A Neural Network that uses Evolutionary Learning

Mario Köppen, Martin Teunis, Bertram Nickolay

Abstract— **This paper proposes a new neural architecture (Nessy) which uses evolutionary optimization for learning. The architecture, the outline of its evolutionary algorithm and the learning laws are given. Nessy is based on several modifications of the multilayer backpropagation neural network. The neurons represent genes of evolutionary optimization, refered to as solutions. Weights represent probabilities and are used for selectioning. The training value of the output layer is set to Zero, the theoretical limit of every cost-oriented optimization, and the crossover operator is replaced by a transduction operator. Mutation is used as usual. Nessy algorithm can be characterized as individual evolutionary algorithm, but as a neural network too. It was designed for image processing applications. A short example is presented, where the discriminative feature of two images is succesfully detected by the proposed evolutionary neural network.**

Keywords— **Neural Learning, Evolutionary Learning Neural Networks, Optimization, Image Processing.**

I. Introduction

THE combination of neural networks and evolutionary computation has been intensively researched in the past years. The main goal of these studies was to adapt the topology of a neural network by a genetic algorithm (GA) that encodes a neural network structure into an individual gene. To name a few of these works: [1], [2] and [3]. The results of these studies are encouraging enough to consider these Neuro-GA systems as a reliable approach for the neural network design. However, this approach offers some disadvantages and narrows the application fields of neural networks. One thing to consider is the use of the neural network training or generalization error as a fitness measure for the individual neural network. This prevents from the estimation of the capabilities of a neural network due to refined learning strategies. Also, Neuro-GA requires a complicated and time consuming data management. It factorizes the effort to adapt a population of individual genes with the effort to train a neural network.

One key idea to overcome these difficulties is to break up the whole learning task into smaller parts and synthesize the neural network from its well-adapted components ([4]). This differs from Neuro-GA in one important aspect: the neural network is not only taken as an individual of a GA, but the GA is modified for the spezial purpose of neural network structural adaptation. Another work from the same author ([5]) went further and gave a new approach by the incorporation of evolutionary learning into a neural network.

In comparison with Neuro-Fuzzy systems this approach has not been fully considered in recent works. Neuro-Fuzzy systems modify neural network structures and learning rules in a manner which results in a fuzzy control performed by these networks. The neural network adopts the fuzzy inference and applies its learning rule to the adaptation of fuzzy rules and/or membership functions.

This paper proposes a new neural network architecture which adopts evolutionary learning. The main ideas behind this approach are: base architecture is the multilayer backpropagation network (MBPN) with three layers, but its weights are regarded as probabilities; these probabilities are applied, not only used for calculations; the main design principle is an individual evolutionary algorithm (IEA), as defined in [5]; and the crossover operator is replaced by a transduction operator. The proposed architecture (Neural Evolutionary Strategy System - Nessy) is outlined in section II, some properties are explained in section III. To learn about the learning abilities of Nessy, section IV gives an optimization for a pattern recognition problem. Finally, section V offers some conclusions and possible modifications.

II. Nessy-Architecture

Nessy is composed of three layers: a solution layer, a generation layer and an output layer (see figure 1). The solution layer maintains a set of solutions for the optimization task as in conventional evolutionary learning. Every solution consists of a gene. The data type of the elements of a gene is not restricted to bits, but all genes must have the same size. The generation layer is of the same size as the number of elements of the gene of a solution. The state of every neuron of the generation layer is a pointer to a solution of the solution layer. The output layer has only one neuron, and it holds the training value Zero.

Solution layer and generation layer are fully connected, every connection has a weight assigned to it. These weights are regarded as probabilities. The symbol p_{ij} stands for the weight of the connection from the generation layer neuron i to the solution layer neuron j. Generation layer and the only output neuron are also connected, but no special weights are assigned.

The interface to the optimization problem, as in conventional evolutionary learning, has three parts: the calculation of the fitness of a solution, the initialization of the solutions and (optional) management of mutation. Once initialized, Nessy works autonomously. The outline of Nessy algorithm is as follows:

1. Every neuron of the generation layer randomly selects

M.Köppen, M. Teunis and B. Nickolay are with the Fraunhofer-Institute for Production Systems and Design Technology (IPK), Pascalstr. 8-9, 10587 Berlin, Germany. E-mail of corresponding author: mario.koeppen@ipk.fhg.de.

Camera-ready paper for the ICEC'97

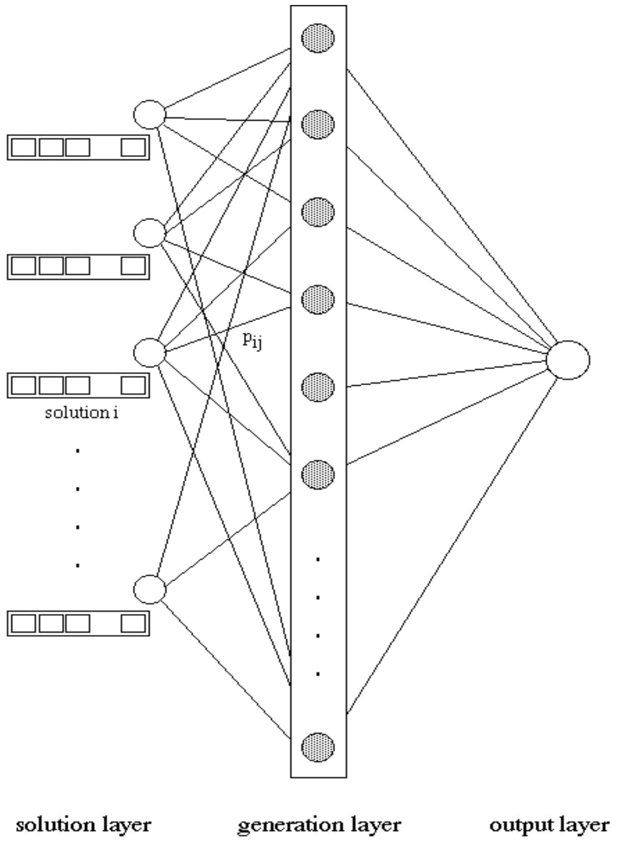


Fig. 1. Nessy network

a solution of the solution layer. This random access is controlled by the probabilities assigned to the weights. E.g. weight p_{23} describes the (relative) probability for generation layer neuron 2 to choose solution 3 for its state.

2. The fitness of all solutions chosen by the generation layer neurons are summed and compared with the training value (Zero). The relative error of every generation layer neuron is backpropagated.
3. The weights of the solution-generation layer connections are updated.
4. The solutions of the solution layer are modified by a transduction operator.
5. The solutions are mutated.

The algorithm in detail:

1. Generation layer neuron i randomly chooses a solution. Solution j is chosen with probability

$$\frac{p_{ij}}{\sum_k p_{ik}}. \tag{1}$$

As a consequence, different generation layer neurons can select the same solution from the solution layer, depending from the weights assigned to this solution. During the adaptation, good solutions get high weights assigned. This improves the probability of this repetitive choice of good solutions in the generation layer and hence of transduction of their genes into the whole population.

2. The backpropagated error of every generation layer neuron is simply the normalized fitness value of its state, i.e.

$$h_i = \frac{f_i}{\sum_k f_k}. \tag{2}$$

The MBPN is a supervised neural network. The desired values of the output layer are given as training values and the neural net adapts its weights in order to compute these given training values from the input values. However, in evolutionary algorithms the desired output is unknown. If the optimization is cost-oriented, i.e. the optimization goal is to minimize the fitness function, and if it is ensured that the fitness value is always non-negative, Zero is the theoretical limit of the optimization. Depending on the optimization problem this value can be reached or not. But, by choosing Zero as training value the solutions are forced to move towards this value and the contradicting learning goals of the MBPN and an evolutionary algorithm can be combined.

3. The update of the weights is a little more complicated. To get a learning rule for probabilities we consider the expectation value of the network output

$$\bar{f} = \frac{\sum_{\{j_i\}\in I^m} \sum_i f_{j_i} \prod_i p_{ij_i}}{\prod_i \sum_k p_{ik}} \tag{3}$$

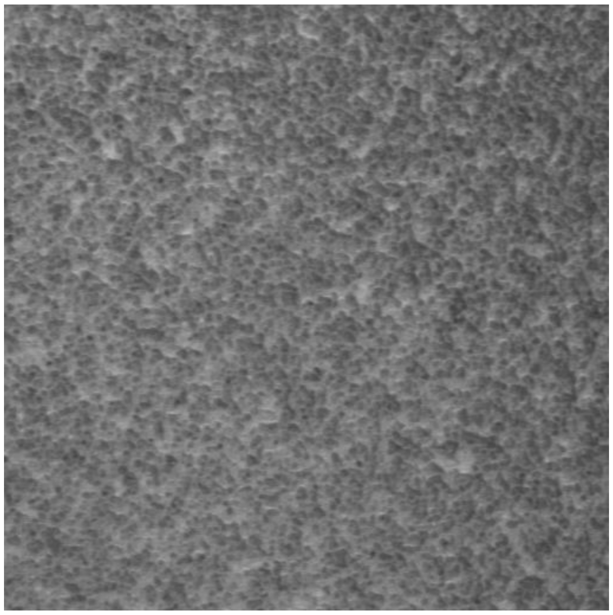
where m stands for the number of generation layer neurons, I is the set of the first n integers (n is the number of solution layer neurons), consequently, $\{j_i\}$ is a set of m indices in the range $1 \ldots n$ that represents one possible choice of solutions of the generation layer neurons. We use gradient learning rule. The partial derivatives of (3) with respect to the p_{ij} contains expectation values of the generation layer neuron states. These values are replaced with its actual values. This results in the learning law

$$p_{ij}^{new} = p_{ij}^{old} - \alpha \frac{f_j - g_i}{O} \tag{4}$$

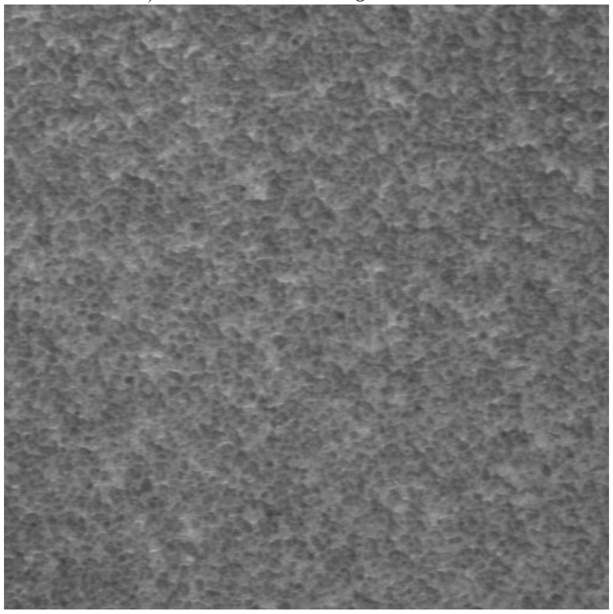
where α is the learning rate, g_i is the fitness value of the state of generation layer neuron i and O the state of the output neuron. The new weights are restricted to the range $[0, 1]$.

4. The modification of every solution is performed fitness-proportionately. Every solution compares its fitness with the fitness of generation layer neuron state i. If the second one is larger than the solutions' fitness, the solution takes gene element i of the generation layer neuron state. This operation replaces the crossover operator of conventional evolutionary learning. Its main task - the creation of new solutions - is also achieved. This operator was introduced by [6] as implantation operation, later ([7]) it was referred to as transduction operator and deduced from bacterial genetics. A bacteria transducts its genes over the whole population (infection).

5. Mutation is necessary if the transduction operator is applied. The whole population would tend to loss their



a) normal distributed granulation



b) too high packed granulation

Fig. 2. Images of sand paper

genetic diversity, because successful genes are copied over unsuccessful ones. Unfortunately, the mutation operation is problem-dependent. In our case we used integer numbers from a limited interval. The mutation is performed by adding a GAUSSIAN distributed random number to the actual gene. Two parameters control the mutation: mutation probability p_μ and the standard deviation of the GAUSSIAN distribution σ_μ. Mutation operation is independent from fitness values.

Altogether, Nessy has one structural parameter (size of solution layer) and three learning parameters (learning rate α, mutation probability p_μ and mutation standard deviation σ_μ).

III. Properties

Nessy is both, a neural network and an evolutionary algorithm. Its a neural network because all calculations are performed locally, i.e. independent from the state of other neurons of the same layer, and because it has a graphical representation. Also, Nessy uses evolutionary learning because its learning is based on random search, because it uses probabilities rather than computes with probabilities, and because solutions are fitness-proportionately modified.

Nessy optimizes the entire population. This is a consequence of its individual evolutionary algorithm, as explained in [5]. IEA optimizes the population by optimizing every individual, not the population as a whole. In IEA every individual is improved by learning from better ones. In our approach this is directly achieved by the transduction operator.

Nessy uses a very simple data management. Its implementation is straightforward.

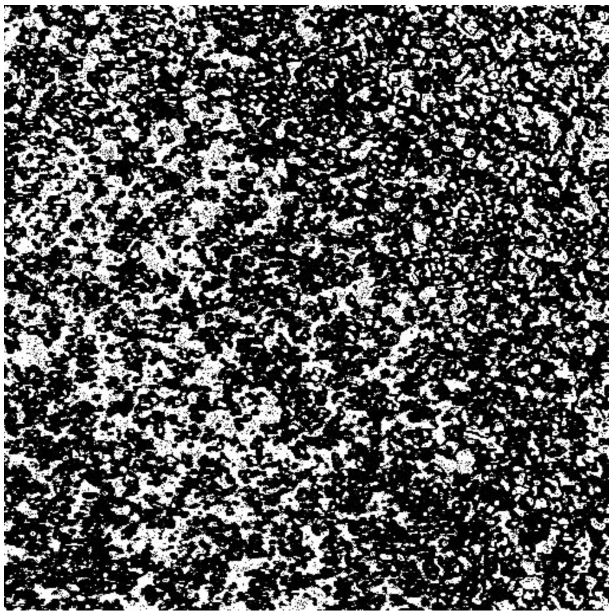
IV. Application example

To prove the learning abilities of Nessy we have chosen a pattern recognition problem. Consider figure 2. There are textures of two sand papers shown. The upper one with normal distributed granulation, the lower one with too high packed granulation. From its visual properties both images are hardly to separate. Greyvalue properties like histogram or co-occurrence matrix are very similar.

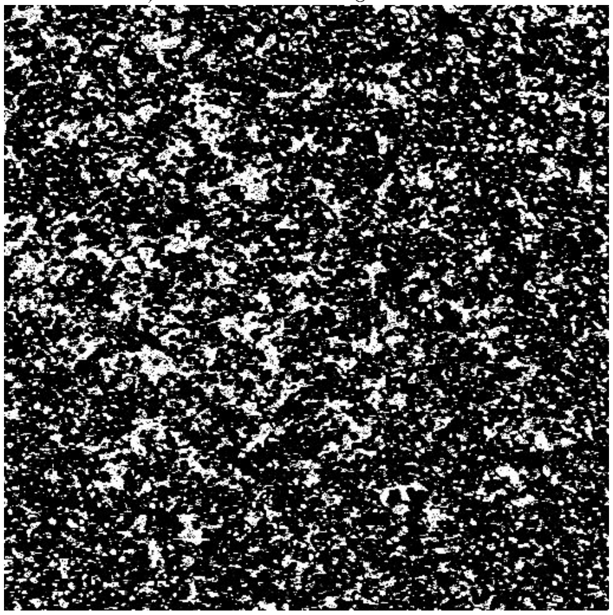
One approach for improving the contrast between both images is to make a greyvalue lookup. For this operation a lookup table is used with 256 entries with values between 0 and 255. The entries are numbered from 0 to 255. In the processing image every pixel with greyvalue g is replaced with the value of entry g in the lookup table. A lot of useful image processing operations can be characterized by such a lookup table (e.g. inversion, normalization, linearization, squaring, bit-clearing, gamma correction). The optimization goal is to find a lookup table which, applied on both images, generates a great difference in the histogram of both images. This problem is well-suited for Nessy. Every solution is represented by a gene of size 256, one gene element for one entry.

The initialization of the network is performed by setting all solutions equal to the identical lookup table (i.e. entry g has value g). The fitness of a solution is calculated as the inverse of the mean squared error between both modified histograms. The size of the solution layer was set to 10. Different settings for the mutation parameters were tested. A good choice for the mutation probability was 0.01. The mutation standard deviation is not a critical parameter. While a large value slows down the adaptation process, it is worth for the later stages of the optimization. Finally, a value of 10 was chosen, which gave a good compromise.

The full run was performed for 30,000 cycles. The modified images are shown in figure 3. The difference between both images has increased. Figure 4 shows three lookup tables generated by different runs. The height of a vertical bar indicates the replacing greyvalue, the horizontal posi-



a) normal distributed granulation



b) too high packed granulation

Fig. 3. Contrast-improved images

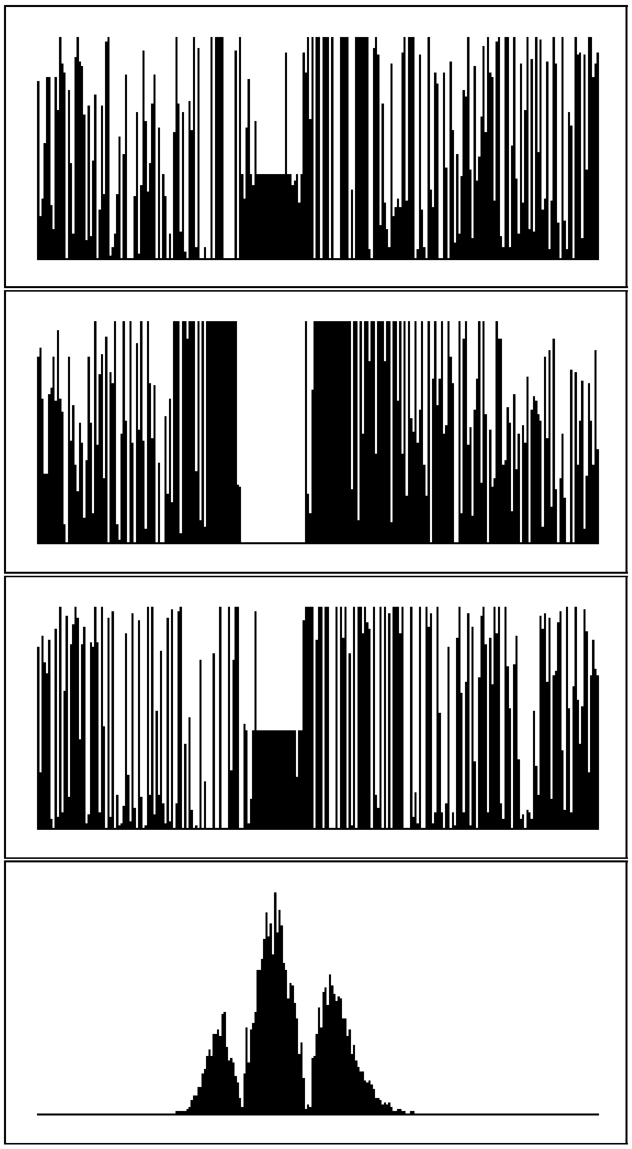


Fig. 4. Adapted lookup tables of three different runs and the absolute difference of the greyvalue histograms of both images in figure 2

tion of the bar indicates the replaced greyvalue (0 on the left, 255 on the right).

All of them have the plateau in the middle region in common. The optimal solution maps all greyvalues in the range 100 to 120 onto the same greyvalue (no matter which one it is). Most of the remaining greyvalues are "thrown" to either 0 or 255. If we consider the histogram of absolute differences between the histograms of both images (see bottom of figure 4), the maximal difference between both images is in that greyvalue range. Hence, the optimization procedure uses that discriminating property of both images to improve the contrast between them.

V. Conclusions

A new neural architecture (Nessy) has been proposed which uses evolutionary learning. The learning ability of Nessy has been proved with a pattern recognition example. Nessy is a handsome architecture for the application of evolutionary computation. No external data management is necessary.

Nessy can be modified in different ways. Similar to MBPN, batched learning can be introduced. This improves the reliability of the estimated expectation values in formula (4). Also, the modification of solutions can be performed on a group of solutions. This allows the application field of Nessy to be extended to combinatorial optimization problems. Another possible modification is mutation control by learning history. If the fitness values of the whole population does not change over a longer period, the mutation standard deviation should be increased.

If the population drifts into the higher fitness regions, it should be decreased. Future work on Nessy will explore the effects of these modifications. Also, Nessy should be applied to real optimization problems and compared with other evolutionary learning methods.

References

[1] Takeda, F., et al., *A Paper Currency Recognition Method by a Neural Network Using Masks and Mask Optimization by GA*, Proc. of WWW'94, Nagoya, Japan, 1994, pp. 125-129.

[2] Galic, E., Höhfeld, M., *Improving the Generalization Performance of Multi-Layer-Perceptron with Population-Based Incremental Learning*, in: Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.-P., (eds.), "Parallel Problem Solving from Nature - PPSN IV", Lecture Notes in Computer Science 1141, Springer Verlag, Berlin, Germany, 1996, pp. 740-750.

[3] Stepniewski, S.W., Keane, A.J., *Topology Design of Feedforward Neural Networks by Genetic Algorithm*, in: Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.-P., (eds.), "Parallel Problem Solving from Nature - PPSN IV", Lecture Notes in Computer Science 1141, Springer Verlag, Berlin, Germany, 1996, pp. 771-780.

[4] Zhao, Q., *Co-Evolutionary Algorithm: A New Approach to Evolutionary Learning*, Proc. of IIZUKA'96, Iizuka, Japan, 1996, pp. 529-523.

[5] Zhao, Q., Higuchi, T., *Efficient learning of NN-MLP based on individual evolutionary algorithm*, Neurocomputing 13(1996), 2-4, pp. 201-215.

[6] Furuhashi, T., Nakaoka, K., Uchikawa, Y., *A New Approach to Genetic Based Machine Learning and an Efficient finding of Fuzzy Rules*, Proc. of WWW'94, Nagoya, Japan, 1994, pp. 114-122.

[7] Hashiyama, T., Furuhashi, T., Uchikawa, Y., *Design of Fuzzy Controllers for Semi-Active Suspension Generated through the Genetic Algorithm*, Proc. of ANNES'95, Dunedin, New Zealand, 1995, pp. 166-169.