Coevolutionary Feature Learning for Object Recognition

Krzysztof Krawiec* and Bir Bhanu

Center for Research in Intelligent Systems
University of California, Riverside, CA 92521-0425, USA
{kkrwiec,bhanu}@cris.ucr.edu

Abstract. In this paper, we consider the task of automatic synthesis/learning of pattern recognition systems. In particular, a method is proposed that, given exclusively training raster images, synthesizes complete feature-based recognition system. The proposed approach is general and does not require any assumptions concerning training data and application domain. Its novelty consists in procedural representation of features for recognition and utilization of coevolutionary computation for their synthesis. The paper describes the synthesis algorithm, outlines the architecture of the synthesized system, provides firm rationale for its design, and evaluates it experimentally on the real-world task of target recognition in synthetic aperture radar (SAR) imagery.

1 Introduction

Most real-world learning tasks concerning visual information processing are inherently complex. This complexity results not only from the large volume of data that one usually needs to process, but also from its spatial nature, information incompleteness, and, most of all, from the vast amount of hypotheses (concerning training data) to be considered in the learning process. Therefore, a design of recognition system that is able to learn consists in a great part in limiting its capabilities. To induce useful hypothesis on one hand and avoid overfitting to the training data on the other, the learning system must observe some assumptions concerning training data and hypothesis representation, known as inductive bias and representation bias, respectively. In visual learning, however, these biases have to be augmented by an extra ‘visual bias’, i.e., knowledge related to the visual nature of the information being subject to the learning process. A part of that is general knowledge concerning vision (background knowledge, BK), for instance, basic concepts like pixel proximity, edges, regions, primitive features, etc. However, often a more specific domain knowledge (DK) related to particular task/application (e.g., fingerprint identification, face recognition, SAR target detection, etc.) is also required (e.g., the interpretation of scattering centers in SAR imagery).

Contemporarily, most recognition methods make intense use of DK to attain competitive performance level. This is however a two-edged sword, as the more DK the method uses, the more specific it becomes and the less general and transferable is the

* On a temporary leave from Poznań University of Technology, Poznań, Poland.

© Springer-Verlag Berlin Heidelberg 2003
knowledge it acquires. The contribution of such over-specific methods to the overall body of knowledge of Machine Learning, Computer Vision, and Pattern Recognition is questionable.

Therefore, we focus on general-purpose visual learning that requires only BK. In the approach proposed here, the key characteristics of BK are (i) representation of the synthesized systems/hypotheses in the form of information processing chain that extends from the input image to the final recognition, and (ii) assumption on presence of building blocks (modules) in the chain. These suppositions, motivated by biological analogs, enable us to break down learning into components so as to cope with the complexity of recognition task. In particular, high-level building blocks like 'classifier' are specified explicitly; others, however, like 'features', emerge autonomously during the learning process. The success of our learning method depends on its ability to exploit and discover the inherent modularity of the problem at hand.

It is to be noted that the ability to identify building blocks is a necessary, but not a sufficient, precondition for successful learning/synthesis task. By analogy, arranging neurons into layers in an artificial neural network scales down the learning task by reducing the number of architectures that are considered, but does not provide explicit learning rule(s) for particular layers. To enforce learning in each identified building block, we need an evaluation function that spans over the space of all potential solutions and guides the learning process. Unfortunately, when no a priori definition of module's 'desired output' is available, this requirement is hard to meet. This is why we propose to employ cooperative coevolution [10], a variety of evolutionary algorithm, that allows breaking down the problem into subproblems without explicitly specifying objectives for each of them. This paper focuses on cooperation that takes place at feature level.

2 Related Work and Contributions

No general methodology has been developed so far that effectively automates the process of recognition system synthesis. Several methods have been reported in the literature; they include blackboard architecture, case-based reasoning, reinforcement learning, and automatic acquisition (learning) of models, to mention the most predominant. The paradigm of evolutionary computation (EC) has found applications in image processing and analysis. It has been found effective for its ability to perform effective global parallel search in high-dimensional search spaces and to resist the local optima problem. However, in most approaches the learning/adaptation is limited to parameter optimization. Relatively few results have been reported [4,8,12,13], that perform deep visual learning, i.e., with learner being able to synthesize and manipulate entire recognition system.

The major contribution of this paper is a novel method that, given exclusively training raster images, synthesizes complete feature-based recognition system. The proposed approach is general and does not require any assumptions concerning training data and application domain. Its novelty consists in (i) procedural representation of features for recognition and (ii) utilization of coevolutionary computation for their synthesis.
3 Rationale for the Synthesis Algorithm and System Design

3.1 Preliminary Assumptions

We make the following assumptions that are of methodological nature and do not affect the generality of the proposed approach.

(a) We follow the feature-based recognition paradigm and split the object recognition process into two fundamental modules: feature extraction module and decision making/recognition module. The novelty of the proposed approach consists in an extensive learning process that aims at optimizing the way the former module extracts features from an input image, prior to the learning that takes place in the recognition module.

(b) The synthesis of the recognition system adopts the learning-from-examples scheme and relies exclusively on a finite training set \( D \) of images, which is assumed to be a representative sample from the universe \( U \). In particular, we observe the supervised learning setting and assume that the training data are partitioned into finite number of decision classes \( D_i \), i.e.

\[
\bigcup_i D_i = D, \forall i \neq j \ \ D_i \cap D_j = \emptyset.
\]

(c) Inspired to some extent by the constructive induction [7] research in machine learning, we view the feature extraction process in a procedural way. In particular, we assume that a single feature extraction procedure is a chain of \( n \) primitive, possibly parameterized, operations (building blocks), which are essentially calls to functions from predefined fixed library/collection of \( m \) such functions. A feature extraction procedure accepts an image as input and yields single scalar value as the result. A set of one or more feature extraction procedures forms a feature vector/set \( S \) (representation), and is essentially equivalent with the feature extraction module introduced in (a).

3.2 Learning as Optimization Process - Complexity Issues

A well-designed representation/feature set is clearly a necessity for high recognition rate, and that is why its design is usually so demanding and resource-consuming. To automate that process and include it into the learning loop, we formulate it as a search (optimization) problem in the discrete space of all representations \( \Omega \), with each search state corresponding to a unique set of feature extraction procedures \( S \in \Omega \). We assume also that an evaluation function \( f: \Omega \times U \rightarrow \mathbb{R} \) is given, such that, given the training data \( D \), \( f(S,D) \) is an estimate of probability of correct recognition for \( S \) for all examples \( U \). Without loss of generality, from now on we assume that \( f \) is maximized, and that its lower bound (the worst value) is 0.

In the above setting, given the training data \( D \), the task of synthesizing globally optimal feature set (representation) \( S^* \) can be formalized as:

\[
S^* = \arg \max_{S \in \Omega} f(S,D)
\]
When no assumptions are made concerning the nature of \( f \), the task of finding \( S^\ast \) has exponential time complexity. To prove that, it is enough to show that the size of the search space \( \Omega \) is an exponential function of representation size; this manifests itself at the following two levels:

- **Single feature** level. Let us assume for simplicity, that the primitive operations are parameter-free. Then, the number of different feature extraction procedures is \( m^n \), where \( m \) is the size of the library of primitive operations, and \( n \) stands for the length of the feature extraction procedure.

- **Feature set** level. Given an upper bound on the representation size \( k \) (number of features), the total number of feature sets to be considered is

\[
|\Omega_k| = \sum_{i=0}^{k} \binom{m^n}{i}.
\]  

(2)

This prohibitively large number disables the use of exhaustive search algorithm even for relatively small values of \( m, n, \) and \( k \). Other search techniques that would possibly reduce the time complexity, like branch and bound, make some assumptions concerning the characteristic of the function \( f \) being optimized (e.g., local upper/lower bounds, global convexity). In our learning-from-examples setting, however, \( f \) is essentially given by the training data \( D \) and no useful assumptions concerning its nature can be made.

Heuristic or metaheuristic search is, therefore, the only plausible method that can be applied to the synthesis task posed as above and that can yield reasonably good suboptimal solutions \( S_\ast, f(S_\ast D) < f(S' D), f(S_\ast D) >> 0 \), in polynomial time. In fact, for some problems the solutions found during the heuristic search may even be globally optimal; however, as we don't know the upper bound of recognition performance, we cannot discern it from the suboptimal ones.

### 3.3 Rationale for the Use of Coevolution

To search the space of representations \( \Omega \), we propose to use the cooperative coevolution (CC) [10], a variety of recognized metaheuristics of genetic algorithm. In formal terms, the choice of this particular method is irrelevant, as, according to Wolpert’s ‘no free lunch’ theorem [16], a hunt for an universal, best-of-all metaheuristics is futile. More formally, let us define a search algorithm as an iterative process that, at each step maps its current state (a set of \( p \) points in the search space) onto a new state. Then, given any pair of search algorithms \( a_1 \) and \( a_2 \),

\[
\sum \mathcal{P}(\tilde{c} | f, p, a_1) = \sum \mathcal{P}(\tilde{c} | f, p, a_2).
\]  

(3)

where \( f \) is a fitness function and \( \tilde{c} \) is the histogram of fitness. As a result, the average performance of any metaheuristic search over a set of all possible fitness functions is the same.

In real world, however, not all fitness functions are equally probable. Most real problems are characterized by some features that make them specific. The practical utility of a search/learning algorithm depends, therefore, on its ability to detect and
benefit from those features. In particular, the complexity of the problem and the way it may be decomposed are such features.

In the last few years, coevolution has been reported as a promising approach to handle the increasing complexity of problems posed in artificial intelligence and related disciplines. In particular, its collaborative variety, the cooperative coevolutionary algorithm (CC) [10], besides being appealing from the theoretical viewpoint, has been reported to yield interesting results in some experiments [14,1]. The basic feature that makes CC different from EC is that, instead of having just single population of individuals, in CC one maintains several of them, with individuals in populations encoding only a part of the solution to the problem. Therefore, individuals cannot be evaluated independently; they have to be (temporarily) combined with some representatives from the remaining populations to form a solution, called hereafter organism O, that can be evaluated. This joint evaluation scheme forces the individuals from particular populations, and, as a result, the entire populations, to cooperate. In other words, it is an organism, not an individual, that corresponds to the search state S in the formalism introduced in Section 3.2 (O ≡ S). Except for this evaluation step, the evolution proceeds in each population independently (see Table 1).

**Table 1. Comparison of EC and CC algorithms (major differences in boldface)**

<table>
<thead>
<tr>
<th>Evolutionary Computation (EC)</th>
<th>Cooperative Coevolution (CC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>solution = an individual in population</td>
<td>solution = an organism composed of individuals selected from different populations</td>
</tr>
<tr>
<td>initialize population loop</td>
<td>initialize populations loop</td>
</tr>
<tr>
<td>evaluate individuals</td>
<td>evaluate organisms and assign fitness to individuals in populations</td>
</tr>
<tr>
<td>store best individual</td>
<td>store best organism for each population</td>
</tr>
<tr>
<td>select mating candidates recombine parents and use their offspring as the next generation</td>
<td>select mating candidates recombine parents and use their offspring as the next generation</td>
</tr>
<tr>
<td>until stopping condition return best individual</td>
<td>until stopping condition return best organism</td>
</tr>
</tbody>
</table>

Cooperative coevolution provides the possibility of breaking up the complex problem into components without specifying explicitly the objectives for them. The manner in which the individuals from populations cooperate emerges as the evolution proceeds. In our opinion, this makes CC especially appealing to the problem of synthesis of recognition systems, where the overall target is well defined, but there is no a priori knowledge about what should be expected at intermediate stages of processing, or such knowledge requires an extra effort from the designer.

Recently, some advances have been made in the area of theoretical foundations of coevolution. Most work done so far focuses on attempting to prove that the behavior of coevolution is similar to that of regular evolution. For instance, it has been shown in [6], that, when some assumptions are made regarding parameters (number of populations and population size), coevolutionary algorithms exhibit the same type of dynamics as EC.
3.4 Multi-agent Approach and Decision-Level Fusion

Facing the suboptimal character of representations synthesized by the evolutionary process, we incorporate in our approach multi-agent methodology that aims to compensate for that imperfection and allows us to boost the overall performance.

The search for performance improvement by approaching the problem with multiple algorithms/agents has a long history in AI-related disciplines. In particular, the so-called compound- or meta-classifiers became an important research issue in PR and ML during the last decade. Many results concerning stacked generalization, mixture of experts, bagging, etc. (see, e.g., [1]), indicate that employing many agents and aggregating their outcomes may significantly improve the (recognition) performance in comparison to the single-agent approach.

The basic prerequisite for the agents' fusion to become beneficial is their diversification. This may be ensured by using homogeneous agents with different parameter setting, or homogenous agents with different training data (e.g. bagging), or heterogeneous agents, to mention only a few most popular approaches. In the approach proposed here, the diversification is naturally provided by the random nature of the genetic search. In particular, there are at least two approaches that seem to be reasonable methods for multiple-agent acquisition from genetic search:

- exploiting different well-performing systems synthesized during a single genetic search,
- exploiting best systems synthesized during many genetic searches that started from different initial states (initial populations).

Though computationally more expensive, the latter of the techniques provides better performance of particular agents and better differentiation of the resulting agents' pool, so it has been adopted in the approach described here. For the sake of simplicity, the agents differ only in the features synthesized by genetic searches; the classifiers used in particular subsystems are homogenous.

4 Technical Approach

For the sake of clarity, let us first strictly distinguish the synthesized recognition system, from the synthesis algorithm (learning algorithm) that leads to its creation. The recognition system takes an image as an input and produces recognition decision (object identity) at its output; on the contrary, synthesis algorithm takes the training data (set of images) as input and yields the recognition system. The synthesis algorithm proceeds in two stages, which correspond to two main components/parts of the synthesized system. In the following subsections we describe the recognition system and the synthesis algorithm.

4.1 Architecture of the Synthesized Recognition System

The result of synthesis is a feature-based recognition system that incorporates data fusion at different levels of processing. The top-level architecture encompasses a set of
subsystems that work in parallel, process the input image $X$ independently, and output recognition decisions that are further aggregated by a simple majority voting procedure into the final decision. The number of subsystems $n_{sub}$ is a parameter set by the designer. All subsystems are homogenous as far as the structure is concerned; they only differ in the features extracted from the input image and the knowledge acquired by the classifier. Thus, for simplicity, all the following description will be concerned with a single subsystem.

![Diagram of the architecture of a single synthesized subsystem]

Fig. 1. The architecture of a single synthesized subsystem

Each subsystem has two major components (see Fig. 1): (i) a collection of feature extraction procedures, and (ii) a trained classifier. In each subsystem, the process of recognition starts with the input image $X$ being fed into the set of feature extraction procedures $S$. The procedures from $S$ yield feature values, which are subsequently gathered to build representation, i.e., a fixed-length vector of feature values $Y(X)$. Finally, that feature vector is passed through the classifier $C$, that yields this subsystem's vote $C(Y(X))$ (class probability distribution).

4.2 The Algorithm for Synthesizing Recognition System

The synthesis of recognition system consists in running independent learning process for each subsystem shown in Fig. 1. Although the synthesis algorithm used is the same for all subsystems, the results are diversified by starting the feature synthesis process from different initial populations. This is technically implemented through initializing the pseudorandom number generator with different values.

For a single subsystem, the learning encompasses two stages: (1) coevolutionary feature synthesis and (2) classifier induction. The following subsections describe both these stages in detail.

4.2.1 Coevolutionary Synthesis of Feature Extraction Procedures: The basic engine for the feature synthesis algorithm employs the search based on cooperative coevolution described in Sect. 3.3. Its result, feature set $S$, is implemented into the synthesized system.

The algorithm, whose overall architecture is shown in Fig. 2, maintains a collection of populations, each being a set of individuals. Each individual $I$ encodes a single image processing/feature extraction procedure and, given an input image $X \in D$, yields a vector $y(I,X)$ of scalar feature values. For clarity, this encoding and execution of feature extraction program is detailed in a separate Sect. 4.2.2.
This coevolutionary search proceeds in all populations independently, except for the evaluation phase. To evaluate an individual \( I_j \) from population \( j \), we first provide for the remaining part of the representation. For this purpose, representatives \( I^*_i \) are selected from all remaining populations \( i \not= j \). A representative \( I^*_i \) of \( i \)th population is defined here in a way that has been reported to work best [14]: it is the best individual w.r.t. the evaluation done in the previous generation. In the first generation of evolutionary run, however, since no prior evaluation data is given, it is a randomly chosen individual.

Then, \( I_j \) is temporarily combined with representatives of all the remaining populations to form an organism

\[
O = \langle I^*_1, \ldots, I^*_{j-1}, I^*_j, I^*_{j+1}, \ldots, I^*_{n_{sub}} \rangle,
\]

that corresponds to search state \( S \) (see Section 3.2). Then, the feature extraction procedures encoded by individuals from \( O \) are run for all images \( X \) from the training set \( D \). The scalar feature values \( y \) computed by them are grouped, building the compound feature vector \( Y \):

\[
Y(X) = \langle y(I^*_1, X), \ldots, y(I^*_{j-1}, X), y(I^*_j, X), y(I^*_{j+1}, X), \ldots, y(I^*_{n_{sub}}, X) \rangle.
\]

Feature vectors \( Y(X) \), computed for all training images \( X \in D \), together with the images' decision class labels constitute the dataset:

\[
\langle Y(X), i : \forall X \in D, \forall D \rangle
\]

Finally, cross-validation, i.e. multiple train-and-test procedure is carried out on these data. For the sake of speed, we use here fast classifier \( C_{rlt} \) that is usually much simpler than classifier \( C \) used in the final synthesized system (see Fig. 1). The resulting predictive recognition ratio becomes the evaluation of the organism \( O \), which is sub-
sequently assigned as the fitness value to $f()$ the individual $I_j$, concluding its evaluation process:

$$f(I_j, D) = f(O, D) = \frac{\text{card}\left(\left\{ (Y(X), i), \forall D_j, \forall X \in D_j : C(Y(X)) = i \right\} \right)}{\text{card}(D)} \quad (7)$$

Using this evaluation procedure, the coevolutionary search proceeds until some stopping criterion (usually considering computation time) is met. The best synthesized organism/representation $S$ becomes the part of the feature extraction module presented in Fig. 1.

4.2.2 Representation of Feature Extraction Procedures: As it has been already mentioned in Section 3, we assume that (i) basic, general-purpose building blocks are given a priori to the synthesis process, and (ii) that an individual feature extraction procedure is a chain/sequence of such blocks. These assumptions provide the system with basic background knowledge BK (however, not domain knowledge, DK) that speeds up the convergence of the search process.

Though the overall feature synthesis process relies on cooperative coevolution, for representing the feature extraction procedures as individuals in evolutionary process, we adopted a variety of Linear Genetic Programming (LGP) [1], a hybrid of genetic algorithms (GA) and genetic programming (GP), as the one that seems to meet these assumptions best. The individual's genome, i.e. the internal encoding of solution it represents, is a fixed-length string of numbers. The genome is interpreted as a sequential program composed of (possibly parameterized) basic operations that work on images and scalar data. This LGP-like representation combines advantages of both GP and GA, being procedural and at the same time more resistant to the destructive effect of crossover that may occur in 'regular' GP.

The above mentioned operations are effectively calls to image processing and feature extraction functions. They work on registers, i.e. working variables, and may use them both as input as well as output arguments. Image registers store processed images, whereas real-number registers keep intermediate results or scalar features. Each image register is single-channel, has the same dimensions as the input image, and maintains a single rectangular ROI that may be used by an operation as a mask. For simplicity, both the number of image registers as well as the number of real-number registers are controlled by the same parameter $n_{reg}$.

Technically, each individual is a fixed-length string of bytes $0..255$, with each chunk of 4 consecutive bytes encoding a single operation with the following elements:

- operation code (opcode),
- ROI flag – decides whether the operation should be global (work on the entire image) or local (limited to rectangular region of interest (ROI)),
- ROI size (ignored if ROI flag is 'off'),
- arguments – numbers (identifiers) of registers to fetch input data and store the result.

An exemplary operation is morphological opening (operation code) using rectangular ROI (ROI flag 'on') of size 14 (ROI size) on the image fetched from image register
#4 (pointed by argument #1), and storing the result in image register #5 (pointed by argument #2).

There are currently 70 operations implemented in the system. They mostly consist of calls to functions from Intel Image Processing and OpenCV libraries, encompass image processing, ROI-related operations, feature extraction, and arithmetic and logic operations.

In the above settings, the processing of a single input image (example) \( X \in D \) by the LGP procedure encoded in an individual \( I \) proceeds as follows (see Fig. 3):

![Diagram of LGP procedure with annotations](image)

**Fig. 3.** Illustration of genome interpretation during procedure execution (genome length 12, one real-number register)

1. **Initialization** of register contents:
   - Each of the \( n_{\text{reg}} \) image registers is set to \( X \). The ROIs of images are set to consecutive local features (here: bright 'blobs') found in the image, so that ROI in the \( i^{th} \) image register encompasses \( i^{th} \) local feature.
   - Real-number registers are set to the coordinates of corresponding ROIs; in particular, real-number registers \( 2i \) and \( 2i+1 \) store the coordinates of the \( i^{th} \) image ROI.

2. **Execution**: the operations encoded by \( I \) are carried out one by one, with intermediate results passed from operation to operation by means of image and real-number registers (see example in Fig. 3).

3. **Interpretation**: the scalar values \( y_j(I,X), j=1,\ldots,n_{\text{reg}} \) contained in the \( n_{\text{reg}} \) real-value registers at the end of procedure’s execution are interpreted as the output yielded by \( I \) for image \( X \). There values are gathered to form an individual’s output vector

\[
y = \left< y_1(I,X), \ldots, y_{n_{\text{reg}}}(I,X) \right>
\]

that is subject to further processing described in Sect. 4.2.1.
4.2.3 **Classifier Induction**: The result of the first stage of learning is the best representation \( S \) (see Fig. 1) synthesized in coevolutionary process. The second stage of learning is consists in (a) computing the compound feature vector \( Y(X) \) for all the training examples \( X \in D \), and (b) training the classifier \( C \) on the resulting data set.

This process resembles the classifier induction that takes place in evaluation process described in 4.2.2. However, this time the entire set \( D \) is used for classifier training, as no more performance estimation is required. Secondly, as this learning is single-event, a more sophisticated induction algorithm \( C \) may be used (as compared to the classifier \( C_{th} \) used in the evaluation function).

5 **The Experimental Results**

The primary objective of the computational experiment was to evaluate the overall performance of the approach and verify its scalability with respect to the number of decision classes.

5.1 **Parameter Setting**

Table 2 shows parameter settings used for the feature synthesis process. As far as the second stage of learning is concerned (see Fig. 1), a compound classifier \( C \) has been used to boost the recognition performance. In particular, \( C \) implements the '1-vs.-all' scheme, i.e. is composed of \( l \) base classifiers (where \( l \) is the number of decision classes), each of them working as discriminator between a single decision class and all the remaining classes. To aggregate their voices, a simple voting procedure is used. Support vector machines with polynomial kernels of degree 3 have been employed as base classifiers. To train them, we used fast sequential minimal optimization algorithm [9] with complexity constant set to 10.

The training time has been set to 4000 seconds to estimate the quality of results the proposed method is able to attain in a limited time. Let us stress that this demand of computational resources concerns learning only; in testing, the trained system recognizes a single object in a few dozens of milliseconds.

5.2 **The Data and the Task**

The proposed approach has been tested on the demanding task of target recognition in synthetic aperture radar (SAR) images. The difficulties associated with this recognition task are:

- poor visibility of objects - most of them are reflected as sets of scattering centers only (no line features are present for these man-made objects at 1 foot resolution of data),
- low persistence of features under rotation, and
- high level of noise.
Table 2. Parameter setting for feature synthesis process concerning single decision class

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutation operator</td>
<td>one-point, probability 0.1</td>
</tr>
<tr>
<td>Crossover operator</td>
<td>one-point, probability 1.0, cutting allowed at every point</td>
</tr>
<tr>
<td>Selection operator</td>
<td>tournament selection with tournament pool size = 5</td>
</tr>
<tr>
<td>Number or registers (image and numeric) (n_{reg})</td>
<td>2</td>
</tr>
<tr>
<td>Number of populations (n_{sub})</td>
<td>4</td>
</tr>
<tr>
<td>Genome length</td>
<td>40 bytes (10 operations)</td>
</tr>
<tr>
<td>Single population size</td>
<td>200 individuals</td>
</tr>
<tr>
<td>Classifier (C_{fs}) used for feature set evaluation</td>
<td>decision tree inducer C4.5</td>
</tr>
<tr>
<td>Time limit for evolutionary search</td>
<td>4000 seconds (Pentium 1.4 GHz processor)</td>
</tr>
<tr>
<td>Number of subsystems (n_{sub})</td>
<td>10</td>
</tr>
</tbody>
</table>

The MSTAR public database [11] containing real images of several military targets taken at different aspects/azimuths has been used as the source of images for the experiments described in this section. The original images have the same spatial resolution of one foot, but different sizes, so they have been cropped to 48×48 pixels. Only the magnitude part of the complex images has been used. No other form of preprocessing (e.g., image enhancement) has been applied.

To investigate the scalability of the proposed approach w.r.t. to the problem size, we defined several datasets with increasing number of decision classes for 15-deg. depression angle. The smallest problem considered concerned \(l=2\) decision classes: BRDM2 and ZSU. Then, the consecutive problems were created by adding the decision classes up to \(l=8\) in the following order: T62, Zil131, a variant A04 of T72 (T72#A04 in short), 2S1, BMP2#9563, and BTR70#C71 (see Fig. 4).

For \(i^{th}\) decision class, its representation \(D_i\) in the training data \(D\) consists of two subsets of images (for cross-validation training and testing; see Section 4.2.1) sampled uniformly from the original database with respect to 6-degree azimuth step. Training set \(D\) contains therefore always \(2*(360/6)=120\) images from each decision class, so its total size is \(120*i\). The corresponding test set \(T\) contains all the remaining images (for given target and elevation angle) from the original MSTAR collection, i.e.

\[
T = \bigcup_{i=1}^{i} D_i ,
\]

where \(\overline{D}\) stands for complement with respect to the MSTAR database. In this way, the training and test sets are strictly disjoint.
5.3 Results

Figure 5a presents the performance of the proposed synthesis approach on the test data as a function of the number of decision classes for the case of forced recognition. It may be easily observed, that, as new decision classes are added to the problem, the recognition falls down very slowly. The major drop-offs occur when T72 tank and 2S1 self-propelled gun (classes 5 and 6, respectively), are added to the training data; this is probably due to the fact that these targets are visually similar to each other (e.g., both have gun turrets, see Fig. 4) and significantly resemble the T62 tank (class 3). On the contrary, introducing consecutive targets 7 and 8 (BMP2 and BTR60) did not affect much the performance.

Figure 5b shows the receiver operating characteristic (ROC) curves obtained by modifying the confidence threshold that controls the voting procedure. Again, the presented results vote in favor of our method: the ROC curves do not drop suddenly as the false positive ratio decreases. Therefore, high probability of correct identification (PCI) may be obtained when accepting some rejection rate (e.g., for 4 decision classes, PCI=0.99 when accepting ~0.12 rejection rate).
6 Conclusions

In this contribution, we provided experimental evidence for the possibility of synthesizing, without or with little human intervention, a feature-based recognition system which recognizes 3D objects at the level of recognition ratio comparable to handcrafted solutions and maintains performance as the number of objects to be recognized increases. Let us emphasize that these encouraging results have been obtained in the demanding field of SAR imagery, where the acquired images only roughly depict the underlying 3D structure of the object.

There are several major factors that contribute to the overall high performance of the approach. First of all, in the feature synthesis phase, it manipulates entire feature extraction procedures, as opposed to many approaches reported in the literature, which are usually limited to learning meant as parameter optimization. This allows for learning/developing sophisticated features, which are novel and sometimes very different from expert’s intuition. Secondly, the paradigm of coevolution allows us to decompose the task of representation (feature set) synthesis into several semi-independent, cooperating subtasks. In this way, we exploit the inherent modularity of the learning process, without the need of specifying explicit objectives for each developed module. And thirdly, the fusion at feature and decision level helps us to aggregate, sometimes contradictory, information sources and build a recognition system that is close to perfection with a bunch of imperfect components at hand.

As no domain-specific knowledge has been used, this approach may be extended to other visual learning tasks at low expense of time and effort. This includes also unsupervised learning, as this is only matter of making appropriate changes to the fitness function. On the other hand, the method’s background knowledge may be easily tailored to the task by modifying the library of elementary operations.
Acknowledgements. We would like to thank the authors of software packages: ECJ [5] and WEKA [15] for making their software publicly available. This research was supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the U. S. Government. The first author is supported by the Polish State Committee for Scientific Research, research grant no. 8T11F 006 19.

References