International Journal of Computational Intelligence and Applications
Vol. 5, No. 3 (2005) 305–320
© Imperial College Press
ICP Imperial College Press
www.icpress.co.uk

USING COMPETITIVE CO-EVOLUTION TO EVOLVE BETTER PATTERN RECOGNISERS

TARAS KOWALIW, NAWWAF KHARMA*, CHRISTOPHER JENSEN,
HUSSEIN MOGHNIEH and JIE YAO

Department of Electrical and Computer Engineering
Concordia University, 1455 de Maisonneuve Blvd. W. Montréal
Québec, Canada. H3G 1M8
**kharma@ece.concordia.ca*

Revised 6 June 2005

We present a system for the automatic synthesis of classifiers. The CellNet system for generating binary pattern classifiers is used as a base for further experimentation. As in the original CellNet software, we evolve pattern recognisers (hunters). However, in this version called CellNet Co-Ev, we also evolve the patterns (prey) in a competitive co-evolution. Patterns evolve through the application of camouflage functions, which are used to obscure the data naturally found in the database. The addition of this competitive co-evolution yields a larger and more varied database, artificially increasing the difficulty of the classification task. Application to the CEDAR database of handwritten characters shows an increase in the reliability of the evolution of recognisers, as well as in the elimination of over-fitting, relative to the original CellNet software.

Keywords: Pattern classification; character recognition; genetic algorithms; competitive co-evolution; CellNet.

1. Introduction and Review

The aim of project CellNet is the creation of a software system capable of automatically evolving complete pattern recognisers from arbitrary pattern sets; that is, to minimise the amount of expert input required for the creation of pattern recognisers given some set of target data. This is an ambitious goal, requiring much additional work. This paper describes a step towards this goal: the original CellNet system is augmented with a competitive co-evolution, towards the aim of improving reliability and allowing for an artificial inflation of the provided data set. Unlike the original CellNet system, the approach described in this paper allows for the co-evolution of patterns; that is, images in the database are augmented with a genome which specifies the application of camouflage functions — these are topologically invariant functions which obscure the original pattern in the image, by introducing deformations, noise, etc. In doing so, the classification task becomes more difficult, leading to the evolution of pattern recognisers which are more reliable. Additionally, the

problem of over-fitting,[a] a source of constant worry in nearly all pattern recognition tasks, is eliminated.

A Pattern Recognition System is almost always defined in terms of two functionalities: the description of patterns and the identification of those patterns. The first functionality is called feature extraction, and the second, pattern classification. Since there are many types of patterns in this world, ranging from the images of fruit flies to those of signatures and thumb prints, the focus of most research endeavours in pattern recognition has rightfully been directed towards the invention of features that can capture what is most distinctive about a pattern. This leads to two overlapping areas of research associated with features: feature selection and feature creation. Feature selection has to do with choosing a small set of features from a larger set, in order to maximise the rate of recognition of an associated classifier, and simultaneously reducing the (computational) cost of classification. Feature creation, on the other hand, has to do with the creation or growth of complex features from a finite set of simpler ones, also for the benefit of an associated classifier. Both areas of research are reviewed below.

Many topics refer to Evolutionary Computation, especially Genetic Algorithms (GAs); the interested reader may refer to Mitchell's text for an introduction,[9] or to the paper by Sarker *et al.* for a more recent review with discussion of some applications.[12]

1.1. *Feature selection*

Siedlecki[14] was the first to suggest the use of GAs for feature selection. Ten years later, Kudo[8] demonstrates the superiority of GAs over a number of traditional search & optimisation methods, for sets with a large number of features. Vafaie[16] shows that a GA is better than *Backward Elimination* in some respects, including robustness. Guo[6] uses a GA for selecting the best set of inputs for a neural network used for fault diagnosis of nuclear power plant problems. Moser[10] surveys previous attempts at speeding up the action of GAs, and then proposes two new architectures of his own: VeGA, which uses parallelism to simultaneously evaluate different sets of features (on one computer), and DVeGA, which utilises an adaptive load balancing algorithm to use a network of heterogeneous computers to carry out the various evaluations. Shi[13] applies a GA to the selection of relevant features in handwritten Chinese character recognition. Fung[4] uses a GA to minimise the number of features required to achieve a minimum acceptable hit-rate, in signature recognition. Yeung[16] succeeds in using a GA in tuning four selectivity parameters of a *Neocognitron*, which is used to recognise images of handwritten Arabic numerals. Current research often involves the use of GAs in ensembles of classifiers, as in Ref. 5.

[a]Over-fitting is a common problem in machine learning: The tendency of classifiers to perform with high accuracy on data during training, but with less accuracy on other data. It may be measured by contrasting accuracy on the training data against some previously unseen independent test data.

1.2. *Feature creation*

The first paper that addresses feature creation appears to be [Ref. 15]. In his work, Stentiford uses evolution to create and combine a set of features (into a vector), which is compared to reference vectors, using nearest-neighbour classifiers. A single feature is a set of black- or white-expecting pixel locations. The results were generated using a training set of 30,600 printed characters and a test set of 10,200 printed characters. The error rate, on the test set, was 1.01%, and was obtained using 316 nearest-neighbour classifiers automatically generated by the evolutionary system. More recently, Brumpy *et al.*[3] used evolution to create features for use in analysing satellite and other remote-sensing captured images. Specifically, the researchers defined a genetic programming scheme for evolving complete image processing algorithms for, for example, identifying areas of open water; that is, a general framework in which both feature creation and classification was created, allowing a Genetic Algorithm to find the best algorithm for recognition. Their system called GENIE successfully evolved algorithms that were able to identify the intended ground features with 98% accuracy. Co-evolutionary methods have been used for the creation of descriptions of patterns or images, as in Veenman *et al.*,[17] but in the context of multi-agent systems; while a powerful methodology for many applications, multi-agent approaches are too slow for our present purposes.

1.3. *Evolvable pattern recognition systems*

Given successful systems for feature creation, it is a logical step to attempt to create whole pattern recognition systems; that is, systems which automatically perform all work involved in the pattern recognition process. After all, the hardest part of successful pattern recognition is the selection/creation of the right set of features. Indeed, initial examples of such systems seem to have focused on the evolution of features or feature complexes, which are useful for classification. Two systems described below to do this. CellNet[6] blurs the line between feature selection and the construction of binary classifiers out of these features. HELPR[10] also evolves feature detectors, but the classification module is completely separate from the feature extraction module and is trained rather than evolved. Other differences exist, but, both attempts are the only systems, that we know of, that aim at using artificial evolution to synthesise complete recognition systems (though currently for different application domains), with minimum human intervention.

CellNet is an ongoing research project aiming at the production of an autonomous pattern recognition software system, for a large selection of pattern types. Ideally, a CellNet operator would need little to no specialised knowledge to operate the system. To achieve this, CellNet divides the problem (and hence the solution) into two parts: feature creation and classifier synthesis.

The long-term goal of the CellNet project is the inclusion of methodologies for the simultaneous evolution of both features and classifiers (Cells and Nets); at present, however, a set of hard-coded features are used. Some interesting

perspectives are offered as to how an evolvable feature creation framework may be structured. The most interesting of these suggestions is the use of a pre-processing routine for deconstructing images using techniques inspired by pre-attentive vision in humans — from there, a wide array of possible machine learning techniques may be used.

In its current form, CellNet is capable of evolving classifiers for a given set of patterns. To achieve this it uses a specialised genetic operator: Merger. Merger is an operator, somewhat similar to that used in MessyGAs,[4] designed to allow the algorithm to search increasingly larger feature spaces in an incremental manner. This is different from a normal GA, where the dimensionality of the search space is fixed at the start of the run. CellNet is cast as a general system, capable of self-adapting to many handwritten alphabets, currently having been applied to both Arabic and Indian numbers (using the CEDAR database, and a second database collected in Montreal[1]). This paper may be viewed as an extension of the CellNet system — as such, CellNet is described in more detail in Sec. 2.

HELPR is composed of two modules: a features extraction module and a classification module. The classification system is not evolvable; only the feature extractor is. The feature extraction module is made of a set of feature detectors. Its input is the raw input pattern and its output is a feature vector. The system is designed to handle signals (not visual patterns or patterns in general).

Each feature extractor has two stages: a transformation network followed by a capping mechanism. A transformation network utilises a set of morphological operations (such as Open and Erode) with structuring elements in addition to a small number of arithmetic operations (such as addition and multiplication). As such a transformation network (or TN) represents a mixed morpho-arithmatic expression which transforms an n-element digitised input signal into an n-element output signal. The purpose of the TN is to highlight those features of the input signal that are most distinctive of the target classes. On the other hand, the purpose of the capping mechanism (which is a single-layer of perceptrons) is to transform the n-element signal coming out of the TN into a k-element array of scalars, where k is the number of target classes defined by the user. Ideally, every element will return $a + 1$ for a single class and -1 for the rest.

The most interesting features of this work is that (a) it adopts a holistic approach to the evolution of pattern classifiers: it boldly states, starting with the title, that such a goal is possible; (b) it demonstrates that it is possible for a computerised system to automatically evolve a pattern classifier, which can be used for a real-world application; (c) it augments the standard evolutionary processes in very creative ways to produce a customised system, which perform well and (e) it suggests that it is possible, using a small set of morphological operations (in addition to simple arithmetic), to characterise a wide range of signals. There is no doubt that this paper, as well as the CellNet architecture are examples of a new trend in pattern classification: that of the autonomous pattern recogniser. However, there is also no doubt that both works still have a number of deficiencies to deal with.

Both HELPR and CellNet only evolve part of the system: other parts of the system are hard-coded. In addition, both systems have a large number of parameters that are manually set by an expert user. Finally, both systems only work on specific input signals/images; the promise of extension is there, but not yet realised. This is why the following set of problems must be dealt with:

(1) **Feature Creation** (or Detection) mechanisms that are suited to large sets of classification problems. This is to eliminate the need for serious re-configuration or worse still, re-engineering, every time a different application domain is targeted.
(2) **Elimination of Parameters**, many of which need to be set by the user before the system is exploited. These parameters include: probability distributions types, probability values, population size, number of feature extractors and so on.
(3) **Thorough Testing** of the evolved system against a diverse set of pattern databases (and not just Radar signatures or Arabic Numerals), and doing so, without subjecting the system to any significant amount of re-configuration.

This paper may be viewed as a step towards the realisation of points 2 and 3. Co-Evolution is demonstrated to help eliminate the problem of over-fitting in the creation of a classifier, hence eliminating the need for an expert to manually determine the correct parameters. Additionally, the camouflage routines presented aid in the diversification of a given pattern database, allowing for greater variance in any given set of data. That is, by artificially making the problem harder during the training period, better results are obtained in terms of reliability and potential application to the real world, without any database-dependent expert input required.

2. Hunters and Prey

2.1. *Hunters*

CellNet hunters were first introduced in Ref. 7, a reader interested in their formal representation is urged to consult that source — our implementation is identical. However, given they are an entirely new and rather complicated representations for pattern recognisers, we offer an informal explanation of their structure, along with some illustrative examples.

2.1.1. *Feature functions*

The basis on which Hunters are built is a set of normalised feature functions. The functions (all applied to thinned figures) used by CellNet Co-Ev are [parameterised histograms, central moments, Fourier descriptors, Zernike moments, normalised width (of a bounding box), normalised height, normalised length, number of terminators, number of intersections]. However, for the purposes of explanation, we

shall assume that our examples use only two: F_1 and F_2. This assumption is made for ease of visual representation of a 2-dimensional Feature Space, and is easily generalised.

2.1.2. *Hunters*

A Hunter is a binary classifier — a structure which accepts an image as input, and outputs a classification. The fitness function which is used to drive the Genetic Algorithm determines which digit the agent classifies; for example, assuming our fitness function specifies that we are evolving Hunters to recognise the digit "ONE", a Hunter which returns "yes" given an image will implicitly be stating that the image is a "ONE", as opposed to "NOT-ONE", i.e. any other digit. We will refer to these classes as the primary class and the non-primary class. Hence, a Hunter outputs a value from {PRIMARY, NON-PRIMARY, UNCERTAIN} when presented with an image.

A Hunter consists of Cells, organised in a Net. A Cell is a logical statement — it consists of the index of a Feature Function, along with bounds. Every cell is represented by the following format:

Feature Function $\boldsymbol{F}_i$	Bound $\boldsymbol{b}_1$	Bound $\boldsymbol{b}_2$

Provided with an image I, a cell returns TRUE, if $\boldsymbol{b}_1 < \boldsymbol{F}_i\ (I) < \boldsymbol{b}_2$, and FALSE otherwise.

A Net is an overall structure which organises cells into a larger tri-valued logical statement. That is, a Net is a logical structure which combines the TRUE/FALSE values of its constituent Cells to return a value from {PRIMARY, NON-PRIMARY, UNCERTAIN}.

The structure of a Net is as follows: A Net is a tree, with a voting mechanism as its root node. At depth 1, there are a set of one or more Chromosomes.

Chromosomes consist of trees which begin with a Class Bit — this bit determines whether or not the Chromosome votes for "PRIMARY" or "NON-PRIMARY". Following the Class Bit is a tree of one or more Cells, connected into a logical structure by AND and NOT nodes. A Chromosome may be represented as a string as follows:

Class BIT C	[NOT]	$Cell_1$	AND	[NOT]	$Cell_2$	AND	...

Hence, the latter part is a logical statement, which returns TRUE or FALSE. A Chromosome will return $\boldsymbol{C}$ if the logical statement returns TRUE — otherwise it will remain silent.

A Net is a collection of such Chromosomes, connected via a Vote mechanism. The Vote mechanism collects input from each Chromosome (although some Chromosomes will remain silent), consisting of a series of zero or more values of "PRIMARY" or "NON-PRIMARY". The Vote mechanism will tally the number of each, and output the majority, or "UNCERTAIN" in the case of no input or a tie.

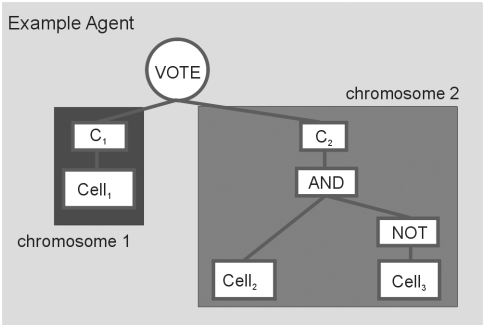

Fig. 1. Tree diagram of the example agent.

For example, consider the following Example Agent, specified by the two Chromosomes — chromosome1 and chromosome2:

chromosome1:	C_1	$Cell_1$			
chromosome2:	C_2	$Cell_2$	AND	NOT	$Cell_3$

This Example Agent may be drawn as a tree, shown in Fig. 1.

A Hunter is a Net — that is, a Hunter is an organised collection of one or more Cells, which when presented with an image, will return one of "PRIMARY", "NON-PRIMARY" or "UNCERTAIN". The complexity of a Hunter is the number of cells it contains, regardless of organisation.

2.1.3. *Examples of hunters of complexity one*

The following are some examples of Hunters with complexity one, and interpretations in a two-dimensional feature space. Assume the Primary class is "ONE", and consists the non-primary class is "NOT-ONE".

Our first Hunter, A_1, consists of a single cell in a single chromosome — it is illustrated in Fig. 2. It is instructive to consider the Feature Space of all images on the basis of Feature F_1 and F_2 — every image maps to an $\langle x,y \rangle$ coordinate in this space, and hence may be drawn in a unit square. Agent A_1 may be viewed

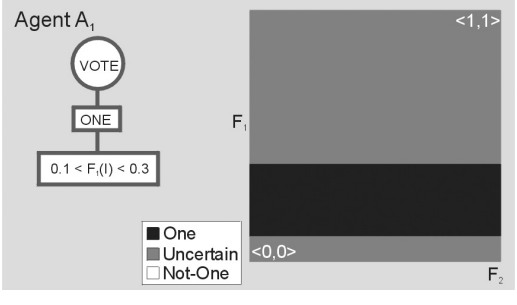

Fig. 2. Agent A_1 and its partition of Feature Space.

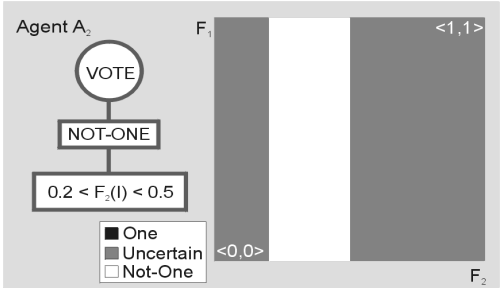

Fig. 3. Agent A_2 and its partition of Feature Space.

as a statement which partitions Feature Space into three disjoint sets — this is also illustrated in Fig. 2. A second Hunter, A_2, is illustrated in Fig. 3; this agent's partition of the same feature space is also illustrated.

2.1.4. *Merger*

Thus far, we have given examples only of Hunters with complexity one — this is the state of the CellNet Co-Ev system when initialised. What follows is a system of cooperative co-evolution which generates agents of increasing complexity.

Cooperative Co-evolution is achieved through the inclusion of a new Genetic Operator, augmenting the typical choices of Crossover and Mutation. This new operator, Merger, serves to accept two Hunters and produce a single new Hunter of greater complexity. The complexity of the merged Hunter will be the sum of the complexities of the parents.

Merger operates at the level of Chromosomes — when merging two Hunters, Chromosomes are paired randomly, and merged either Horizontally or Vertically. Vertical Merger simply places both Chromosomes in parallel under the Vote mechanism — they are now in direct competition to determine the outcome of the Vote. Horizontal Merger, on the other hand, combines the two Chromosomes to produce a single and more complex Chromosome, where the two original Chromosomes are connected via a AND or AND-NOT connective. Hence, Horizontal Merger serves to refine a particular statement in the vote mechanism.

There are several decisions made when selecting which type of Merger is undertaken, and how connections are made in the Horizontal case — these decisions are under the control of two Affinity bits which are associated with each Chromosome. These Affinity bits ensure that all decisions are under genetic control, and may be selected for. For more detail, refer to Ref. 7.

2.1.5. *Horizontal and Vertical Merger*

The Cooperative Co-evolution of Hunters, as realised through Merger is a technical process, more easily explained visually. We reconsider agents A_1 and A_2 of Sec. 3.1.3, considering their children through the Merger operator.

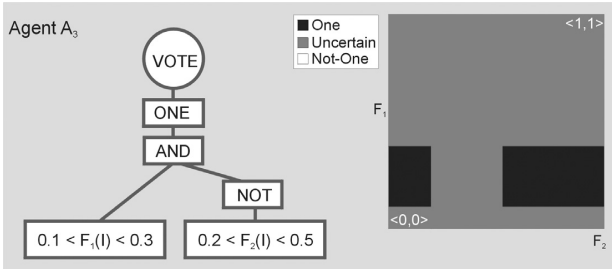

Fig. 4. Agent A_3 and its partition of Feature Space.

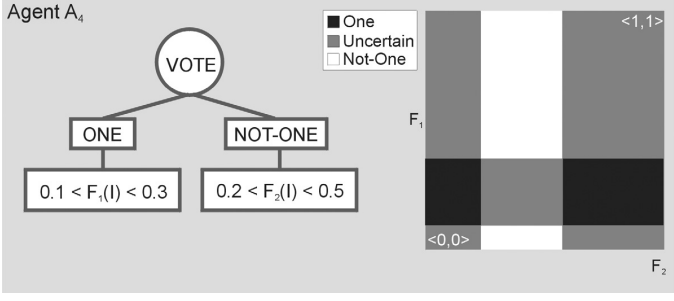

Fig. 5. Agent A_4 and its partition of Feature Space

Consider the Horizontal Merger of Hunters A_1 and A_2 — here, we produce agent A_3 by combining the Chromosomes of A_1 and A_2 into one new one, linked via an AND connective. As is visible in Fig. 4, Horizontal Merger may be viewed as the refinement of a partition created by two Chromosomes.

In contrast, consider the Vertical Merger of these same two Hunters, producing agent A4 — in this case, the Chromosomes are combined directly under the Vote mechanism. As shown in Fig. 5, Vertical Merger may be loosely be viewed as the union of the partitions generated by two Chromosomes.

2.2. *Prey*

In a divergence from the methodology in the original CellNet,[6] the second population in the CellNet Co-Ev system consist of Prey; Prey are primarily images, drawn from the CEDAR database of handwritten digits. In addition to the image, Prey disguise themselves via a set of camouflage functions, controlled genetically.

A Prey consists of a simple genome — an image index and a series of bits:

Image Index I	Bit b_1	Bit b_2	...	Bit b_k

The Image Index points to a particular image in the database. The Bits are Boolean parameters, indicating whether a particular camouflage function is to be applied or not.

Prey exist to be passed to Hunters for classification — prior to this, however, all camouflage functions specified by the series of bits in a Preys genome are applied — hence, a Hunter views a transformed version of the original Image specified by the Image Index.

Camouflage functions used by the CellNet Co-Ev system consist of {Salt&Pepper, Scaling, Translation, Rotation} (Made explicit in Appendix A). These particular functions were chosen as they were topologically invariant (save Salt-Pepper, unlikely to have an effect). Parameters for the functions were chosen such that simultaneous application of all to an image would still yield a human-readable image.

Crossover for a Prey is single-point and fixed-length, occurring within the series of bits. Mutation flips a single bit in the series. This scheme was chosen as it appears to be the simplest example of a GA on a bit string, and closest to the "standard" method.

2.3. *Agent fitness and competitive co-evolution*

The relationship between Hunters and Prey in the CellNet Co-Ev system is defined through their interaction in the Genetic Algorithm. Both populations are initially spawned randomly. For each generation, each agent is evaluated against the entirety of the opposing population. Explicitly:

Let h be a member of the Hunter population H, p a member of the Prey population P. For each generation, each Hunter h attempts to classify each Prey p — let

$$classAttempt(h,p) = \begin{cases} 1; & h \text{ correctly classfies } p \\ 0.5; & h \text{ responds uncertain} \\ 0; & h \text{ incorrectly classifies } p. \end{cases} \tag{1}$$

Then the accuracy_{train} of a hunter h is

$$accuracy_{train}(h) = \frac{1}{p} \sum_{p \in P} classAttempt(h,p). \tag{2}$$

Note that $accuracy_{train}$ differs from the traditional definition of the accuracy of a classifier. Later, when discussing the Validation Accuracy of a Hunter, we shall use a different measure on an independent validation set of (non-camouflaged) images, im in I.

$$classAttempt_{validation}(h,im) = \begin{cases} 1; & h \text{ correctly classifies } im \\ 0; & \text{otherwise.} \end{cases} \tag{3}$$

Leading to the familiar measure of accuracy, which we shall call $accuracy_{validation}$.

$$accuracy_{validation}(h) = \frac{1}{I} \sum_{im \in I} classAttempt_{validation}(h,im). \tag{4}$$

Fitness of a hunter is defined as

$$fitness(h) = accuracy^2_{train}(h) - \alpha \cdot complexity(h) \tag{5}$$

where α is a system parameter designed to limit Hunter complexity, and $complexity(h)$ is the number of cells in Hunter h.

In contrast, the fitness of a Prey p is defined as

$$fitness(p) = \frac{1}{H} \sum_{h \in H} (1 - classAttempt(h, p)) \tag{6}$$

which is proportional to the inverse of fitness for Hunters.

As is evident from the relation between Hunters and Prey, the two populations exist in a state of Competitive Co-Evolution. The purpose of the addition of Camouflage functions to the database of images is two-fold:

(1) It artificially increases the size of the database of images provided, by creating subtle changes in existing images. Hence, the system has a broader palette of training data.
(2) The dynamics of the populations creates a more difficult problem for the Hunter population — not only do Hunters have to deal with an artificially large number of agents, it is precisely the transformation which they most often fail to recognise which will comprise the bulk of the next population of Prey. Hence, a Hunter population's weak points are emphasised.

3. Data and Analysis

The CellNet system was tested using the CEDAR database of handwritten numbers. Unlike previous experiments, only one pattern recognition task was attempted, although it is expected that scale-up to other handwritten languages, such as Indian or Chinese digits, is still possible. The system was configured to generate five binary hunters for each digit — these are labelled $\boldsymbol{h.x.y}$, where x is the digit number, and y an index from 0 to 4. Each hunter was trained using a base set of 250 training images, and tested via an independent set of 150 validation images.

All hunters were developed using identical system parameters, although training and validation images for each run were chosen randomly from a pool of 1,000 images. The hunters were placed in a genetic algorithm, using fitness-proportional selection and elitism (for definitions, refer to Mitchell's introductory text [9]). Parameters were a rate of crossover of 0.8, a rate of mutation of 0.02, a rate of merger of 0.02, and a rate of elitism of 0.1. The complexity penalty used in fitness was set to 0.0002. Prey were evaluated similarly — fitness-proportional selection, elitism of 0.1, crossover of 0.8 and mutation of 0.02.

Each run was executed for a maximum of 250 generations, outputting data regarding validation accuracy each 10 generations. A typical run may be seen in the evolution of the $h.0$ hunters, as illustrated in Fig. 6. Training and validation accuracies are very close, although validation accuracy tends to achieve slightly higher levels — this behaviour is typical of all digits. This is in contrast to previous experiments involving CellNet on handwritten characters, where overfitting

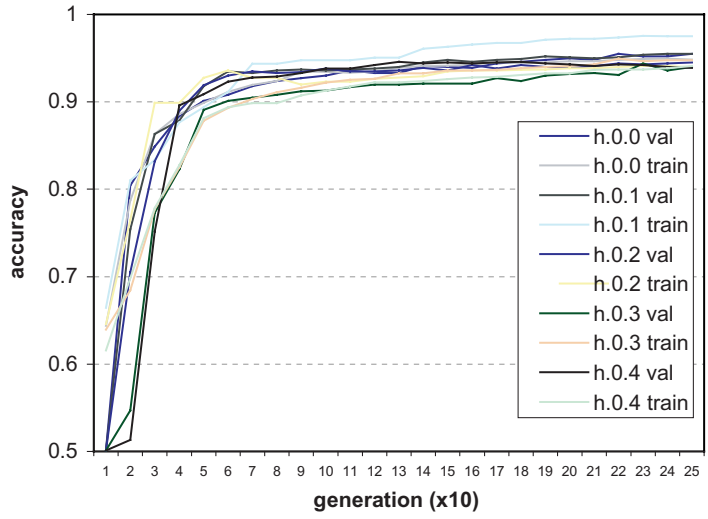

Fig. 6. Maximum training (*light lines*) and validation (*dark lines*) accuracies for the $h.0$ hunters.

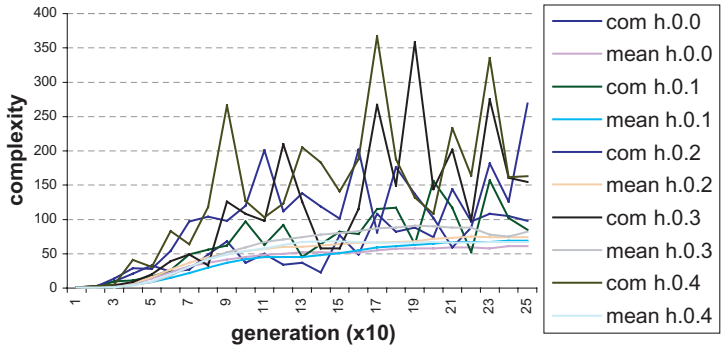

Fig. 7. Complexity of most fit agents (*dark lines*), and mean complexity (*light lines*) for the h.0 runs.

of approximately 2–3% was reported consistently.[6] It is also noted that in initial generations of the runs, overfitting is common, as it can clearly be seen that the training plots are more accurate than the validation plots; this initial bonus, however, disappears by generation 60, where the validation plots overtake. However, also in contrast to previous experiments, complexity is vastly increased — in the case of the zero digit, mean complexity jumps from approximately 35 cells to approximately 65 cells, while the complexity of the most accurate agent jumps from 40 cells to seemingly random oscillations in the range of 50 cells to 350 cells. Figure 7 shows the complexities of the most accurate agents and the mean for the $h.0$ runs.

Table 1. Maximum training and validation accuracies for the binary classifiers.

	h.dig.0		h.dig.1		h.dig.2		h.dig.3		h.dig.4		Mean		
	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Diff
0	0.951	0.955	0.977	0.955	0.946	0.945	0.951	0.944	0.941	0.946	0.953	0.949	+0.004
1	0.992	0.981	0.984	0.971	0.992	0.982	0.987	0.977	0.990	0.984	0.981	0.979	+0.002
2	0.906	0.906	0.935	0.932	0.895	0.904	0.881	0.896	0.906	0.920	0.905	0.912	−0.007
3	0.922	0.910	0.894	0.919	0.894	0.910	0.895	0.908	0.906	0.926	0.902	0.915	−0.013
4	0.944	0.941	0.972	0.967	0.957	0.962	0.956	0.962	0.957	0.952	0.957	0.957	+0.000
5	0.890	0.919	0.935	0.937	0.899	0.919	0.919	0.922	0.894	0.914	0.907	0.922	−0.015
6	0.914	0.941	0.925	0.940	0.923	0.953	0.945	0.917	0.931	0.923	0.928	0.935	−0.007
7	0.937	0.934	0.937	0.954	0.954	0.954	0.946	0.940	0.961	0.939	0.947	0.944	+0.003
8	0.900	0.914	0.933	0.918	0.932	0.939	0.875	0.905	0.914	0.931	0.911	0.921	−0.010
9	0.882	0.911	0.938	0.944	0.915	0.917	0.918	0.926	0.924	0.939	0.915	0.927	−0.012
											mean		**− 0.006**

Table 2. Percentage agreement in errors made by classifiers by digit.

Digit	0	1	2	3	4	5	**6**	7	8	9	Mean
Number of errors	38	22	62	58	34	80	58	55	73	29	60.9
Agreement	0.26	0.05	0.40	0.52	0.47	0.19	0.19	0.35	0.25	0.25	0.29

Table 1 shows the maximum training and validation accuracies for each binary hunter. The final columns compute the means for the validation and training accuracies for each class of hunter, and compare the difference. It is shown that the mean difference between training and validation data is −0.006, implying slight ***underfitting*** of the classifiers to the training data.

Finally, a series of experiments was undertaken regarding the classifications of the evolved binary classifiers. The scope of these experiments was the determination of the relative independence of the errors made by the classifiers when classifying images. Hence, our goal was a measure of the variance found between the errors of the hunters for each particular digit.

Each hunter evaluated a set of 300 previously unseen images — a note was made for each error. Each classifier for any particular digit then had an associated error list of images, These lists were contrasted, computing the total number of errors (for all 5 hunters) and the percentage of the list shared by two or more hunters. These results are shown in Table 2. It is evident that there is much variance between the errors made by the various hunters.

4. Conclusions

Relative to the original CellNet experiments, our augmented system performs admirably. The creation of binary classifiers is accomplished for each digit, showing little variance between results. This is contrasted against the original CellNet project's use of 30 runs to find a good classifier.

Additionally, the problem of over-fitting has been eliminated through the inclusion of competitive co-evolution. The inclusion of a genome for patterns and camouflage functions for diversification has resulted in a more difficult problem for the classifiers, increasing overall performance.

Finally, it has been demonstrated that although the reliability of the system's ability to generate classifiers has been improved, the produced classifiers' error sets are largely independent. This matter is crucial for the creation of multi-classifiers (combinations of the binary classifiers to form a single multiple-class recogniser), a step which a practitioner may wish to take. The independence of the error rates of the classifiers implies that several hunters for each class may be used in a bagging or bootstrapping technique, methods which are expected to improve the accuracy of the overall multi-classifier.

These results represent a significant step forward for the goal of an autonomous pattern recogniser: competitive co-evolution and camouflage is expected to aid in the problem of over-fitting and reliability without expert tuning, and also in the generation of a larger and more diverse data set.

Appendix A: Camouflage Functions

Translation Left – similarily for Up, Down and Right

```
translation = translationRatio * width;
If (translation > left most pixel width value)
     then translation = left most pixel width value
Pass over the pixels (j, j)in the image
If (pixels stays within the bound after translation)
     set pixels(i, j) to pixels(i, j +translation)
else
     set pixel(i, j) = 0
```

Rotation CounterClockwise – similarily for Clockwise

```
Create an image with pixels(x, y) set to 0
Pass over the pixels (i, j) of the original image
If (pixels (i, j) not equal to 0)
     set x = (int)(j*cos(rotationAngel*PI/180) - (height-1)
             *sin(rotationAngel*PI/180));
     Set y = height-1-(int)(j*sin(rotationAngel*PI/180)
           +(height-1-i)*cos(rotationAngel*PI/180));
if (x, y) is within image bounds
     set pixels(x, y) = 1
else
     set pixels(x, y) = 1
```

Salt and Pepper

```
Set pepper ratio to pr = 1.0- noisePercent/100.0;
Generate random number between -1.0 and +1.0
random = 2.0*(rand()-RAND{_}MAX/2.0)/RAND{_}MAX;
if (pixel not equal to 0)
      if (random > pr)
             set pixels(i, j) to 1
```

Scaling

```
Create temporary image with pixels(x, y) set to 0
Pass over the pixels (i, j) of the original image
if (pixels(i, j) not equal to 0)
    if (s{_}factor*i<height)
        if (s{_}factor*j<width)
            set temporary image pixels(s_factor*I,
                s_factor*j) = 1.
```

References

1. Y. Al-Ohali, M. Cheriet and C. Suen, Databases for recognition of handwritten Arabic cheques, *Pattern Recogn.* **36** (2003) 111–121.
2. S. P. Brumby, J. Theiler, S. J. Perkins, N. R. Harvey, J. J. Szymanski, J. J. Bloch and M. Mitchell, Investigation of image feature extraction by a genetic algorithm, in *Proc. SPIE 3812* (1999) 24–31.
3. G. Fung, J. Liu and R. Lau, Feature selection in automatic signature verification based on genetic algorithms, *ICONIP'96* (1996), pp. 811–815.
4. D. E. Goldberg, B. Korb and K. Deb, Messy genetic algorithms: motivation, analysis, and first results, *Complex Syst.* **3**(5) (1989) 493–530.
5. S. Gunter and H. Bunke, Optimization of weights in a multiple classifier handwritten word recognition system using a genetic algorithm, *Electronic Letters on Computer Vision Image Analysis* **3**(1) (2004) 25–41.
6. Z. Guo and R. Uhrig, Using genetic algorithms to select inputs for neural networks, in *Proc. COGANN'92* (1992) 223–234.
7. N. Kharma, T. Kowaliw, E. Clement, C. Jensen, A. Youssef and J. Yao, Project CellNet: evolving an autonomous pattern recogniser, *Int. J. Pattern Recogn. Artif. Intell.* **18**(6) (2004).
8. M. Kudo and J. Sklansky, A comparative evaluation of medium- and large-scale features selectors for pattern recognition, *Kybernetika* **34**(4) (1998) 429–434.
9. M. Mitchell, *An Introduction to Genetic Algorithms* (MIT Press, 1998)
10. A. Moser, A distributed vertical genetic algorithm for feature selection, *ICDAR '99* (1999), pp. 1–9 (late submissions' supplement).
11. M. Rizki, M. Zmuda and L. Tamburino, Evolving pattern recognition systems, *IEEE Trans. Evol. Comput.* **6**(6) (2002) 594–609.
12. R. Sarker, J. Kamruzzaman and C. Newton, Evolutionary optimization (EvOpt): a brief review and analysis, *Int. J. Comput. Intell. Appl.* **3**(4) (2003) 311–330.
13. D. Shi, W. Shu and H. Liu, Feature selection for handwritten Chinese character recognition based on genetic algorithms, *IEEE Int. Conf. Syst. Man Cybernet.* **5** (1998) 4201–4206.

14. W. Siedlicki and J. Sklansky, On automatic feature selection, *Int. J. Pattern Recogn.* **2** (1998) 197–220.
15. F. W. M. Stentiford, Automatic feature design for optical character recognition using an evolutionary search procedure, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-7**(3) (1985) 349–355.
16. H. Vafaie and K. De Jong, Robust feature selection algorithms, in *Proc. IEEE Int. Conf. Tools Artif. Intell.* (1993) 356–363.
17. C. Veenman, M. Reinders and E. Backer, A cellular co-evolutionary algorithm for image segmentation, *IEEE Trans. Image Process.* **12**(3) (2003) 304–316.