A model for the dynamic interaction between evolution and learning

Bernhard Sendhoff, Martin Kreutz

1999

Preprint:

This is an accepted article published in Neural Processing Letters. The final authenticated version is available online at: https://doi.org/[DOI not available]

published in:
Neural Processing Letters 10(3), pages 181-193, 1999.

A model for the dynamic interaction between evolution and learning

Bernhard Sendhoff
(bernhard.sendhoff@neuroinformatik.ruhr-uni-bochum.de) and
Martin Kreutz
(martin.kreutz@neuroinformatik.ruhr-uni-bochum.de)
Institut für Neuroinformatik
Ruhr-Universität Bochum
D-44780 Bochum, Germany

Abstract. The interaction between learning and evolution has elicited much interest particularly among researchers who use evolutionary algorithms for the optimization of neural structures. In this article, we will propose an extension of the existing models by including a developmental phase – a growth process – of the neural network. In this way, we are able to examine the dynamical interaction between genetic information and information learned during development. Several measures are proposed to quantitatively examine the benefits and the effects of such an overlap between learning and evolution. The proposed model, which is based on the recursive encoding method for structure optimization of neural networks, is applied to the problem domain of time series prediction. Furthermore, comments are made on problem domains which associate growing networks (size) during development with problems of increasing complexity.

Keywords: evolution,learning,ontogeny,neural development,structure optimization

1. Introduction

The processes of evolution and learning although very different in their physical realizations are similar when analyzed at an abstract level as processes which extract information from the environment to design and optimize structures, e.g. neural structures. Of course, adaptation in both cases operates on different time scales and combination of both approaches imply different strategies in the way in which the processes ultimately enhance the probability of survival and reproduction of organisms. Artificial neural networks (ANN) and evolutionary algorithms (EA) are both models of their natural counterparts (biological nervous systems and biological evolution respectively) and their prime properties are adaptation on large time scales (EA) and learning (ANN). It therefore seems natural to combine both systems to analyze the intrinsic interdependencies of evolution and learning.

In Darwinian or better still in Neo-Darwinian evolution knowledge acquired by the individual during its lifetime cannot be transferred into

its genome and subsequently passed on to the next generation, as it is explicitly incorporated in Lamarckian evolution. However, there is an indirect influence, first noted by Baldwin (1896), whereby individuals which are able to adapt better to their environments increase their reproduction probability. This implies that the learning process influences the reproduction efficiency. Besides the biochemical problems of reverse transcription of information, it might also be disadvantageous for the stability of the organism against changing environments. Lamarckian evolution would speed up the adaptation process and promote the early independence of the offspring from the parents by providing more explicit information about the environment in the genotype. At the same time however, it would be more difficult to change "hard-wired" information during periods of rapid environmental change (Sasaki and Tokoro, 1997). This line of argument has been supported by experiments (Hinton and Nowlan, 1987; Maynard-Smith, 1987; Fontanari and Meir, 1990; Belew, 1990), for both sexual and asexual reproduction. The authors applied a direct coding method to a simple neural network, using the symbol set (1,0,?), where connections are represented by '1', absent connections by '0' and connections which are specified during learning, here guessing in several trials, by '?'. One of the main results is that learning increases the robustness of the evolutionary process to mutation at the expense of a drastic reduction of the selection pressure towards the correct genetic determination of the whole structure, which would imply elimination of all ?-alleles.

Another model for the connection between learning and evolution in neural networks has been studied (Nolfi et al., 1994a; Nolfi and Parisi, 1997) by applying an "ecological" neural network, which incorporates a control module supervising and predicting its own actions, to robot control. Nolfi et al. conclude that evolution and learning influence each other and that evolution combined with learning optimizes the adaptability of the network.

Although several other proposals have been suggested for the inclusion of an ontogenetic development in neural network optimization (e.g. Belew, 1993, Gruau, 1993), only a few have been used for the analysis of evolution and learning; a problem which has already been acknowledged by Hinton and Belew. The genotype–phenotype mapping representing the developmental process, which is itself influenced by the active learning process of the organism, is expected to have a major influence on the connection between learning and evolution. The models to date mainly consider a static coupling of evolution and learning: the influence of evolution ends when learning starts. A dynamic coupling of evolutionarily determined properties and learning has been considered by Gruau and Whitley (1993) and by Nolfi et al. (1994b), although in a

very different framework from the one which is proposed in this work. In Gruau and Whitley 1993 learned information is backcoded into the genotype which is exploited for the development of several individual networks. This model is closely related to Lamarckian evolution (as noted by the authors) and, as such, is conceptually different from the approach proposed here. The work by Nolfi et al. (1994b) introduces a dynamic interaction between environment and genotypic information at the level of neurite growth and does not aim to propose an analysis of the *interaction* between the different sources of information during development. The *quantitative* exploration of this dynamical process and the relative influence of the two adaptation processes at different developmental stages poses an interesting augmentation and extension of these models and constitutes the main aim of this article.

In Section 2 we will introduce a model for a genotype–phenotype mapping representing neural development which is partly influenced by the genetically determined information and by the learning process. In order to analyze the interaction between evolution and learning we will introduce several measures in Section 3 and apply them to the problem of time series prediction in Section 4. The paper concludes with a discussion and an outlook on future work.

2. Outline of the model

In order to be able to analyze the dynamical interaction between learning and evolution, we need a model which incorporates the genotype phenotype map as a developmental process in the evolutionary optimization. The recursive encoding method of neural networks (Sendhoff and Kreutz, 1998; Sendhoff, 1998), which is based on the grammar encoding by Kitano (1990), describes such a growth process of the neural structure. In addition to the specification of the structure, an initialization of the weights is possible which is "hidden" in the developmental process and which shares similarities with models of the biological specification of neural connections (Gierer, 1988). Our model is shown in Figure 1. The genetic representation describes the growth process which at the same time is influenced by intermediary learning. In the next section we will define a genetic representation which incorporates a growth process as a model for the ontogenetic development and which allows the observation of the influence of genetic information and learning on the neural structure, which itself represents the phenotype.

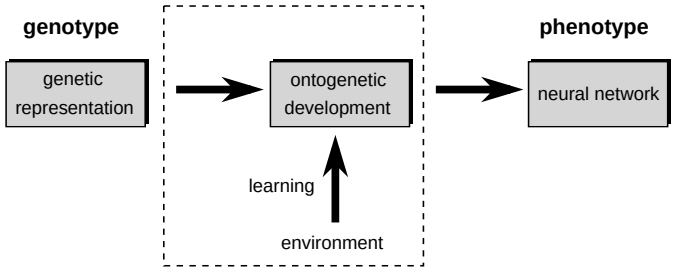

Figure 1. Scheme of the proposed model including neural development (ontogeny). The main purpose is to analyze the interaction between the two arrows inside the dotted box. Note that no transverse flow of information is present – the model is strictly Neo-Darwinian.

2.1. The recursive encoding scheme

In the recursive encoding, the connection matrix of a feed-forward network without a layered structure is developed in different stages. The elements of the connection matrix define the initial weights between neurons; in the case of a zero element the neurons are not connected. Since the matrix is feed-forward, only the upper triangular part is needed for the specification of the network. The diagonal elements define the threshold values for each neuron. The developmental process of the matrix is specified by a mapping from a first vector S_C (small chromosome) with integer elements to a second vector L_C (large chromosome), also with integer elements. In each step, each element of the connection matrix is replaced by a 2×2 matrix of new elements. In Figure 2 the growth of the connection matrix is shown. The following notation will be used for the recursive encoding:

$a^k_{\mu\nu}$	element at position (μ, ν) of the connection matrix in the kth recursion step. If the element is non-zero, it represents the initial weight between neuron μ and ν ($a^k_{\mu\nu} \in \{0, \ldots, N_{sym}\}$).
R	the number of recursion steps
k	the recursion step, $(k = 1, \ldots, R)$
N_{sym}	maximum integer number permitted in (S_C, L_C)
$N(a^k_{\mu\nu})$	the index of the first element in S_C, which is identical to $a^k_{\mu\nu}$
d_{S_C}	dimension of the vector S_C (small chromosome)
d_{L_C}	dimension of the vector L_C (large chromosome)
s_i, l_i	elements of the vector S_C and L_C

At each recursion step k, the index $N(a^k_{\mu\nu})$ of the *first* element in S_C, which is identical to the connection matrix element $a^k_{\mu\nu}$, is determined; for example, index $N(a^k_{\mu\nu} = 7) = 3$ in Figure 3 (a). The element is then

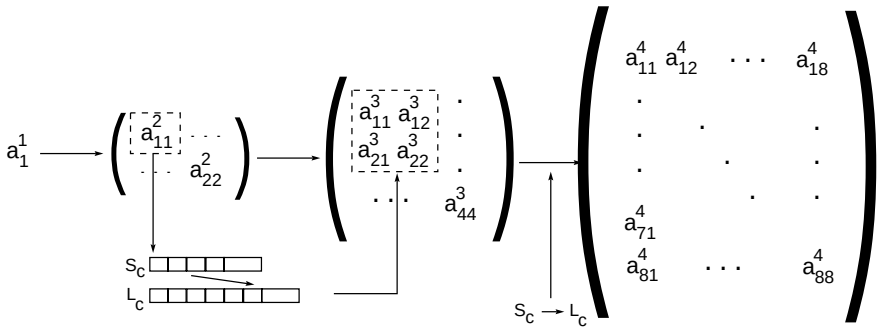

Figure 2. Scheme of the recursive development of the connection matrix up to a size of 8×8. In each step, each element is replaced by a 2×2 matrix via the mapping $S_C \rightarrow L_C$

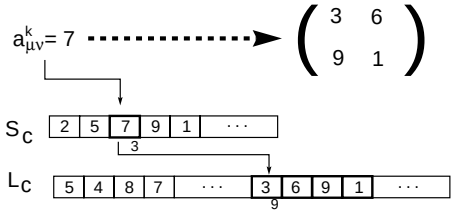

Figure 3. One element is replaced by four elements in the recursion step via the small chromosome $S_C \rightarrow$ large chromosome L_C mapping.

replaced by the four elements at the positions

$$\begin{aligned}(\, 4 \cdot (N(a_{\mu\nu}^k) - 1) + 1, \quad & 4 \cdot (N(a_{\mu\nu}^k) - 1) + 2, \\ 4 \cdot (N(a_{\mu\nu}^k) - 1) + 3, \quad & 4 \cdot (N(a_{\mu\nu}^k) - 1) + 4 \,) \end{aligned} \tag{1}$$

in the large chromosome L_C. Figure 3 (a) shows the replacement of an element $a_{\mu\nu}^k = 7$ by the four elements $(3, 6, 9, 1)$ at the positions $(9, 10, 11, 12)$. Should $a_{\mu\nu}^k$ not be identical to any element of S_C, it is replaced by four so-called terminal symbols (in the notation of integer strings, the most convenient choice is zero). The terminal symbol (zero) denotes that no connection exists between neuron μ and ν. The terminal symbol is in turn always replaced by another four terminal symbols in a recursion step. Finally, the connection matrix is simplified by deleting all neurons which do not contribute to the network output. Thus, the main components of the recursive encoding are:

- In each step, each element of the connection matrix is replaced by a 2×2 matrix whose elements are specified by equation (1).
- If the element is not identical to any entry of S_C, the elements of the 2×2 matrix are given by $(0, 0, 0, 0)$.

– In each step, 0 is replaced by $(0, 0, 0, 0)$.

– R steps are carried out.

In its simplest form the recursive encoding develops a connection matrix from a vector S_C with elements $s_i \in \{1, \ldots, N_{sym}\}$ and a vector L_C with elements $l_i \in \{1, \ldots, N_{sym}\}$ with the described mapping. The connection matrix is then translated into a neural network in the following way. If $a^R_{\mu\nu} = 0$, neuron μ and neuron ν are not connected. All other integer values $a^R_{\mu\nu} \in \{1, \ldots, N_{sym}\}$ are mapped to an interval $[-\delta, \delta]$. The activation function $tanh(x)$ is used for all neurons.

This *basic* scheme has been successfully applied to network optimization for time series prediction by Sendhoff and Kreutz (1998). Theoretical analysis of the encoding scheme concerning the dependence of the probability of the zero symbol on the coding parameters has been carried out in Sendhoff 1998. Furthermore, extensions of the basic scheme to guarantee certain properties of the encoding, like completeness, have been proposed, i.e. additional negative integers are introduced, which map to the zero symbol *after* the replacement process.

2.2. The incorporation of pre-learned weights in the recursive encoding

In order to investigate the interaction between learning and genetically determined development, it is necessary to extend the replacement scheme. Firstly, we note that each connection matrix during development (i.e. all three matrices in Figure 2) can represent a neural network at different *ontogenetic stages* which can learn specific tasks. Secondly, we have to realize a transfer of pre-learned information from one ontogenetic stage to the next. Therefore, in this scheme it must be possible to include the network connection matrix after simplification and after the learning process at step $(k-1)$ in the connection matrix at step k. Two different methods can be used to achieve this; see Figure 4. Firstly, in each step the connection matrix is built in the original form using the *basic* recursive encoding method described in the last section including the initial weight specification. In addition, the network matrix from the previous ontogenetic stage is copied to the upper left part of the new matrix. Therefore, replacing some of the genetically determined initial weights by the pre-learned weights of the previous ontogenetic stage. Alternatively, the network matrix is split up into four parts of equal size, which are copied to the four edges of the connection matrix at step k. In this way, the network grows *in the middle*, which means that input and output neurons and their connections are not changed during the growth process. Only hidden neurons are added in the developmental

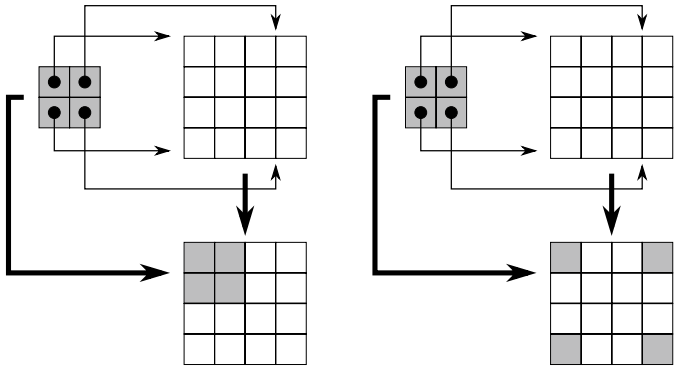

Figure 4. The extension of the matrix replacement scheme of the basic recursive encoding to include learning of networks represented by matrices of earlier developmental steps. Left: The original replacement scheme is used to build the matrix at step k, however the network connection matrix with *learned* weights from step $(k-1)$ are copied to the 1st quadrant of the new matrix. Right: The four quadrants of the network matrix from step $(k-1)$ are copied to the upper left, lower left, upper right and lower right section of the new matrix.

phase. The differences both with respect to performance and to the qualitative behaviour of the interaction measures (Section 3), between the schemes Figure 4 (left) and (right) are marginal, see Sendhoff 1998. In the following experiments the left scheme will be used. Using one of these replacement schemes in the developmental process an overlap between the learning process (connection matrix which is copied from the last growth step) and the genetically determined structure and weight initialization is therefore realized.

3. The interaction between learning and evolution

The interaction between genetically predetermined weights and weights which have been learned in an earlier phase is realized as follows. The neural network grows in the developmental process by applying the recursive encoding method. The network has to carry out tasks as soon as it reaches a minimal size[1] during the developmental phase. The network is trained for 75 cycles[2], after each replacement step the trained weights are transferred to a part of the enlarged network of the next step. Therefore, after each developmental step the structure and the weights of the network are determined partially by the trained network of the last step and partially by the genetic information contained in the vectors (*chromosomes*) (S_C, L_C). If the matrix is not simplified, i.e.

[1] In the experiments in Section 4.1 the minimal number of neurons is six.
[2] One cycle corresponds to the presentation of all training and/or test pattern.

no neurons are removed, one quarter of the matrix originates from the previously trained network and three-quarters from the replacement rules of the step, although in the experiments this relation can be very different. The matrix is forced to grow, i.e. replacement schemes in which the network does not expand in later steps, due to the deletion of neurons, are removed from the population. In this way, the border case, i.e. the network at step $(k+1)$ is identical to the one from step k, which would merely result in doubling the learning cycles for this network, cannot occur.

Since the overall network structure has changed, it is by no means clear to what extent the enlarged network at step $(k+1)$ will benefit from the fact that part of its weights from network k have already been learned. The Evolution–Learning interaction was analyzed by studying the relative influence of the pre-learned weights (from step k) on the prediction error of the network at step $(k+1)$, both before (head start, *HS*) and after (final relation, *FR*) learning. $\mathcal{E}_{wt}^{va}(c)$ refers to the network error after cycle c *with* weight transfer and $\mathcal{E}_{nt}^{va}(c)$ to the network error after cycle c *without* weight transfer on the validation data set.

$$HS = \frac{\mathcal{E}_{nt}^{va}(c=0)}{\mathcal{E}_{wt}^{va}(c=0)} \tag{2}$$

$$FR = \frac{\mathcal{E}_{nt}^{va}(c=75)}{\mathcal{E}_{wt}^{va}(c=75)} \tag{3}$$

Additionally, the "weight difference relation" (*WD*) between the pre-learned and the genetically determined weights was recorded. The weight difference relation takes the relative changes of the weights induced by the learning process into account. In order to calculate these measures, each individual had to be trained once with weight transfer from previous steps and once without weight transfer[3]. The following notation will be used:

- $\mathcal{M}^{learn}$ contains all index tuples of the connection matrix which refer to weights transferred from the previous network.
- $\mathcal{M}^{gene}$ contains all index tuples of the connection matrix determined by the recursive encoding method.
- d_M is the dimension of the connection matrix
  d_M^t is the dimension of the transferred matrix.

[3] If on average four networks are evaluated twice, eight networks have to be trained per individual. These computations were carried out on the Cray T3E parallel computer of the *Höchstleistungsrechenzentrum Jülich, Germany* using 80 processors. The computation time of eight hours corresponds to roughly 40 days on a Sun UltraSparc II workstation.

– $a_{\mu\nu}$ and $a^L_{\mu\nu}$ are the matrix elements (weights) before and after learning, respectively.

Now, the weight difference relations can be defined as follows:

$$WD_1 = \frac{\mathrm{card}(\mathcal{M}^{gene})}{\mathrm{card}(\mathcal{M}^{learn})} \frac{\sum_{(\mu,\nu)\in\mathcal{M}^{learn}} |a_{\mu\nu} - a^L_{\mu\nu}|}{\sum_{(\mu,\nu)\in\mathcal{M}^{gene}} |a_{\mu\nu} - a^L_{\mu\nu}|}, \tag{4}$$

$$WD_2 = \frac{d_M}{d^t_M} \frac{\mathrm{card}(\mathcal{M}^{gene} \cup \mathcal{M}^{learn})}{\mathrm{card}(\mathcal{M}^{learn})} \frac{\sum_{(\mu,\nu)\in\mathcal{M}^{learn}} |a_{\mu\nu} - a^L_{\mu\nu}|}{\sum_{(\mu,\nu)\in\mathcal{M}^{gene}\cup\mathcal{M}^{learn}} |a_{\mu\nu} - a^L_{\mu\nu}|}.$$

4. Experiments for time series prediction

In order to investigate the interaction between learning and evolution based on the measures introduced in the previous section, we applied the neural networks to the problem of time series prediction of the Lorenz system of three coupled differential equations in its chaotic domain (Lorenz, 1984). The data was generated with a 4th order Runge-Kutta method ($dt = 0.05$) and split up into a training, a test and a validation data set each with 500 three-dimensional input/output vectors. Standard back-propagation with a momentum term was used for learning and the mean squared error on the validation set for the fitness function.

Each individual of the population in the evolutionary algorithm consists of four chromosomes: the recursive encoding vectors S_C and L_C; the three input weights to the first three neurons and the coding vector $C = (R, d_{S_C}, N_{sym})$ which contains the three coding parameters and which is optimized on a larger time scale. A more detailed discussion on the evolution of the coding itself via the optimization of C is presented in Sendhoff and Kreutz 1998 and Sendhoff 1998. The mutation operators on the recursive encoding vectors are position dependent, the crossover operator for S_C is uniform and for L_C four tuples are exchanged, which is chosen according to the developmental process. The probabilities for the different operators are summarized along with all other parameters in Table I. A deterministic (μ, λ) selection with one elitist (Bäck, 1996) has been applied.

4.1. One step prediction

In the experiment, each network in the developmental phase has to carry out the same task, namely one step prediction of the Lorenz chaotic time series. The results are shown in Figure 5. The values of

Table I. Parameter values for the experiments for the one step prediction of the Lorenz time series with the recursive encoding method and back-propagation learning. p_m and p_c, respectively, denote the probabilities of mutation and crossover for the different chromosomes, S_C, L_C, the coding C and the input weights *input*.

parameter	**value**	**parameter**	**value**
learning rate	0.0175	momentum term	0.75
max. (R, d_{S_C}, N_{sym})	(7,50,50)	(μ, λ)	$(20, 80)$
weight interval	$[1.0, -1.0]$	$p_c(S_C, L_C)$	0.75
$p_m(S_C)$	$2 \cdot (d_{S_C})^{-1}$	$p_m(L_C)$	$2 \cdot (d_{L_C})^{-1}$
$p_m(C)$, $p_c(C)$	0.05, 0.05	$p_m(input)$, $p_c(input)$	0.33, 0.75

FR for the best network (5 (c) & (d)), show on average a sixfold increase of performance due to the weight transfer from the previously learned network. All developmental factors (FR, HS, WD_1, WD_2) are averaged over the number of network evaluations during the development. The final error relation *FR* increases over the epochs, which indicates that the selection pressure is towards the incorporation of information from previously trained structures in order to enhance the networks performance *after* the new learning process. The values of the head start *HS* in Figure 5 (b) show that the transferred weights enhance the network's performance also *before* the new learning process, but at a lower rate (on average 1.3 compared to 2.0 for *FR*) and that there is no selection pressure to increase this factor (on average and for the best individual). Therefore, it seems that the previously learned weights are better starting values for the new learning process, whilst the increase of the performance of the network initially (before learning) is limited. This is supported by the values of the weight difference relations WD_1, WD_2 (5 (b)). Although the change in the genetically determined weight values during the learning process is larger than the change of the weights from the previously trained network, the difference is rather small (both WD_1 and WD_2 assume values between 0.8 and 0.9 on average). Therefore, the main advantage from the weight transfer, which is used in the developmental process, seems to be the initial starting point in weight space and not the usage of the information for the actual task which is contained in the pre-learned weights.

There is another interesting result from combining learned and genetically determined weights. In Figure 5 (c), the values of *FR* are plotted logarithmically and we note that the relation in the final error fluctuates in some epochs to very large values (up to 1000, compared with the typical average of 2.0). In the standard recursive encoding

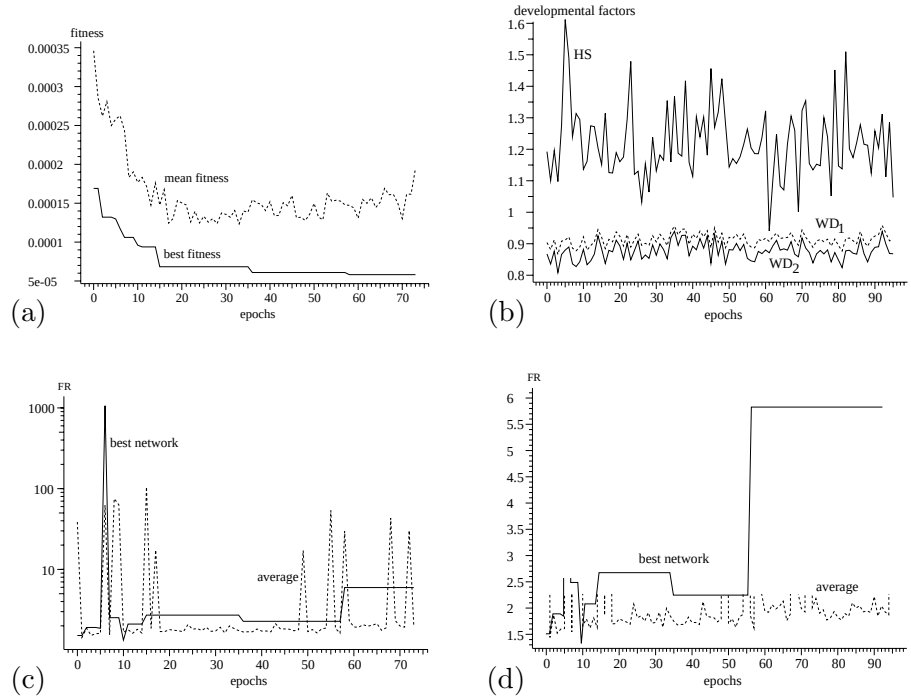

Figure 5. (a) The fitness values for the "pre-learned encoding scheme". (b) The developmental factors HS, WD_1 and WD_2 for the population average and (c) the final error relation FR on a logarithmic scale and (d) the same FR values excluding the fluctuations.

method, it is possible that the genetic weight initialization of larger networks is particularly poor resulting in a final error which is of magnitudes larger than average. This is due to the coarseness of the weight initialization (cf. Section 2.1 on the weight initialization in the recursive encoding method). The phenomenon does not seem to occur if previously trained weights are transferred. Since this is less likely to take place in small networks, the first network to be evaluated, whose weights are only determined genetically, is not prone to such fluctuations of the error. The transfer of weights in the next network evaluations can then prevent these fluctuations. Subsequently, the combination of pre-learned weights with genetically determined weights seems to increase the stability of the network's performance during the developmental phase. Since the selective pressure only acts on the final performance of the network which develops according to the "pre-learned encoding scheme", these effects can occur even in the best network of the population.

5. Discussion and Outlook

In this article we proposed a model for the interaction of evolution and learning which includes a developmental phase (ontogeny) of a neural network and which therefore allows the analysis of the effect of an overlap of information accumulated in the weights during short term adaptation (learning) and genetically determined information of the structure and the weight initialization during long term adaptation (evolution). Based on the experimental data we can conclude that both processes can benefit from such an approach. The overall performance of the evolutionary algorithm is increased and more importantly the individual network can exploit already learned behaviour by transfering the information during the growth process into a new network structure. The performance of the new neural network before learning is increased, albeit quantitatively not by the same amount as after learning. This seems to indicate that the pre-learned weights constitute better starting points for the new learning process. Another remarkable effect is that of increased stability of the network structure during the growth process due to the transfer of weights. In this way the neural structure can grow without running the risk of intermediate failure which is not directly exposed to selective pressure in our model.

Although the genotype–phenotype mapping used in our model is far from being biologically plausible, some of the above comments might well be applicable to the neural development of the brain, where it seems reasonable that an overlap between learning and genetically determined growth process exists. This points to an interesting extension of the proposed model, namely, the inclusion of plasticity during the learning process.

Another experiment, which has already been carried out to some extent in Sendhoff 1998, addresses the variation of the task of the neural network between different developmental steps, i.e. to demand from the neural structure that it solves more and more complex problems during the development. In time series prediction, one possible extension is to let the network iterate, i.e using its own output as input for a limited number of steps, with an increasing iteration time before unperturbed external data is used again. The observation is that, on the one hand, the pre-learned information is used in a slightly different way (*HS* increases and *FR* decreases relative to the results of Section 4) and that on the other hand larger networks do not necessarily correspond to more difficult problems. That is to say, that the evolutionary optimization attempts to keep the growth process of the neural structure as small as possible during development. In this way, previous learned weights can

be exploited more efficiently and the overall network size is kept small even at later development stages.

It therefore seems sensible to use problems for which it is more likely that larger networks correspond to higher complexity. The domain of robotics as proposed by Gruau (1995) might provide a useful area of application. Finally, it is interesting to note that the process of modeling a system using an incremental approach for both the model and the difficulty of the task has also been used successfully for the non-parametric estimation of densities in statistics. In the "method of sieves", see the work by Geman and Hwang (1982) and references therein, the optimization of the density estimation is carried out for a subset of the parameter space. Iteratively, the size of the subset increases with the sample size.

Acknowledgements

The authors would like to thank W. von Seelen, C. von der Malsburg, C. Igel and P. Stagge for stimulating discussions on the subject and for proofreading the manuscript and one of the anonymous reviewers for pointing out additional references. The Höchstleistungsrechenzentrum Jülich, Germany is acknowledged for providing the computation time on the Cray T3E massively parallel supercomputer and for their expert advice. The project is supported by the German Ministry of research under grant SONN II 01IB701A0 and grant LEONET 01IB802C4.

References

Bäck, T.: 1996, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

Baldwin, J.: 1896, 'A new factor in evolution'. *American Naturalist* **30**, 441–451.

Belew, R.: 1990, 'Evolution, learning, and culture: Computational metaphors for adaptive algorithms'. *Complex Systems* **4**, 11–49.

Belew, R.: 1993, 'Interposing an ontogenetic model between genetic algorithms and neural networks'. In: J. Cowan (ed.): *Advances in Neural Information Processing Systems*, Vol. 5. pp. 99–106.

Fontanari, J. and R. Meir: 1990, 'The effect of learning on the evolution of asexual populations'. *Complex Systems* **4**, 401–414.

Geman, S. and C. Hwang: 1982, 'Nonparametric maximum likelihood estimation by the method of sieves'. *The Annals of Statistics* **10**(2), 401–414.

Gierer, A.: 1988, 'Spatial organization and genetic information in brain development'. *Biological Cybernetics* **59**, 13–21.

Gruau, F.: 1993, 'Genetic synthesis of modular neural networks'. In: S. Forrest (ed.): *Genetic Algorithms: Proceedings of the 5th International Conferences (ICGA)*. San Mateo, CA, pp. 318–325.

Gruau, F.: 1995, 'Automatic definition of modular neural networks'. *Adaptive Behavior* **3**(2), 151–183.

Gruau, F. and D. Whitley: 1993, 'Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect'. *Evolutionary Computation* **1**(3), 213–233.

Hinton, G. and S. Nowlan: 1987, 'How learning can guide evolution'. *Complex Systems* **1**, 495–502.

Kitano, H.: 1990, 'Designing neural networks using genetic algorithms with graph generation system'. *Complex Systems* **4**, 461–476.

Lorenz, E.: 1984, 'Irregularity: A fundamental property of the atmosphere'. *Tellus* **A**(36), 98–110.

Maynard-Smith, J.: 1987, 'When learning guides evolution'. *Nature* **329**, 761–762.

Nolfi, S., J. Elman, and D. Parisi: 1994a, 'Learning and evolution in neural networks'. *Adaptive Behavior* **3**(1), 5–28.

Nolfi, S., O. Miglino, and D. Parisi: 1994b, 'Phenotypic plasticity in evolving neural networks'. In: D. Gaussier and J. Nicoud (eds.): *Proceedings of the First International Conference From Perception to Action*. pp. 146–157.

Nolfi, S. and D. Parisi: 1997, 'Learning to adapt to changing environments in evolving neural networks'. *Adaptive Behavior* **5**(1), 75–98.

Sasaki, T. and M. Tokoro: 1997, 'Adaptation toward changing environments: Why Darwinism in nature?'. In: P. Husbands and H. Inman (eds.): *Fourth European Conference on Artificial Life*. pp. 145–153.

Sendhoff, B.: 1998, *Evolution of Structures – Optimization of Artificial Neural Structures for Information Processing*. Aachen: Shaker Verlag. Doctoral dissertation.

Sendhoff, B. and M. Kreutz: 1998, 'Evolutionary optimization of the structure of neural networks using a recursive mapping as encoding'. In: G. Smith, N. Steele, and R. Albrecht (eds.): *Artificial Neural Nets and Genetic Algorithms – Proceedings of the 1997 International Conference*. pp. 370–374.