# A Combined Neural and Genetic Learning Algorithm

Lampros Tsinas and Bernd Dachwald

Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
85577 Neubiberg, Germany

## Abstract

*Neural networks and genetic algorithms are well-known representatives of learning procedures. In this paper a hybrid procedure, which combines both concepts, is introduced. Its functionality is presented on a typical pattern recognition problem.*

## Introduction

An optical sensor system (CCD-camera) constitutes the basis for a successful recognition of important features in the restricted environment of an autonomous mobile robot (ATHENE) and for the behavior-based navigation for this "intelligent" technical system. ATHENE provides a good platform for the development of learning, vision-equipped robots [Wershofen, Graefe 93]. Such a representative of the nearest generation of robots, should be able to process its multisensorial input data (in this case only video data) at a higher "intelligence" level. Furthermore, it should be able to realize and recognize important parts of its "world section". Analogous to human abilities, it should also be able to learn some (or all) of the information processing steps (feature extraction, object recognition, navigation, situation perception, etc.).

The backpropagation algorithm (BP) [Rumelhardt et al. 86] is at present the most popular learning algorithm for multilayered feedforward neural networks. It constitutes the basis of many investigations, further developments and applications ([Pomerleau 92], [Tsinas, Graefe 93]). Other network-models, like ART [Grossberg 82] and LVQ [Kohonen 84] have also proved their abilities. In the last three decades - inspired by Darwin's theory of evolution and parallel to the development of neuro-science - the development of genetic algorithms (GAs) proceeded [Holland 92]. The basic idea of these algorithms is, that in a set of individuals exposed to selection, fitter individuals produce more offspring than less fit individuals, and consequently make their way in the struggle for life. Applications of GAs cover a wide spectrum ([Schulze-Kremer 93], [Jakob et al. 92]). A combination of both categories of learning methods - exploiting the specific advantages of BP and GAs - may result in a more powerful, more robust and probably faster learning procedure. First results in this direction are presented in [Petridis et al. 92].

We have examined the efficiency of some learning algorithms for the automatic evaluation of visual information. Therefore a hybrid method, consisting of BP and a genetic algorithm, was developed. This learning algorithm (CoNGA: Combined Neural and Genetic Algorithm) was tested on a typical classification problem and compared with BP. The learning algorithm, the learning task and some results of our experiments are described below.

0-7803-1899-4/94 $4.00 

# CoNGA

CoNGA (Combined Neural and Genetic Algorithm) is a learning algorithm which combines distinct learning methods. In recent years many experimentation was carried out about the training of neural networks by genetic algorithms and numerous papers present advancements and results ([Montana, Davis 89], [Petridis et al. 92], [Whitley, Hanson 89]). In contrast - but not antagonism - to the known work, the following features can be found within CoNGA in combination:

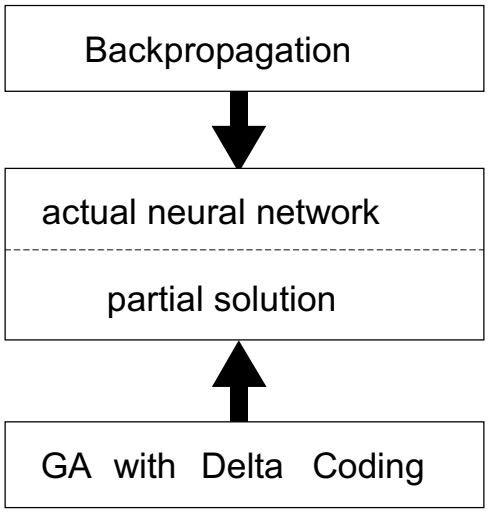


**Fig. 1**
The hybrid nature of CoNGA.

a) In contrast to the traditional binary encoding, the parameters to optimize are encoded as *floating-point strings*.
b) Instead of the generally used generational reproduction, where all individuals of the population reproduce themselves at every time step, the *steady-state reproduction-technique* is preferred and applied. Using this technique, at one time step only two individuals create offspring, and therefore less expenditure of computation is required.
c) In contrast to the ordinarily used fitness proportional selection-technique we allocate the reproductive trials according to the rank of the individuals in the population (*linear ranking selection*) [Whitley 89].
d) The search strategy of *Delta Coding* [Whitley et al. 91] is employed and modified for floating-point strings. Delta Coding is based on the idea that a string can also express a distance away from some previous partial solution. This partial solution is the best known solution so far. Using Delta Coding, at any time only a dynamically selected subspace (a so-called *n*-dimensional hypercube, where *n* is the number of parameters) of the whole search space - constructed around the most recent partial solution - is explored. After the population has converged to one solution, it is reinitialized and the new solution is stored and taken as the center of the new (reduced or enlarged) hypercube.
e) *One-point crossover*, *uniform crossover* [Syswerda 89] and *crossover nodes* [Montana 91] are used as genetic operators. The application probabilities of the genetic operators are dynamically adapted during the computation [Davis 91]. Because of the reinitialisation of the population, every time it has converged, a mutation-operator is not necessary [Whitley et al. 91].
f) *The GA* described by a) to e) *is combined with the standard backpropagation algorithm* (Fig. 1). During the training CoNGA can switch from BP to the GA (and back). This should be done, if the learning effort of the current method stagnates (low convergence) and shows no significant improvement of the learning error in a definite number of time steps (i.e. if it gets stuck into a local minimum). This allows the algorithm to escape from local minima, because the BP-to-GA-switch spans the search space wider (hopefully wide enough to escape the local minimum).

The basic requirement for the application of the GA is the mapping of the learning status of the neural weight-architecture to a string. The learning status is completely described by the weights and biases of the neurons. For that purpose a defined part of the string is assigned to each neuron in the network. This part of the string stores the weights from the respective neuron to all neurons of the previous neuron layer and the bias of the respective neuron. This mapping results in a string that is identical to the learning status of a neural network.

## Learning paradigm

A typical task in image processing is the recognition of objects. A even more complex task is to identify objects which have different shapes or views. In first approximation such object-views are handled as independent objects. One example – chosen because of its view-varieties only – is the recognition of dice-views. A die has six faces, but because the views of the faces with two, three and six dots are not invariant in 90(-rotation, we have to consider nine different views which have to be recognized clear as "Eigenviews" (Fig. 3). For the recognition task we have at first computed the greyvalue-gradients over five image-rows of digitized images of the different dice-views. We took 35 values per row which would be enough for the classification. In some other experiments we took larger sized greyvalue-gradient sets. But sets with more than 35 values per row (or more than five rows) include redundant information. To keep the computation time at a moderate level, we have chosen the smallest size for the feature set. That leads to a feature vector with 35(5 = 175 components. The problem was examined with BP, the plain GA and with the hybrid "trainer" CoNGA.

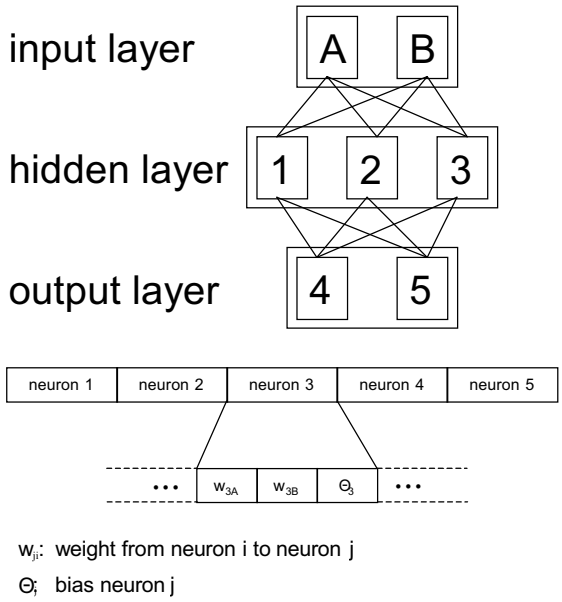


**Fig. 2**
The coding of a neural network as a string.

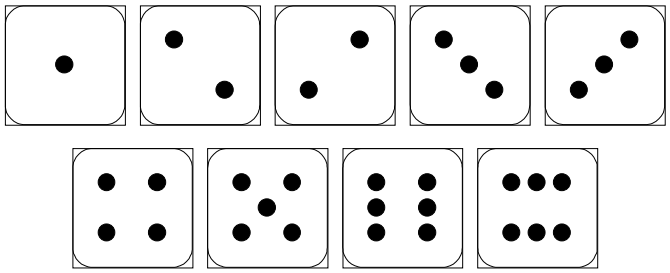

**Fig. 3**
The nine views of the six faces of a die.

For the classification of the nine different object-views a neural network with 175 input neurons, one hidden layer with ten neurons and one output layer with six neurons was selected. Each output neuron matches to one of the faces (including the double views). During the training

175(10 + 10(6 = 1810 weights and 10 + 6 = 16 biases

have to be "learned". The input values (gradients) range from 0 to 129. The learning goal is, that only the correct output neuron is active (output = 1.0) and all other output neurons are inactive (output = 0.0). The database for the training consists of nine example-views.

Good parameters for BP were selected through trial-and-error (learning rate s = 0.4 and momentum μ = 0.4). The populations contain thirty individuals for the plain GA-training and ten individuals for the CoNGA-training. Using (different) random initialization-values for the weight-structure of the network we made five independent training runs. The training results are pictured in Fig. 4, 5 and 6, respectively for the BP-, the plain GA- and the CoNGA-training.

Fig. 4 and 5 show clearly, that the plain GA converges faster than BP in all five program runs. Because of time-claim only five independent training runs were made and therefore we cannot guarantee that anyone of the learning procedures may exhibit a better learning quality in some other run, but the results are characteristic for each training. On one run the hybrid CoNGA-algorithm was able to reduce the learning error to a value less than 10-6 after only 7500 time steps. It is very unlikely, that randomly selected initial points (individuals) in the hypercube centered around the best BP-solution exhibit a better performance. Due to that, every BP-to-GA-switch leads to a peak in the curve (Fig. 6). (At present) non-optimum switch-criteria from the BP- to the GA-part of CoNGA and back cause CoNGA to be outperformed by the plain GA in the other four runs. The results show, that CoNGA can be a powerful, sensible extension or supplementation to other learning methods.

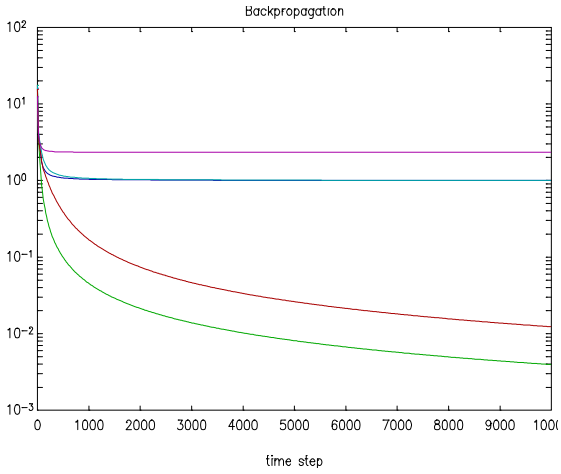


**Fig. 6**
Backpropagation-training.

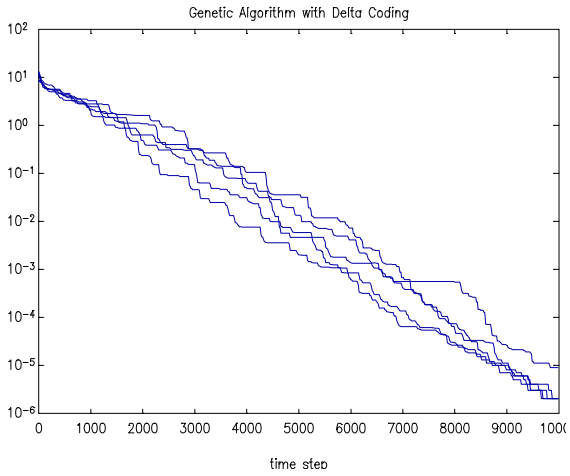


**Fig. 5:**
GA-training.

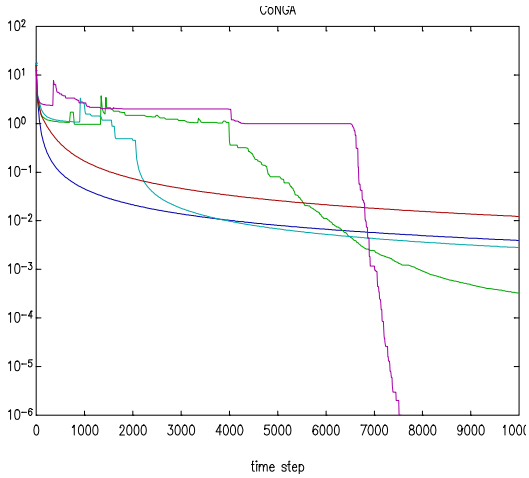


**Fig. 4**
CoNGA-training.

## Conclusions

A new hybrid learning method for network-like structures was described. It consists of neural learning by the backpropagation algorithm and applies evolutionary, genetic operators. The learning behavior of the algorithm was tested on a characteristical image processing example. It was

able to prove its potential and allows to assume, that it may exhibit also good performance for other than image processing problems or signal processing problems in general.

The development and the experiments demonstrate, that GAs (and CoNGA) have to be further examined. Clear statements about the optimal switch-criteria from BP to the GA (and back) cannot be made at the time. We think of an automatic selection of the (much enough) learning parameters. But it is ambiguous, how that could be realized. Sure, one possibility is any kind of trial-and-error method, perhaps assisted by some knowledge-based and problem-driven adaptation mechanism. The way from the labs to the industries departments is still long.

# References


**Davis, L. (1991)**: Handbook of Genetic Algorithms. Van Nostrand Reinhold, 1991.

**Goldberg, D.E. (1989)**: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, 1989.

**Grossberg, S. (1982)**: The Adaptive Brain. Vols. I and II. Reidel Press, 1982.

**Holland, J.H. (1992)**: Adaption in Natural and Artificial Systems. MIT Press, 1992.

**Jakob, W.; Gorges-Schleuter, M.; Blume, C. (1992)**: Application of Genetic Algorithms to Task Planning and Learning. In Parallel Problem Solving from Nature, 2, Männer, R. and Manderick, B. (eds.). Elsevier Science Publishers B.V., 1992, pp 291-300.

**Kohonen, T. (1984)**: Self-Organization and Associative Memory. 2nd ed., Springer-Verlag, 1984.

**Michalewicz, Z. (1992)**: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 1992.

**Montana, D.J.; Davis, L. (1989)**: Training Feedforward Neural Networks Using Genetic Algorithms. In Proceedings of the 1989 International Joint Conference on Artificial Intelligence. Los Altos, 1989. Morgan Kaufmann Publishers, pp 762-767.

**Petridis, V.; Kazarlis, S.; Papaikonomou, A.; Filelis, A. (1992)** A Hybrid Genetic Algorithm for Training Neural Networks. In Aleksander, I., Taylor, J. (eds.), Artificial Neural Networks 2. North Holland, 1992, pp 953-956.

**Pomerleau, D. A. (1992)**: Progress in Neural Network-based Vision for Autonomous Robot Driving. In Proceedings of the IEEE Symposium Intelligent Vehicles '92. Detroit, pp 391-396.

**Rumelhardt, D. E.; Hinton, G. E.; Williams, R. J. (1986)** Learning Internal Representations by Error Propagation. In Parallel Distributed Processing. MIT Press, 1986.

**Schulze-Kremer, S. (1993)**: Genetic Algorithms for Protein Tertiary Structure. In Machine Learning: ECML-93, Vienna, 1993; Brazdil, P.L. (ed.), Springer Verlag, Lecture Notes in Artificial Intelligence 667, pp 262-279.

**Syswerda, G. (1989)**: Uniform Crossover in Genetic Algorithms. In Schaffer, J.D., (ed.), Proceedings of The Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, 1989, pp 2-9.

**Tsinas, L.; Graefe, V. (1993)**: Coupled Neural Networks for Real-time Road and Obstacle Recognition by Intelligent Road Vehicles. In International Joint Conference on Neural Networks, IJCNN '93. Nagoya, October 1993.

**Wershofen, K.-P.; Graefe, V. (1993)**: Ein sehender mobiler Roboter als Experimentierplattform zur Erforschung des maschinellen Lernens. In 9. Fachgespräche über Autonome Mobile Systeme, AMS '93. München, October 1993, pp 115-126.

**Whitley, D. (1989)**: The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation Of Reproductive Trials is Best. In Schaffer, J.D., (ed.), Proceedings of The Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, 1989, pp 116-123.

**Whitley, D.; Hanson, T. (1989)**: Optimizing Neural Networks, Using Faster, More Accurate Genetic Search. In Schaffer, J.D., (ed.), Proceedings of The Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, 1989, pp 391-397.

**Whitley, D.; Mathias, K.; Fitzhorn, P. (1991)**: Delta Coding: An Iterative Search Strategy for Genetic Algorithms. In Belew, R.; Booker, D., (eds.), Proceedings of The Fourth International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, 1991, pp 77-84.