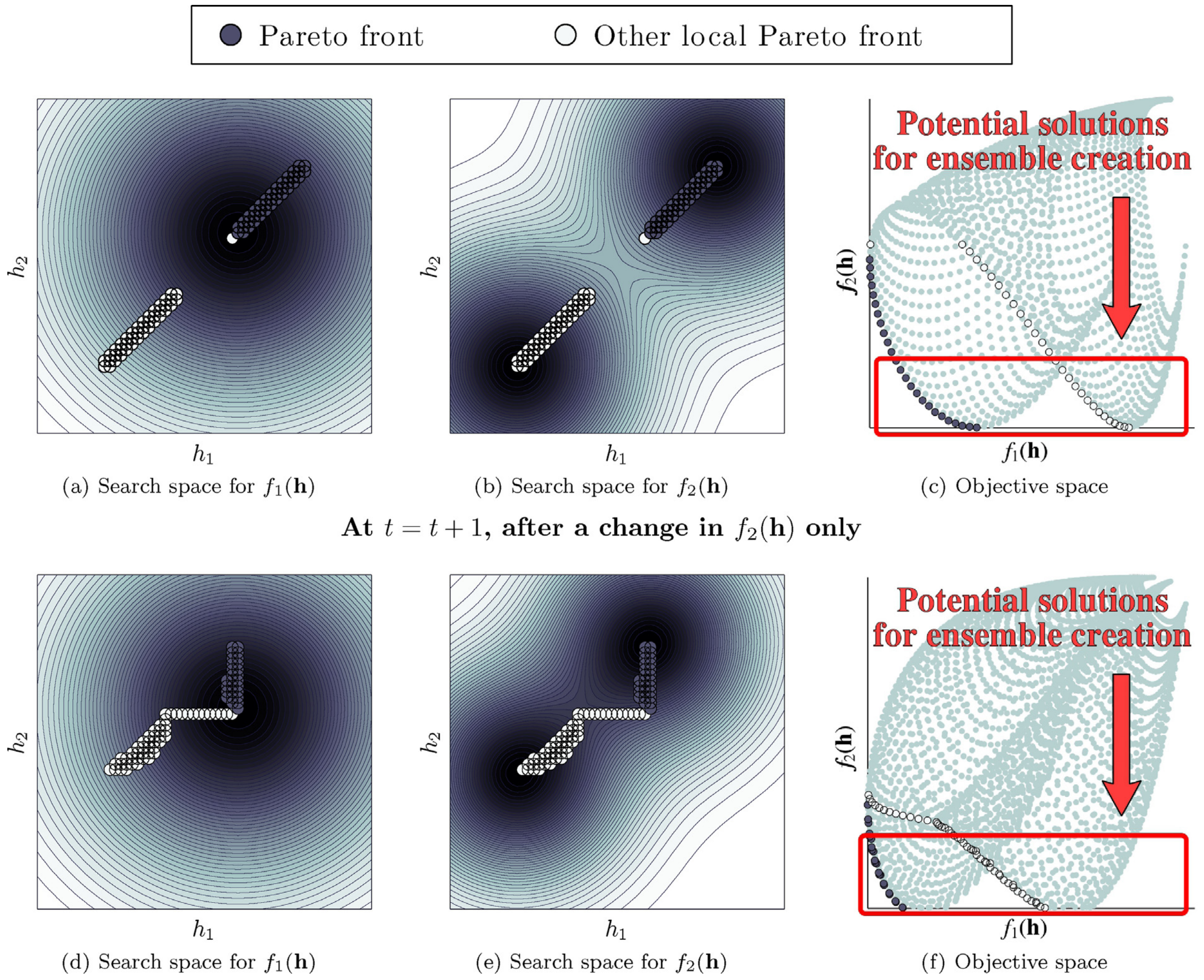

Fig. 5. Position of local Pareto fronts in both search spaces and the objective space. Obtained with a grid, true optimal solutions are illustrated by the dark circles and other locally Pareto-optimal solutions with light circles. While the goal in a MOO is to find the optimal Pareto front (dark circles), another goal of the AMCS ADNPSO module is to search both search spaces to find solutions that are suitable for classifiers ensembles. For instance, if at a time t, $f_1(\mathbf{h})$ and $f_2(\mathbf{h})$ respectively correspond to $f_s(\mathbf{h}, t)$ and $f_e(\mathbf{h}, t)$, these would be solutions in the red rectangle in (c) and (f) (with low generalization error and for a wide range of FAM network F_2 sizes). Even if, at a time $t = t + 1$, change occurs for only one objective function (e), the entire objective space is affected and the problem must be considered dynamic. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

facial models are designed and updated over time, as new reference data becomes available. An ADNPSO incremental learning strategy – integrating an aggregated dynamical niching particle swarm optimization (ADNPSO) algorithm and a specialized archive – is proposed to evolve heterogeneous classifier ensembles in response to new reference data. Each particle in the optimization environment corresponds to a FAM network in the classification environment, and the ADNPSO incremental learning strategy evolves a swarm of classifiers such that both FAM generalization error rate and network size are minimized.

Particles are guided by the ADNPSO algorithm. As with the DNPSO algorithm [39], the ADNPSO algorithm is also able to detect and track many local optima in a type III dynamic optimization environment. In addition, it exploits several objective functions while maintaining genotype diversity in the search spaces, in particular around local optima. However, unlike existing multi-objective optimization (MOO) algorithms (such as NSGA, MOEA, and MOPSO), optimization does not rely on the objective space – it exploits information available in the search space to determine fitness values and future search directions for each solution. This results in an optimization algorithm that is influenced by different objectives. It is aimed at generating pools of classifiers with high *genotype* and *phenotype diversity*, rather than purely solving a MOO problem that provides the optimal Pareto front. It does so by (1) maintaining diversity of solutions around the local optima in each search space and (2) adjusting the position of each solution according to the different objective functions, to allow converging toward different local Pareto fronts.

Since particles are constantly moving with certain randomness and at great speed, a specialized archive of solutions divides the objective space according to FAM network structural complexity (i.e., F_2 layer size), and for each division, it captures non-dominated solutions locally along with their associated networks. This effectively maintains a pool of classifiers with high phenotype diversity. From this pool, a greedy search algorithm selects an ensemble of classifiers within the archive on the basis of accuracy, and *both* genotype and phenotype diversity.

The DPSO-based incremental learning strategy developed in [14] has been modified to evolve heterogeneous ensembles in a MOO framework. In particular, the ADNPSO strategy differs from previous research (1) in the way networks are associated with each particle, (2) in the definition of the initial FAM network conditions used to estimate fitness, and (3) in the addition of a specialized archive to store solutions.

This rest of this section provides additional details on the new ADNPSO strategy, including the ADNPSO algorithm, specialized archive to store locally non-dominant solutions according to complexity, and learning strategy used to integrate those components to the AMCS.

4.1. ADNPSO incremental learning strategy

An ADNPSO incremental learning strategy (Algorithm 1) is proposed to evolve FAM networks according to multiple objectives and accumulates a pool of FAM networks in the specialized archive (see Section 4.3), and ensemble selection. During incremental learning of a data block D_t, FAM hyperparameters, parameters and architecture are cojointly optimized such that the generalization error rate and network size are minimized. Based on the hypothesis that maintaining diversity among particles in the optimization environment implicitly generates diversity among classifiers in the classification environment [14], properties of the ADNPSO algorithm are used to evolve diversified heterogeneous ensembles of FAM networks over time.

Algorithm 1. ADNPSO incremental learning strategy.

Inputs: An AMCS and new data sets D_t for learning.
Outputs: A pool of accurate FAM networks with different complexity phenotype diversity.

```
    Initialization:
1:  • Set the swarm and archive parameters,
    • Initialize all $(O+2)N$ networks: $\mathrm{FAM}_{n,o}$, $\mathrm{FAM}_n^{\mathrm{start}}$, and $\mathrm{FAM}_n^{\mathrm{start}}$,
    • Randomly initialize particles positions and velocities, and set
    ADNPSO iteration counter at $\tau = 0$.

    Upon reception of a new data block $D_t$, the following
    incremental process is initiated:
    • Update the fitness of networks associated to the personal best
    positions:
2:  for each particle $n$, where $1 \leq n \leq N$ do
3:    for each objectives $o$, where $1 \leq o \leq O$ do
4:      Train and validate $\mathrm{FAM}_{n,o}$ with $D_t^{\mathrm{t}}$ and $D_t^{\mathrm{v}}$ respectively.
5:      Estimate $f_o(\mathbf{h}_{n,o}^*, t)$ using $D_t^{\mathrm{f}}$.

    • Update the specialized archive:
6:  Update the accuracy of each solution in the archive.
7:  Remove locally dominated solutions form the archive.
8:  for each particle $n$, where $1 \leq n \leq N$ do
9:    for each objectives $o$, where $1 \leq o \leq O$ do
10:     Categorize $\mathrm{FAM}_{n,o}$.
11:     if $\mathrm{FAM}_{n,o}$ is a non-dominated solution for its network size
          domain then
12:       Add the solution to the specialized archive.
13:       Remove solutions in the archive that are locally dominated
          by $\mathrm{FAM}_{n,o}$.

    • Optimization process:
14: while the optimization algorithm does not reach its stopping
        condition do
15:   Update particle positions according to the ADNPSO algorithm
        (Eq. (8)).
      —Estimate fitness and update personal best positions:
16:   for each particle $n$, where $1 \leq n \leq N$ do
17:     $\mathrm{FAM}_n^{\mathrm{est}} \leftarrow \mathrm{FAM}_n^{\mathrm{start}}$
18:     Train $\mathrm{FAM}_n^{\mathrm{est}}$ with validation using $D_t^{\mathrm{t}}$ and $D_t^{\mathrm{v}}$.
19:     Estimate $f_o(\mathbf{h}_n(\tau), t)$ of each objective using $D_t^{\mathrm{f}}$.
20:     for each objective $o$, where $1 \leq o \leq O$ do
21:       if $f_o(\mathbf{h}_n(\tau), t) < f_o(\mathbf{h}_{n,o}^*, t)$ then
22:         $\{\mathbf{h}_{n,o}^*, \mathrm{FAM}_{n,o}, f_o(\mathbf{h}_{n,o}^*, t)\} \leftarrow \{\mathbf{h}_n(\tau), \mathrm{FAM}_n^{\mathrm{est}}, f_o(\mathbf{h}_n(\tau), t)\}$.

      - Update the specialized archive:
23:     Categorize $\mathrm{FAM}_n^{\mathrm{est}}$
24:     if $\mathrm{FAM}_n^{\mathrm{est}}$ is a non-dominated solution for its network size
          domain then
25:       Add the solution to the specialized archive
26:       Remove solutions in the archive that are locally dominated
          by $\mathrm{FAM}_n^{\mathrm{est}}$.
27:   Increment iterations: $\tau = \tau + 1$.

    • Define initial conditions for fitness estimation with $D_{t+1}$ :
28: for each particle $n$, where $1 \leq n \leq N$ do
29:   $\mathrm{FAM}_n^{\mathrm{start}} \leftarrow \mathrm{FAM}_n^{\mathrm{est}}$
```

At a time t, and for each particle n, the current particle position is noted $\mathbf{h}_n$, along with its personal best values on each objective function o, $\mathbf{h}_{n,o}^*$. The values estimated on the objective functions and the best position of each particle are respectively noted $f_o(\mathbf{h}_n, t)$ and $f_o(\mathbf{h}_{n,o}^*, t)$. For O objectives, and the ADNPSO algorithm presented in Section 4.2 that uses N particles, a total of $(O+2)N$ FAM networks are required. For each particle n, the AMCS stores:

1. O networks $\mathrm{FAM}_{n,o}$ associated with $\mathbf{h}_{n,o}^*$ (particle n personal best position on objective function o),
2. the network $\mathrm{FAM}_n^{\mathrm{start}}$ associated to the current position of the each particle n after convergence of the optimization process at time $t-1$, and
3. the network $\mathrm{FAM}_n^{\mathrm{est}}$ obtained after learning D_t with current position of particle n (noted $\mathbf{h}_n$).

While $\text{FAM}_n^{\text{start}}$ represents the state of the particle before learning D_t, $\text{FAM}_n^{\text{est}}$ is the state of the same particle after having explored a position in the search space, and it is used for fitness estimation.

Particle positions are then randomly initialized within their allowed range. When a new D_t becomes available, the optimization process begins. Using the new data and for all objectives o, fitness associated with the best position of each particle ($f_o(\mathbf{h}^*_{n,o}, t)$) is updated along with each network $\text{FAM}_{n,o}$ (lines 3–5). The archive is then updated (lines 6–13). Accuracy of the solutions in the archive is also updated and checked for non-dominance. The archive is then filled accordingly with the networks $\text{FAM}_{n,o}$.

During the initialization process (line 1), the swarm and all FAM networks are initialized. Particle positions are randomly initialized within their allowed range. When a new D_t becomes available, the optimization process begins.

Networks associated with the best position of each particle ($\text{FAM}_{n,o}$) are incrementally updated using the new data, along with their fitnesses $f_o(\mathbf{h}^*_{n,o}, t)$ (lines 3–5). Network in the archive and their fitnesses are also updated in the same manner (lines 6–13). Since accuracy corresponds to dynamic optimization problem, Algorithm 1 verifies if solutions then still respect the non-dominant criteria of the specialized archive. Afterward, the specialized archive is filled accordingly using the networks $\text{FAM}_{n,o}$.

Optimization then continues were it previously ended until the ADNPSO algorithm converges (lines 14–27). During this process, the ADNPSO algorithm explores the search spaces (line 15). It then copies $\text{FAM}_n^{\text{start}}$ to redefine the state of FAM_{est} prior learning at a time t, trains the latter using $\mathbf{h}_n$, and estimates its fitness (lines 17–19). For each objective o, the best position ($\mathbf{h}^*_{n,o}$) and its corresponding fitness ($f_o(\mathbf{h}^*_{n,o}, t)$) and network ($\text{FAM}_{n,o}$) are updated if necessary (lines 20–26). In the cases of equality between $f_o(\mathbf{h}_n, t)$ and $f_o(\mathbf{h}^*_{n,o}, t)$, the network that requires the least resources (F_2 nodes) is used. Each time fitness is estimated at a particle's current position, FAM_{est} is categorized according its network size and added to the archive if it is non-dominated for its F_2 size domain (lines 23–26). Finally, the iteration counter is incremented (line 27).

Once optimization converges, networks corresponding to the last position evaluated of every particle ($\text{FAM}_n^{\text{est}}$) are stored in $\text{FAM}_n^{\text{start}}$ (lines 28 and 29). These networks will thus define the swarm's state prior learning data block D_{t+1}.

During this Algorithm 1 each time fitness is estimated, FAM networks are trained using the training data set D_t^t under five different random pattern presentation orders to minimize the impact of pattern presentation order at a time t. Since the primary objective is accuracy, $\text{FAM}_n^{\text{est}}$ is the network that yields the lowest error rate and the fitness for each objective is defined according the latter.

4.2. Aggregated dynamical niching PSO

Particle swarm optimization (PSO) is a population-based stochastic optimization technique that is inspired by social behavior of bird flocking or fish schooling. By associating a classifier to each particles, PSO is a powerful tool to cojointly optimize swarms of classifiers hyperparameters, parameters, and architecture. With PSO, each particle corresponds to a single solution in the optimization space, and the population of particles is called a swarm. Unlike evolutionary algorithms (such as genetic algorithms), each particle always stores its best position and the best position of its surrounding. In a mono-objective problem and at a discrete iteration τ, particles move through the hyperparameter space and change their positions $\mathbf{h}(\tau)$ under the guidance of Φ sources of influence [32]:

$$\mathbf{h}(\tau+1) = \mathbf{h}(\tau) + w_0(\mathbf{h}(\tau) - \mathbf{h}(\tau-1)) + \sum_{\phi=1}^{\Phi} r_\phi w_\phi(\mathbf{h}_\phi - \mathbf{h}(\tau)), \quad (6)$$

where ϕ is the index of a source of influence, r_ϕ a random number, w_0 an inertia weight, and w_ϕ the weights indicating the importance each influence. With this formalism, each particle (1) begins at its current location, (2) continues moving in the same direction it was going according to an inertia weight w_0, and (3) is attracted by each source of influence according to a random weight w_ϕ.

PSO algorithms evolve the swarm according to a *social influence* (i.e., their neighborhood previous search experience) and a *cognitive influence* (i.e., their own previous search experience). For instance, with a canonical PSO algorithm, Eq. (7) becomes

$$\begin{aligned}\mathbf{h}(\tau+1) = {} & \mathbf{h}(\tau) + w_0(\mathbf{h}(\tau) - \mathbf{h}(\tau-1)) + r_1 w_1(\mathbf{h}_{\text{social influence}} \\ & - \mathbf{h}(\tau)) + r_2 w_2(\mathbf{h}_{\text{cognitive influence}} - \mathbf{h}(\tau)). \end{aligned} \quad (7)$$

Although originally developed for static optimization problems, PSO formalism has been adapted to suit the nature of the optimization problem at hand. For instance, it 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 occurs in the optimization environment. The latest particle swarm optimization algorithms developed to insure diversity in the swarm are presented in [16,36,40,44]. Change detection and memory adjustment mechanisms for DPSO are presented in [5,8,28,60].

PSO algorithms have also been adapted for MOO in three ways by (1) defining social and cognitive influences according to a fitness function based on the notion of Pareto-dominance (see Fig. 4a), (2) storing non-dominated solutions in an archive, and (3) managing phenotype diversity in the objective space. A review of multi-objective particle swarm optimization (MOPSO) algorithms is given in [50]. Most of these approaches use a global best topology and focus on moving particles according to the Pareto front rather than local optima in the search space. Under the hypothesis that many solutions will be stored in an archive, they also use classic archive that considers only global Pareto-optimality.

To generate a pool of classifiers, ADNPSO uses the same approach as mono-objective optimization algorithms, and defines influences in the different search spaces, with the objective functions. This is achieved by reformulating the general PSO definition (Eq. (7)) according to two objectives: error rate ($f_e(\mathbf{h}, t)$) and network F_2 size ($f_s(\mathbf{h}, t)$) of each FAM network. Each particle will then move according to a cognitive and social influence for both objectives (see Fig. 6), formally defined by:

$$\begin{aligned}\mathbf{h}(\tau+1) = {} & \mathbf{h}(\tau) + w_0(\mathbf{h}(\tau) - \mathbf{h}(\tau-1)) \\ & + r_1 w_1(\mathbf{h}_{\text{social influence, error rate}} - \mathbf{h}(\tau)) \\ & + r_2 w_2(\mathbf{h}_{\text{cognitive influence, error rate}} - \mathbf{h}(\tau)) \\ & + r_3 w_3(\mathbf{h}_{\text{social influence, network size}} - \mathbf{h}(\tau)) \\ & + r_4 w_4(\mathbf{h}_{\text{cognitive influence, network size}} - \mathbf{h}(\tau)), \end{aligned} \quad (8)$$

As previously showed in Fig. 5, the rational behind this approach is that when several local optima are present in different search spaces, non-dominated solutions tend to be located in regions between local optima of the different objectives. By adjusting the weights w_ϕ, a swarm may be biased according to one objective or

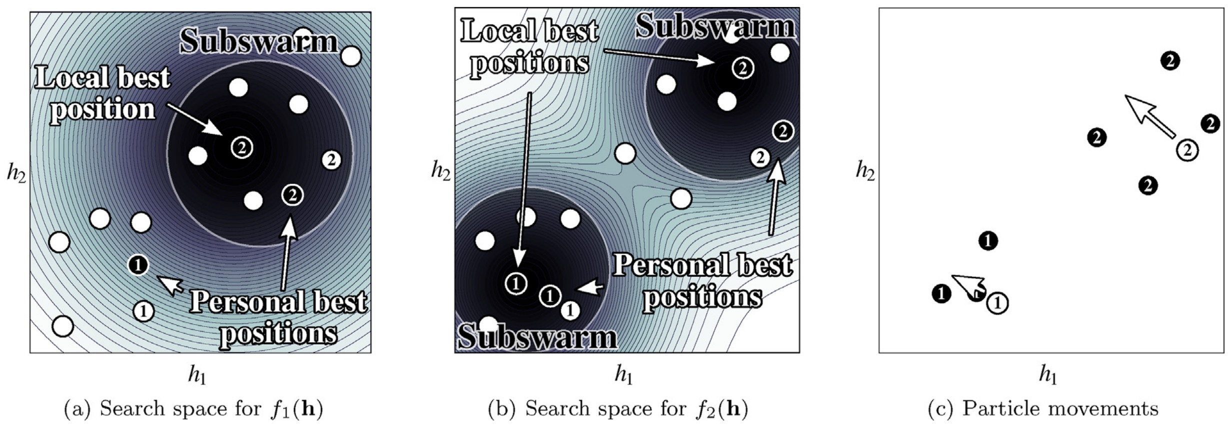

Fig. 6. An illustration of influences in the search spaces and resulting movements. Given the same objective functions used in Fig. 5, two particles in a swarm (white circles), and their social and cognitive influences (black circles), let subswarms have a maximal size of 5 particles. Both particles 1 and 2 have cognitive influences in both search spaces, yet particle 1 is not part of any subswarm for $f_1(\mathbf{h})$. Unlike particle 2, it has no social influence for this objective and ADNPSO sets $w_1 = 0$ when computing its movement with Eq. (8).

even divided in three subpopulations: (1) one that specializes in accuracy (w_1 and $w_2 > w_3$ and w_4), (2) one that specializes in complexity (w_1 and $w_2 < w_3$ and w_4), and (3) a generalist subpopulation that put both objectives on equal footing ($w_1 = w_2 = w_3 = w_4$).

Social influences of both objectives are managed by creating subswarms that adapt the DNPSO local neighborhood topology [39] to multiple objectives. While DNPSO creates subswarms dynamically around the *current position* of local best particles (i.e., particles with a personal best position that has the best fitness in their neighborhood), ADNPSO uses the memory of these local best particles. Social influences are then *personal best position* of local best particles computed independently for both objectives. As shown in Fig. 6, by limiting the size of each subswarm, particles can be excluded of these subswarms for none, one, or both objectives. For the objective that was excluded, a particle is said to be “free” and its social influence is removed by setting the weights $w_1 = 0$ and/or $w_3 = 0$ when computing Eq. (8) (depending for which objective(s) the particle is “free”). This way, a poor compromise can be avoided, and conflicting influences can then be managed simply by limiting the maximal size of each subswarm.

The DNPSO local neighborhood topology offers many ways to insure particle diversity in the search space [40]. It is also adapted to also maintain cognitive (i.e., personal best) diversity among particles within each subswarm. The ADNPSO algorithm defines a distance Δ around local best positions of each objective. Every time a particle moves with the distance Δ from the detected local optima of one objective, the personal best value of that particle is erased (i.e., “loses its memory”). It then moves only according to the other objectives by setting designated weights to 0. Since MOO problems generally have conflicting objectives, this results in particles that move away from the local optima when they are within the distance Δ from each other.

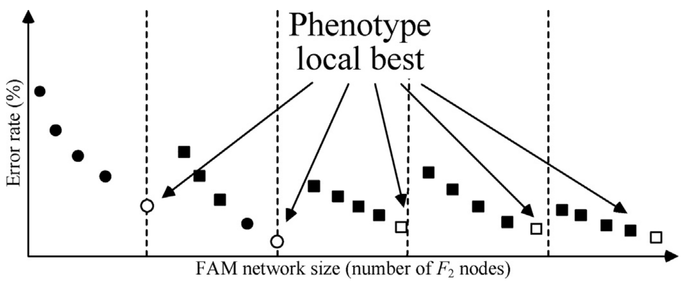

Fig. 7. Illustration of the specialized archive of solutions in the objective space. The FAM network size objective is segmented in different domains (or slices of complexity), where both Pareto-optimal (circles) and locally Pareto-optimal (squares) solutions are kept in the archive. The local best are defined as the most accurate network of each size domain.

The computational cost of the ADNPSO algorithm depends on the time needed to create subswarms and to manage particle positions during the optimization process. For each particle, it must (1) sort the rest of the swarm according to a distance metric to define the neighborhood and (2) sort the neighbors by fitness to find the local best. In contrast, as it is explained in Section 4.1, each time fitness is estimated, a FAM neural network must be trained and tested. Since, managing the swarm with the ADNPSO algorithm requires a computational cost significantly lower than that necessary to update the fitness of the swarm, it should not be considered when applied to the AMCS.

4.3. *Specialized archive and ensemble selection*

A specialized archive is introduced in the AMCS framework to store a pool of classifier such that phenotype diversity in the objective space is maintained according to FAM network size, and as framework for ensemble selection. In the ADNPSO incremental learning strategy (Section 4.1), the archive regroups FAM networks associated with each solution found in the search space according to their structural complexity (i.e., number of F_2 nodes) and stores them in to create a pool of classifiers among which ensembles can be selected.

Since the AMCS is applied to an ill-defined pattern recognition problem, where a limited amount of reference data is used for system design, both objectives are discrete functions, and the error rate is prone to over fitting. In this context, the specialized archive is used to (1) insure phenotype diversity in the objective space according to FAM network size and (2) as framework for ensemble selection. As shown in Fig. 7, the archive categorizes FAM networks associated with each solution found in the search space according to their F_2 layer size and stores them to create a pool of classifiers among which ensembles can be selected. Although, this imply keeping dominated solutions in the objective space for a MOO formulation, using a specialized archive ensures storing classifiers with a wide phenotype diversity in structural complexity.

When a new block D_t of reference samples becomes available at a time t, the swarm evolves in the search spaces and the performance of each network is re-evaluated using a mixture of new data and old data (in the LTM). However, since no further training occurs once they are stored in the archive, the size of the FAM networks in the archive never changes. Compact classifiers obtained in earlier blocks may remain in the archive over time as new data is learned incrementally.

With the ADNPSO algorithm, a genotype local best topology is used to define neighborhoods and zones of influence for the different particles in the search space. The same principle is applied in the objective space for ensemble selection. The most accurate FAMs of each network size domain are considered as phenotype local best solutions. Classifiers are selected to create an initial ensemble that is completed with a second selection phase that uses a greedy search process (introduced in [14]) to increase classifier diversity by maximizing their genotype diversity. The result of the overall selection process is an ensemble with:

1. high phenotype diversity of FAM network sizes, where networks of different structural complexity are considered, even though the estimated generalization capabilities of the some networks are not necessarily the highest or Pareto-optimal and
2. diversified classifiers in both feature and decision spaces [14].

Unlike other approaches in the literature, the proposed AMCS does not consider time as a factor to add/remove a classifier from the ensemble. It uses the notions of dominance and phenotype diversity. If classifiers become obsolete in time due to a decrease in their accuracy, they will lose their dominant position and eventually be erased from the archive. On the other hand, although they remain in the archive, solutions that do not increase ensemble diversity are never selected.

5. Experimental methodology

This paper focuses on the appearance-based classification aspect of the face recognition system by replacing the classification module and biometric data base (in Fig. 1) by the proposed AMCS. The rest of the system relies on classical algorithms. As recognition is performed with an AMCS based on the FAM classifier, the response for each successive ROI is a binary code (equals to “1” for the predicted class, and “0”s for the others). For a video sequence of a given length, the predicted class label C_k is the one with the highest accumulated response obtained for each ROI (i.e., a majority vote between predictions for each individual ROI).

The rest of this section describes the procedure utilized to perform proof-of-concept experiments, including data bases, incremental learning scenarios, experimental protocol, and performance indicators.

5.1. Video data bases

The first data base was collected by the Institute of Information Technology of the Canadian National Research Council (IIT-NRC) [22]. 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 15 s are captured at a rate of 20 frames/s with an Intel web cam 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, the same setup, almost the same background, and each face occupies between 1/4 and 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 varies from class to class, ranging from 40 to 190 for one video sequence.

The second video data base is called Motion of Body (MoBo), and was collected at Carnegie Mellon University under the HumanID project [26]. Each video sequence shows one of 25 different individuals 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 Sony DXC 9000 cameras, with a resolution of a 640×480 pixels, are positioned at different locations around the individuals. Only the video sequences with visible faces were kept: full frontal view and both sides with an angle of about 70° with the full frontal view.

Segmentation is performed using the well known Viola–Jones algorithm included in the OpenCV C/C++computer vision library. In both cases, regions of interest (ROIs) produced are converted in gray scale and normalized to 24×24 images where the eyes are aligned horizontally, with a distance of 12 pixels between them. Principal Component Analysis is then performed to reduce the number of features. For the IIT-NRC data base, the 64 features with the greatest eigenvalues are extracted and vectorized into $\mathbf{a} = \{a_1, a_2, \ldots, a_{64}\}$, where each feature $a_i \in [0, 1]$ is normalized using the min–max technique. Learning is done with ROIs extracted from the first series of video sequences (1527 ROIs for all individuals) while testing is done with ROIs extracted from the second series of video sequences (1585 ROIs for all individuals). The ROIs obtained with the MoBo data base were processed with Local Binary Pattern and Principal Component Analysis to produce 32 feature vectors, also normalized using the min–max technique. ROIs from sequences for each type of walk and view are divided in two; the first half is used for learning and the second half, for testing. This yields a total of 36,374 learning patterns and 36,227 test patterns. In both cases, the number of features was fixed after error convergence with a 1NN classifier trained on the learning data bases and tested on the test data base. Moreover, to insure that no false positives are present during training and testing, the ROIs have then been manually filtered.

5.2. Incremental learning scenarios

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

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

5.2.2. 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 of an individual are captured in a video sequence, and then learned by the system to refine its internal models. To assess AMCS 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 $2 \le t \le K+1$, 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.

5.3. Experimental protocol

The performance of the proposed DPSO learning strategy is evaluated and compared with various techniques to generate and select

Table 1
Parameters for ADNPSO.

Parameter	Value
Swarm's size N	60
Weights $\{w_0, w_\phi\}$	$\{0.73, 2.9\}$
Maximal number of subswarms	∞
Maximal size of each subswarm	4
Neighborhood size	5
Minimal distance between two local best particles	0.1
Minimal velocities of free particles	0.0001

classifiers during supervised incremental learning of data blocks D_t. The DPSO parameters used for both experiments are shown in Table 1. Weight values $\{w_0, w_\phi\}$ were defined as proposed in [32], and to detect a maximal number of local optima, no constraints were considered regarding the number, the maximal size of each subswarm is set at 4. Since Euclidean distances between particles are measured with the DPSO algorithm, the swarm evolves in a *normalized* $\mathbb{R}^4$ space to avoid any bias due to the domain of each hyperparameter. Before being applied to FAM, particle positions are denormalized to fit the hyperparameter domain. For each new blocks of data D_t, the ADNPSO optimization process is set to either stop after 10 iterations without improving the performance of either generalization error rate of network size, or after maximum 100 iterations.

Learning is performed over ten trials using tenfold cross-validation with the LTM used as specified in [13]. The proportion of D_t assigned to the LTM, and the maximal number of patterns for each class present in the LTM, are respectively set to $\lambda_D = 1/6$ and $|C_k|_{LTM} = 20$. 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 the LTM to estimate the fitness of each particle during the DPSO algorithm (D_t^f). Between successive training epochs, the presentation order of training patterns is changed randomly. Within each trial, five different replications are performed using different class presentation orders, for a total of fifty replications.

The simulations evaluate the performance achieved in both scenarios for incremental learning of new data blocks D_t, where AMCSs employ the DPSO learning algorithm and selection ensemble discussed in Section 4 – ADNPSO ← the networks in the specialized archive corresponding to the phenotype local best plus a greedy search that maximizes genotype diversity [14]. This system is compared to AMCSs using the DPSO learning strategy used with different optimization algorithms and ensemble selection techniques, in particular:

1. DNPSO ← the ensemble of FAM networks associated to the local best positions found with the mono-objective DNPSO algorithm plus a greedy search that maximizes genotype diversity, also within the swarm [14],
2. MOPSO ← the entire archive obtained with the DPSO incremental learning strategy employed with a multi-objective PSO algorithm that uses the notion of dominance to guide particles toward the Pareto optimal front [11], and
3. GBEST ← the FAM network corresponding to the DNPSO global best solution.

For references, the performance is also given for the batch learning methods:

4. PSO_B ← an AMCS that uses the entire swarm of FAMs trained with a canonical PSO batch learning strategy [23], and
5. *k*NN ← a single *k*NN classifier.

The MOPSO algorithm was used with the same applicable parameters than with the proposed ADNPSO, and with a grid size of 10 (for further details, see [11]). Moreover, at a given time t, batch learning consists of initializing the system, and learning all the data blocks D_t accumulated thus, $B_t = D_1 \cup \cdots \cup D_t$ [23]. In the context of a face recognition application, using Principal Component Analysis for feature extraction and selection, and *k*NN for classification is equivalent to the well known Eigenfaces method [56].

5.4. Performance evaluation

The average performance of AMCSs is assessed in terms of generalization error rate achieved with video-sequences, and resource requirements. The *generalization error rate for a single ROI* is the ratio of incorrect predictions over all test set predictions, where each ROI is tested independently. Classification decisions produced for a single ROI are considered to be the most conservative performance metric, and are used for fitness estimation during Algorithm 1.

For the video-based face recognition application, *generalization error rate for video sequences* (over two or more ROIs), is the result of the fusion between the tracking and classification module (see Section 2). Given video sequences, it is the ratio of incorrect predictions over all predictions obtained with a majority vote among all class accumulated binary responses from the AMCS over a fixed number of regions of interest (ROIs). For unbalanced data bases (i.e., video sequences of different length), classification rate for a number of frames exceeding the length of shorter sequences is computed with predictions obtained with all ROIs of the latter.

The identification capabilities of the AMCS are also evaluated with cumulative match curves moon01. These curves estimate the ranking capabilities of a classification system during an identification application by verifying if the correct prediction is within the best ranks.

Resource requirement of AMCSs that employ the DPSO incremental learning strategy is measured in terms of *compression*, that is, the average number of training patterns, contained in all D_t^t presented to the AMCS, per category prototype in the network. For a single FAM network, compression refers to the average number of training patterns per neuron in the F_2 layer, and for ensembles, it is the total number of F_2 layer nodes for all FAM networks in the ensemble. The higher the compression, the better. Since learning with *k*NN consists of memorizing the training data set D_t^t, compression with that network is always one.

6. Results and discussion

The objective of the AMCS and ADNPSO incremental learning strategy is to provide a face recognition system a mean to perform accurate predictions in real time. To illustrate this, this section first compares the accuracy and structural complexity of ensembles evolved using the DPSO learning strategy described in Section 4 (ADNPSO) with other AMCSs that perform incremental learning and batch references methods.

To give more insight on the effect of the different optimization methods on pools of classifier generation, Section 6.2 presents the evolution of the swarm and archive in the objective space during the update learning scenario. The resulting swarm and specialized archive obtained with ADNPSO are compared with those obtained when incremental learning is guided mono-objective optimization (DNPSO), and classic MOO (MOPSO).

6.1. Performance during video-based face recognition

Figs. 8 and 9 present the video-based generalization error rate according to the number of ROI used to perform recognition at

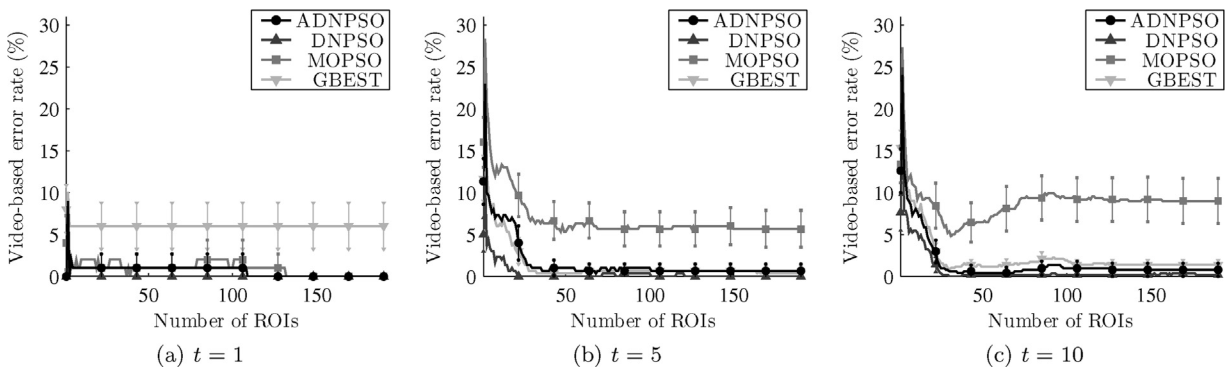

Fig. 8. Evolution of the video-based error rate versus the number of ROIs used to identify individuals of the IIT-NRC data base during the *enrollment* incremental learning scenario. Performance is shown at different points in time and error bars correspond to the 90% confidence interval.

different points in time for both incremental learning scenarios. With each ROI, evidence in the form of FAM network outputs (binary codes) is accumulated and used to establish a ranking through majority voting. When classes are enrolled incrementally, class decision boundaries become more complex in time. Accuracy obtained with few ROIs then decreases, while the number of ROIs necessary to achieve a video-based error comparable to 0% increases. On the other hand, the video-based error rate obtained after updating classes through incremental learning decreases over time, as new blocks of data become available. Moreover, when the AMCS has knowledge of the whole classification problem when adding new data, ensembles obtained are more robust as it can eventually achieve perfect accuracy.

When recognition is performed by accumulating responses of AMCSs over few ROIs (15) during incremental learning, the cumulative match curves in Figs. 10 and 11 are used to assess their ranking capabilities. Ideally, with perfect accuracy, the correct class is always ranked first. Ambiguity regarding predictions acts according to the error rate. For the enrollment learning scenario, ranking capability diminishes in time when the classification environment becomes complex, and during update, it increases with the knowledge regarding individual class distributions.

However, when the number of ROIs used to perform recognition increases, the video-based error rate tends to increase at the end of each sequence. When both learning and test sequences of the IIT-NRC data base were recorded, the individuals were all initially facing the camera, giving a full frontal image of their face. As they start moving, changing his facial orientation and expression, different facial views, corresponding to data points in unexplored regions of the feature space, are presented to the system. Recognizing an individual toward the end of a video sequence is thus more difficult. Until all classes are updated, correct predictions for each ROI accumulated at the beginning of the test sequences are surpassed by the wrong predictions accumulated with the subsequent ROIs.

The accuracy of different methods is compared in Table 2 with the number ROIs needed to reach an error rate comparable to 0% and the corresponding error rate. The higher level of accuracy achieved by proposed AMCSs and a real time estimation on the speed at which this performance is attained. An AMCS driven by ADNPSO can achieve a video-based error rate comparable to 0% within a time frame similar to that obtained with mono-objective DNPSO incremental learning strategy and batch learning approaches. In the worst case (the update scenario), the proposed method needs 10 additional ROIs than when mono-objective optimization is used with either incremental (DNPSO) or batch learning (PSO_B) to reach an error rate comparable to 0%. Assuming ideal tracking performances and a camera that acquires video sequences at a rate of 30 frames/s, this represents around a third of a second. On the other hand, using MOO or the single global best solution during mono-objective optimization gives less robust solutions and, in those cases, the AMCS's error rate is never comparable to 0%. Results are similar with the MoBo data base, except that AMCSs with the proposed DPSO-based strategy require fewer ROIs to achieve an error rate similar to 0% and that a single FAM network can also achieve this level of accuracy.

Table 3 presents a comparison of accuracy obtained with other video-based face recognition systems in literature that perform *batch learning* on both IIT-NRC and MoBo data bases. With the exception of [2] with the IIT-NRC data base and [65] with the MoBo data base, the AMCS with the proposed ADNPSO learning strategy outperforms all other systems. Regardless of the scenario, the AMCS with ADNPSO must accumulate about 1 s of video stream to achieve an error rate of 0% after incremental learning of the entire MoBo data base. In comparison, after performing *batch learning* of

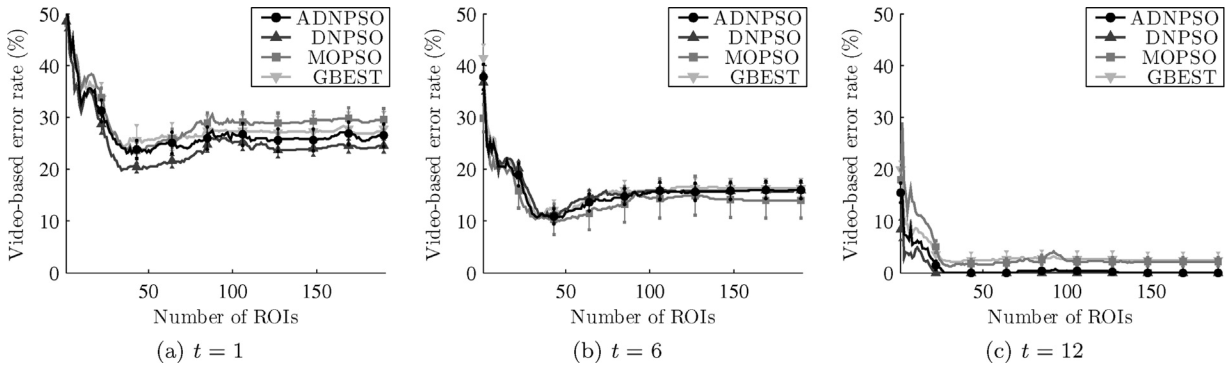

Fig. 9. Evolution of the video-based error rate versus the number of ROIs used to identify individuals of the IIT-NRC data base during the *update* incremental learning scenario. Performance is shown at different points in time and error bars correspond to the 90% confidence interval.

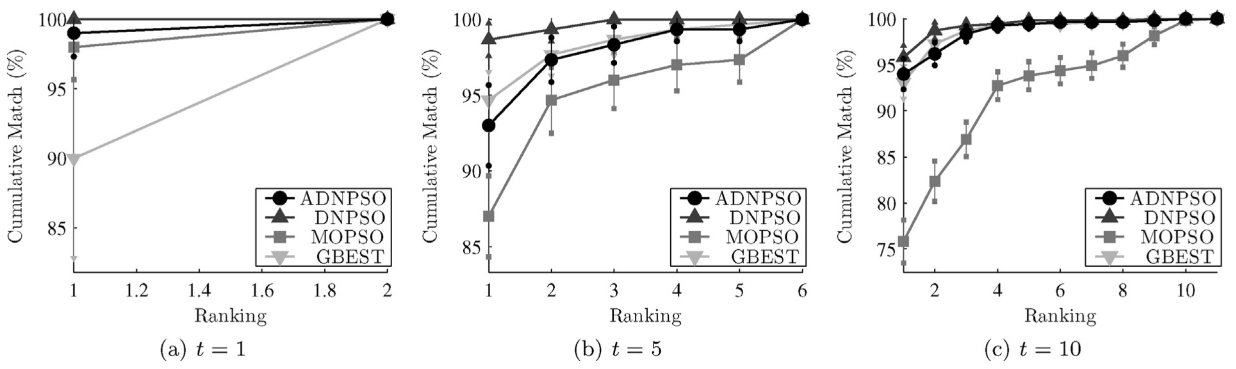

Fig. 10. Cumulative match curves obtained during the *enrollment* incremental learning scenario at different points in time when 15 ROIs are used to perform recognition. Performance is shown at different points in time and error bars correspond to the 90% confidence interval. During enrollment, the maximal rank increases with the number of classes present in the system.

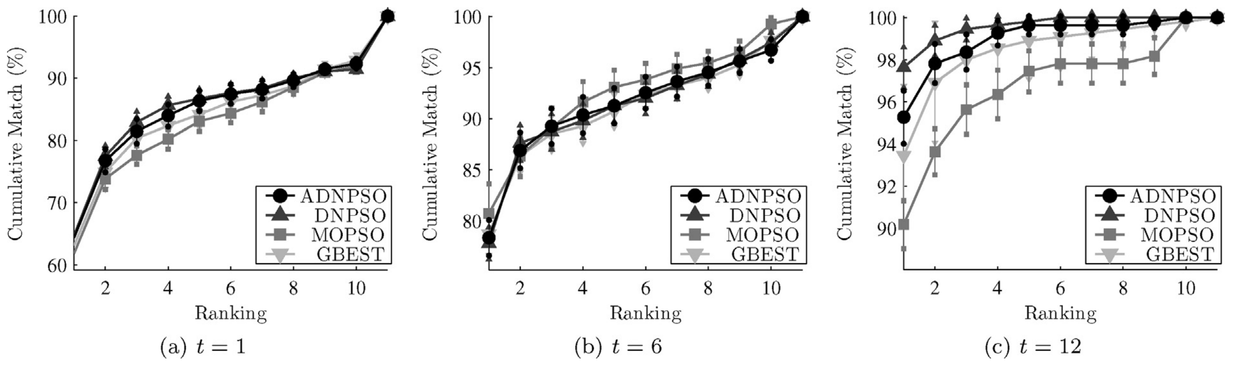

Fig. 11. Cumulative match curves obtained during the *update* incremental learning scenario at different points in time when 15 ROIs are used to perform recognition. Performance is shown at different points in time and error bars correspond to the 90% confidence interval. During enrollment, the maximal rank increases with the number of classes present in the system.

Table 2
Minimal average error rate and number of ROIs necessary to achieve a generalization error rate *comparable* to 0% for video-based face recognition. Results shown are obtained after learning the entire IIT-NRC and MoBo data bases through the both learning scenarios. The mention "never" indicates that the method never achieves an error rate comparable to 0%.

Type of learning	Incremental				Batch	
Method	ADNPSO	DNPSO	MOPSO	GBEST	PSO$_B$	*k*NN
IIT-NRC data base						
Enrollment learning scenario						
Minimal av. error rate	0.6 ± 0.7	0 ± 0	5 ± 3	2.1 ± 1	0 ± 0	0 ± 0
Nb. of ROIs to reach 0%	27	22	Never	Never	20	23
Update learning scenario						
Minimal av. error rate	0 ± 0	0 ± 0	1.2 ± 0.6	0.8 ± 0.5	0 ± 0	0 ± 0
Nb. of ROIs to reach 0%	31	20	Never	Never	20	23
MoBo data base						
Enrollment learning scenario						
Minimal av. error rate	0 ± 0	0 ± 0	0.5 ± 0.2	1.2 ± 1.4	0 ± 0	0 ± 0
Nb. of ROIs to reach 0%	30	28	Never	25	30	16
Update learning scenario						
Minimal av. error rate	0 ± 0	0 ± 0	3 ± 1	0.3 ± 0.3	0 ± 0	0 ± 0
Nb. of ROIs to reach 0%	27	24	never	25	30	16

Table 3
Comparison of the DPSO-based learning strategy with other authors on the IIT-NRC and MoBo data bases. Classification rates were obtained for recognition on video sequences.

IIT-NRC data base

Proposed syst.	Arandjelovic et al. (2005)	Gorodnichy (2005)	Tangelder et al. (2006)	Wang et al. (2009)
100%	100%	95%	95%	93%

MoBo data base

Proposed syst.	Cevikalp et al. (2010)	Hadid et al. (2004)	Liu et al. (2003)	Wang et al. (2008)	Zhou et al. (2003)
100%	98%	94%	99%	94%	100%

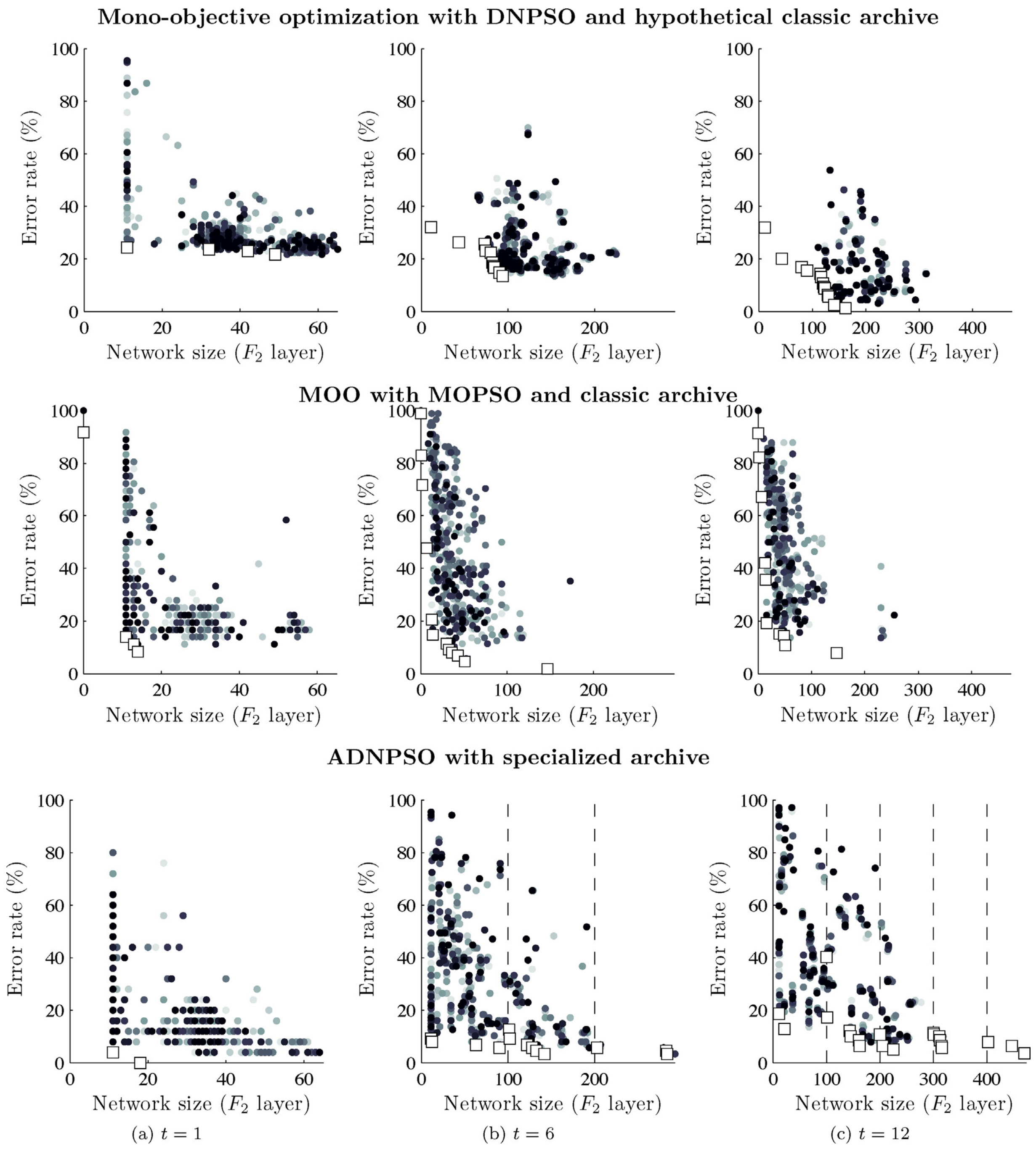

Fig. 12. Objective space during the update incremental learning scenario. Circles show evolution of the swarm during its evolution at a time t, and squares illustrate solutions stored (or would be stored for mono-objective optimization) in the archive. Light and dark circles respectively indicate the position of each particle at the start and end of the optimization process.

the MoBo data base, [65] achieved the same result by accumulating classifier responses for 0.5 s. While [2] also obtained a 0% video-based error rate, the number of accumulated response, to achieve this is not available.

Previous result showed that an AMCS driven by ADNPSO can achieve generalization capabilities comparable to 100% with few ROIs. Table 4 depicts the structural complexity indicators only for ensembles that yielded error rates comparable to 0%. On this aspect, ADNPSO resulted in ensembles with less base classifiers, where the average structural complexity (compression) of each member is lower (higher), and thus less overall ensemble complexity. Not only are ensembles smaller, but also the average ensemble member obtained with ADNPSO uses only a fraction of the classifiers present in the archive. Given the limited number of training samples available in the worst case (the update scenario), designing ensembles with all reference samples in the IIT-NRC and MoBo data bases yield ensembles composed of an average of $J = 870$ and $J = 5700$ F_2 layer nodes, respectively. Each time a query sample is presented to the face recognition system, the FAM choice function (Eq. (1)) is evaluated for each F_2 layer node. For each query sample, FAM predictions have a time complexity of $O(IJ)$, where the number of input features has been fixed here to $I = 64$. For each camera with a frame rate of 30 frames/s in a moderately cluttered scene (ten people maximum), the system will process a worst case of 300 ROIs/s.

Table 4
Structural complexity indicators of AMCSs that always give error rates comparable to 0%. Results are given after incremental learning of both data bases and learning scenarios. Complexity is evaluated in terms of ensemble size, average network compression, and total compression of the entire ensemble. The arrows serve as reminders that lower ensemble sizes and higher compressions indicate better results. Each cell is presented with the 90% confidence interval, and the best values are given in bold.

Type of learning	Incremental		Batch	
Method	ADNPSO	DNPSO	PSO_B	*k*NN
IIT-NRC data base				
Enrollment learning scenario				
Ensemble size (↓)	4.5 ± 0.4	19.4 ± 0.7	60 ± 0	**1 ± 0**
Average comp. (↑)	**9.3 ± 0.7**	6.7 ± 0.3	2.2 ± 0.2	1 ± 0
Total comp. (↑)	**2.1 ± 0.2**	0.34 ± 0.02	0.037 ± 0.003	1 ± 0
Update learning scenario				
Ensemble size (↓)	5.5 ± 0.4	19.5 ± 0.7	60 ± 0	**1 ± 0**
Average comp. (↑)	**7.4 ± 0.4**	5.8 ± 0.2	2.2 ± 0.2	1 ± 0
Total comp. (↑)	**1.4 ± 0.2**	0.30 ± 0.03	0.037 ± 0.003	1 ± 0
MoBo data base				
Enrollment learning scenario				
Ensemble size (↓)	7.5 ± 0.5	23.3 ± 0.7	60 ± 0	**1 ± 0**
Average comp. (↑)	**50 ± 6**	23 ± 2	3.6 ± 0.1	1 ± 0
Total comp. (↑)	**6.7 ± 0.7**	1.0 ± 0.1	0.060 ± 0.004	1 ± 0
Update learning scenario				
Ensemble size (↓)	5.5 ± 0.8	19.4 ± 0.8	60 ± 0	**1 ± 0**
Average comp. (↑)	**28 ± 1**	18.6 ± 0.8	3.6 ± 0.1	1 ± 0
Total comp. (↑)	**5.1 ± 0.5**	0.9 ± 0.1	0.060 ± 0.004	1 ± 0

By today's standard this can be easily accomplished in 1/300 s on a standard desktop computer.[2] Compared to the batch method learning methods, Table 4 shows that incremental learning performed with PSO-based learning strategies provides simpler models that batch learning. However, only the proposed ADNPSO gives comparable accuracy *and* higher compression than *k*NN after performing incremental learning on both data bases and during both scenarios. Although ADNPSO and *k*NN compressions are on the same scale, it should be noted that using the latter with J recognition category (i.e., training samples) implies computing the Euclidean distance for each J category and ranks the solutions to find the best k, a time complexity of $O(kIJ\log(J))$.

6.2. Swarm and archive evolution during optimization

To give an example on how pools of classifier are generated and ensembles are selected, Fig. 12 gives an example of the swarm's evolution in the objective space during the update incremental learning scenario. It compares mono-objective optimization (DNPSO), formal MOO (MOPSO), and the proposed ADNPSO scheme for the replication that yielded error rate for single ROIs closest to the average.

During mono-objective optimization with the DNPSO algorithm, networks in the swarm evolve according only to accuracy. When learning data from complex classification environments, FAM networks then tend to continuously grow its F_2 layer to maintain or increase its accuracy. The swarm then tends to move downward in the objective space, while neglecting the search for potential lighter solutions that could also provide networks accurate enough to be included in an ensemble.

If influences are defined in the objective space with the MOSPO algorithm, Fig. 12 shows that using classifiers such as FAM introduces a bias in the swarm's movements toward structural complexity. While the MOPSO algorithm theoretically considers both objectives equally, when used to evolve FAM networks, Fig. 12 shows that it is easier to find non-dominated solutions with smaller F_2 layer sizes than with lower error rate. In time, the MOPSO algorithm directs most particles in the different search spaces such as mostly minimizing FAM F_2 layer size, thus limiting the search capabilities for accurate solutions.

On the other hand, the proposed ADNPSO directs subswarms of particles according to either accuracy, network size, or both at the same time (see Fig. 6). Although this creates a swarm of particles that could successfully fill a classical archive, the specialized archive insures that the most accurate solutions of different network sizes are stored. As results presented earlier showed, this creates suitable pools and ensembles of FAM neural networks. Still, if several classes are added in time and the classification environment becomes more complex, it could become necessary to redefine the specialized archive's boundaries to accommodate such changes.

7. Conclusion

In this paper, an ADNPSO incremental learning strategy is proposed to evolve heterogeneous ensembles of classifiers in response to new reference data during video face recognition. This strategy is applied to an AMCS where all parameters of a swarm of FAM neural network classifiers (i.e., a swarm of classifiers), each one corresponding to a particle, are co-optimized such that both error rate and network size are minimized. To provide a high level of accuracy over time while minimizing the computational complexity, the AMCS integrates information from multiple diverse classifiers, where learning is guided by an aggregated dynamical niching PSO (ADNPSO) algorithm that optimizes networks according to both these objectives. By using the specialized archive, local Pareto-optimal solutions detected by the ADNPSO algorithm can also be stored and combined with a greedy search algorithm to create ensembles based on accuracy, phenotype and genotype diversity.

Overall results indicate that using information in the search space of each objective (local optima positions and values), rather than in the objective space, permits creating pools of classifiers that are more accurate and with lower computational cost. This results in ensembles that give an accuracy comparable to that obtained with mono-objective optimization and batch learning methods. However, this is achieved with only a fraction of the computational cost (between 16% and 20% depending on the data base and learning scenario used).

However, the proposed AMCS is designed to observe small amounts of learning data under several perspectives with a swarm of classifiers, so that it can perform in the context of a real video-based face recognition application. Although it can be performed

[2] This statement is valid assuming a moderate number of individuals populating a scene and cameras feeding the face recognition system.

off-line, while predictions can afterward be performed on-line, the learning process can become long when applied if data acquisition conditions are more constrained and data is available in large amounts (such as with the MoBo data base). To circumvent this problem, future work should then consider focusing on characterizing reference learning samples with different quality measures. To disambiguate concepts and further reduce FAM network structural complexity, available data could then be filtered according their level of quality so that learning is performed only with suitable samples. In this context, the utility of the LTM used in the AMCS could also be redefined. Rather than using the LTM only for validation purposes, it could also use these quality measures to select reference samples and keep a representative snapshot of the data distributions at a time *t*. This way networks in the swarm could be reinitialized if they bring no new knowledge during the learning process.

Acknowledgment

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

References

[1] H. Abdulsalam, D. Skillicorn, P. Martin, Classification using streaming random forests, IEEE Transactions on Knowledge and Data Engineering 23 (1) (2011) 22–36.

[2] O. Arandjelovic, R. Cipolla, A methodology for rapid illumination-invariant face recognition using image processing filters, Computer Vision and Image Understanding 113 (2) (2009) 159–171.

[3] M. Barry, E. Granger, Comparison of ARTMAP neural networks for classification for face recognition from video, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Orlando, USA, August, 2007, pp. 2256–2261.

[4] A. Bifet, E. Frank, G. Holmes, B. Pfahringer, Accurate ensembles for data streams: combining restricted hoeffding trees using stacking, in: Proceedings of the Asian Conference on Machine Learning, Tokyo, Japan, 2010, pp. 225–240.

[5] T. Blackwell, J. Branke, Multi-swarm optimization in dynamic environments, in: Proceedings of the Conference Applications of Evolutionary Computing, Coimbra, Portugal, April, 2004, pp. 489–500.

[6] A. Blum, Empirical support for winnow and weighted-majority algorithms: results on a calendar scheduling domain, Machine Learning 26 (1) (1997) 5–23.

[7] L. Breiman, Pasting small votes for classification in large databases and on-line, Machine Learning 36 (1) (1999) 85–103.

[8] A. Carlisle, G. Dozier, Tracking changing extrema with adaptive particle swarm optimizer, in: Proceedings of the World Automation Congress, Orlando, FL, USA, June, 2002, pp. 265–270.

[9] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, IEEE Transactions on Systems, Man, and Cybernetics C 3 (5) (1992) 698–713.

[10] L.-F. Chen, H.-Y. Liao, J.-C. Lin, Person identification using facial motion, in: Proceedings on Image Processing, vol. 2, 2001, pp. 677–680.

[11] C. Coello, G. Pulido, M. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 256–279.

[12] J.-F. Connolly, E. Granger, R. Sabourin, An adaptive ensemble of fuzzy ARTMAP neural networks for video-based face classification, in: Proceedings of the IEEE Congress on Evolutionary Computation, July, 2010, pp. 1–8.

[13] J.-F. Connolly, E. Granger, R. Sabourin, An adaptive classification system for video-based face recognition, Information Sciences 192 (1) (2012) 50–70.

[14] J.-F. Connolly, E. Granger, R. Sabourin, Evolution of heterogeneous ensembles through dynamic particle swarm optimization for video-based face recognition, Pattern Recognition 45 (7) (2012) 2460–2477.

[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA. ii. Evolutionary Computation, IEEE Transactions on 6 (2) (2002) 182–197.

[16] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Science 178 (15) (2008) 3096–3109.

[17] G. Edwards, C. Taylor, T. Cootes, Improving identification performance by integrating evidence from sequences, in: IEEE Proceedings on Computer Vision and Pattern Recognition, 1999, pp. 486–491.

[18] H. Ekenel, L. Szasz-Toth, R. Stiefelhagen, Open-set face recognition-based visitor interface system, in: M. Fritz, B. Schiele, J. Piater (Eds.), Computer Vision Systems, Volume 5815 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2009, pp. 43–52.

[19] A.P. Engelbrecht, Fundamental of Computational Swarm Intelligence, John Wiley & Sons, Chichester, UK, 2005 (Chapter 7.2).

[20] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Advances in Artificial Intelligence, Springer-Verlag, New York, 1999, pp. 286–295.

[21] V. Ganti, J. Gehrke, R. Ramakrishnan, Mining data streams under block evolution, ACM SIGKDD Explorations Newsletter 3 (1) (2002) 1–10.

[22] D.O. Gorodnichy, Video-based framework for face recognition in video, in: Second Workshop on Face Processing in Video in Proceedings of the Conference on Computer and Robot Vision, Victoria, Canada, May, 2005, pp. 325–344.

[23] 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.

[24] E. Granger, D. Prieur, J.-F. Connolly, Evolving ARTMAP neural networks using multi-objective particle swarm optimization, in: 2010 IEEE Congress on Evolutionary Computation (CEC), july 2010, vol. 1, pp. 1–8.

[25] 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.

[26] R. Gross, J. Shi, The CMU motion of body (MoBo) database, 2001, Technical Report CMU-RI-TR-01-18, Carnegie Mellon University, 2001.

[27] S. Grossberg, Nonlinear neural networks: principles, mechanisms, and architectures, IEEE Transactions on Neural Networks 1 (1) (1988) 17–61.

[28] X. Hu, R.C. Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, Honolulu, USA, May, 2002, pp. 1666–1670.

[29] 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.

[30] X. Jiang, W. Ser, Online fingerprint template improvement, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (8) (2002) 1121–1126.

[31] M. Kapp, R. Sabourin, P. Maupin, Adaptive incremental learning with an ensemble of support vector machines, in: International Conference on Pattern Recognition, Istanbul, Turkey, August, 2010, pp. 4048–4051.

[32] J. Kennedy, Some issues and practices for particle swarms, in: Proceedings of the IEEE International on Swarm Intelligence, Honolulu, USA, April, 2007, pp. 162–169.

[33] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, Journal of Machine Learning Research 8 (1) (2007) 2755–2790.

[34] L.I. Kuncheva, Classifier ensembles for changing environments, in: Proceedings of the International Workshop on Multiple Classifier Systems, Cagliari, Italy, 2004, pp. 1–15.

[35] R. Li, T.R. Mersch, O.X. Wen, A. Kaylani, G.C. Anagnostopoulos, Multi-objective memetic evolution of art-based classifiers, in: Proceedings of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, July, 2010, pp. 1–8.

[36] X. Li, J. Branke, T. Blackwell, Particle swarm with speciation and adaptation in a dynamic environment, in: Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, USA, July, 2006, pp. 51–58.

[37] 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.

[38] L.L. Minku, A.P. White, X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Transactions on Knowledge and Data Engineering 22 (5) (2010) 730–742.

[39] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, DNPSO: a dynamic niching particle swarm optimizer for multi-modal optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, Hong Kong, China, May, 2008, pp. 26–32.

[40] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, Evaluating the performance of DNPSO in dynamic environments, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Singapore, October, 2008, pp. 12–15.

[41] K. Nishida, Learning and detecting concept drift, PhD Thesis, Hokkaido University, Sapporo, Japan, 2008.

[42] D.F. Olivieira, A.M.P. Canuto, M.C.P. de Souto, Use of multi-objective genetic algorithms to investigate the diversity/accuracy dilemma in heterogeneous ensembles, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Atlanta, USA, June, 2009, pp. 1238–1245.

[43] N.C. Oza, Online ensemble learning, PhD Thesis, University of California, Berkeley, CA, 2000.

[44] E. Özcan, M. Yýlmaz, Particle swarms for multimodal optimization, in: Proceedings of the International Conference on Adaptive and Natural Computing Algorithms, Warsaw, Poland, April, 2007, pp. 366–375.

[45] N. Poh, W. Rita, J. Kittler, F. Roli, Challenges and research directions for adaptive biometric recognition systems, in: M. Tistarelli, M. Nixon (Eds.), Advances in Biometrics, Volume 5558 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2009, pp. 753–764.

[46] 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 C 31 (4) (2001) 497–508.

[47] M.A. Potter, K.A.D. Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evolutionary Computation 8 (2000) 1–29.

[48] T. Rashid, A heterogeneous ensemble network using machine learning techniques, International Journal of Computer Science an Network Security 9 (8) (2009) 335–339.

[49] A. Rattani, Adaptive biometric system based on template update procedures, PhD Thesis, University of Cagliari, Cagliari, Italy, 2010.

[50] M. Reyes-Sierra, C.A.C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, International Journal of Computational Intelligence Research 2 (3) (2006) 287–308.
[51] F. Roli, L. Didaci, G. Marcialis, Adaptive biometric systems that can improve with use, in: N.K. Ratha, V. Govindaraju (Eds.), Advances in Biometrics, Springer, London, 2008, pp. 447–471.
[52] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intelligent Data Analysis (IDA) 11 (1) (2006) 1–28 (special issue on Knowledge Discovery from Data Streams).
[53] J. Stallkamp, H. Ekenel, R. Stiefelhagen, Video-based face recognition on real-world data, in: IEEE 11th International Conference on Computer Vision 2007, ICCV 2007, vol. 1, October, 2007, pp. 1–8.
[54] W.N. Street, Y.S. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 377–382.
[55] A. Tsymbla, M. Pechenizkiy, P. Cunningham, S. Puuronen, Dynamic integration of classifiers for handling concept drift, Information Fusion 9 (1) (2008) 56–68.
[56] M.A. Turk, A.P. Pentland, Eigenfaces for recognition, Journal of Cognitive Neurosicence 1 (3) (1991) 71–86.
[57] U. Uludag, A. Ross, A. Jain, Biometric template selection and update: a case study in fingerprints, Pattern Recognition 37 (7) (2004) 1533–1542.
[58] G. Valentini, Ensemble methods based on bias-variance analysis, PhD Thesis, University of Genova, Genova, Switzerland, 2003.
[59] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept drifting data streams using ensemble classifiers, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2003) 226–235.
[60] H. Wang, D. Wang, S. Yang, Triggered memory-based swarm optimization in dynamic environments, in: Applications of Evolutionary Computing, Valencia, Spain, April, 2007, pp. 637–646.
[61] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101.
[62] Z. Xingquan, W. Xindong, Y. Ying, Dynamic classifier selection for effective mining from noisy data streams, in: IEEE Proceedings of the International Conference on Data Mining, 2004, pp. 305–312.
[63] X. Zhang, Y. Gaoa, Face recognition across pose: a review, Pattern Recognition 42 (11) (2009) 2876–2896.
[64] W.Y. Zhao, R. Chellappa, P.J. Phillips, A.P. Rosenfeld, Face recognition: a literature survey, ACM Computing Surveys 35 (4) (2003) 399–458.
[65] S. Zhou, V. Krueger, R. Chellappa, Probabilistic recognition of human faces from video, Computer Vision and Image Understanding 91 (2003) 214–245.
[66] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation, 3 (November (4)) (1999) 257–271.
[67] I. Zliobaite, Learning under concept drift: an overview, CoRR, abs/1010.4784, 2010.