CellNet Co-Ev: Evolving Better Pattern Recognizers Using Competitive Co-evolution

Taras Kowaliw[1], Nawwaf Kharma[2], Chris Jensen[2], Hussein Moghnieh[2], and Jie Yao[2]

[1] Computer Science Dept., Concordia University, 1455 de Maisonneuve Blvd. W.
Montreal, Quebec, Canada H3G 1M8
taras.kowaliw@utoronto.ca

[2] Electrical & Computer Eng. Dept., Concordia University, 1455 de Maisonneuve Blvd. W.
Montreal, Quebec, Canada H3G 1M8
kharma@ece.Concordia.ca

Abstract. A model for the co-evolution of patterns and classifiers is presented. The CellNet system for generating binary classifiers is used as a base for experimentation. The CellNet system is extended to include a competitive co-evolutionary Genetic Algorithm, where patterns evolve as well as classifiers; This is facilitated by the addition of a set of topologically-invariant camouflage functions, through which images may disguise themselves. This allows for the creation of a larger and more varied image database, and also artificially increases the difficulty of the classification problem. Application to the CEDAR database of hand-written characters yields both an increase in reliability and an elimination of over-fitting relative to the original CellNet project.

1 Introduction and Review

Our goal is to build a system for evolving recognizers for arbitrary problem spaces utilizing as little expert input as possible. This is an ambitious goal, requiring much additional work. This paper describes a step toward that goal: Using the CellNet system [Kharma et al], we implement co-evolution utilizing a set of camouflage functions for the given pattern database. Our results will be shown to contribute towards this overall goal.

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.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 1090–1101, 2004.

1.1 Feature Selection

Siedlecki in [12] is the first paper to suggest the use of GA for feature selection. Ten years later, Kudo [6] demonstrates the superiority of GA over a number of traditional search & optimization methods, for sets with a large number of features. Vafaie [13] shows that a GA is better than Backward Elimination in some respects, including robustness. Guo [4] uses a GA for selecting the best set of inputs for a neural network used for fault diagnosis of nuclear power plant problems. Moser [8] proposes two new architectures of his own: VeGA, which uses parallelism to simultaneously evaluate different sets of features (on one computer), and DveGA, which utilizes an adaptive load balancing algorithm to use a network of heterogeneous computers to carry out the various evaluations. Shi [10] applies a GA to the selection of relevant features in hand-written Chinese character recognition. Fung [2] uses a GA to minimize the number of features required to achieve a minimum acceptable hit-rate, in signature recognition. Yeung [14] succeeds in using a GA in tuning four selectivity parameters of a Neocognitron, which is used to recognize images of hand-written Arabic Numerals.

1.2 Feature Creation

The first paper to speak about feature creation appears to be [Stentiford, 1985]. 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 good; the error rate, was 1.01%, and was obtained using 316 nearest-neighbour classifiers automatically generated by the evolutionary system. More recently, a group of researchers including Melanie Mitchell used evolution to create features for use in analyzing satellite and other remote-sensing captured images [1]. Specifically, the researchers defined a genetic programming scheme for evolving complete image processing algorithms for, for example, identifying areas of open water. Their system called GENIE successfully evolved algorithms that were able to identify the intended ground features with 98% accuracy.

1.3 Evolvable Pattern Recognition Systems

It was natural for researchers to move from feature creation to attempt to create whole pattern recognition systems. After all, the hardest part of successful pattern recognition is the selection/creation of the right set of features. Indeed, the first such systems seem to have focused on the evolution of features or feature complexes, which are useful for classification. Two systems described below do this. CellNet [5] blurs the line between feature selection and the construction of binary classifiers out of these features. HELPR [9] also evolves feature detectors, but the classification module is completely separate from the feature extraction module. Other differences exist, but, both attempts are the only systems, that we know of, that aim at using artificial evolu-

tion to synthesize 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 specialized knowledge to operate the system. CellNet will eventually achieve this through the inclusion of methodologies for the simultaneous evolution of features and selectors (Cells and Nets); However, at present, 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 deconstructing images using techniques inspired by Pre-Attentive Vision.

However, at present, CellNet is an evolvable (existing) feature selector and classifier synthesizer, which uses a specialized genetic operator, Merger (a.k.a Cooperative Co-Evolution). Merger is an operator, somewhat similar to that used in MessyGAs [3], designed to allow the algorithm to search a larger space than a regular GA. CellNet is cast as a general system, capable of self-adapting to many hand-written alphabets, currently having been applied to both Latin and Indian alphabets – as such, a more powerful engine is required than in preceding approaches.

HELPR is composed of a feature 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 current 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. The transformation network utilizes a set of morphological operations with structuring elements in addition to a small number of arithmetic operations. As such, a transformation network (tn) represents a mixed expression which transforms an n-element digitized 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 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.

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 realized. This is why the following set of problems must be dealt with the:

1. Provision of a Feature Creation-Selection mechanism that is suited to a large class of classification problems. This is to eliminate the need for re-engineering, every time a different application domain is targeted.
2. Elimination of the many parameters that need to be set by the user before the system is run. These include items such as: probability distributions, probability values, size of population, number of detectors (feature extractors) and so on.

3. Testing any resulting system against a diverse set of pattern databases (and not just Radar signatures or Arabic Numerals), and doing so, without subjecting the system to the slightest amount of re-configuration.

This paper may be viewed as a step towards the realization 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.

2 Hunters and Prey

2.1 Hunters

CellNet hunters were first introduced in [5], 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 representation for pattern recognizers, 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 normalized feature functions; The functions (all applied to thinned figures) used by CellNet CoEv are {parameterized histograms, central moments, Fourier descriptors, Zernike moments, normalized width (of a bounding box), normalized height, normalized length, number of terminators, numer of intersections}, as in [5]. 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 the Feature Space, and is easily generalized.

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 recognize 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, organized 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 organizes 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, a chromosome 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.

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	$Cell_3$	

A Hunter is a Net – that is, a Hunter is an organized 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 organization.

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 the non-primary class is "NOT-ONE".

Our first Hunter, A_1, consist of a single cell in a single chromosome – It is illustrated in Fig. 2.1.3.1. 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 <x,y> coordinate in this space, and hence may be drawn in a unit square. Agent A_1 may be viewed as a statement which partitions Feature Space into three disjoint sets – this is also illustrated in

Fig. 2.1.3.1. A second Hunter, A2, is illustrated in Fig. 2.1.3.2; This agent's partition of the same feature space is also illustrated.

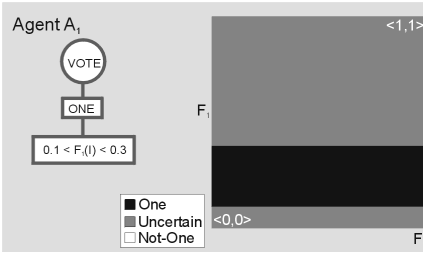

Fig. 2.1.3.1. Agent A_1 and its partition of Feature Space

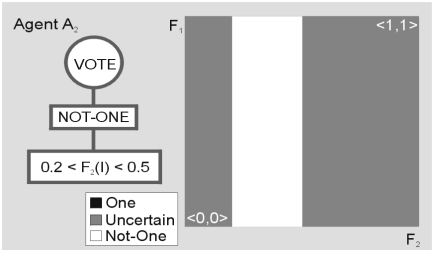

Fig. 2.1.3.2. Agent A_2 and its partition of Feature Space

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 initialized. 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, Mutation and Elitism. 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.

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 re-consider agents A_1 and A_2 of Section 3.1.3, considering their children through the Merger operator.

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. 2.1.5.1, Horizontal Merger may be viewed as the refinement of a partition created by two Chromosomes.

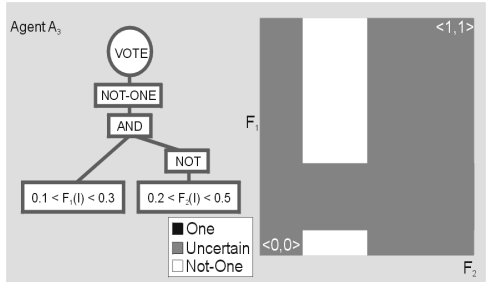

Fig. 2.1.5.1. Agent A_3 and its partition of Feature Space

In contrast, consider the Vertical Merger of these same two Hunters, producing agent A_4 – in this case, the Chromosomes are combined directly under the Vote mechanism. As shown in Fig. 2.1.5.2, Vertical Merger may be loosely be viewed as the union of the partitions generated by two Chromosomes.

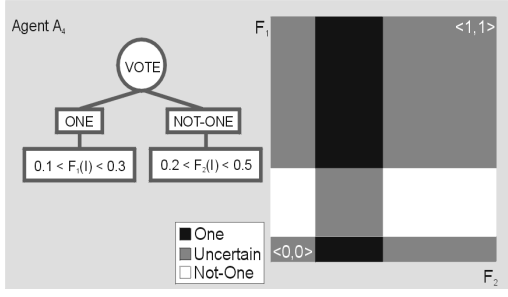

Fig. 2.1.5.2. Agent A4 and its partition of Feature Space

2.2 Prey

In a divergence from the methodology in [5], the second population in the CellNet Co-Ev system consist of Prey; Prey are primarily images, drawn from the CEDAR database of hand-written 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}. 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.

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

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

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*.

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

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

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

where alpha 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 recognize which will comprise the bulk of the next population of Prey. Hence, a Hunter population's weak points are emphasized.

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 labeled ***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 1000 images. The hunters were placed in a genetic algorithm, using fitness-proportional selection and elitism. 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. 3.1. 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 hand-written characters, where overfitting of approximately 2-3% was reported consistently [5]. 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. Fig. 3.2 shows the complexities of the most accurate agents and the mean for the h.0 runs.

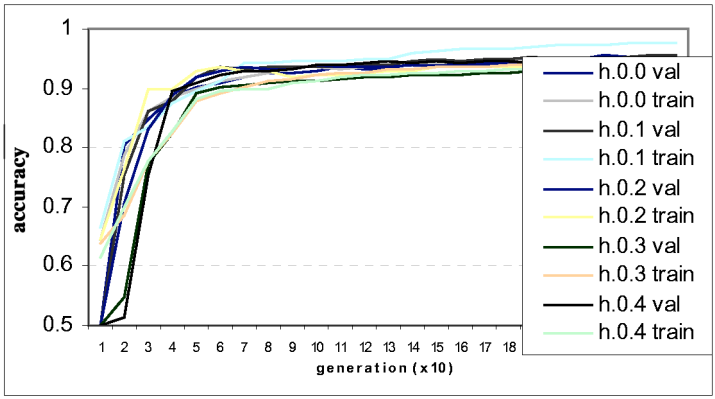

Fig. 3.1. Maximum training (*light lines*) and validation (*dark lines*) accuracies for the h.0 hunters.

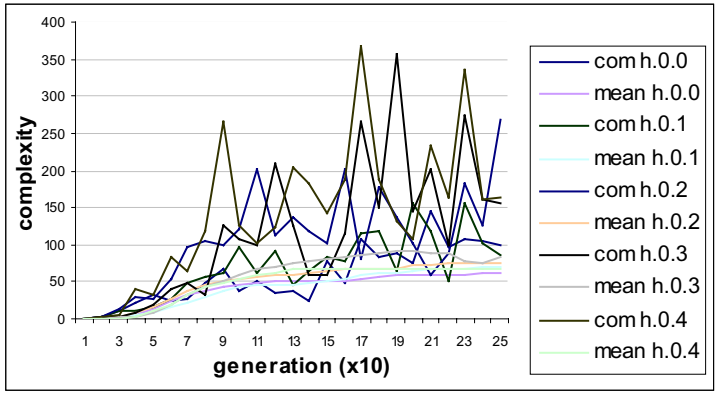

Fig. 3.2. Complexity of most fit agents (*dark lines*), and mean complexity (*light lines*) for the h.0 runs.

Table 3.1 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.

Table 3.1 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		
dig	**train**	**valid**	**train**	**valid**	**train**	**valid**	**train**	**valid**	**train**	**valid**	**train**	**valid**	**diff**
0	.951	.955	.977	.955	.946	.945	.951	.944	.941	.946	.953	.949	+.004
1	.992	.981	.984	.971	.992	.982	.987	.977	.990	.984	.981	.979	+.002
2	.906	.906	.935	.932	.895	.904	.881	.896	.906	.920	.905	.912	-.007
3	.922	.910	.894	.919	.894	.910	.895	.908	.906	.926	.902	.915	-.013
4	.944	.941	.972	.967	.957	.962	.956	.962	.957	.952	.957	.957	+.000
5	.890	.919	.935	.937	.899	.919	.919	.922	.894	.914	.907	.922	-.015
6	.914	.941	.925	.940	.923	.953	.945	.917	.931	.923	.928	.935	-.007
7	.937	.934	.937	.954	.954	.954	.946	.940	.961	.939	.947	.944	+.003
8	.900	.914	.933	.918	.932	.939	.875	.905	.914	.931	.911	.921	-.010
9	.882	.911	.938	.944	.915	.917	.918	.926	.924	.939	.915	.927	-.012
											mean		-.006

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 4.2; It is evident that there is much variance between the errors made by the various hunters.

Table 3.2. Percentage agreement in errors made by classifiers by digit

Digit	**zero**	**one**	**two**	**three**	**four**	**five**	**six**	**seven**	**eight**	**nine**	**mean**
number of errors	38	22	62	58	34	80	58	55	73	129	60.9
agree-ment	0.26	0.05	0.40	0.52	0.47	0.19	0.19	0.35	0.25	0.25	0.29

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 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 recognizer), 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 recognizer: 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.

References

1. Brumby, S.P., Theiler, J., Perkins, S.J., Harvey, N.R., Szymanski, J.J., Bloch J.J., Mitchell, M. Investigation of Image Feature Extraction by a Genetic Algorithm. In: Proc. SPIE 3812 (1999) 24—31.
2. Fung, G., Liu, J., & Lau, R. Feature Selection in Automatic Signature Verification Based on Genetic Algorithms, ICONIP'96, (1996) pp. 811-815
3. Goldberg, D. E., Korb, B. and Deb, K., Messy genetic algorithms: Motivation, analysis, and first results. In Complex Syst., vol. 3, no. 5, pp. 493-530, (1989).
4. Guo, Z., & Uhrig, R. Using Genetic Algorithms to Select Inputs for Neural Networks, Proceedings of COGANN'92. (1992) pp. 223-234
5. Kharma, N., Kowaliw, T., Clement, E., Jensen, C., Youssef, A., Yao, J. Project CellNet: Evolving an Autonomous Pattern Recognizer, Int. Journal of Pattern Recognition and Artificial Intelligence (In publication). http://www.ece.concordia.ca/~kharma/ResearchWeb/CellNet/cn.pdf
6. Kudo, M., & Sklansky, J. A Comparative Evaluation of Medium- and Large-Scale Features Selectors for Pattern Recognition in Kybernetika, V.34, No. 4, (1998) pp. 429-434.
7. Mitchell, M. An Introduction to Genetic Algorithms, MIT Press (1998)
8. Moser, A. A Distributed Vertical Genetic Algorithm for Feature Selection, ICDAR '99. pp1-9 (late submissions' supplement) (1999).
9. Rizki, M., Zmuda, M., Tamburino, L., Evolving pattern recognition systems, IEEE Transactions on Evolutionary Computation, volume 6, issue 6 , (2002) pp. 594-609
10. Shi, D., Shu, W., & Liu, H. Feature Selection for Handwritten Chinese Character Recognition Based on Genetic Algorithms, IEEE Int. Conference on Systems, Man, and Cybernetics, V.5, (1998) pp. 4201-6.
11. Siedlicki, W., & Sklansky, J. On Automatic Feature Selection, International Journal of Pattern Recognition, 2:197-220 (1998).
12. Stentiford, F. W. M. Automatic Feature Design for Optical Character Recognition Using an Evolutionary Search Procedure, IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. PAMI-7, No. 3, (1985) pp. 349- 355.
13. Vafaie, H., & De Jong, K., Robust Feature Selection Algorithms, Proceedings of the IEEE International Conference on Tools with Artificial Intelligence. (1993) pp. 356-363.
14. Yeung, D., Cheng, Y., Fong, H., & Chung, F., Neocognitron Based Handwriting Recognition System Performance Tuning Using Genetic Algorithms, IEEE Int. Conference on Systems, Man, and Cybernetics, V.5, (1998) pp. 4228-4233.