Genetic Encoding Strategies for Neural Networks

Philipp Köhn
University of Tennessee — Universität Erlangen-Nürnberg
Dreibergstr. 5, 91056 Erlangen, Germany
kohn@cs.utk.edu

Abstract

The application of genetic algorithms to neural network optimization (GANN) has produced an active field of research. This paper proposes a classification of the encoding strategies and it also gives a critical analysis of the current state of development.

The idea of evolving artificial neural networks (NN) by genetic algorithms (GA) is based on a powerful metaphor: the evolution of the human brain. This mechanism has developed the highest form of intelligence known from scratch. The metaphor has inspired a great deal of research activities that can be traced to the late 1980s (for instance [15]). An increasing amount of research reports, journal papers and theses have been published on the topic, generating a continously growing field.

Researchers have devoloped a variety of different techniques to encode neural networks for the GA, with increasing complexity. This young field is driven mostly by small, independet research groups that scarcely cooperate with each other. This paper will attempt to analyse and to structure the already performed work, and to point out the shortcomings of the approaches.

1 The Problem of GANN

Neural networks proved to be a powerful problem solving mechanism with the ability to learn. The success and speed of training, however, is based on the initial parameter settings, such as architecture, initial weights, learning rates, and others. In GANN systems, the genetic algorithm is used to find the optimal parameter settings for a given task.

We can distinguish two seperate issues for the genetic algorithm: on the one hand architecture optimization, and on the other hand weight training. A few GANN systems focus solely on weight training [27], wheras others focus on architecture [23]. There are also many ways to combine the two issues: The GA may set initial weight information while weight training is left to established learning rules for NN, such as back–propagation [26]. Or, the evolution of the NN can take place in two stages: One that finds the optimal architecture, the other that finds a weight setting for each architecture [10].

2 Weight encoding

A few different complexities of weight encoding have been investigated in this field. The simplest case is one–bit connectivity information [27]. Or, two bits are used for encoding if a connection is disconnected, inhibitory or excitatory [19]. Weights have been encoded as real numbers [21, 22], bit strings [20, 27], or bit strings using grey coding [13]. Maniezzo proposes that the number of encoding bits should increase during the evolution [17], allowing fine tuning at a later stage.

A study suggests that a high number of encoding bits per weight improves GA weight training, whereas grey coding has no impact [13].

3 Architecture encoding

The task to encode the architecture of a network is far more complex than weight encoding. The classical view that the genome is a bit string of fixed length does not easily fit to the complex interconnected structure of a NN. Thus, most recent researchers are using more complex data types.

The different encoding strategies can be classified by their smallest defining unit, or the “allele” [26]: connections, nodes, layers or pathways, or they can be indirectly encoded which is a a completely different approach. The classes are roughly ordered to the complexity of the strategies.

Connection–based encoding – The vast majority of early approaches is connection–based [13, 17, 19, 21, 27]. Here, the genome is a string of weight values or pure connectivity information. This requires a fixed maximal architecture, which is typically either fully–connected or layered.

Node–based encoding – An advantage over connection–based encoding is that more flexibility can be obtained by using nodes as basic units. In this approach, the genome is a string or tree [14] of node information. The code for each node may include relative position, backward connectivity [24], weight values, threshold function [26] and more. Crossover and mutation is mostly restricted to cuts between node information.

Layer–based encoding – With layer–based encoding one can obtain larger networks [8, 9, 16]. The encoding scheme is a very complicated system of descriptions of connectivity between a list of layers. Therefore, special GA operators are required.

Pathway–based encoding – Pathway–based encoding is proposed in [10] for recurrent NN. In this instance, the network is viewed as a set of paths from an input to an output node. Again, special GA operators are used.

Indirect encoding – This completely seperate strategy can be traced back to 1990 [12], with more recent approaches [1, 4] receiving great acclaim in the community. The basic idea is to encode a grammar–based construction program in the genome, instead of directly encoding properties of the NN. Boers and Kuiper used a grammar based on Lindenmayer systems, Gruau applied cellular encoding.

Basically all encoding techniques developed so far, can be classified into one of these classes.

If the genomes are too large, the efficiency of the GA decreases [26], and this problem has to be addressed when larger networks are involved. More complex encoding strategies are motivated by this lack of scalability, because not every connection or node has to be individually represented in the genome.

With increasing complexity and inhomogenity of the genoms, the classical GA operators mutation and crossover are replaced by parameter–oriented operators such as "randomly add neuron" [23], or "drop path" [10].

4 Tasks for GANN systems

Thanks to mutation operators, every optimal NN will be ultimately found by any GANN system. The quality of an encoding strategy, however, has to be measured by the speed of convergence to sufficiently good solutions.

Most papers report good results on small–scale toy problems such as xor, sine, parity, or low–bit adding. The convergence time competes well with back–propagation [28, 13].

The largest problems that have been tackled successfully range from high–bit parity [4], pole–balancing [28, 10], and simplified pattern recognition [1] to high–order classification tasks [26]. If convergence times are reported at all, they are usually quite high: 3 days on 11 workstations [1] or one week [26, in conversation], [10, in conversation] for the mentioned tasks. These are tasks that can be solved faster by simple back–propagation on arbitrary NN chosen by rules of thumb. So, some doubts have to be raised regarding the efficiency of recent GANN systems.

5 Fundamental Problems

In order to improve the performance of GANN systems, some of the fundamental problems of the approach must be resolved.

Lack of theoretical insight – In general, the field lacks theoretical insight. It is rare that general criteria for encoding strategies are proposed, such as by Gruau [4]: completeness (every NN can be encoded) and closure (only meaningful NN are encoded) among others. These are fulfilled by almost all strategies.

Also, there have been only a few properties of GANN systems to be identified and discussed. The argument over the Baldwin effect [5] or the permutation problem (see below) has just started.

Permutation problem – Almost always, the same (or similar) networks may be encoded in quite different ways. The crossover of two different encodings of similar networks can result in a completely divergent NN. This broadly acknowledged problem is also known as structural–functional mapping problem [27] or competing conventions problem [6]. There have been a few proposals to solve it [25, 11, 6]. However, these studies also indicate that the problem has less impact than expected.

GA parameters – There are numerous parameters for the genetic algorithms, the neural networks and the GANN systems that have to be set: population size, mutation and crossover rates, the use of distributed GA, fitness evaluation, learning rate, momentum term, number of epochs, activation functions, encoding details, et cetera.

Although many researchers describe the benefit of the application of GA to NN as the avoidance of rules of thumb in NN design, rules of thumbs and time–intensive experimental optimizations have to be used for these parameters [13]. Still, it is not at all clear, if these parameters are robust or if they depend on the specific problems. In a few of the GANN systems, some of these parameters are encoded in the genome, for instance: the learning rate in [16] or number of encoding bits per weight in [18]. This encoding solves part of the problem at the cost of an increase of the search space and thus the convergence time.

Comparison of encoding techniques – The inexistence of a theoretical account makes the comparison of different GANN systems difficult. The empirical results, however, do not clearly indicate that more complex systems show a better performance.

The fundamental problem of the poor performance of GANN systems on real–world scale tasks has been recognized [28]. One magic word for the solution is modularity [7], but unfortunately encoding strategies with high modularity like indirect encoding, have not been a breakthrough.

Another idea is to partially develop NNs for subtasks and compose them to a more complex structure. This was investigated for different parts of the nervous system of an artificial being [2, 3].

At this stage, however, the efficient application of GANN systems seems to be restricted to (a) weight training problems where no error information is available, such as for the pole–balancing problem, or the recurrent networks, and (b) architecture optimization for classes of problems that can use equal NN architectures.

Acknowledgements

This paper is based on parts of my Masters thesis at the University of Tennessee, Knoxville. I thank my advisor Bruce MacLennan for his support. Φ

References

[1] Egber Boers and Herman Kuiper. Biological metaphors and the design of modular artificial neural networks. Master's thesis, Leiden University, the Netherlands, 1992.

[2] Hugo de Garis. Genetic programming. In *International Joint Conference on Neural Networks*, pages III 511–516. IEEE, 1990.

[3] Hugo de Garis. Circuits of production rule: Gennets — the genetic programming of artificial nervous systems. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 699–705, Innsbruck, 1993.

[4] Frederic Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, Ecole Normale Superieure de Lyon, 1994. ftp://lip.ens-lyon.fr /pub/Rapports/PhD/PhD94-01-E.ps.Z.

[5] Frederic Gruau and Darell Whitley. Adding learning to the celular development of neural networks: evolution and the Baldwin effect. *Evolutionary Computation*, 3(1):213–233, 1993.

[6] Peter J.B. Hancock. Genetic algorithms and permutation problem: a comparison of recombination operators for neural net structure specification. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 108–122, Baltimore, 1992. IEEE.

[7] Bart L.M. Happel and Jacob M.J. Murre. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6/7):985–1004, 1994.

[8] Steven Alex Harp and Tariq Smad. Towards the genetic synthesis of neural networks. In *Third International Conference on Genetic Algorithms*, pages 360–369. Morgan Kaufmann, 1989.

[9] Steven Alex Harp and Tariq Smad. Genetic synthesis of neural network architecture. In *Handbook of Genetic Algorithms*, pages 202–221, 1991.

[10] Christian Jacob and Jan Rehder. Evolution of neural network architectures by a hierarchical grammar–based genetic system. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 72–79, Innsbruck, 1993.

[11] Nachimuthu Karunanithi, Rajarshi Das, and Darell Whitley. Genetic cascade learning for neural networks. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 134–145, Baltimore, 1992. IEEE.

[12] Hiroaki Kitano. Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4:461–476, 1990.

[13] Philipp Koehn. Combining genetic algorithms and neural networks: The encoding problem. Master's thesis, University of Tennessee, Knoxville, 1994. ftp://archive.cis.ohio-state.edu /pub/neuroprose/koehn.encoding.ps.Z.

[14] John R. Koza and James P. Rice. Genetic generation of both the weight and architecture for a neural network. In *International Joint Conference on Neural Networks*, pages II 397–404. IEEE, 1991.

[15] Steve Lehar and John Weaver. A developmental approach to neural network design. In *International Conference on Neural Networks*, pages II 97–104. IEEE, 1994.

[16] Martin Mandischer. Representation and evolution of neural networks. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 643–649. Innsbruck, 1993.

[17] Vittorio Maniezzo. Searching among search spaces: hastening the genetic evolution of feedforward neural networks. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 635–642, 1993.

[18] Vittorio Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions of Neural Networks*, 5(1):39–53, 1994.

[19] Leonardo Marti. Genetically generated neural networks II: searching for the optimal representation. In *IEEE Joint Conference on Neural Networks*, pages II 221–226, 1992.

[20] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hedge. Designing neural networks using genetic algorithms. In *Third International Conference on Genetic Algorithms*, pages 379–384. Morgan Kaufmann, 1989.

[21] D. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *11th International Joint Conference on Artificial Intelligence*, pages 762–767. Morgan Kaufmann, 1989.

[22] David J. Montana. Automated parameter tuning for interpretation of synthetic images. In *Handbook for Genetic Algorithms*, pages 282–311, 1991.

[23] Wolfram Schiffmann, Merten Joost, and Randolf Werner. Synthesis and performance analysis of neural network architectures. Technical Report 16/1992, Universität Koblenz, Germany, 1992. ftp://archive.cis.ohio-state.edu /pub/neuroprose/schiff.nnga.ps.Z.

[24] Wolfram Schiffmann, Merten Joost, and Randolf Werner. Application of genetic algorithms to the construction of topologies for multilayer perceptrons. In *International Joint Conference on Neural Networks and Genetic Algorithms*, pages 675–682, Innsbruck, 1993.

[25] M. Srinivas and L.M. Patnaik. Learning neural network weights using genetic algorithms — improving performance by search space reduction. In *International Joint Conference on Neural Networks*, pages 2331–2336. IEEE, 1991.

[26] David W. White. *GANNet: A Genetic Algorithm for Searching Toplogy and Weight Spaces in Neural Network Design.* PhD thesis, University of Maryland, 1993.

[27] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14:347–361, 1990.

[28] Darell Whitley, Stephen Dominic, Rajarshi Das, and Charles W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284, 1993.