Coupling of Evolution and Learning to Optimize a Hierarchical Object Recognition Model

Georg Schneider, Heiko Wersing, Bernhard Sendhoff, Edgar Körner

2004

Preprint:

This is an accepted article published in Parallel Problem Solving from Nature – PPSN VIII. The final authenticated version is available online at: https://doi.org/[DOI not available]

published in: X. Yao et al. (editors), Parallel Problem Solving from Nature - PPSN VIII, pp. 662-671, 2004

Coupling of Evolution and Learning to Optimize a Hierarchical Object Recognition Model

Georg Schneider, Heiko Wersing, Bernhard Sendhoff, and Edgar Körner

Honda Research Institute Europe GmbH
Carl-Legien-Strasse 30, D-63073 Offenbach/Main, Germany
{Georg.Schneider, Heiko.Wersing}@honda-ri.de

Abstract. A key problem in designing artificial neural networks for visual object recognition tasks is the proper choice of the network architecture. Evolutionary optimization methods can help to solve this problem. In this work we compare different evolutionary optimization approaches for a biologically inspired neural vision system: Direct coding versus a biologically more plausible indirect coding using unsupervised local learning. A comparison to state-of-the-art recognition approaches shows the competitiveness of our approach.

1 Introduction

Evolutionary algorithms provide a general method for system design optimization and their successful combination with neural networks has been shown in various applications [1]. In the work presented here we optimize neural structures applied to object recognition problems. A critical problem in the application of neural vision systems is the introduction of invariance properties, such as translation, scaling and rotation of the input stimuli. We propose to use hierarchical architectural principles, which are inspired by the human vision system. There is strong biological evidence that hierarchical processing is an important principle in the visual cortex [2]. Barlow [3] proposed that these hierarchical neural representations are structured according to the principle of redundancy reduction. Along this line, unsupervised local learning rules were used to obtain visual features similar to the ones found in early visual brain areas [4].

In order to apply evolutionary algorithms to the design of neural systems their structure and parameters must be represented or encoded. Most approaches [1] use *direct* or *explicit coding*, e.g., via a *connection matrix*, where each entry represents a connection between two neurons. Biologically this scheme is implausible as the amount of information needed to be stored in the genome is far too large. This makes *indirect coding* approaches, where not every neuron with every connection is explicitly encoded in the genome, attractive. By using for example a predefined building process which controls the development of the phenotype, the only information which have to be encoded in the genome are process control parameters [5]. The next step is not only to use a set of fixed rules for the development, but an active learning process for the indirect coding. Interesting

approaches which focus on this combination of evolution and learning can be found [6, 7]. This scheme of an indirect coding using local learning rules for the building process of a complex neural system is biologically far more realistic [8, 9]. Few researchers use a form of indirect coding in their evolutionary optimizations of neural networks. Kitano [5] suggests a graph generation grammar to indirectly code neural networks and shows that the indirectly coded networks exhibit a magnitude of speed-up in convergence of the evolutionary optimization. Sendhoff and Kreutz [7] have included a developmental phase - a growth process - in the analysis of the dynamic interaction between genetic information and information learned during development. A strongly neurobiologically inspired approach to the combination of evolution and learning for the design of neural networks has been suggested by Rolls and Stringer [6]. Their optimized networks are restricted to three canonic architectures: pattern association memory, auto-association network and competitive neural network. In summary, most contributions which focus on indirectly coded evolutionary optimization schemes did not approach complex tasks, like 3D object recognition.

In our work presented here we combine biologically inspired hierarchical networks with evolution strategies in a novel way to obtain powerful recognition architectures for general 3D object recognition. Our focus is a comparison of direct versus indirect coding of the features in the visual hierarchy with regard to the generalization capabilities of the network. In the case of the indirect coding, we use a coupling of evolution and different local learning processes. The target value of the optimization is the classification performance of the vision network in an 3D object recognition task. Our vision model architecture is introduced in Section 2. The details of the evolutionary optimization are described in Section 3. We state and discuss the results, including a comparison to other state-of-the-art algorithms, in Section 4. In the last section, we conclude our work.

2 The neural vision system for object recognition

In the following, we define the hierarchical model architecture that we will use for the evolutionary optimization. The model is based on a feedforward architecture with weight-sharing and a succession of feature-sensitive matching and pooling stages (see also [10] for a discussion on the general properties and biological relevance of this architecture). The model comprises three stages arranged in a processing hierarchy (see Figure 1). The input image is presented as a 64×64 pixel image. The S1 layer consists of 4 Gabor feature planes at 4 orientations with a dimension of 64×64 each. The C1 layer subsamples by pooling down to a reso-

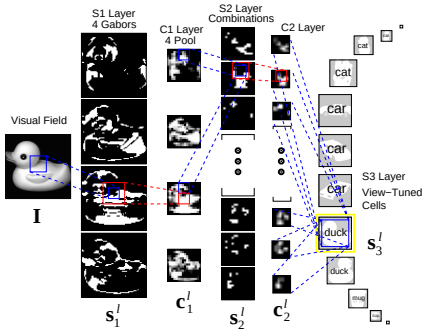

Fig. 1. Sketch of hierarchical network.

lution of 16×16 for each of the 4 S1 planes. The S2 layer contains combination coding cells with possible local connections to all of the C1 cells. The C2 layer pools the S2 planes down to a resolution of 8×8. The final S3 cells are tuned to particular views, which are represented as the activity pattern of the C2 planes for an input image.

The first processing stage consists of a convolution with 4 differently oriented first-order Gabor filters, a Winner-Take-Most (WTM) mechanism between these features and a final threshold function. We adopt the notation, that vector indices run over the set of neurons within a particular feature plane of a particular layer. To compute the response $s_1^l(x,y)$ of a neuron in the first layer S1, responsive to feature type l at position (x,y), first the image vector $\mathbf{I}$ is multiplied with a weight vector $\mathbf{w}_1^l(x,y)$ characterizing the receptive field profile:

$$q_1^l(x,y) = |\mathbf{w}_1^l(x,y) * \mathbf{I}|, \tag{1}$$

where the inner product is denoted by $*$, i.e. for a 10×10 pixel image, $\mathbf{I}$ and $\mathbf{w}_1^l(x,y)$ are 100-dimensional vectors. All neurons in a feature plane l have the same receptive field structure, given by $\mathbf{w}_1^l(x,y)$, but shifted receptive field centers, as in a classical weight-sharing architecture [11]. In a second step, a Winner-Take-Most mechanism is performed with

$$r_1^l(x,y) = \begin{cases} 0 & \text{if } \frac{q_1^l(x,y)}{M} < \gamma_1 \text{ or } M = 0, \\ \frac{q_1^l(x,y) - M\gamma_1}{1-\gamma_1} & \text{otherwise,} \end{cases} \tag{2}$$

where $M = \max_k q_1^k(x,y)$ and $r_1^l(x,y)$ is the response after the WTM mechanism which suppresses sub-maximal responses and provides a model of latency-based competition [10]. The parameter $0 < \gamma_1 < 1$ controls the strength of the competition. The activity is then passed through a simple threshold function with a common threshold θ_1 for all neurons in layer S1:

$$s_1^l(x,y) = \mathrm{H}\big(r_1^l(x,y) - \theta_1\big), \tag{3}$$

where $\mathrm{H}(x) = 1$ if $x \geq 0$ and $\mathrm{H}(x) = 0$ otherwise and $s_1^l(x,y)$ is the final activity of the neuron sensitive to feature l at position (x,y) in the S1 layer. The activities of the first layer of pooling C1-neurons are given by

$$c_1^l(x,y) = \tanh\big(\mathbf{g}_1(x,y) * \mathbf{s}_1^l\big), \tag{4}$$

where $\mathbf{g}_1(x,y)$ is a normalized Gaussian pooling kernel with width σ_1, identical for all features l, and *tanh* is the hyperbolic tangent function. The features in the intermediate layer S2 are sensitive to local combinations of the features in the planes of the previous layer, and are thus called *combination neurons* in the following. We also use the term *feature bank* to denote the set of features of a particular layer. We introduce the layer activation vector $\bar{\mathbf{c}}_1 = (\mathbf{c}_1^1, \ldots, \mathbf{c}_1^K)$ and the layer weight vector $\bar{\mathbf{w}}_2^l = (\mathbf{w}_2^{l1}, \ldots, \mathbf{w}_2^{lK})$ with K=4. Here $\mathbf{w}_2^{lk}(x,y)$ is the receptive field vector of the S2 neuron of feature l at position (x,y), describing connections to the plane k of the previous $C1$ neurons. The combined linear

summation over previous planes is then given by $q_2^l(x,y) = \bar{\mathbf{w}}_2^l(x,y) * \bar{\mathbf{c}}_1$. The weights of these combination neurons are a main target of our evolutionary optimization. After the same WTM procedure with strength γ_2 as in (2), the activity in the S2 layer is given by $s_2^l(x,y) = H(r_2^l(x,y) - \theta_2)$ after thresholding with a common threshold θ_2. The step from S2 to C2 is identical to (4) and given by $c_2^l(x,y) = \tanh(\mathbf{g}_2(x,y) * \mathbf{s}_2^l)$, with Gaussian spatial pooling kernel $\mathbf{g}_2(x,y)$ with range σ_2. The nonlinearity parameters $\gamma_1, \theta_1, \sigma_1, \gamma_2, \theta_2, \sigma_2$ will be subject to evolutionary optimization.

Classification of an input image with C2 output $\bar{\mathbf{c}}_2$ is done by nearest neighbor match to previously stored template activations $\bar{\mathbf{c}}_2^v$ for each training view v. This can be realized e.g. by view-tuned units (VTU) in an additional S3 layer with a radial basis function characteristics according to $s_3^v = \exp(-||\bar{\mathbf{w}}_3^v - \mathbf{c}_2||^2)$ where $\bar{\mathbf{w}}_3^v = \mathbf{c}_2^v$ is tuned to the training C2 output of pattern v. Classification can then be performed by detecting the maximally activated VTU.

3 Evolutionary optimization of the neural vision system

3.1 Evolution strategies

We employ a standard evolution strategy (ES) [12] with a semi-global step-size-adaptation with two different step-sizes to optimize the vision system. This turned out to be sufficient: one step-size for the 6 nonlinearity parameters and one for the combination feature weights, described in more detail in the following sections. In the case of the indirect coding, we need just one step-size since the combination features are optimized by the local learning process. We used discrete recombination for the 6 nonlinearity parameters. The strategy parameters were recombined by a generalized intermediate recombination. In our studies, we used the "ES-typical" deterministic (μ, λ) selection.

3.2 First and second order generalization

For the evaluation of the optimized vision systems we introduce the concept of first and second order generalization (Note that the more common terms test and validation error are not suitable, since we are working on different databases and not on two subsets of one database.), which is displayed in Figure 2. The flow of the evolutionary optimization of the hierarchical neural vision system is the following: We code the vision system into the chromosome (directly for the first two and indirectly for the next three settings). Then we apply evolutionary operators like mutation and recombination to the population. Thereafter, we construct the offsprings – different vision systems – and train these

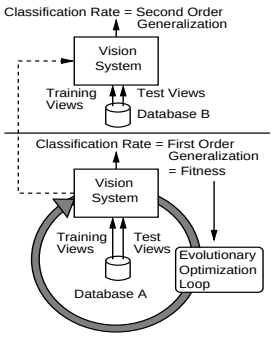

Fig. 2. Concept of first and second order generalization.

using a few views from objects of an image database A. Then we test the systems with the classification of test object views from database A, not contained in the training set. We use the classification rate as the fitness for the following selection of the parents, which constitute the next generation. After a sufficient number of generations we get vision systems which are well structured and successfully classify objects of database A. We call this performance *first order generalization*. With *second order generalization* we denote the ability of the system optimized on database A, to successfully classify objects from a database B, without any changes to features or nonlinearities.

3.3 Direct coding

In the representation of the vision system we differentiate between system nonlinearities and the combination features. The system nonlinearities are 6 parameters which efficiently characterize the quality of the nonlinear processing steps of the system. These are: 1. the WTM selectivities $\gamma_1, \gamma_2 \in [0, 1]$, which control the competition between the different features at the same image location within the same layer, 2. the threshold parameters $\theta_1, \theta_2 \in [0, 3]$, which control the number of neurons firing, and 3. the pooling ranges $\sigma_1, \sigma_2 \in [0.0001, 7]$, which control the sizes of the Gaussian pooling kernels used in layer C1 and C2. The parameters γ_1, γ_2, θ_1, θ_2, σ_1, σ_2 are coded as real values into the chromosome. Additionally, to the system nonlinearities the weights $\bar{\mathbf{w}}_2^l = (\mathbf{w}_2^{l1}, \ldots, \mathbf{w}_2^{l4})$, which define the combination feature bank, are directly coded into the chromosome, $l = 1, ..., L$, where L is the number of S2 feature planes. For comparison with the different indirect codings we explored two different domains for the weights: a non-negative one with $w_{2i}^{lk} \in [0, 1]$ and one with $w_{2i}^{lk} \in [-1, 1]$. The coding of the combination feature bank is organized as follows: We define the size of one feature of the combination feature bank $\bar{\mathbf{w}}_2^l \in I\!R^{36=4\times3\times3}$. Each of the 4 planes of layer C1 corresponding to four different local orientations in the image is convolved with a 3×3 filter. We define w_{2i}^{lk}, with $k = 1, ..., 4$, and $i = 1, ..., 36$ as the ith entry of $\mathbf{w}_2^{lk}$. The optimization was carried out with $L = 9, 36, 50$ features. With 50 features $50 \times 36 = 1800$ values have to be optimized. Thus the full optimization (including also the nonlinearities) took place in a 1806-dimensional $(1800 + 6 = 1806)$ search space.

3.4 Indirect coding

In the indirect coding approach, we still code the nonlinearities directly like described before but we use three different unsupervised local learning processes to determine the weights of the combination feature bank. These are the principal component analysis (PCA), the fast independent component analysis (fastICA) [13] and the non-negative sparse coding scheme (nnSC) [14]. The processes use 1440 randomly selected 3×3 pixel patches of the C1 layer (which contains 4 planes) to learn a filter bank. These patches therefore consist of 36 entries each $(3 \times 3 \times 4 = 36)$. The combination feature bank then consists of the L principal component vectors of the C1 patches for the PCA and of the L independent

component vectors in the case of the fastICA. In the case of the nnSC the bank is the basis for a sparse coding of the patches. In contrary to PCA and fastICA it can be an overcomplete set and $L > 36$ is therefore possible. This learning process is also controlled by a sparsity factor which determines the trade-off between sparsity and reconstruction ability of the patch inputs. We code this parameter in addition to the 6 nonlinearities into the chromosome and therefore perform the optimization in a just 7-dimensional search space. The space for the PCA and ICA optimization is 6-dimensional, as we have here no additional parameters which are needed. In the following, we briefly summarize the procedure of our indirect coding scheme from the genotype to the phenotype to the fitness evaluation: For each genotype in the population do:

1. Construction of the phenotype up to the C1 layer.
2. Generation of C1 layer activations using the database A.
3. Collecting 3×3-patches of the activated C1 layer.
4. Use of unsupervised local learning for the generation of the combination feature bank using the patches (and the sparsity parameter, which is explicitly coded in the genotype, for the case of the nnSC).
5. Construction of the complete phenotype – the vision system – with all nonlinearities and the combination feature bank.
6. Training of the vision system with training views of database A (storing C2 activations as a VTU for each training view)
7. Calculation of the classification rate using test views of database A in a nearest-neighbor classification based on C2 feature output. The result is the fitness of the individual.

4 Results and Discussion

For the evolutionary optimization of the combination features and nonlinearity parameters we used the object database COIL20 [15]. This database contains 20 different objects with 72 images of varying angles of rotation in depth, reaching from 0 to 360 degrees in 5 degree steps. After the vision system is generated according to the parameters in the chromosome, it is trained with 3 views (0, 120 and 240 degrees) of each object, by simply storing the highest order C2 feature activation vectors of each training view (see Section 2). In the test phase, the vision system has to classify 24 remaining views, which are equally distributed between 0 and 360 degrees. These test views are matched in a nearest-neighbor fashion to the stored training vectors. We note that other classifiers like linear discriminators could also be applied, see [10], but the nearest-neighbor approach has the advantage of not requiring an additional weight adaptation on the view-tuned units. The target of the optimization is the determination of the nonlinearities and the combination features in a way that the system will have a minimal classification error after training, i.e., that the first order generalization of the hierarchical vision system is maximal. A further test for the system is the second order generalization. For this test, we use a subset of the COIL100

[15][1] database which contains 100 objects, also with 72 images of continually varying rotation angle. We have to note that 17 objects of the COIL20 are also objects of the COIL100 database. We excluded these objects to ensure a proper measure of the second-order generalization. We call this reduced database of 83 objects *COILselect*. In the following tests, we optimized each setting 10 times for 400 generations using a (7,19)-ES, which we identified to be a good setting in preliminary studies, considering the trade-off between evaluation time and performance. We ran 3 different settings with the number of features contained in the combination feature bank set to $L = 9, 36, 50$ [2].

The results are displayed in Table 1. Note that we focus in the following discussion of the results on the average, and not on the best results achieved with a setting. When we compare the performances with respect to the first order generalization (the errors on the COIL20), we find that the direct coding outperforms the indirect one significantly (with respect to the Student-t test). This can be explained by the fact that the number of degrees of freedom is in the direct case much higher than in the indirect case and therefore the vision system could be adapted to perform particularly well on database A. Among the direct codings the one which allows also negative values for the combination features (denoted with "neg. CF") performs better than the one with non negative values (denoted with "non neg. CF") for the same reason. The possibilities of the vision system with negative values are in that sense higher, that not only the existence of features could be combined, but also the absence of features. The neg. CF setting again performs best with $L = 36$ features. Only 9 features seem to be too few to represent the objects properly. With 50 features the performance degrades compared to 36 because of the larger search space dimension.

Table 1. Results of directly and indirectly coded evolutionary optimization. L=number of features, b=best result, m=mean, and s=standard deviation of 10 runs.

direct coding		error COIL20			error COILsel.		
	L	b	m	s	b	m	s
non-neg. CF	9	7.9	8.6	0.5	24.2	27.6	3.0
	36	7.7	8.3	0.5	23.2	26.5	2.2
	50	7.3	8.6	0.8	23.2	24.8	1.5
neg. CF	9	**6.5**	8.1	1.2	22.8	26.3	2.4
	36	7.1	**7.8**	0.5	22.9	24.2	1.8
	50	7.1	8.1	0.7	22.4	24.3	1.7

indirect coding		error COIL20			error COILsel.		
	L	b	m	s	b	m	s
PCA	9	9.4	10.6	1.1	23.4	25.1	1.2
	36	8.1	9.6	1.1	23.6	25.8	3.1
fast ICA	9	8.5	9.4	0.7	24.4	26.7	1.8
	36	8.8	9.7	1.0	22.5	24.7	2.8
nnSC	9	9.0	10.0	0.6	24.1	26.5	1.6
	36	8.5	9.7	1.3	22.4	**24.1**	1.4
	50	8.8	9.5	0.6	**21.7**	24.2	1.4

[1] We converted the color images of the COIL100 database to grey value images and scaled them down to 64x64 pixels.

[2] except for the indirect coding with PCA and fastICA where L must be less or equal to the data vector dimension.

We now focus on the errors on the COILselect database which give us a measure of how good the vision system performs on an arbitray database - the second order generalization. We find that now both main settings (direct and indirect coding) perform almost equally well. The advantage of the indirect coding is not significant. From this result, we can draw the conclusion, that the unsupervised learning processes have enough freedom and are capable of building up a high performing vision system with regard to second order generalization. Among the indirect coding settings the nnSC performs best, although only positive values for the combination features are allowed in this method. Equally to the first order generalization also here 36 features seem to be adequate as a trade off between adaptability and increasing the search space too much.

In order to understand more about the properties of the optimization problem, it is interesting to find out whether similar sets of combination features have been found during different optimization runs. For a comparison of two sets of combination features we have to define a distance measure D_{CF} which is invariant under permutation of single features (as these permutations have no effect on the fitness). D_{CF} is defined as follows: Starting with the first feature of one of the two CF-sets we search for the closest (in Euclidean metric) feature in the second CF-set. We measure the distance d_1 between these two features and exclude both of them from the sets. Then we repeat this procedure with the next feature of the first set and so forth. To have a symmetric measure, we repeat the process with the two sets interchanged. The resulting distance measure D_{CF} is equal to the sum of all calculated distances divided by two: $D_{CF} = \frac{1}{2}\sum_{i=1}^{2L} d_i$, where L denotes the number of combination features in each set. Using the described distance measure we observed, that there exist a large number of significantly different CF-sets that have similar performance.

In the following we shortly discuss the case: direct coding; $L = 9$; negative CFs. In order to be able to interpret the absolute distance values, we take the best vision system and add Gaussian noise to its CFs. We do this ten times and for three different noise levels. After that, we measure D_{CF} and the misclassification rates of the changed system. For Gaussian noise with a standard deviation of $\sigma_{noise} = 0.025$ the mean distance measure is $\bar{D}_{CF} = 1.4$ and the misclassification rate increased on average from 6.5% to 7.7% (i.e. $\Delta\bar{f} = 1.2$). For $\sigma_{noise} = 0.05(0.1)$, we get $\bar{D}_{CF} = 2.6(5.2)$ and $\Delta\bar{f} = 1.6(3.2)$. Then we calculate $\bar{D}_{CF} = 40.1$ for all pairs of two of the best CF-sets out of the 10 optimization runs. Next, we derive $\bar{D}_{CF}$ for 10 randomly generated CF-sets and get $\bar{D}_{CF} = 40.5$. Comparing the similarities of optimized and random features, we conclude, that the optimized CFs have almost no similarity to each other and seem to be widely spread over the whole parameter space.

To assess the performance of the best result of the indirectly coded evolutionary optimization, we have performed a comparison to a previous, manually tuned version of the vision system (mtVS) [14], and to other state-of-the-art systems. Here we use the results of Roobaert & van Hulle [16], who performed an extensive study on the COIL100 database, comparing support vector machines (SVM), and the eigenspace-based system of Nayar et al. [15] (denoted

Columbia in the table). The results are shown in Table 2, where the number of objects and the number of training views is varied (for less than 100 objects the first n objects are taken). We see, that the evolutionary optimization could effectively improve the performance of the manually tuned vision system (mtVS). Compared to other classifiers the optimized vision system (optVS) is highly competitive and shows superior performance especially in the most difficult cases of the task, where only few training views are available and a high number of objects have to be classified. The results of the nearest-neighbor classifier based on the plain image data (NNC) illustrate the baseline similarity of the images in the database.

Table 2. Comparison of misclassification rates on COIL100 database.

	30 Objects			4 Training Views		
	Training Views			Number of Objects		
Method	36	8	2	10	30	100
NNC	0	7.5	29.5	13.5	18.2	29.9
Columbia	0	4.4	32.9	7.9	15.4	23.0
SVM	0	4.8	29.0	9.0	15.1	25.4
mtVS	0	7.3	28.3	18.4	15.8	23.9
optVS	0	4.4	22.9	12.4	12.9	20.2

5 Conclusion

The work presented here is the first study of the evolutionary optimization of a biologically inspired vision network, which is capable of performing a complex 3D real world object classification task. We compared the optimization using a direct and an indirect coding of the combination feature bank. We showed that the used biologically inspired hierarchical architecture has a very robust behavior, where a lot of different combination feature banks are equally well suited for classification. Therefore, the directly coded evolutionary optimization found good results with a good convergence behavior despite the huge dimensionality of the search space for the direct coding with $L = 50$ features (1806-dimensional). Considering second order generalization, the results are even better than the ones for only 9 features (330-dimensional). For the more difficult COILselect database, 36 or even 50 filters for the combination feature bank seem more adequate [14] and the drawback of a harder optimization is compensated by the enhanced representational capability of the network.

We found, that the coupling of evolutionary search with unsupervised local learning processes yields good results. Hereby we can realize a biologically more sensible encoding and work in a 6 respectively 7 dimensional search space (compared to over 1800 before). Comparing direct and indirect coding, we find that the direct evolutionary optimization yields significantly better results in the first order generalization. This performance advantage stems from specialization to

the database used during evolution as the performance gain cannot be observed for the second order generalization. Here we see even a slight advantage for the indirect coding, which, however, does not have high statistical significance. We also showed that the optimized architecture is highly competitive with other current high-performing recognition methods like support vector machines.

Acknowledgment This work was supported by the BMBF under grant LOKI 01IB001E.

References

1. X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
2. M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.
3. H. B. Barlow, "The twelfth Bartlett memorial lecture: The role of single neurons in the psychology of perception," *Quart. J. Exp. Psychol.*, vol. 37, pp. 121–145, 1985.
4. B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1 ?" *Vision Research*, vol. 37, pp. 3311–3325, 1997.
5. H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*, vol. 4, pp. 461–476, 1990.
6. E. T. Rolls and S. M. Stringer, "On the design of neural networks in the brain by genetic evolution," *Progress in Neurobiology*, vol. 6, no. 61, pp. 557–579, 2000.
7. B. Sendhoff and M. Kreutz, "A model for the dynamic interaction between evolution and learning," *Neural Processing Letters*, vol. 10, no. 3, pp. 181–193, 1999.
8. S. Quartz and T. Sejnowski, "The neural basis of cognitive development: A constructivist manifesto," *Behavioral and Brain Sciences*, vol. 9, pp. 537–596, 1997.
9. A. G. Rust, R. Adams, S. George, and H. Bolouri, "Towards computational neural systems through developmental evolution," in *LNCS*, S. W. et al., Ed., vol. 2036, 2001, pp. 188–202.
10. H. Wersing and E. Körner, "Learning optimized features for hierarchical models of invariant recognition," *Neural Computation*, vol. 15, no. 7, pp. 1559–1588, 2003.
11. K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cyb.*, vol. 39, pp. 139–202, 1980.
12. H.-P. Schwefel and G. Rudolph, "Contemporary evolution strategies," in *Proc. of the Third European Conf. on Artificial Life : Advances in Artificial Life*, ser. LNAI, F. M. et al., Ed., vol. 929. Berlin: Springer Verlag, June 1995, pp. 893–907.
13. A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Computation*, vol. 9, no. 7, pp. 1483–1492, 1997.
14. H. Wersing and E. Körner, "Unsupervised learning of combination features for hierarchical recognition models," in *Int. Conf. Artif. Neur. Netw. ICANN*, J. R. D. et al., Ed. Springer, 2002, pp. 1225–1230.
15. S. K. Nayar, S. A. Nene, and H. Murase, "Real-time 100 object recognition system," in *Proc. of ARPA Image Understanding Workshop*, Palm Springs, 1996.
16. D. Roobaert and M. V. Hulle, "View-based 3d object recognition with support vector machines," in *Proc. IEEE Int. Workshop on Neural Networks for Signal Processing, Madison,USA*. New York, USA: IEEE, 1999, pp. 77–84.