Artificial Intelligence Review (2022) 55:1723–1802
https://doi.org/10.1007/s10462-021-10049-5

Evolutionary design of neural network architectures: a review of three decades of research

Hamit Taner Ünal[1] · **Fatih Başçiftçi**[2]

Accepted: 17 July 2021 / Published online: 27 July 2021

Abstract
We present a comprehensive review of the evolutionary design of neural network architectures. This work is motivated by the fact that the success of an Artificial Neural Network (ANN) highly depends on its architecture and among many approaches *Evolutionary Computation*, which is a set of global-search methods inspired by biological evolution has been proved to be an efficient approach for optimizing neural network structures. Initial attempts for automating architecture design by applying evolutionary approaches start in the late 1980s and have attracted significant interest until today. In this context, we examined the historical progress and analyzed all relevant scientific papers with a special emphasis on how evolutionary computation techniques were adopted and various encoding strategies proposed. We summarized key aspects of methodology, discussed common challenges, and investigated the works in chronological order by dividing the entire timeframe into three periods. The first period covers early works focusing on the optimization of simple ANN architectures with a variety of solutions proposed on chromosome representation. In the second period, the rise of more powerful methods and hybrid approaches were surveyed. In parallel with the recent advances, the last period covers the *Deep Learning Era*, in which research direction is shifted towards configuring advanced models of deep neural networks. Finally, we propose open problems for future research in the field of neural architecture search and provide insights for fully automated machine learning. Our aim is to provide a complete reference of works in this subject and guide researchers towards promising directions.

Keywords Artificial neural networks · Evolutionary computation · Machine learning · Artificial intelligence · Optimization

✉ Hamit Taner Ünal
htaner.unal@gmail.com

[1] Institute of Natural and Applied Sciences, Department of Information Technologies Engineering, Selçuk University, Konya, Turkey

[2] Faculty of Technology, Department of Computer Engineering, Selçuk University, Konya, Turkey

1 Introduction

Artificial Neural Network (ANN) is a computational machine learning model loosely inspired by the human brain. It is typically composed of several processing units (neurons) interconnected in a layered structure (Haykin 1993). This model can demonstrate human-like skills such as image recognition and natural language processing. The learning is established through training of the network with the help of structured data and the utilization of learning algorithms.

Since the invention of *Mark I Perceptron*, the first ANN model by Frank Rosenblatt in 1958 (Rosenblatt 1958), Artificial Neural Networks have been transformed from single-layer models into complex structures consisting of hundreds or even thousands of layers in various architectures. Thus, they have been called *Deep Neural Networks*. The process of training these deep networks is called *Deep Learning*. Thanks to the recent availability of the massive amount of data (Big Data) and advancements in the technology of Graphic Processing Units (GPU), modern deep learning architectures surpass human performance by achieving state-of-the-art results on image classification tasks which helped develop revolutionary technologies such as self-driving cars and cancer diagnosis from x-ray images (Abdel-Zaher and Eldeib 2016; Levine et al. 2019; Rashed and El Seoud 2019; Spielberg et al. 2019).

Extensive experimental data reveal that the success of a neural network for solving a particular problem essentially depends on its architecture (Weiß 1994a). From a simple ANN model to today's highly complex deep structures, designing artificial neural networks is rather a difficult and troublesome task. Even today, network architectures are usually determined manually by domain experts through trial and error. Furthermore, the relationship between network architecture and its performance cannot be formulated. Considering the vast computational resources and amount of time required to search for possible neural architectures, manual methods are undoubtedly infeasible to obtain optimal solutions. This motivated researchers to employ advanced algorithms such as *metaheuristics* to automate this process and improve network performance with better architectures.

It would be quite demanding to conduct a review that examines the optimization of Artificial Neural Network design from a broad spectrum, covering all types of solution methods including metaheuristics and other advanced algorithms. Among many approaches, *Evolutionary Computation*, a set of global-search techniques inspired by the evolution theory became the most popular, offering promising and competitive solutions on a wide range of real-world tasks. For this reason, this review will narrow down the research and our focus will only be on the works that concentrated on combining artificial neural networks and evolutionary algorithms, which are two powerful paradigms of *Artificial Intelligence* (AI).

The first studies aiming to design network architectures with evolutionary methods start in the late 1980s. Over the past 30 years, considerable progress has been achieved. To this end, we made a thorough research and surveyed all relevant papers in this period. By examining the historical progress, we analyzed studies in chronological order and divided the whole timeframe into three periods based on significant achievements and scientific trends. The first period covers initial attempts to evolve simple ANN architectures in a competitive nature to invent efficient strategies for chromosome representation. The second period starts with the introduction of the Neuroevolution of Augmenting Topologies (NEAT) proposed by Stanley and Miikkulainen (2001). NEAT was considered to be a major breakthrough and a key milestone in this field. The second period involves many attempts to improve or outperform NEAT from various aspects. The third period covers the

Deep Learning Era when researchers explored methods to automate the design and configuration of deep neural networks. Figure 1 shows the number of papers published on the evolutionary design of neural network architectures throughout the investigated period and how the focus has been shifted from simple ANN models to Deep Neural Network (DNN) architectures. The works include, but are not limited to journal articles, conference papers, and dissertations. We aimed to include all relevant papers without any selection criteria, such as the number of citations, the Journal's impact factor, etc., and exhaustively searched through all databases available by double-checking with interim review papers in the historical context.

The primary purpose of this study is to present all innovative works by examining novel evolutionary approaches adopted in the design of artificial neural network architectures and to analyze solution strategies comparatively with a special emphasis on various evolutionary computation techniques adopted and the encoding strategies proposed. As such, it has a complementary nature to previous studies. Due to the surge of interest in the subject, many review papers have been published in various intervals until today. The first review paper was published in 1992 by Schaeffer (1992), who examined the early steps and surveyed approaches for encoding strategies. Later on, quite extensive reviews were carried out by Yao in the first decade (1993; 1998; 1999). Further reviews have been published covering up-to-date surveys and comparative analysis (Azzini and Tettamanzi 2011; Balakrishnan and Honavar 1995; Branke 1995; Cantú-Paz and Kamath 2005; Castellani 2013; Castillo et al. 2003; Castillo et al. 2007; de Campos et al. 2015; De Campos et al. 2011; Drchal and Šnorek 2008; Floreano et al. 2008; Vonk et al. 1995b; Weiß 1993; Weiß 1994a; Weiß 1994b; Whitley 1995). The most recent surveys are by Ojha et al. (2017), Chiroma et al. (2017) Stanley et al. (2019), and Baldominos et al. (2020). Due to a shift of interest from conventional neural models to deep architectures, some of the latest surveys concentrate mostly on *Neural Architecture Search* (NAS) methods recently being developed (Elsken et al. 2018b; Wistuba et al. 2019). Although these works provide comprehensive analysis, only a few of these reviews cover the whole spectrum of historical progress. Furthermore, there are still papers that are ignored, not sufficiently examined, or not compared in terms of encoding strategies and various techniques adopted. Despite being recently published, the review paper by Baldominos et al. (2020) doesn't sufficiently cover the latest advances in the evolutionary design of deep neural networks such as *AmoebaNet-A* by Real et al. (2019).

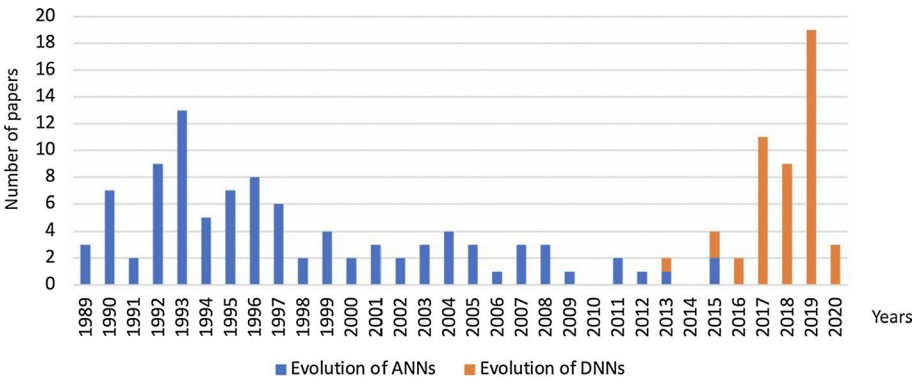


Fig. 1 Number of Papers Published on Evolutionary Design of Neural Network Architectures (1989–2020)

The rapid increase of interest in this subject requires more frequent updates since significant achievements with state-of-the-art results have been reported in the last two years, by utilizing evolutionary approaches. Published review papers are depicted on a timeline in Fig. 2.

To summarize, this paper presents an extensive survey on the evolutionary design of neural network architectures with the following contributions:

- We provide a detailed and systematic review of evolutionary approaches for searching optimal neural network architectures, covering the complete spectrum of historical progress.
- We examine the use of various evolutionary computation techniques such as *Genetic Algorithms* or *Evolutionary Programming* and analyze genetic operations, population initialization methods, and evaluation techniques with a variety of fitness functions.
- We put a special emphasis on chromosome encoding strategies with a comparative analysis of direct and indirect representation approaches, since they have a significant effect on the performance of the optimization process.
- We surveyed not only simple ANN architecture optimization approaches but also recent advances on the evolutionary design of deep neural architectures such as *Convolutional Neural Networks* (CNN).
- We raise open questions for future research on reducing the computational cost of architecture search and providing systems to fully automate machine learning tasks without expert knowledge.

The rest of this review paper is organized as follows: In Sec. 2, we introduce Artificial Neural Networks with biological backgrounds and historical developments. In Sec. 3 we investigate optimization methodology and summarize Evolutionary Computation techniques together with genetic operators applied. In Sec. 4, we made a categorical classification of representation methods and surveyed various encoding strategies with common challenges such as *Competing Conventions Problem*. In Sec. 5, we investigated the historical progress in three periods of development, namely *early works*, *the rise,* and the *deep learning era*. Finally, we conclude the survey in Sec.6.

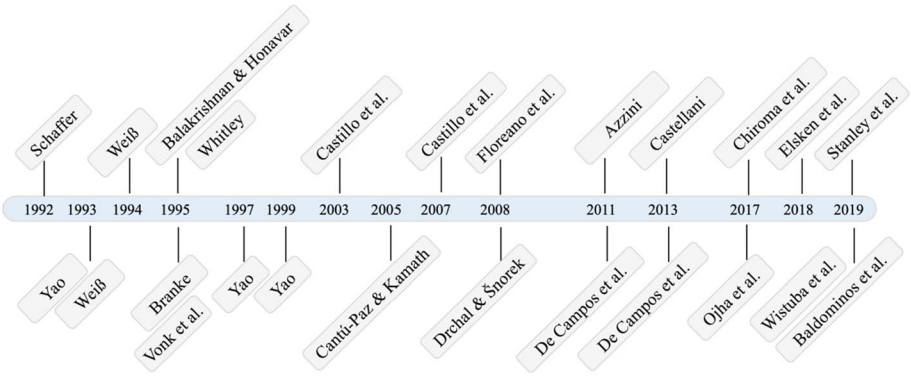


Fig. 2 Review Papers Published on Evolutionary Design of Neural Networks Between 1989 and 2020

2 Artificial neural networks

Artificial Neural Network is an advanced machine learning model inspired by the human brain (Haykin 1993). Designed as a simulation of biological nerve cells, it consists of several neurons interconnected in a layered structure and connections. ANN basically serves as a function for input–output mapping of a particular problem. A basic skeleton of an ANN has an input layer which acts as the collection point of sensors from the external world and an output layer that produces an output value as a function of received inputs. Between the input layer and the output layer, there are hidden layers with arbitrary depth and width, accommodating a determined number of processing elements (neurons) and connections. This part is usually considered as a "Black Box" since no one can explain the effect of its structure for a given problem. A typical Artificial Neural Network with two hidden layers and a single output is depicted in Fig. 3.

An artificial neuron can be defined as the weighted sum of incoming signals transformed by an activation function (Floreano et al. 2008) (Eq. 2). The connection between two neurons acts as a variable multiplier, commonly referred to as *synaptic weights*. The training is carried out to determine the best values for these weights. This mathematical model can generalize with the help of the sample data fed to it, thereby realizing learning-related skills such as classification and regression. The artificial neural network model with only one hidden layer can theoretically approximate any non-linear continuous function. With this feature, it is defined as a *Universal Function Approximator* (Cybenko 1989; Funahashi 1989; Hecht-Nielsen 1987; Hornik 1991; Kolmogorov 1957).

2.1 Biological Motivation

The human brain accommodates a huge network of biological nerve cells, called neurons. When this highly complex structure is examined closely, it can be seen that neurons are connected to other neurons through dendrites, synapses, and the axon. The signals obtained from the input unit called *dendrites* are processed inside the cell and transferred to other neurons with the help of *axons* and *synapses* (Fig. 4). The nerve cell to which the signal is transferred likewise transfers the signal transmitted to it to the next neuron. Neurons that act as a kind of "*activation*" sometimes strengthen the signals they receive by transferring them to the next neuron (*excite*) and sometimes stop it by inhibiting (*all-or-none*).

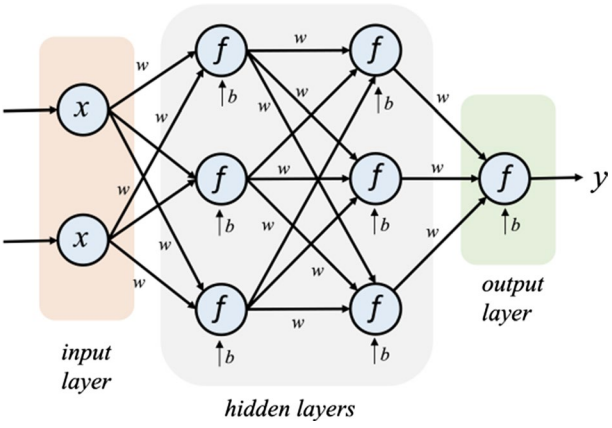


Fig. 3 A typical Artificial Neural Network with two hidden layers and a single output

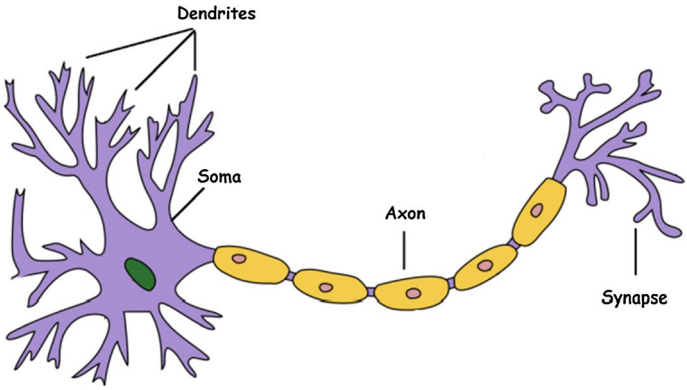


Fig. 4 Biological Nerve Cell

In 1943, McCulloch and Pitts (1943) laid the foundations of Artificial Neural Networks by creating a model of the biological nerve cell (Fig. 5). In 1958, Frank Rosenblatt invented the machine called Mark I Perceptron (Rosenblatt 1958). In the Perceptron project funded by the American Navy, Rosenblatt aimed to recognize and classify simple geometric shapes by mechanically creating artificial neurons. Despite the simplicity of perceptron, the project has been described by the *New York Times* as the "embryo of an electronic computer that will be able to see, speak, write, walk, multiply and be conscious of its existence '' in the near future (Baldominos et al. 2020).

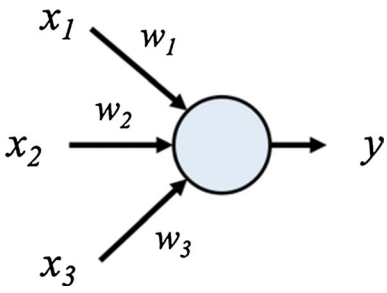
The artificial neurons in Perceptron can be defined as a binary device with a threshold. It receives inputs from excitatory or inhibitory synapses. The neuron becomes active if the sum of weighted inputs exceeds its threshold. It can also be expressed as a function that maps its input x to an output value $f(x)$:

$$f(x) = \begin{cases} 1 \; if \; w \cdot x + b > 0, \\ 0 \; otherwise \end{cases} \tag{1}$$

where w is a vector of weights and $w \cdot x$ is the dot product of $\sum_{i=1}^{m} w_i x_i$, where m is the number of inputs, and b is the bias (Fig. 6).

Networks, where activation is started from the inputs and flowed through hidden layers and towards output, is called a *feedforward neural network*. Likewise, the output of

Fig. 5 McCulloch-Pitts Neuron



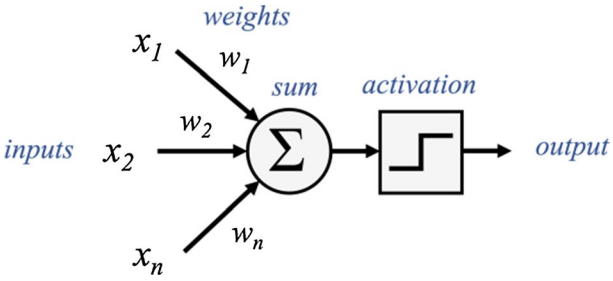


Fig. 6 Rosenblatt's Perceptron, 1958 (with step activation)

a neuron in a feed-forward neural network can be determined as the sum of its weighted inputs squashed with an activation function:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2}$$

where $x_1, x_2, ...x_n$ are input signals, $w_1, w_2, ...w_n$ are connection weights, b is the bias, and f is the activation function (e.g., sigmoid, tanh, etc.).

Some networks may have a feedback loop, where output is redirected to its input for discovering or responding to temporal dependencies (Fig. 7). These types of networks are commonly used for natural language processing and are called *Recurrent Neural Networks* (RNN) (Stanley 2004).

2.2 Multi-layer perceptron (MLP)

Although Rosenblatt's perceptron became popular and created excitement, it was only able to provide solutions to linear functions. In 1969, Minsky and Papert published an article called *Perceptrons* that proved the inadequacy of this model by revealing in all aspects that *Perceptron* could not approximate non-linear functions such as XOR (Minsky and Papert 1969). With this paper, the AI Winter, a period of great decrease of research and investments in artificial intelligence which would last until the mid-1980s, has started.

The development that made artificial neural networks popular again was the discovery of a gradient-based training method called *Backpropagation* (Rumelhart et al. 1986).

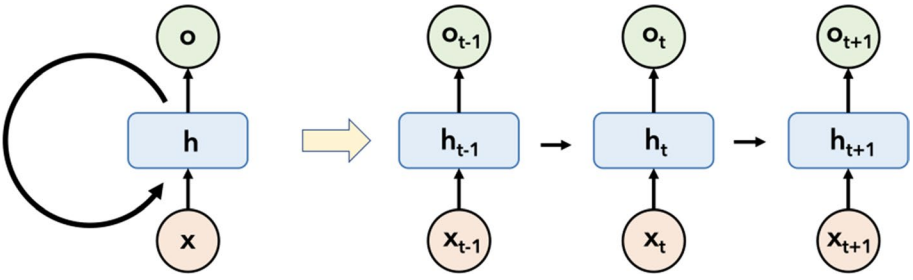


Fig. 7 A Recurrent Neural Network with a feedback connection. The diagram shows an RNN influencing its hidden state h with x as input and o as output for a sequence of time steps. It is typically used for natural language processing such as speech recognition or machine translation

Backpropagation updates the synaptic weights by taking the derivation of the network output error in multiple layers of artificial neural networks (also called *Multi-Layer Perceptron (MLP)*) based on the *delta rule* (Werbos 1974). It facilitates the training of Artificial Neural Networks in a very short time. This method is still the most commonly used network training method. Thus, multi-layered and fully feedforward neural networks became popular again as an effective machine learning model and started to produce solutions to many real-world problems, including non-linear functions. By the end of the 1980s many models have been developed, some of which are still in use today. *Hopfield networks* (Hopfield 1982), *Vector Quantization Models (LVQ)* (Kohonen 1995), *Adaptive Resonance Theory (ART)* (Carpenter and Grossberg 1986) *Self-organizing models* (SOM) (Kohonen 1989), *Elman Network* (Elman 1990), *Support Vector Machines* (Cortes and Vapnik 1995) and *Radial Based Networks* (Park and Sandberg 1991) have been introduced to the literature as different variants of Artificial Neural Networks.

2.3 Towards deep architectures

In 1980, Fukushima (1980) laid the foundations of Convolutional Neural Networks (CNN), with the *Neocognitron* inspired by Hubel and Wiesel's studies on neuroscience (1959, 1962). This model consisted of two layers, similar to the visual cortex in the brains of mammals. In the first layer, rough features of images such as corners and edges were detected, and in the second layer, more detailed processing and classification were carried out. Then, LeCun (1989, 1990a) introduced the first handwriting character recognition software in the late 1980s by using the MNIST dataset to train his model (Fig. 8). In addition, developments in the field of natural language processing (NLP) led to the development of advanced methods for processing Recurrent Neural Networks (RNN), bringing techniques such as LSTM in the late 90 s (Hochreiter and Schmidhuber 1997).

Fig. 8 Samples from the MNIST Dataset used for handwriting recognition. Sample digits for training were taken from American Census Bureau employees and for testing were taken from American high school students

Training of Artificial Neural Networks was a difficult and time-consuming process even in the 2000s when processor technology was making progress. As the number of layers in the network increases, so does the number of parameters to calculate, which required more processing power and higher memory in multi-layer structures called deep neural networks. The second AI Winter lasted until 2006, when Geoffrey Hinton, one of the pioneers of the field, published his groundbreaking work, showing that training of deep networks can be carried out in a more reasonable time with the structure called Deep Belief Nets (DBN). This development helped speed up the research on Artificial Neural Networks again and led to the blossom of the AI boom (Hinton et al. 2006; Hinton and Salakhutdinov 2006).

Overcoming obstacles in front of the Deep Neural Networks increased interest in this area and led to new research directions. Prof Fei Fe Li from Stanford University argued that the algorithms had reached maturity even many years ago, but the available datasets were still very poor. Fei Fei Li and his colleagues created a dataset consisting of thousands of categories and millions of images from the internet with the work they started in 2007, called ImageNet (Deng et al. 2009). This dataset was the largest dataset ever created in the world. The biggest feature that distinguishes ImageNet from other data sets was that the categories were hierarchically subdivided according to the *WordNet* system (Fig. 9).

As of 2009, a competition started to be organized for image recognition using the ImageNet dataset (ImageNet Large Scale Visual Recognition Challenge - ILSVRC). The researchers started to compete with the deep neural network models they developed to obtain the highest accuracy to recognize images in ImageNet, and this race led to one of the most important developments in the field of Deep Learning in 2012. Alex Krizhevsky and his colleagues have made the biggest leap in artificial intelligence by halving the error rate achieved on ImageNet with the deep neural network model they named AlexNet (Krizhevsky et al. 2012). No such improvement was expected because the researchers estimated that the average error rate of around 25% could improve by 1% each year. With the model they developed, AlexNet made a great leap of advance which could take approximately 12 years. The model they created implemented many recent innovations such as ReLU activation, GPU usage, and dropout. Furthermore, they proved that deep networks with millions of parameters and connections do better, as opposed to shallow ones. This success brought interest in the competition and deep learning research to the highest level, allowing a new spring of artificial intelligence to begin. In the ImageNet contest, successive records

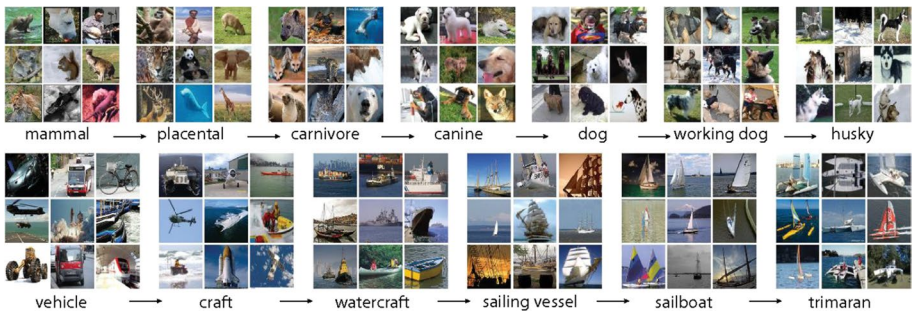


Fig. 9 ImageNet Dataset organized in WordNet hierarchy. WordNet links words into semantic relations to be used in computational linguistics and natural language processing

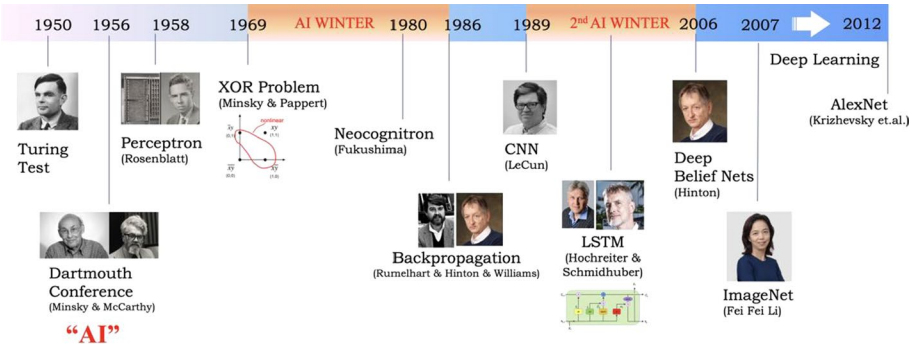


Fig. 10 Key milestones in the development of Artificial Neural Networks

were achieved in the proceeding years. Key milestones in the development of Artificial Neural Networks are depicted in Fig. 10.

3 ANN optimization

The optimization of Artificial Neural Networks has been studied from various aspects. These are mainly architectural design, connection weights, learning algorithm, node transfer functions, determination of initial weights, optimization of the input layer, and optimization of learning parameters e.g., learning rate, or momentum. To summarize, every variable in the artificial neural network model can be optimized in several ways. However, it is possible to combine these optimization areas into two main sub-categories. The first one is *network design* and the second one is *network training*.

3.1 Optimization of ANN architectures

Architecture optimization in Artificial Neural Networks is mainly concerned with the optimization of structural parameters such as the number of layers, number of neurons in each layer, and connections scheme. The selection of node activation functions, which is studied

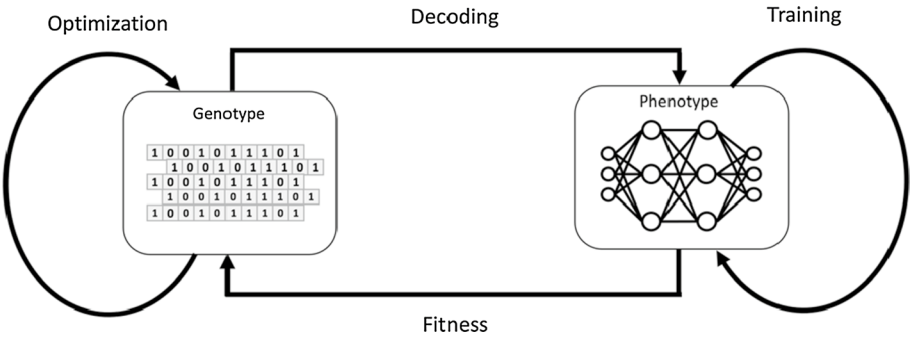


Fig.11 General Flow of ANN Optimization

separately in some works, is actually an area of architecture design. On the other hand, the process of reducing input parameters, defined as input layer optimization, is an area that falls into the field of data science, and cannot be defined as architecture optimization. The general flow of an ANN optimization is depicted in Fig. 11.

3.2 Optimization of synaptic weights

The backpropagation method is the most common and most effective method of finding optimal weights. It is a gradient-descent-based algorithm which aims to minimize the total mean square error between actual output and desired output. In every iteration, this error is used to guide the algorithm to find optimal weight values for the desired output. Although being very effective, the BP has a tendency to be trapped at local minima and quite often causes the vanishing or exploding gradients problem. Furthermore, in some real-world problems, it could be inefficient due to the structure of the error surface. For best results, the user is required to select the best hyper parameters, such as learning rate, momentum, and batch size, which further necessitates another heuristics.

Therefore, other methods have also been proposed to train Artificial Neural Networks. Initial attempts were aimed to increase the performance of BP by introducing gradient-based variations using sophisticated algorithms such as Conjugate Gradient (Charalambous 1992; Fletcher and Reeves 1964; Hestenes and Stiefel 1952) and Quasi-Newton methods (Dennis and Moré 1977; Huang 1970; Nocedal and Wright 2006). In order to improve convergence, adaptive learning rates were applied to some applications (Barzilai and Borwein 1988). Later, nature-inspired metaheuristics were thoroughly investigated and experimented with as competitive alternatives to Backpropagation. These approaches include but not limited to Evolutionary Computation techniques such as Genetic Algorithms (Gonzalez-Seco 1992; Gupta and Sexton 1999; Montana and Davis 1989; Sexton and Gupta 2000), Evolutionary Strategies (Greenwood 1997), and Differential Evolution (Ilonen et al. 2003), popular optimization methods such as Simulated Annealing (Sexton et al. 1999), Artificial Bee Colony Algorithm (Karaboga and Akay 2007), Particle Swarm Optimization (Roy et al. 2013). Due to the surge of interest in the field of Artificial Intelligence, many other techniques were also applied, including Fuzzy Sets (Juang et al. 1999), APPM (Artificial Photosynthesis and Phototropism Mechanisms) (Cui et al. 2012) as an alternative solution to BP.

3.3 Simultaneous optimization of architecture and weights

The general goal of an ANN optimization process is to achieve the best generalization. During optimization, every candidate solution is evaluated by simply training the network by using BP or other methods, thus obtaining the error rate on test data. It would require vast computational resources and time to achieve optimal architectures, by iterating through possible solutions. Addressing this phenomenon, many researchers aimed to optimize both architectures and weights at the same time to save computation costs.

3.4 Invasive and non-invasive approaches

The typical strategy of ANN architecture optimization is to search for better models and evaluate the algorithm by training the candidate solutions using gradient-based methods

such as backpropagation. Many researchers did not follow this computationally intensive path and aimed to optimize both architecture and weights simultaneously. Thus, these two different approaches formed the classification of approaches as invasive and non-invasive. *Non-invasive* refers to the former approaches where architecture is optimized and weights are obtained by BP-like algorithms, while *invasive* refers to the latter approaches (Palmes et al. 2005).

3.5 Methodology

In Artificial Neural Networks, it is possible to consider network architecture design as a search problem in the architectural space. The average error obtained for each architecture creates a surface in this search space. Proposed methods aim to find the lowest (or highest) point on this surface (Liu and Yao 1996a). According to Miller et al. (1989), this surface is:

- *infinitely large*: because there is no limit on the number of neurons and connections that can be used.
- *nondifferentiable*: because decision variables are discrete.
- *complex and deceptive*: because there is no direct relationship between network performance and network size, and similar architectures may yield different performance.
- *multimodal*: because networks with different topologies can give the same result.

For this reason, finding the ideal architecture in artificial neural networks is too difficult or impossible even for small networks to be solved by conventional methods. As Miller et al. (1989) express, *"the network design stage remains something of a black art"*.

3.5.1 Generalization and architecture

A considerable amount of reports in the literature state that the speed and generalization ability of a neural network usually depends on its complexity (Weiß 1994a). For example, a deep ANN having a large number of hidden layers and nodes will provide more accurate output for the training data but may demonstrate poor generalization for unknown test data, which is a phenomenon called *overfitting* (Yen and Lu 2000; Zhang and Muhlenbein 1993). In this case, the network simply memorizes training samples and noise in the training data, destroying the capability of the network to generalize (Fiszelew et al. 2007). On the other hand, a smaller network with only a few hidden layers and neurons usually has poor learning ability and may not be able to approximate the function. Some researchers followed the principle of *Occam's Razor*, which states that simpler models should be preferred to unnecessarily complex ones (Thorburn 1918; Zhang and Muhlenbein 1993; Zhang and Mühlenbein 1993). There are several approaches to identify an optimal and efficient neural network. These are SIC (Schwarz Information Criterion), AIC (Akaike Information Criterion), and PMDL (Predictive Minimum Description Length). Although used by many researchers all the above methods have significant drawbacks and weaknesses.

3.5.2 Conventional methods

Conventional techniques such as brute force, enumerative, or random search only provide low-quality solutions over very limited options. Constructive and destructive methods are among the classical approaches introduced in the early years. Constructive methods, as

the name suggests, aims to obtain the ideal topology by gradually expanding the model starting from a minimal architecture. For example, the cascade-correlation approach is an example of constructive methods (Fahlman and Lebiere 1990). On the other hand, destructive methods start from a large architecture first and then are followed by removing neurons or pruning the connections on this architecture. Therefore, they are often referred to as *pruning* methods. A popular example of destructive methods is LeCun's Optimal Brain Damage (OBD) (1990b). Although these methods address the problem of structuring ANN models, they investigate restricted topological subsets rather than the entire surface of possible ANN architectures (Fischer and Leung 1998). The inefficiency of conventional methods has led researchers to exploit global search algorithms. Mostly inspired by natural phenomena, *Metaheuristics* are usually applied in such circumstances where the search space is infinitely large. A general characteristic of Metaheuristics is that they can obtain an optimum solution to very difficult problems in a reasonable time.

3.5.3 Metaheuristics

Metaheuristics are stochastic/non-deterministic global optimization methods that are generally inspired by nature, the swarm of animals, or daily life. Although they are classified in different ways, they generally appear in three different types: single solution-based, population-based, or hybrid (Fig. 12) (Blum and Roli 2003; Dréo et al. 2006; Ojha et al. 2017). While some of them have memory features, some others are memoryless approaches.

3.5.3.1 Single solution based metaheuristics As it can be understood from its name, these methods proceed with only one solution during the search. Examples of single solution-based metaheuristics are *Simulated Annealing* (SA) (Kirkpatrick et al. 1983), which simulates the warming and cooling processes of substances in the metallurgical industry, and *Tabu Search* (TS) (Glover 1989, 1990) inspired by the phenomenon of taboo in human behavior. Furthermore, local search algorithms such as *Variable Neighborhood Search* (VNS) (Mladenović and Hansen 1997) and *Greedy Randomized Adaptive Search* (GRASP) (Feo et al. 1994) also fall into this class.

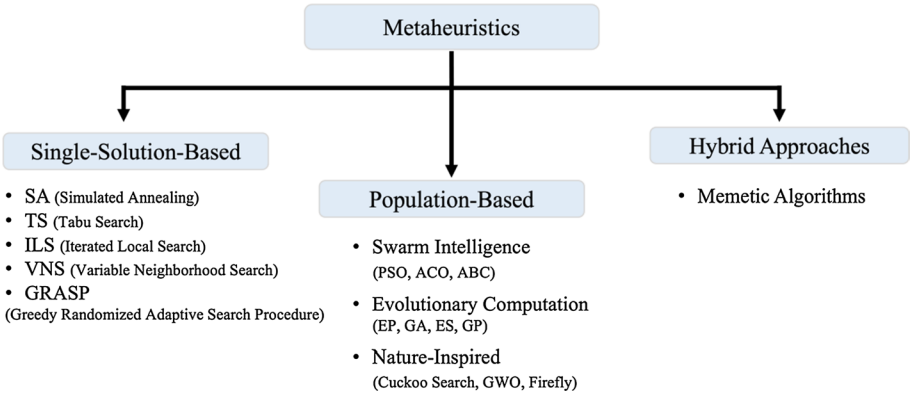


Fig. 12 Classification of Metaheuristics

3.5.3.2 Population-based metaheuristics Population-based algorithms are global optimization methods that are mostly inspired by nature and based on the principle of performing the search with more than one candidate solution on every iteration. Unlike single-solution-based approaches, they have global search capability. Evolutionary computational approaches are among the most popular population-based methods. It is based on the *survival of the fittest* principle of evolution theory. Another example of population-based approaches is *Swarm Intelligence*. The most common methods in this approach are *Particle Swarm Optimization* (PSO) (Eberhart and Kennedy 1995), *Ant Colony Optimization* (ACO) (Dorigo et al. 1996), and *Artificial Bee Colony Algorithm* (ABC) (Karaboga 2005) which are inspired by self-organized behaviors of animals such as fish, birds, bees and ants (Ojha et al. 2017). In this method, a random swarm is created initially, and the behavioral patterns of the swarm help the search move to directions of possible solutions. For example, flocks of birds in PSO form a weight vector and search the entire search space for food and direct the flock towards the food source (good solutions). Similarly, ACO is inspired by the ant swarm looking for food and leaving pheromone in the direction of the food source. As the level of pheromone increases, the search is steered to better solutions. Many other algorithms, inspired by nature, have been developed. These are including, but are not limited to *Gray Wolf Optimization* (Mirjalili et al. 2014), *Cuckoo Search* (Yang and Deb 2009), and *Firefly algorithms* (Yang 2009).

3.5.3.3 Hybrid metaheuristics Another important paradigm in metaheuristics is hybrid approaches. In the hybrid approach, called a *memetic algorithm*, the strategy is to combine more than one global or local search algorithm to obtain a stronger algorithm. Conventional or local search algorithms reach the results fairly quickly, while the risks of trapping into local minimum are higher. On the other hand, population-based global search methods are slower but have the ability to reach the global minimum. It is aimed to obtain more effective results with the synergy of these two approaches.

3.6 Evolutionary computation

Evolutionary computation is a set of global optimization techniques that have been widely used for training and automatically designing neural networks (García-Pedrajas et al. 2003). It is undoubtedly the most popular and successful population-based metaheuristic optimization paradigm inspired by biological evolution (Sun et al. 2019c). Throughout its historical development, several evolutionary approaches have been proposed including *Genetic Algorithms (GA), Evolutionary Programming (EP), Genetic Programming (GP), Evolutionary Strategies (ES),* etc., among which GAs became the most popular due to their biological grounds and superior performance in solving various optimization problems in a reasonable time. A general classification of Evolutionary Computation methods is depicted in Fig. 13.

Evolutionary approaches mimic natural selection, adaptation to the environment, and survival of the fittest principles of biological evolution. Similar to the evolution of living organisms in nature, they aim to reach a global solution by improving the candidates called individuals within each population in each generation. Thus, without having any *a priori* information, it can simultaneously search many points in the architecture space in parallel and reach the optimum solution in a short time without trapping into the local minimum. This makes it one of the most successful methods for architectural design in artificial

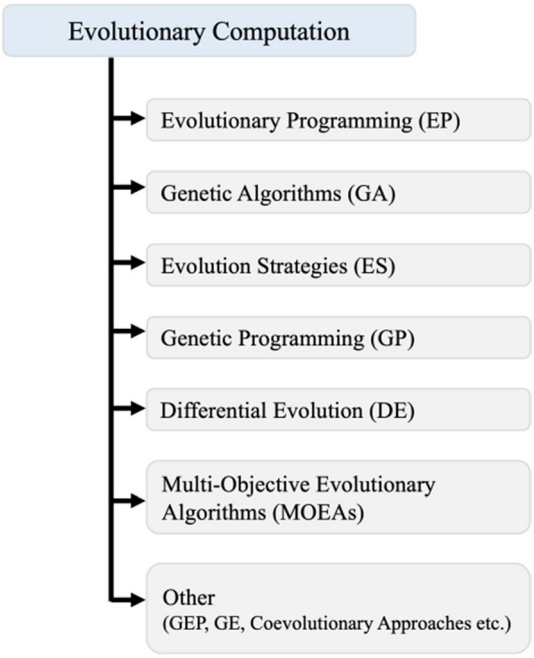


Fig. 13 Classification of Evolutionary Computation Methods

neural networks. Therefore, the newly formed discipline of evolutionary computation-based methods used for network design or network training in artificial neural networks is called *Neuroevolution* (Stanley et al. 2019).

In evolutionary computation, all features of an individual in the population are encoded on the chromosome as in DNA encoding. This encoding can be binary, real number, or categorical. The encoded chromosome is called the *genotype*, while decoded features are called the *phenotype*. Each value encoded in the chromosome is called *alleles*. Evolutionary Computation is divided into various sub-disciplines: evolutionary programming, genetic algorithms, genetic programming, evolution strategies, and differential evolution. Although they all mimic the natural processes of biological evolution and having many features in common, there are some methodological differences. For example, only selection and mutation operators are used in evolutionary programming, while genetic algorithms use all genetic operators such as selection, crossover, and mutation. In addition, in the sub-discipline of genetic programming, reproduction is tree encoding instead of binary or real-coded (Bäck et al. 1997; Baldominos et al. 2020; Spears et al. 1993).

3.6.1 Evolutionary programming (EP)

Evolutionary Programming focuses on the evolution of various parameters of fixed computer programs. It was proposed by Fogel et al. (1964; 1962; 1966). The basic approach in the optimization of these parameters is the selection and random mutation in generations. In this method, the crossover operator is not applied. With this feature, it is less affected by encoding restrictions. For many authors, EP is the most suited paradigm of evolutionary computation for evolving ANNs (Angeline et al. 1994; García-Pedrajas et al. 2003).

3.6.2 Genetic algorithms (GAs)

Genetic algorithms are an evolutionary global optimization technique introduced by John Holland in 1975 (De Jong 1975; Goldberg and Holland 1988; Holland 1975; Mitchell 1998). It has been applied to a wide variety of problems and demonstrated superior performance. Unlike the other evolutionary approaches, GA incorporates a 'crossover' operator to imitate the effect of sexual reproduction (Jones 1993). However, in artificial neural network design, some researchers avoided the crossover operator. This is due to a permutation problem or a phenomenon called *competing conventions*. In this problem which will be detailed in the next section, it is observed that chromosomes with different encoding produce the same mathematical output. This creates an undesirable situation in terms of optimization.

3.6.3 Evolutionary strategies (ES)

In this approach, a vector consisting of real numbers is subject to evolution by using selection and mutation operators. This paradigm was introduced in the 1970s by Rechenberg (1973) and Schwefel (1977). It uses representations independent of the natural problem and uses only selection and mutation as operators. Later on, *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) has been developed which can take the results of each generation, and adaptively increase or decrease the search space for the next generation (Hansen and Ostermeier 1996, 1997).

3.6.4 Differential evolution (DE)

Differential Evolution was developed by Storn and Price (1997) to overcome various deficiencies in evolutionary approaches. The method they proposed is non-differentiable, nonlinear, and has parallelization capability which can handle multi-model cost functions easily. It has fewer control parameters and good convergence features. The vector generation scheme of DE leads to a rapid increase in population vector distances if the target function surface is flat. This "divergence feature" prevents the DE from progressing very slowly in shallow areas of the objective function surface and ensures rapid progression after the population passes through a narrow valley.

3.6.5 Genetic programming (GP)

Genetic Programming is an extension of Genetic Algorithms, invented by Cramer (1985) and further developed by Koza (Koza 1992, 1995). Genetic Programming enables machines to automatically build computer programs (Gruau 1994). Koza used LISP, which is a tree-based programming language to evolve compute programs to solve several tasks. A LISP program can be defined as a rooted and labeled tree called the *S expression*. LISP functions are represented as labels and leaves are labeled with constants or inputs. Computed *S expression* values form the output. Crossover of two parent trees is accomplished by cutting a sub tree from one parent and pasting to another as a replacement. As the key researcher on this paradigm, John Koza applied this paradigm for generating neural networks and optimizing both architectures and weights (Koza and Rice 1991).

3.6.5.1 Gene expression programming (GEP) Gene Expression Programming was invented by Ferreira (2001; 2006), as a variation to Genetic Algorithms. Unlike GAs, in which individuals of a population are linear strings of fixed length, and unlike GP, in which individuals are nonlinear entities of different sizes and shapes (parse trees), GEP incorporates both, encoding individuals first as a linear string of fixed length, then representing them as expression trees (ET). Expression trees are encoded into a linear form by using *Karva* language and the encoded tree is then called a K-expression. GEP allows the creation of *multiple genes*, each coding for a program in a small size or sub-expression tree. Ferreira also used GEP to evolve ANNs, claiming his algorithm is very well suited, producing valid structures all the time.

3.6.5.2 Grammatical evolution (GE) Grammatical Evolution is an evolutionary search framework, typically used to generate computer programs defined through context-free grammar, which describes the syntax of expressions (Noorian et al. 2016). It is introduced by Ryan, Collins, and O'Neil (1998) in 1998. GE is designed to evolve programs in any language by using a variable-length linear genome and adopts BNF (*Backus Naur Form*) to express the grammar in the form of production rules. When compared to GP, it is more flexible since the user is able to constrain the way in which the program symbols are assembled together (Drchal and Šnorek 2008). Later it was improved by Lourenço et al. (2016) as structured grammatical evolution (SGE) to address the redundancy and locality issues in GE and consisted of a list of genes, one for each non-terminal symbol (Assunçao et al. 2017).

3.7 Genetic operators

Evolutionary Algorithms typically apply genetic operators namely: Initialization, Selection, Reproduction (crossover), and Mutation. More recently elitism is introduced to improve performance on some real-world tasks. Inspired by biology, these operators are essential tools to obtain global optimum for a given problem, and the performance of the algorithms mainly depends on how these operators are exploited.

3.7.1 Generating the initial population

In evolutionary approaches, first of all, a population of determined size is generated. Each individual in the population represents a solution to the problem. The population size is one of the important parameters affecting the solution. The large selection of the population size increases the diversity while bringing extra calculation costs. Selecting rather a small population size causes the search area to narrow. The generation of the initial population is usually carried out randomly. This allows starting from different points in the solution space.

3.7.2 Fitness function and evaluation

The convergence of the individuals in a population is evaluated by the fitness function. For this reason, the fitness value of the genotype is calculated. In Genetic Algorithms, the fitness function is unique to the problem. The fitness represents how suitable the individuals are for the solution. The performance expected from genetic algorithms is related to the precise determination of the fitness function.

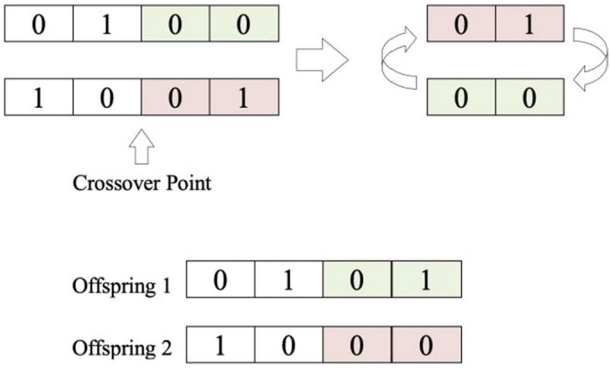


Fig. 14 Crossover operation on a binary string

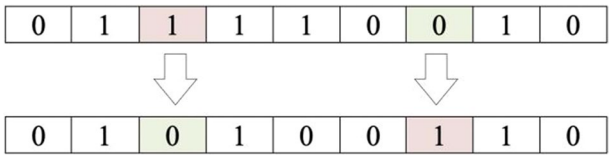


Fig. 15 Mutation on a binary string

3.7.3 Selection

The selection process eliminates individuals with low quality and transfers better individuals to the next step, reproduction. Based on the Darwinian principle of *survival of the fittest*, individuals with higher fitness are more likely to win during selection, although the process involves randomness in nature. There are many methods in the literature for selection. Among them, the roulette wheel, tournament selection, and rank method are the most frequently used.

3.7.4 Reproduction (Crossover)

The reproduction process is executed by crossover which simulates the sexual generation of a child, or offspring from two parents (Koehn 1994). Individuals selected for reproduction produce offspring who share the common characteristics of their parents. It is algorithmically accomplished by taking and combining some parts of two parents and forming the child (Fig. 14). How the crossover is carried out may vary depending on the structure of the encoding and the nature of the problem. The most commonly used crossover methods are single-point, two-point, arithmetic, and uniform crossover. This step, which seems to be not making sense at first glance, determines new solution candidates that bring us closer to the optimal solution. In genetic algorithms, generally, the entire population is not subject to crossover operation. Only, a determined part of the whole is taken to the crossover.

3.7.5 Mutation

Mutation in nature is the change or degradation of a DNA molecule that is found in the nucleus of the living cell and enables the emergence of hereditary properties. Some possible causes of mutation are radiation, X-ray, ultraviolet, sudden temperature changes, and

degradation as a result of chemistry. The mutation is very rare and takes place in a very small part of the chromosomes. In genetic algorithms, the mutation is a small, structural change in chromosomes similar to the natural phenomenon (Fig. 15). As in selection and crossover strategies, the ultimate goal in solving the problem is to reach an optimal solution without getting caught in local optima. There is always a possibility that the genetic algorithm solution will get trapped in a local solution. A way to eliminate this possibility is to *mutate* some chromosomes. This increases the chances of obtaining the ultimate optimum solution.

3.7.6 Elitism

Although the selection strategy seeks to find good candidates, some powerful individuals among the population can also be eliminated, as the process will proceed randomly. Elitism is applied in order to prevent losing good solutions and ensure that the strongest candidates can be transmitted to the next generation in absolute terms. Although criticized for its tendency to converge prematurely, the elitist strategy was used in many studies and produced encouraging results. What is important here is to determine the number of population members to be separated by elitism. Care should be taken to select the most suitable ratio considering that the high amount can reduce the diversity, resulting in a local minimum.

3.8 Multi-objective evolutionary algorithms

The only goal in architectural optimization in Artificial Neural Networks is not high accuracy or good generalization. An algorithm with good generalization capability but high computational cost is not suitable for many real-world problems with time and hardware constraints. Therefore, in addition to network performance, algorithms require to meet more than one criterion such as model size and computational complexity. Multi-Objective Evolutionary Algorithms, which were put forward for such problems, were also preferred in the architectural optimization of artificial neural networks.

A cost function with two contradictory objectives usually comes with an objective causing the other objective to get dominated. Thus, a nondominated solution is called a

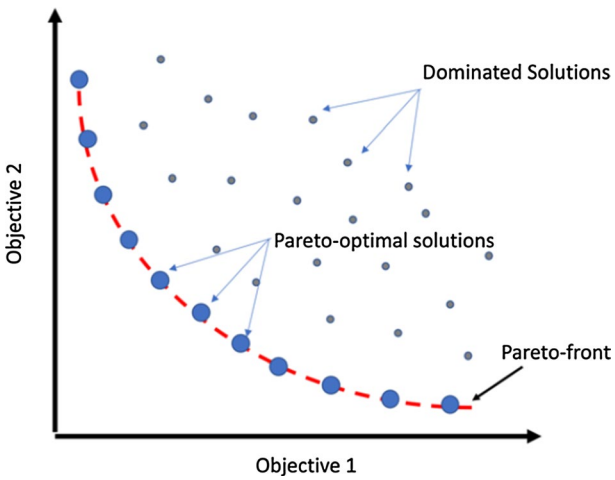


Fig. 16 Pareto-front for two objectives

Pareto-optimal solution. All Pareto-optimal solutions are also called *Pareto-front*. A plot of *Pareto-front* with two objectives is depicted in Fig. 16.

Multi-objective evolutionary algorithms are generally examined in two categories. These are *non-Pareto-based* or *Pareto-based* multi-objective approaches. *Non-Pareto-based* approaches work on the principle of aggregating the cost of criteria that make up the objective function. Here the user adds a multiplier that determines the weight of the criterion she/he wants. Then, net fitness is calculated with a weighted sum. On the other hand, in *Pareto-based* approaches, the criteria are handled as a whole and the solutions that make up the *Pareto-front* are presented to the user.

Multi-Objective Evolutionary Algorithms emerge with the introduction of the Vector Evaluated Genetic Algorithm (VEGA) by Schaffer (1985; 1986). Later, MOGA (Fonseca and Fleming 1993) and NSGA (Srinivas and Deb 1994) were developed. These algorithms were based on population diversity based on the individual selection, non-dominated sorting, and fitness sharing mechanism. Later, fast non-dominated sorting and elitism-based external archive strategies were adopted. NSGA-II has been one of the most successful studies in the literature working on this principle (Deb et al. 2002). In addition, SPEA (Zitzler and Thiele 1999), SPEA2 (Zitzler et al. 2001), and PAES (Knowles and Corne 1999) have been successfully implemented on various problems. For further research on these works, the reader can refer to (Zhou et al. 2011) and (Zhang and Xing 2017) for detailed surveys.

3.9 Coevolutionary approaches

Although they are powerful, evolutionary algorithms may perform poorly in problem types where search space is very large. This is more prohibitive, especially when the fitness function cannot be fully expressed. Researchers employ coevolutionary methods in such situations. The co-evolutionary algorithm is a type of evolutionary algorithm, where the fitness function depends on the relationship between individuals in the population. Relationships between individuals are evaluated and fitness is determined. In other words, there is relative fitness instead of directly calculated fitness. This shows that the coevolutionary algorithm is significantly different from the classical evolutionary algorithm (Azzini and Tettamanzi 2006; Potter and De Jong 1994; Potter and De Jong 1995; Wiegand 2003). Coevolutionary algorithms are basically divided into two sub-categories as *Cooperative* and *Competitive*.

3.9.1 Cooperative coevolution

In Cooperative Coevolution, every individual in the population contributes in cooperation with other individuals to solve the big problem. In order to obtain a general solution, all individual solutions must be brought together.

3.9.2 Competitive coevolution

In competitive coevolution, individuals evolve in competition with each other. In this competition, individuals with high fitness survive according to Darwin's survival of the fittest principle, while those with low fitness disappear. These types of algorithms can be

explained as follows: Let's consider two models that have a predator–prey relationship with each other. Considering that one of these models is a network that performs sorting or pattern recognition, and the other model is a mechanism that generates input for this network, the first model will try to recognize better in the process of evolution, and the other network will produce more difficult inputs for the first network. In this way, they will help each other to reach a global solution (Hillis 1990).

4 Representation

The most important aspect of an evolutionary design is undoubtedly genetic representation. Representation is the method that describes how the genetic *chromosome*, in other words, the *genotype* is encoded and how to transform an encoded genotype into the explicit form of a feature string, called the *phenotype*. Genetic encoding directly affects the speed and efficiency of the solution process. For this reason, an effective encoding mechanism will be one of the most important factors that determine network performance. Although named differently by different authors, there are basically two representation methods (Floreano et al. 2008; Gruau 1994; Yao 1993). These are *direct encoding* and *indirect encoding* (Fig. 17).

4.1 Direct encoding

Direct encoding, also called *strong representation* (Miller et al. 1989) or *high-level encoding* (Schiffmann et al. 1993), is a method in which structural parameters of ANN architecture are directly encoded in the chromosome. The connections between each node forming the network and the connections between these nodes are often expressed as binary with the help of a connection matrix. For instance, an *NxN* matrix can represent an ANN architecture with N nodes, where c_{ij} indicates the existence or non-existence of a connection from node i to node j. We can use $c_{ij}=1$ to indicate an existing connection and $c_{ij}=0$ to indicate no connection. The final chromosome will be formed by concatenating the matrix rows (Fig. 18).

The main advantage of direct representation is that each parameter can be expressed explicitly. In addition, since it does not require any special encoding, its conversion from

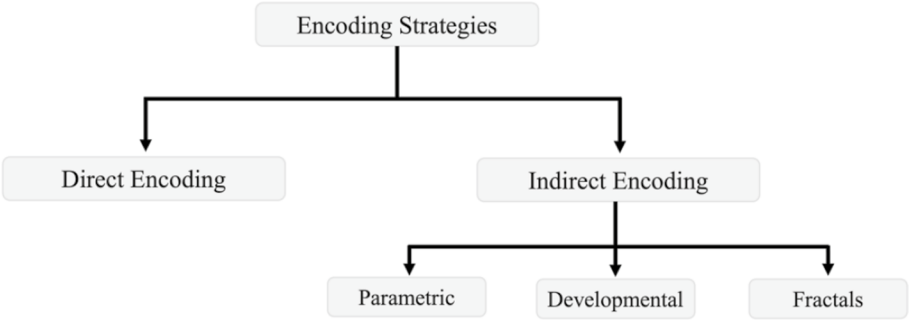


Fig. 17 Classification of Encoding Strategies

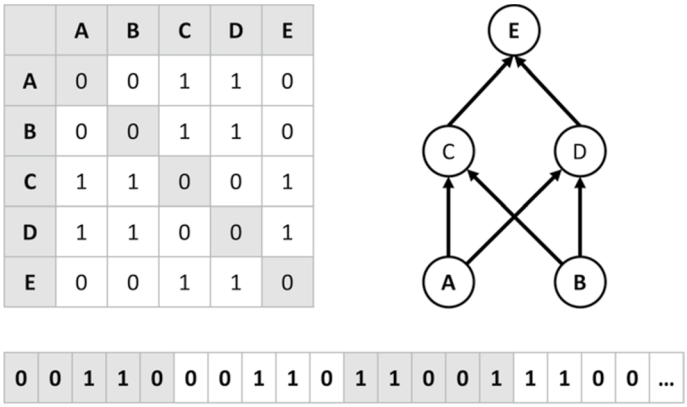


Fig. 18 Direct Representation of a neural network. An *NxN* matrix representing the connectivity scheme of ANN is expressed as binary. In this matrix, 1 indicates the existence of connection, and 0 indicates no connection

genotype to phenotype or phenotype to genotype is very fast. On the other hand, since the chromosome that will form as the network model grows will expand exponentially, it is preferred only in small-size networks. If there is enough prior domain knowledge about the network, for example, if the network is known to be fully connected feedforward, only a smaller chromosome can be obtained by encoding the number of layers and the number of nodes in each layer to the chromosome. In cases where direct representation is preferred, extra care is required when using evolutionary operators because model integrity may be impaired in operations such as crossover, leading to the creation of infeasible child networks (Stanley 2004).

4.2 Indirect encoding

The representation approach in which the Artificial Neural Network model is expressed in various production rules and systems is called *low level*, *weak*, *recipe,* or *indirect encoding* (Branke 1995; Schiffmann et al. 1993). The structural features that make up the network are encoded (or generated) using various parameters or developmental rewriting rules. Also, not all features of the network model need to be specified. The chromosome structure can be reduced by encoding only important parameters.

The emergence of indirect encoding is motivated by biological phenomena. While human DNA is home to only 30.000 chromosomes, there are billions of neurons and trillions of connections between the neurons in the human brain (Mjolsness et al. 1989). This is the basic indication that DNA somehow encodes the human brain indirectly. In order for a compact encoding to be possible in this way, the structures formed must be highly regular.

Throughout history, indirect encoding has been implemented in various ways and used for the evolutionary design of Artificial Neural Network architecture. Yao (Yao 1993) classifies indirect encoding into three categories. These are:

- *Parametric (Blueprint) encoding*, which encodes parameters for constructing connectivity.

- *Developmental Rules*, rewriting grammars and nature-inspired systems.
- *Fractals* from biology.

According to Stanley (2004), the effectiveness of indirect encoding originates from *gene reuse*. By using multiple times of a single gene at different developmental stages, an effective representation as in DNA can be obtained. Stanley examined indirect encoding in two categories. In the first category, he argued that phenotypic structures were created with repeating patterns and that the same pattern was repeated with a structural theme, while in the second category, the same was used to create different developmental pathways. He added that in the second category, different structures can be expressed in different locations. He stated that numerous left/right symmetries in vertebrates and numerous receptive fields in the visual cortex are biological examples of this type of encoding.

4.2.1 Parametric representation

Parametric representation, which is also known as *Blueprint* encoding, is one of the earliest forms of indirect encoding, in which the properties of Artificial Neural Network architecture are encoded with several parameters. These parameters define the number of layers, the number of neurons in each layer, and how neurons connect with each other. The most important advantage is that large models can be expressed with relatively small chromosomes. On the other hand, it is restricted to a range of architectures and may not achieve modular architectures (Gruau 1994). The pioneering example of this type of encoding is the work of Harp et al. (1989). Later on, Hancock (1993), Dodd (1990), and Mandischer (1993) used similar encoding techniques.

4.2.2 Developmental approaches

In the developmental representation, which is also defined as a grammatical encoding (Kitano 1990) in the early studies, the artificial neural network architecture is expressed by previously determined *production* or *growth* rules. The most important feature of the method is that it is scalable, abstract, and modular. Thus, even very large networks can be represented hierarchically with compact chromosomes. This representation form is a biologically plausible encoding method. According to Boers and Kuiper (1992), this encoding approach is expressed in *recipes* instead of Blueprints. Living organisms have a very modular structure and this modularity creates tissues and organs of cells of the same type by repeating each other by following certain growth rules (Dawkins 1986). The cooking process of this *recipe* can be defined as the *ontogenesis* of an organism, in which the rules of splitting or specialization of cells are encoded in the genome (Grönroos 1998). Establishing developmental rules in the creation of artificial neural network architecture is usually carried out with recursive equations or graph generation rules. The first examples of this type of encoding are Mjolsness's (1989) and Kitano's (1990) work. In the following years, Gruau's Cellular Encoding (1994) and Luke and Spector's Edge Encoding (1996).

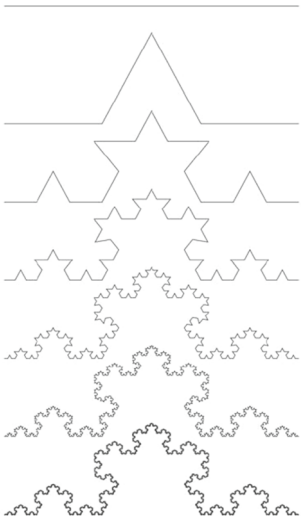


Fig. 19 The first six iterations of the Koch-Snowflake (redrawn from Mandelbrot (1982))

4.2.3 Fractals

Fractals are endless development patterns inspired by biological organisms. Popularized by Benoit Mandelbrot (1982), Fractals are created by repeating a simple process in an infinite loop. They often start with a simple geometrical object and a rule for modifying the object leading to a complex structure. One of the earliest and most popular descriptions of a fractal is Koch-snowflake (shown in Fig. 19), which begins with an equilateral triangle and then replaces the middle third of every line segment with a pair of line segments that form an equilateral bump (Koch 1906).

A fractal representation of ANN connectivity has been proposed by Merrill and Port (1991), arguing that they are biologically more plausible than growth rules. They also claimed that strong evidence exists about parts of the human body (such as lungs) having fractal structures.

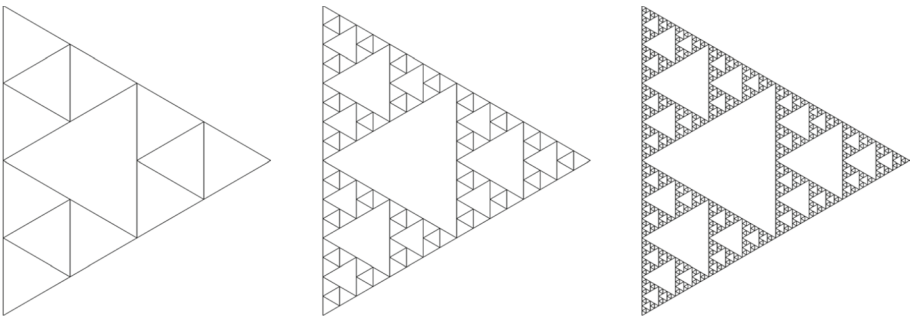


Fig. 20 The Sierpinski triangle drawn using an L-system. It is a fractal with the overall shape of an equilateral triangle, subdivided recursively into smaller equilateral triangles

4.3 Lindenmayer systems (L-Systems)

L-systems are a special class of fractals that mathematically models biological growth in multicellular organisms, especially plants. L-systems are introduced and developed by Aristid Lindenmayer (1971), a Hungarian theoretical biologist and botanist at the University of Utrecht. L-system grammars create *production rules* and morphological description strings applied on the starting axiom that consists of symbols with associated numerical values (Lee et al. 2005). The process of applying these rules is called string re-writing, so highly complex morphologies can be built with relatively simple rules. L-systems are especially suitable for describing fractal structures such as cell divisions in biological organisms and modeling the growth of plants in computer graphics (Lee et al. 2005). A popular example of an L-systems is *Sierpinski Triangle* (Fig. 20). Many researchers developed artificial neural network architectures optimized with evolutionary algorithms and inspired by L-systems for representation (Boers and Kuiper 1992; Gruau 1994; Kitano 1990).

4.4 Artificial embryogeny (AE)

Stanley and Miikkulainen (2003) introduced the term *Artificial Embryogeny* by combining artificial evolutionary systems that utilize the developmental process of embryos in nature. In their taxonomic study, they collected all the developmental processes including Artificial Ontogeny (Bongard and Pfeifer 2001), Computational Embryogeny (Bentley and Kumar 1999), Cellular Encoding (Gruau 1994), and Morphogenesis (Jakobi 1995) under one term. Thus, they created a framework for future studies and emphasized that indirect coding will have an important place in the evolution of artificial neural networks.

4.5 Other Nature-Inspired Approaches

Neural networks are viewed by many authors in a broader biological context of artificial life. Inspired by the features of neural development in animals, Nolfi and Parisi (1997; 1994) developed an innovative method encoding neural network architectures into genetic strings. In this model, the neurons are represented with coordinates in a two-dimensional space. The connections are defined by allowing *axon tress* to grow in the forward direction from neurons. These trees were basically L-system fractals generated from the grammar. This work was further developed by Cangelosi et al. (1994) by adding cell division and migration rules to grow neuron population rather than the direct encoding of each neuron to chromosome. Later Cangelosi and Elman (1995) simulated a model of regulatory ontogenetic development of artificial neural networks. In their simulation, network growth is controlled by genes that produce elements regulating the activation, inhibition, and delay of neurogenetic events. In another nature-inspired study, Dellaert and Beer (1994; 1996) described a model based on *Boolean networks* to evolve autonomous agents with developmental processes. The reader may refer to (Cangelosi et al. 2003) for an extensive survey of studies on biologically inspired neural development.

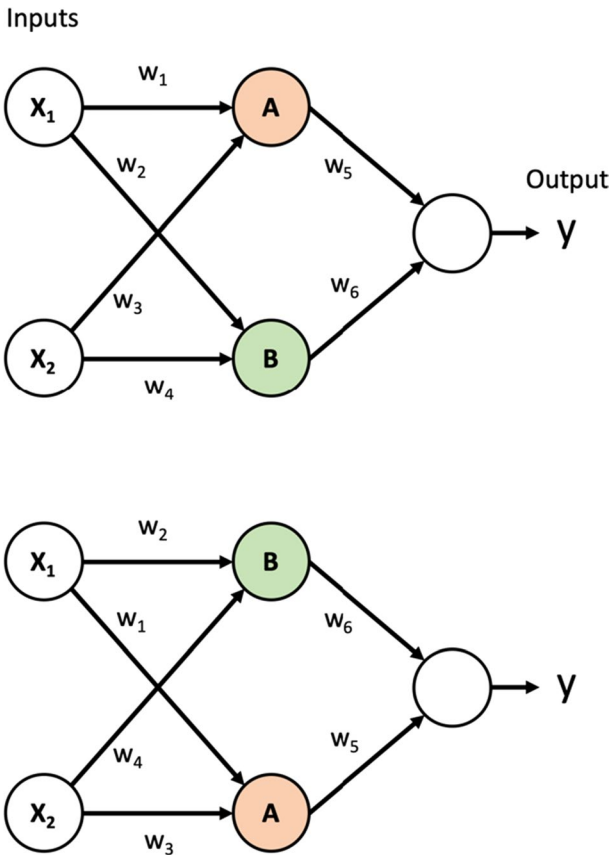


Fig. 21 Competing Conventions Problem. The permutation of nodes in the hidden layer does not change the network function

4.6 Competing conventions problem

Competing Conventions problem, also called *Permutation problem* (Radcliffe 1993) or *Structural–Functional Mapping Problem* (Whitley et al. 1990), is one of the key problems that arise in the optimization of Artificial Neural Networks using evolutionary methods. During the genetic process, some individual solutions in the population, which are completely different with their genotype and phenotype but functionally equivalent produce the same output. This phenomenon makes the evolutionary optimization process unnecessarily slow and causes child networks obtained with crossover to have infeasible or lower fitness. In two separate network models shown in Fig. 21, the permutation of the nodes of the hidden layer does not change the function of the network.

4.7 Noisy fitness evaluation problem

Due to the stochastic nature of random weight initialization, the fitness evaluation of ANN architectures is noisy unless weights are optimized simultaneously (Yao and Liu 1995). The transformation of genotype to phenotype together with the network training returns a fitness which would undoubtedly be different for initial weights generated

randomly. This gets worse when an indirect encoding is adopted for representation because developmental rules are not deterministic. Some authors opted to evolve both architecture and weights at the same time to alleviate this problem. Another approach is to train each architecture several times with different initial weights, and then take the best result to calculate fitness. However, this will lead to a massive increase in computation time (Fiszelew et al. 2007).

4.8 Ensembles

The primary objective of Artificial Neural Networks is to provide generalization. A network that achieves high accuracy on the training set may perform poorly on test data, which has not been previously introduced. On the other hand, the aim of evolutionary methods is optimization. The fitness function of artificial neural networks optimized by evolutionary methods aims for high accuracy with training data. However, the global minimum obtained for the highest accuracy does not necessarily mean the best generalization has been achieved. In the population, there may be other individuals with lower fitness but higher generalization ability. In such cases, ensemble methods are used to obtain the best generalization.

5 Historical progress

We investigated the historical progress in three periods. In the first period, we investigated the early works by explaining the roots of diverse ideas for chromosome representation. In the second period, the emergence and rise of more advanced methods were surveyed. In the last period, recent advances in the deep learning era were reviewed.

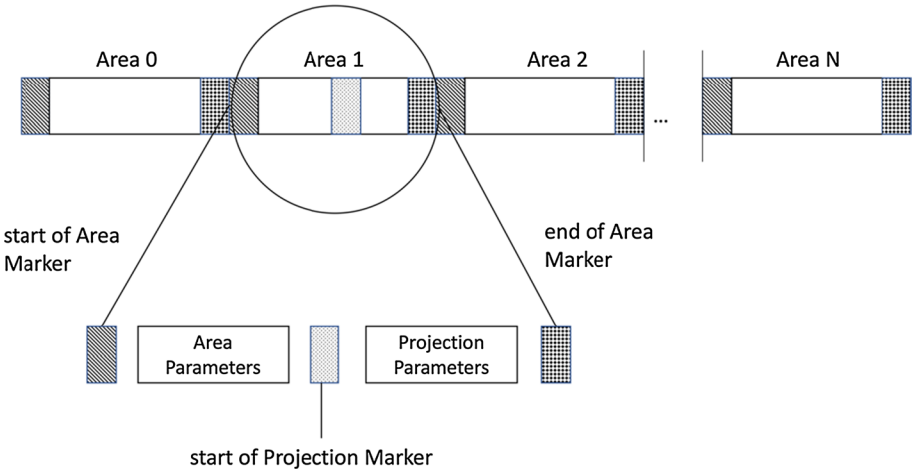


Fig. 22 Network Blueprint Representation (redrawn from Harp et al. (1990))

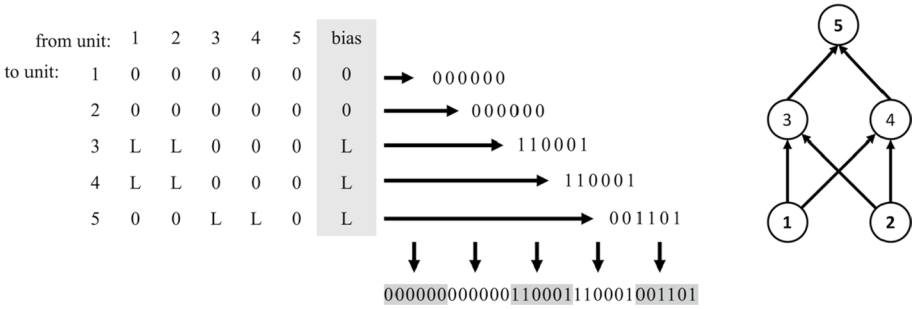


Fig. 23 Connectivity Constraint Matrix (redrawn from Miller et al. (1989)). The first N columns of matrix C specify the constraints on the connections between the N units, while the final (N + 1) column contains the constraints for the threshold biases of each unit. Here 1 indicates connection, 0 indicates no connection and L indicates learnable connection

5.1 Early work

Research on the evolutionary design of ANN architectures begins with Harp et al.'s *NeuroGENESYS* (Guha et al. 1988; Harp et al. 1989, 1990), based on the parametric indirect coding. Introducing a high-level scheme called *Blueprints*, the authors defined a variable-length binary chromosome string consisting of several parameters such as the number of layers, layer size, and connections. In this scheme, each segment called *area* refers to a set of nodes in the network. Each area includes relevant parameters and *projections* to several other areas. The start and end of those segments have *markers*, which help to align of genotype during the crossover (Fig. 22). This enabled the representation of larger networks with smaller chromosomes. A disadvantage of this encoding is that it can only search for architecture within a limited subset.

On the contrary, Miller et al. (1989) proposed a direct-encoding-based approach, which they define as a *strong representation*. This model, called *Innervator*, represents the artificial neural network with a simple connection matrix. In this matrix, the connection from each node to another node is defined with 1 if a connection exists and with 0 if a connection does not exist. Then the GA chromosome is built by concatenating the rows in this matrix. The most important advantage of this direct encoding approach suggested by Miller et al. is that it can search all possible network architectures (feasible and infeasible) in the search space without any restrictions. However, it has a big disadvantage that the chromosome length will increase as the network model grows. Thus, it can only be applied to small networks (Fig. 23).

$$S \to \begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad A \to \begin{pmatrix} c & p \\ a & c \end{pmatrix} \quad B \to \begin{pmatrix} a & a \\ a & e \end{pmatrix} \quad C \to \begin{pmatrix} a & a \\ a & a \end{pmatrix} \quad D \to \begin{pmatrix} a & a \\ a & d \end{pmatrix}$$

$$a \to \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad b \to \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad c \to \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad e \to \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad p \to \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Fig. 24 Graph generation rules used for generation of the 2–2-1 XOR network (redrawn from Kitano, 1990)

One of the earliest studies implementing a nature-inspired indirect encoding is the graph generation grammar introduced by Kitano (1990). In his approach, Kitano defined a context-free production grammar to create the connection matrix of the artificial neural network using a modified *L-system* and created the network structure with 2×2 recursive iterations starting from an axiom matrix of 1×1. Thus, by creating a simple model of neurogenesis in nature, he demonstrated that large networks can also be expressed with the help of very small chromosomes (Fig. 24). Although employing conventional search methods, Mjolsness (1989) described a similar compact encoding scheme where the connection matrix of a network is specified by recursive application of developmental patterns.

The beginning of the 1990s has witnessed many other studies in which architectures are generated automatically by evolutionary approaches. Wilson (1989) conducted experiments on perceptrons and analyzed the performance of models obtained using GA. Schifmann et al. (1990) investigated the relations between network structure and classification ability of BP. They utilized the so-called *BP-Generator* based on a mutation-only evolutionary strategy to create ANN architectures and compared their performance with standard BP-nets. Later, they extended their approach with a crossover operator (Schiffmann et al. 1992; 1993). Hintz and Spofford (1990) proposed a combination of ANN and GA that optimizes the network by evolving the number of neurons, weights, and connections. Another study that optimizes artificial neural network architecture using GA is Dodd's (1990) *Structured Neural Network* model. He proposed an approach that simultaneously optimizes generalization ability and network compactness for a pattern recognition problem classifying dolphin sounds, with a parametric indirect encoding.

In another pioneering work, an invasive approach was taken by Whitley et al. (1990) optimizing both the weight and architecture of ANN. Contrary to augmenting topologies, they started from a fully connected and already trained network and utilized a modified GA, which they call the *GENITOR* algorithm to find connections to be pruned. A significant amount of training time was saved by initializing the pruned network using the weights in the starting network (Branke 1995). Similarly, Höffgen et al. (1990) used GA to minimize networks for better generalization. In the same year, Hancock and Smith (1990) developed *GANNET* to specify the structure of BP-network and implemented their method on real-world problems. Later Hancock (1992b) proposed a GA-based approach to pruning the connections of BP-trained neural networks, similar to Whitley et al. (1990), and

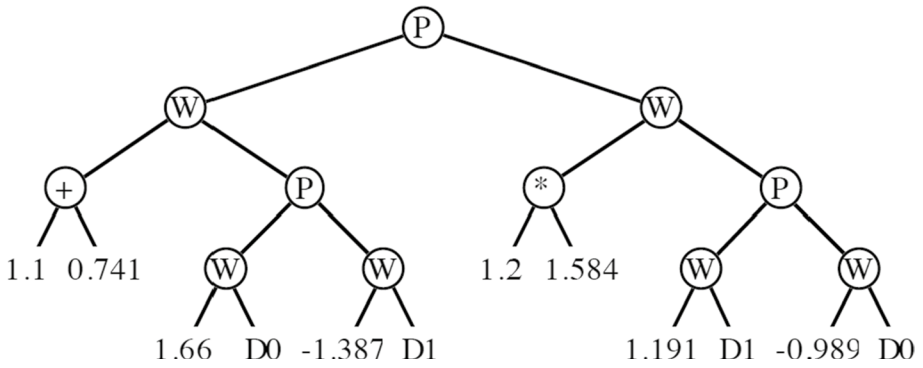


Fig. 25 A graphical depiction of a LISP S-Expression as a rooted tree, representing a neural network for XOR problem (redrawn from Koza & Rice (1991)). Here the root is a linear threshold processing function P. W is the weight function with D0 and D1 as inputs

explored solutions to the *permutation problem* (Hancock 1992a; Hancock 1993). These early works were thoroughly investigated and compared by various authors (Balakrishnan and Honavar 1995; Branke 1995; Radcliffe 1990; Rudnick 1990; Schaffer et al. 1992; Weiß 1994a; Whitley 1995).

As the key researcher of the famous Genetic Programming approach, Koza (1989) had shown that computer programs can be evolved to perform a particular task by extending genetic algorithms applied to tree-based programming languages like LISP. In 1991, Koza and Rice (1991) used this *Genetic Programming* paradigm to obtain both weights and architecture of a neural network and applied it to the problem of the one-bit adder (Fig. 25). This differs from the previous approaches in that the network architecture, as well as the weights, are encoded in the chromosome and they are trained simultaneously. Later, Vonk et al. (1995c) proposed GPNN *(Genetically Programmed Neural Network)* which implemented the same method on toy problems and extended it within another work reviewing recent works of the time (Vonk et al. 1995b).

Optimizing both weight and architecture was also an objective by Marshall (1991). However, instead of optimizing both at the same time, he adopted a different approach by first evolving the network parameters with GA and using BP to calculate fitness. Once an optimum structure is achieved, he optimized weights with GA. Dasgupta and McGregor (1992) described a *Structured Genetic Algorithm (sGA)* to optimize both architecture and weights with a hierarchical two-level direct representation, where high-level genes activate or de-activate sets of lower-level genes. In this approach, the high-level part of the chromosome encodes the connectivity scheme while the low-level encodes weights and biases with binary strings. Robbins et al. (1993) used *GANNET* which adopts a direct encoding approach where each gene directly represents the presence or absence of a connection in the network. Although producing long strings, which are not suitable for large networks, their prototype was capable of designing, implementing, and evaluating a variety of multilayer perceptrons while outperforming conventional methods such as random search, hill-climbing, and parallel hill climbing.

Early works had many innovative approaches for chromosome representation. Fullmer & Miikkulainen (1992) proposed a *marker-based encoding* scheme which is inspired by the biological structure of DNA. In this approach, markers are used to separate individual node definitions, containing all information about a node such as its identification, initial activation value, and a list of values which specifies its input sources and weights. This enabled the use of recurrent nodes and nodes without an input, acting as bias nodes which are normally used in BP learning. Later this approach was extended by Moriarty & Miikkulainen (1993; 1995a; 1995b), who developed new game-playing strategies based on the evolution of ANNs. They described the marker-based chromosome as a continuous circular entity and improved the predecessor work by defining only hidden nodes, which enabled more compact encoding when the output layer is large. The flexibility of location-independent alleles gave the genetic algorithm more freedom to explore useful schemata. Their solution was able to discover new strategies in *Othello*, a popular strategy board game in Japan, similar to Go. In a preliminary study, Gruau (1992; 1993) introduced *Cellular Encoding* designed with a cell rewriting developmental process to improve Kitano's (1990) graph generation grammar. This sophisticated indirect encoding approach was claimed by the author to be biologically more plausible, compact, modular, and abstract. It evolves both architecture and weights represented in binary form with target functions as boolean functions. Later, in his doctoral dissertation (1994), he presented Cellular Encoding as a *machine language* for neural networks and published many other works implementing and

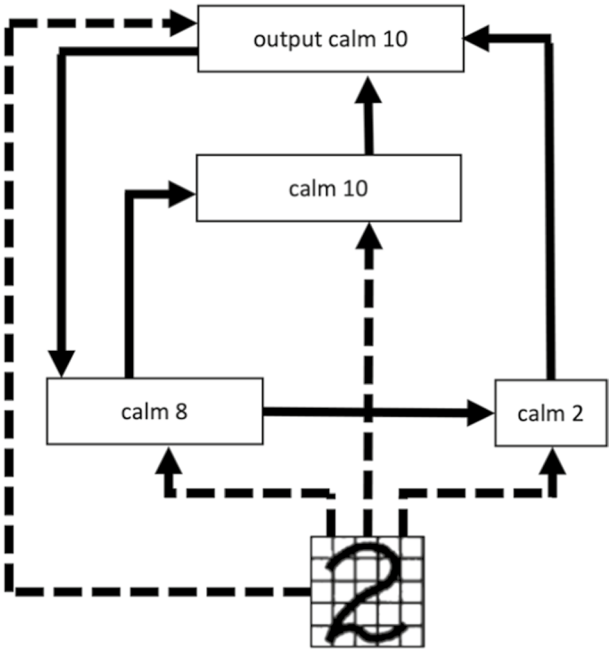


Fig. 26 The modular structure of the winning network after GA search of CALM (redrawn from Happel and Murre (1992)). Dashed arrows indicate learning input connections while solid arrows indirect unidirectional learning connections between modules

comparing his approach with other encoding strategies (Gruau and Quatramaran 1997; Gruau et al. 1996; Whitley et al. 1995).

After Kitano and Gruau, many successful studies were followed inspired by biological developmental patterns. Since the brain is considered to be a highly modular structure, a significant amount of work has been devoted to understanding the cooperative interaction between these modules in the visual system of a mammalian brain. Happel and Murre (1992) explored such modular constraints on neural networks and used CALM (*Categorization and Learning Modules*), a GA-based algorithm to search for suitable modular architectures. In an extension to their initial work (Happel and Murre 1994), they proposed a modular neural network framework to model the brain's global and local structural regularities. Instead of beginning with a fully connected network, they build a modular structure with sparse connections. In a case study of recognizing handwritten digits, a significant improvement has been observed in the generalization performance (Fig. 26). Also inspired by the brain, Elias (1992) described a connection pattern, modeled after morphologically complex biological neurons which are evolved with GA and implemented over analog electronic hardware constructed from artificial dendritic trees which exhibit a spatiotemporal processing capability. In another biologically motivated work by Bornholdt and Graudenz (1992), arbitrary connections, including asymmetric, backward directed, and feedback loops among the neurons of a generally diluted model-brain was allowed. They divided the neurons of the model into three groups: input neurons, cortex neurons, and output neurons. The architecture of the cortex, which is not grouped into layers, is designed to be arbitrary. Similarly, Jacob and Rehder (1993) proposed a hierarchically structured connectionist model, inspired by the intuition of a network designer who builds architectures in mainly two stages. In the first stage, a course connectivity structure is evolved. In the second stage network, architecture is fine-tuned through learning and adaptation by the specialized task. They used context-free grammar to represent net connectivity and optimized both weights

and architecture of neural networks. Inspired by Kitano's work and aiming to develop a universal network generator, Voigt et al. (1993) described a model called *Building Blocks in Cascades Learning* (BBC-Algorithms) and its extension with an evolutionary framework called BBC-EVO algorithm based on *L-systems*. Their evolutionary framework was based on the classical *Evolution Strategy* with self-adaptation of strategy parameters. Later they extended their work with BBC-EA (Born and Santibánez-Koref 1995; Born et al. 1994), discussing the structuring task as an example of the pseudo-boolean optimization problem.

Measuring the effects of representation was an interest by several authors including Marti (1992). He used GAs to obtain parameters to generate neural networks and analyzed their behaviors with various genome representations. Karunanithi et al. (1992) adopted a constructive approach named *Genetic Cascade Learning* as an intuitive solution to competing convention problems. Their proposed method combined GA and the properties of the *Cascade-Correlation* learning algorithm (Fahlman and Lebiere 1990) by adding one hidden unit at a time. Alba et al. (1993a; 1993b) built a three-level genetic ANN design, where the top-level defines structure, the middle layer defines connectivity and the lowest level sets the weights. They developed a tool called GRIAL *(Genetic Research in Artificial Learning)* to apply various GA techniques and used PARLOG, a *Concurrent Logic Language* to implement GA and ANN behavior in GRIAL, which attains intra-level distributed search and parallelism. Following the principle of natural evolution and growth, Boers et al. (1992) described a reverse engineering model of the mammalian brain using *L-systems* combined with GA to design ANN architectures, trained with BP. With the help of an indirect representation scheme based on production rules, they were able to reduce the computation cost with modular and scalable architectures. Following a different path, Braun and Weisbrod (1993) utilized a direct representation scheme for a constrained architecture space. Aiming to overcome competing conventions problem, they proposed ENZO, a genetic algorithm-driven neural network generator which evolves both the architecture and weights for specific problems. Mandischer (1993) developed a representation scheme to construct backpropagation networks using GA. It is based on layers, which structures the network as a list of network parameters and layer blocks. Another attempt to optimize both the architecture and weights of a neural network was reported by White & Ligomenides (1993) using a node-based encoding. In their proposed algorithm, which they call *GANNet*, they used a distributed GA with multiple populations to discover and protect the best individuals among subpopulations. Another survey was carried out by Koehn (1994) in his master thesis, investigating various encoding strategies which influence GAs and ANNs.

Although GA has dominated the area of research as a powerful optimization method, other evolutionary computation techniques were also on the focus of some researchers. McDonnell and Waagen (1993) evolved connectivity and weights of ANN simultaneously, by using evolutionary programming (EP) as a stochastic search method, considering the task as a combinatorial optimization problem. One of the most cited works taking a non-GA approach is by Angeline et al. (1994). They implemented *GNARL* (GeNeralized Acquisition of Recurrent Links) which adopts a modified EP by using only selection and mutation operator to optimize both architecture and weights of recurrent networks. They argued that GAs are not well suited for the evolution of networks due to the *deceptive* nature of the crossover operator. In a relatively unusual approach, Oliker et al. (1993) proposed *distributed genetic algorithm*, which operates separately on each of the hidden and output layer nodes and adopts a fitness function which considers the overall network error and estimation of the network's convergence capability. It creates multiple populations of GA which work together to build an optimum network structure while allowing parallel processing

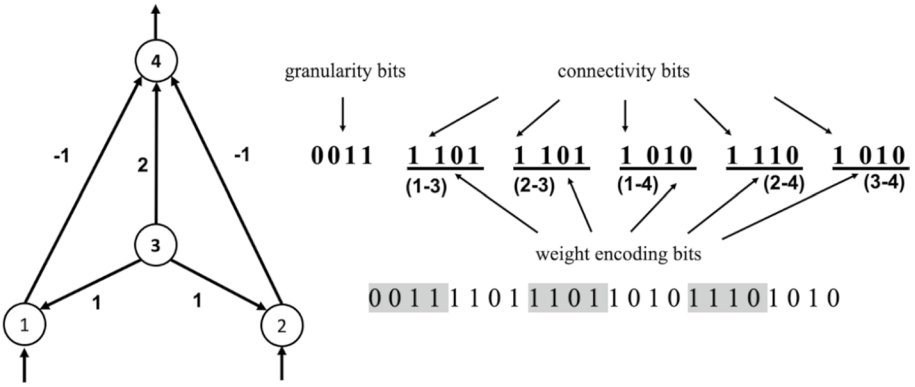


Fig. 27 Granularity encoding scheme of ANNA ELEONORA (redrawn from Maniezzo (1994)). Connectivity is represented with presence/absence bits. The first byte of the string indicates the number of bits (*the granularity*) for corresponding weight

and reducing the search space. Another modification of GA was proposed by Zhang and Mühlenbein (1993; 1993), called *Breeder Genetic Programming* (BGA) to optimize both architecture and weights at the same time. The authors define BGA as a recombination of ES and GAs, having a search process mainly driven by recombination and using variable-size chromosomes, which is a typical characteristic of GP. It employs *Occam's Razor* (Thorburn 1918) in the fitness function, which states *simpler models should be preferred to unnecessarily complex ones*. Thus, their proposed method establishes an optimal trade-off between the performance and size of the network by allowing partial and direct connections between distinct layers, encouraging smaller architectures.

The success of evolutionary approaches had intensified research on this topic by the mid-1990s. Maniezzo (1994) proposed ANNA ELEONORA (Evolutionary Learning Of Neural Optimal Running Abilities) based on a parallel genetic algorithm to evolve the architecture and weights of a neural network. He used an extended direct encoding scheme, similar to Miller et al. (1989), where each connection is represented directly but also followed by relevant weights in binary form. The variable-length chromosome structure features *granularity encoding* which is a control parameter designating the number of bits in the first byte of the string (Fig. 27).

Another innovative approach inspired by biology is the model of *morphogenesis* by Michel and Biondi (1995). They defined both structure and weights of a neural network by morphogenesis, starting with a single cell, and establish connections allowing modular recurrent networks by mimicking protein synthesis regulation in biology. Wong and Goh (1994) used GENETICA-A, which is a GA-based neural network optimization module to search for the best possible network architecture. They applied overlapped tree structures as chromosome representation and described the fitness function based on the generalization performance. This helped overcome overfitting and enabled finding the best networks in the selection. In his doctoral dissertation, Sałustowicz (1995) introduced the *Semantic Changing Genetic Algorithm* (SCGA) and *Unit-Cluster Model* for automatically constructing feedforward neural networks. Aiming to find problem-dependent modular topologies and speed up the optimization, he extended simple GA and described a second-level genetic coding with bit strings. The main idea of his indirect encoding strategy is to translate the bit strings into meaningful symbol strings using a self-adapting symbol table.

Then, the symbol strings can be decoded to phenotype. Modularity is ensured via Unit-Cluster Model with *backbone* connections between clusters. He used BP for the training of the networks and calculate fitness. Tang et al. (1995) proposed a structural genetic algorithm called *Structural Genetic Trained Neural Network* (SGTNN) to optimize the architecture and weight of a neural network with a hierarchical chromosome model partitioning the genes into *control* genes and *connection* genes. They described higher-level control genes represented in binary strings to govern the architecture and connection genes with real values to represent weights and biases.

Along with the proliferation of the studies in this field, hybrid innovations started to appear which combined the advantages of various approaches. Vonk et al. (1995a) took Kitano's grammar encoding (1990) to design a neural network architecture and adopted Dasgupta and McGregor's *Structured Genetic Algorithm* (sGA) to evolve weights (Dasgupta and McGregor 1992). They described the chromosome with a two-level hierarchy, where the top-level represents the structural part as matrix rewriting rules and the bottom level represents the parametric part with real-valued strings. Cho and Shimohara (1996) proposed another modular approach which adopted the ideas and methodologies of Artificial Life based on the neuropsychological evidence proving the modular structure of human information processing system. By using a tree-structured genetic encoding to express connections between modules, they applied GA to make ANNs evolve their own structure autonomously. Rudolph (1996) defined the general paradigms and theoretical foundations of ANN architecture design, reaching the conclusion that GAs are the most convenient approach to achieve the best generalization performance. Using a direct representation, he described a specifically designed GA and a modified fitness function based on the theory of *dimensionally homogenous functions* (Rudolph 1995), to achieve optimally generalizing neural networks. In another study, Stepniewski and Keane (1996) stressed the

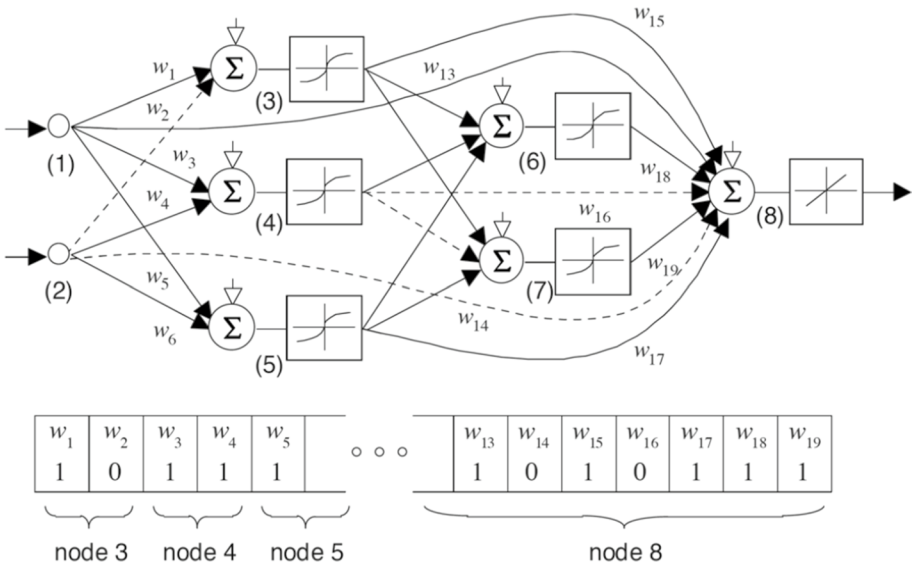


Fig. 28 Network architecture and its encoding used by GA (redrawn from Stepniewski and Keane (1996)). All hidden units have sigmoid activation and output has a linear $(f(x)=x)$ activation function. The authors used a direct representation scheme where a direct connection is possible from each hidden and input node to the output

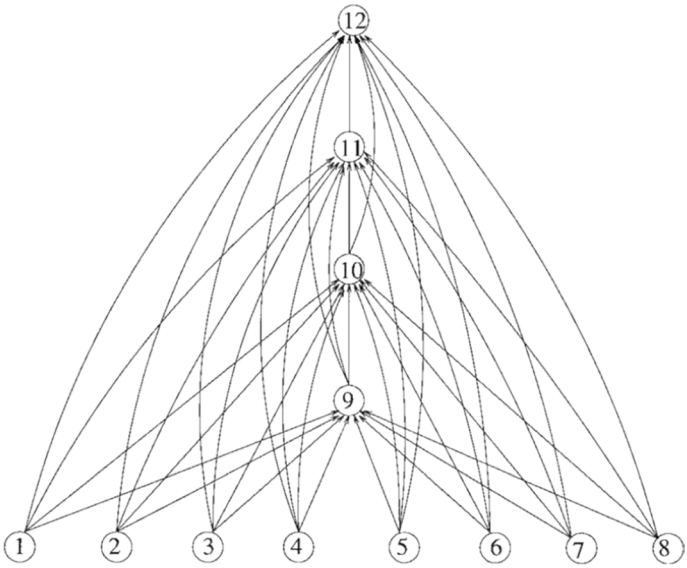


Fig. 29 An example of an optimum network evolved by EPNet for the eight-parity problem (redrawn from Yao and Liu (1997))

challenges when using GA to build ANN architectures and proposed solutions to optimize the process. Arguing the importance of penalizing model complexity and simplification of the structure, they developed relevant procedures to increase the efficiency of GA. Additionally, they formulated a novel objective function to measure the generalization ability (Fig. 28).

Evolutionary programming approaches were also popular due to their flexibility in using mutation as the sole genetic operator while not requiring the variables to be encoded. It also doesn't utilize crossover, which is argued by some authors to have a deteriorating effect on optimizing ANN structures. Fang and Xi (1997) optimized the architecture and weights of neural networks by using EP. They further extended their work to design recurrent neural networks with a new type of connection, what they called *delayed links*. Yao and Liu (1995) used EP to evolve neural networks for medical applications. Both authors published two additional papers in 1996. In the first paper, they used EP with new mutation operators to evolve the structure and weights of general neural networks with different nodes (Liu and Yao 1996a). In their second paper, they proposed an EP-based algorithm, named as *Population-Based Learning Algorithm* (PBLA) to find both architecture and weights of a neural network. Unlike the other previous approaches, they introduced *partial training* when evaluating the population, which resulted in substantial savings in computational cost (Liu and Yao 1996). The same authors further developed *EPNet* (Yao and Liu 1997), which evolves both architecture and weight simultaneously, alleviating the noisy fitness evaluation problem (Fig. 29). They put special emphasis on network behavior rather than its circuitry and encouraged parsimonious structures by architectural mutations.

An innovative encoding strategy was proposed by Luke and Spector (1996) in 1996, namely *Edge Encoding*. They addressed the weaknesses of *Cellular Encoding (CE)* by Gruau (1992, 1993, 1994) and described a new alternative approach for ANN representation. Similar to CE, Edge Encoding used S-expression-based GP techniques to evolve

arbitrary graph structures. However, it differs from CE, in growing graphs by modifying the *edges*, while CE modifies *nodes*. Furthermore, it favors graphs with fewer interconnections in contrast to the highly connected structure of CE. In the same year, another breakthrough was achieved by Moriarty and Miikkulainen (1996) with a new enhanced learning method called *Symbiotic Adaptive Neuro-Evolution* (SANE). Unlike conventional evolutionary approaches, SANE evolved a population of *neurons* instead of *networks*. Each individual neuron in the population represents only a *partial solution* and establishes a connection with the other nodes to construct a complete network. This *symbiotic evolution* approach helps genetic algorithms search diverse areas of the solution space concurrently. The authors evaluated SANE using the inverted pendulum problem and obtained superior performance when compared to the two-layer *Adaptive Heuristic Critic* of Anderson (1989), the *Q-learning* method of Watkins and Dayan (1992), and the GENITOR of Whitley et al. (1990). Further, they carried out empirical studies to demonstrate the effectiveness of their proposed method and improved SANE with Hierarchical SANE, which integrates two levels of evolution in a single framework (Richards et al. 1998). The works followed stressed the advantages of *cooperative coevolutionary* approaches, mostly inspired by *implicit fitness sharing* (Horn et al. 1994; Smith et al. 1993) which promotes diversity through performing parallel searches in decompositions of the architecture space (Moriarty and Miikkulainen 1997). Later in 1998, Richards et al. (1998) used SANE to explore the evolution of networks capable to play the game of GO on small boards with no pre-programmed knowledge and achieved promising results for full-scale GO. Later, Gomez and Miikkulainen (1999) proposed another neuroevolution method called Enforced Sub-populations (ESP) as an enhancement to SANE. Similar to SANE, ESP also evolved neurons instead of full networks. However, it differs from SANE in that a subset of neurons, rather than individual neurons, form the complete architecture (Fig. 30). It also allows the evolution of recurrent networks.

It is remarkable that only eight years after the first attempt to combine ANN and GAs, there were a significant number of researchers working on this topic and proposing innovative solutions in terms of representation and algorithms. De Carvalho (1997) described a simple GA approach to automate the design of neural networks by representing the structure with key parameters. Bebis et al. (1995; 1997) proposed the *coupling of GA* and *weight elimination* to prune larger size networks while preserving generalization ability. They expressed the network size and complexity in terms of the number of connections and used a special fitness function which takes into account both network size and the generalization performance. With a similar goal, Zhang and Ohm (1997) proposed a new representation scheme called *neural trees* to create parsimonious neural networks. They used a hybrid evolutionary approach in which the architectures are evolved by Genetic Programming and other parameters by a local search based on the *breeder genetic algorithm* (BGA). Similarly, Opitz and Shavlik (1997) presented REGENT (*REfining, with Genetic*

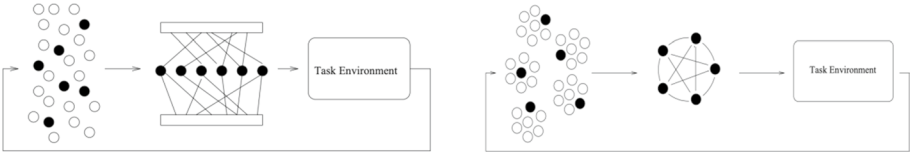


Fig. 30 A comparison of SANE and ESP (SANE is depicted on the left, and ESP is depicted on the right). The ESP suggests the segregation of neurons into sub-populations to form a complete network (redrawn from Gomez and Miikkulainen (1999))

Evolution, Network Topologies) focusing on better generalization while concentrating on connectionist theory-refinement systems to make effective use of problem-specific knowledge and better exploit available computing power. The method they described mainly differs from the other approaches in that it adopts *Lamarckian Evolution*, a theory based on the inheritance of characteristics acquired during a lifetime (Whitley et al. 1994). They implemented this theory to pass trained network weights to the offspring.

Eggenberger (1997) proposed a biologically inspired model based on an *artificial genetic regulatory system* (AGRS) to develop the architecture of ANNs. As an abstraction of the natural process, the AGRS controlled epigenetic development such as cell division, cell death, and cell differentiation. As a result, the architecture of an ANN was constructed by the dynamics of ARGS. Further, he investigated the mechanisms of axonal behavior in the brain and drew an analogy with artificial networks while modeling the chemical reactions when building connections (Eggenberger 2000). Fischer and Leung (1998) brought together the strengths of GA and BP to propose an evolutionary approach called GENNET (GENetic evolution of computational neural NETworks). By implementing a direct representation, they used GA to design the topology and utilized *Computational Neural Network Simulator* for backpropagation learning. In his master's thesis, Grönroos (1998) evolved neural network architectures by adopting and comparing various encoding methods proposed by Miller et al. (1989), Kitano (1990), Nolfi, and Parisi (1991), and Cangelosi et al (1994). Pujol and Poli (1998) optimized both architecture and weights of a neural network using *Parallel Distributed Genetic Programming* (PDGP) which represented programs as graphs based on a two-dimensional grid, arguing the potential of the excessive growing size of the chromosome in standard GP. They proposed a dual (linear and 2D) representation which they claim to avoid this problem by constraining all chromosomes having the same number of nodes. Their graph encoding allowed different kinds of crossover enabling the swap of subgraphs, while preserving the structure of functional components.

By the end of the 1990s, many researchers published a comparative analysis of various approaches and encoding schemes. Siddiqi and Lucas (1998) compared Kitano's (1990) matrix rewriting with simple direct encoding and found out that, contrary to the claimed superiority of indirect encoding, direct encoding did not get worse with the increase of network size and performed at least as well as the matrix rewriting graph generation system. In another comparative study, Aho et al. (1999) proposed a biologically motivated approach combining GA and L-systems to build neural network structures. On a 2-dimensional cell–matrix, they produced L rules automatically by using Genetic algorithms and they defined, what they call as "age" which controls the number of firing times to apply each rule. Although computationally intensive, they were able to find efficient structures. Another indirect encoding was proposed by Lock and Giraud-Carrier (1999) utilizing real-valued alleles to evolve both architecture and training parameters of BP-trained, fully connected feed-forward neural network using GAs. Their work differs from other invasive approaches by not evolving the weights explicitly. Instead, they evolved a single weight spread parameter for weight initialization. Then, they used BP to update weights.

The natural advantage of population-based evolutionary approaches is to keep the whole set of solutions in each generation. This helps the utilization of ensemble methods for obtaining the optimum generalization for the problem. Yao and Liu (1996) took this approach by combining all individuals in the last generation and formed an ensemble to achieve the final result. According to experiments they conducted; this approach produced better results than any isolated networks. Another remarkable study utilizing ensembles is Opitz and Shavlik's (1996; 1999) ADDEMUP (Accurate anD Diverse Ensemble-Maker

giving United Predictions). They used Genetic Algorithms to generate a population of neural networks which were combined to form an ensemble to ensure the best accuracy.

With the beginning of the new millennium, many other innovative approaches followed. Yen and Lu (2000, 2002) sought to address deficiencies regarding the design of multi-layer feed-forward neural networks and proposed a hierarchical encoding strategy composed of high-level and low-level gene segments. Their HGA-NN (Hierarchical Genetic Algorithm-based Neural Network) method is designed in a way that high-level segments have control genes determining the states (activated or deactivated) of genes in low-level segments. The evolution comes to play to add or delete hidden layers and neurons with a switch-on/off scheme. They also evolved weights and biases simultaneously. In another study, Arifovic and Gençay (2001) proposed a model selection methodology using Genetic Algorithms. Their work was among the first to utilize *elitist* strategies to improve performance. Their direct encoding approach designed the chromosome with several parameters including the structure and BP. Motivated from biology, Boozarjomehry and Svrcek (2001) developed GADONN (Genetic Auto-Design of Neural Networks) to automatically design neural networks based on Genetic Algorithms and L-systems. Their proposed method used a context-free L-system to encode the developmental rules in the genotype. This indirect approach grew the axiom in one direction rather than two dimensions, allowing it to scale better.

Many researchers referred to living organisms in the context of multi-tasking and multi-learning ability and believed that this can only be accomplished by the modular structure of the brain. Ferdinando et al. (2001) were amongst them to construct an evolutionary neural network structure and aimed to address the problem of interference when networks are given more than one task. They compared invasive and non-invasive approaches by conducting simulations and found out that a non-invasive strategy performs better on modular architectures. In another innovative study, Moon and Kong (2001) proposed a block-based neural network (BBNN) to optimize the structure and the weights at the same time by using Genetic Algorithms. In their model, networks were composed of individual blocks in a two-dimensional array with four variable input/output nodes and connection weights. The blocks could have four different configurations. In order to facilitate the use of digital hardware such as field programmable logic arrays (FPGA), they restricted the weights to integers and represented structure and weights in bit strings with a 2D direct encoding. Mizuta et al. (2001) also used GA to optimize both structure and weights of a neural network by proposing a multi-objective approach on fitness function to encourage simple models. A summary of early approaches was listed in Table 1. Distribution Graph for Evolutionary Computation Techniques Adopted is given in Fig. 31 and Distribution Graph for Representation Methods is given in Fig. 32.

5.2 The rise

The early works in the field of Neuroevolution are often concerned about encoding strategies and evolutionary approaches proposing innovative genetic operators, fitness functions, and connection schemes. Most of the studies with indirect encoding were inspired by biology and tried to mimic a natural way of designing neural networks with better generalization capabilities. Although indirect representation methods provided encouraging results, one of the most prominent works in this field came with a direct encoding approach, namely NEAT, which stands for *NeuroEvolution of Augmenting Topologies* by Stanley and Miikkulainen (2001; 2002). As the name suggests, NEAT starts from a very simple

Table 1 Early approaches to evolutionary design of neural networks (until NEAT is proposed in 2001)

Authors	Year	Target*	Method**	Encoding	Reference
Harp et al *NeuroGENESYS*	1989	A	GA	Indirect	(1989)
Miller et al *Innervator*	1989	A	GA	Direct	(1989)
Wilson *Perceptron-GA*	1989	A	GA	Direct	(1989)
Kitano *Graph L-System*	1990	A	GA	Indirect	(1990)
Schifmann et al *BP-Generator*	1990	A	ES	Indirect	(1990)
Hintz & Spofford *GA/NN*	1990	AW	GA	Direct	(1990)
Dodd *Crossover*	1990	A	GA	Indirect	(1990)
Whitley et al *GENITOR*	1990	AW	GA	Direct	(1990)
Höffgen et al	1990	A	GA	Direct	(1990)
Hancock & Smith *GANNET*	1990	A	GA	Direct	(1990)
Koza & Rice *Genetic Programming*	1991	AW	GP	Tree Based	(1991)
Marshall *GA*	1991	AW	GA	Direct	(1991)
Dasgupta & McGregor *sGA*	1992	AW	GA	Direct	(1992)
Fullmer	1992	AW	GA	Direct	(1992)
Gruau *Cellular Encoding*	1992	AW	GA	Indirect	(1992)
Schiffman et al *BP-Generator*	1992	A	GA	Indirect	(1992)
Happel & Murre *CALM*	1992 1994	AW	GA	Direct	(1992; 1994)
Elias *ADT*	1992	AW	GA	Direct	(1992)
Bornholdt & Graudenz *GARFIELD*	1992	A	GA	Direct	(1992)
Marti	1992	A	GA	Direct & Indirect	(1992)
Karunanithi et al *GCL*	1992	AW	GA + Cascade Correlation	Direct	(1992)
Robbins et al *GANNET*	1993	A	GA	Direct	(1993)
Alba et al *GRIAL*	1993	AW	GA	Direct	(1993a; 1993b)

Table 1 (continued)

Authors	Year	Target*	Method**	Encoding	Reference
Boers et al	1993	A	GA	Indirect	(1992; 1993)
Braun & Weisbrod *ENZO*	1993	AW	GA	Direct	(1993)
Jacob & Rehder	1993	AW	GA	Indirect	(1993)
Mandischer	1993	A	GA	Direct	(1993)
McDonnell & Waagen	1993	AW	EP	Direct	(1992)
Oliker et al	1993	AW	Distributed GA	Direct	(1993)
Voigt et al *BBC-EVO*	1993	A	ES	Indirect	(1993)
Zhang & Mühlenbein *BGP*	1993a, 1993b	AW	ES + GA + GP	Indirect	(1993a; 1993b)
White and Ligomenides *GANNet*	1993	AW	Distributed GA	Direct	(1993)
Moriarty & Miikkulainen	1993	A	GA	Indirect	(1993)
Maniezzo *ANNA ELEONORA*	1994	AW	Parallel GA	Direct	(1994)
Wong & Goh *GENETICA-A*	1994	A	GA	Direct	(1994)
Angeline et al *GNARL*	1994	AW	EP	Direct	(1994)
Born et al *BBC-EA*	1994	A	ES	Indirect	(1994)
Michel & Biondi	1995	AW	GA	Indirect	(1995)
Sałustowicz *SCGA*	1995	A	GA	Indirect	(1995)
Tang et al *SGTNN*	1995	AW	sGA	Direct	(1995)
Vonk et al	1995	AW	sGA	Indirect	(1995a)
Vonk et al *GPNN*	1995	AW	GP	Tree-based	(1995c)
Born & Santibánez-Koref *BBC-EA*	1995	A	ES	Indirect	(1995)
Yao & Liu *EP-Net*	1995	AW	EP	Direct	(1995)
Cho & Shimohara	1996	A	GP	Tree-based	(1996)
Rudolph	1996	A	GA	Direct	(1996)
Stepniewski & Keane	1996	A	GA	Direct	(1996)
Liu & Yao	1996	AW	EP	Direct	(1996a)
Liu & Yao *PBLA*	1996	AW	EP	Direct	(1996)

Table 1 (continued)

Authors	Year	Target*	Method**	Encoding	Reference
Luke & Spector *Edge Encoding*	1996	A	GP	Tree-based	(1996)
Moriarty & Miikkulainen *SANE*	1996	A	GA	Indirect	(1996)
Moriarty & Miikkulainen *Hierarchical SANE*	1996	A	GA	Indirect	(1996)
Yao & Liu *EPNet with Ensembles*	1996	AW	EP	Direct	(1996)
Fang & Xi	1997	AW	EP	Direct	(1997)
De Carvalho *NeurEvol*	1997	A	GA	Direct	(1997)
Bebis et al *GA_BP_WE*	1995, 1997	AW	GA	Direct	(1995; 1997)
Zhang et al *GP-BGA*	1997	A	GP-BGA	Indirect	(1997)
Opitz & Shavlik *REGENT*	1997	A	GA	Direct	(1997)
Yao & Liu *EPNet*	1997	AW	EP	Direct	(1997)
Fischer & Leung *GENNET*	1998	AW	GA + BP	Direct	(1998)
Pujol & Poli *PDGP*	1998	AW	Parallel Distributed GP	Indirect	(1998)
Gomez & Miikkulainen *ESP*	1999	AW	GA	Indirect	(1999)
Aho et al	1999	A	GA	Indirect	(1999)
Lock & Giraud-Carrier	1999	A + Initial Weights	GA	Indirect	(1999)
Opitz & Shavlik *ADDEMUP*	1999	A	GA	Direct	(1996; 1999)
Yen & Lu *HGA-NN*	2000	AW	GA	Direct	(2000; 2002)
Boozarjomehry & Svrcek *GADONN*	2000	A	GA	Indirect	(2001)
Moon & Kong *BBNN*	2001	AW	GA	Direct	(2001)
Mizuta et al	2001	AW	GA	Direct	(2001)

*A: Architecture Optimization, AW: Architecture and Weights Optimization, **GA: Genetic Algorithms, EP: Evolutionary Programming, GP: Genetic Programming, ES: Evolutionary Strategies

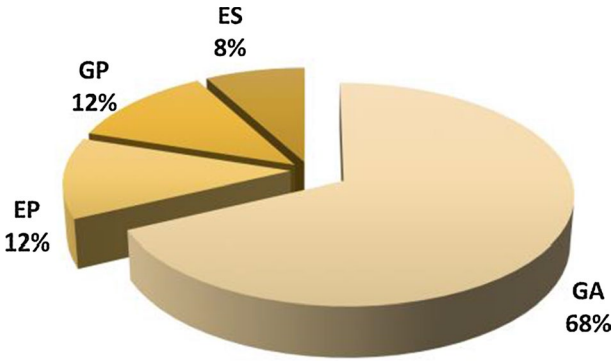


Fig. 31 Distribution Graph for Evolutionary Computation Techniques Adopted (1989–2001)

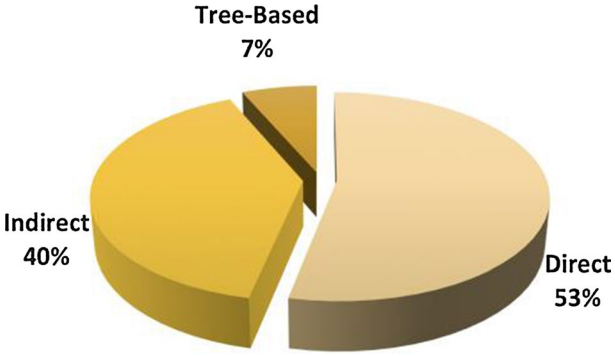


Fig. 32 Distribution Graph for Representation Methods (1989–2001)

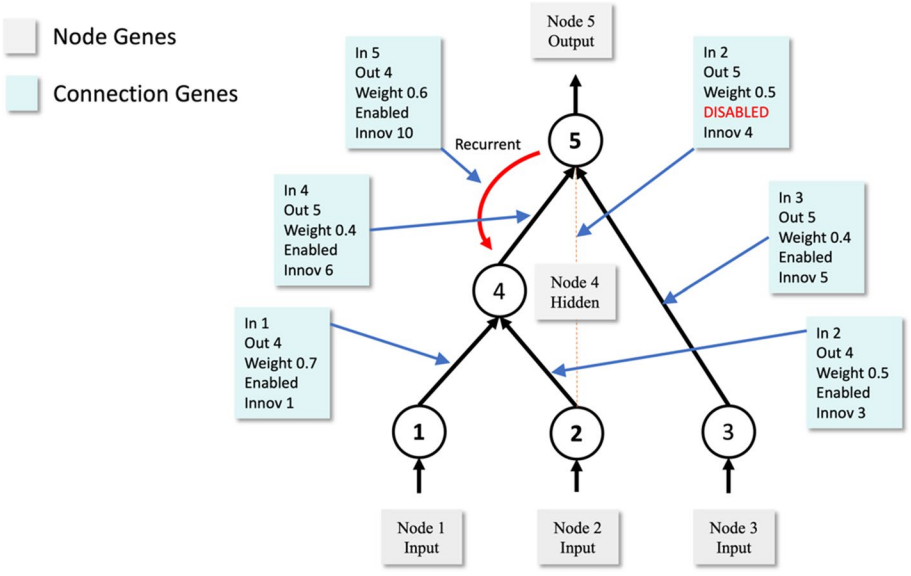


Fig. 33 An example of Genotype to Phenotype mapping of NEAT. The genetic representation of the network architecture is composed of genes which have innovation numbers marking their inception. Redrawn from Stanley and Miikkulainen (2001)

architecture and grows the neural network as necessary to meet the desired performance. The authors aimed to address competing convention problems by introducing an innovative crossover strategy through historical markings. In this way, the marking tells the crossover operator which parts of the networks have common features, thus can be swapped. They also figured out a way to prevent the early elimination of recent modifications to network architecture through a mechanism what they call *speciation*. An example of the Genotype to Phenotype mapping of NEAT is depicted in Fig. 33.

NEAT was a major breakthrough and had a significant impact in the field of Neuroevolution. Since its publication, it is applied to many real-world tasks and is still being experimented by many researchers, even today. The strongest argument made by the authors is that evolving structure along with the connection weights can significantly enhance network performance. They named such invasive approaches as TWEANNs (Topology and Weight Evolving Neural Networks) and proposed networks starting from the minimal structure and gradually increasing complexity. Unlike some previous studies, NEAT ensures network parsimony by gradually augmenting models instead of penalizing network complexity. Furthermore, innovation is protected through *speciation* which allows the organism to compete primarily within their own niches. The main challenge in augmenting topologies is that modification to the network results in lower fitness and causes new architectures to die easily among the population. The authors of NEAT were able to overcome this by providing the opportunity to new individuals to optimize their structure within the niche. Another innovative technique used in NEAT is *explicit fitness sharing* through historical markings, which helps track and measure the similarity of genes during genetic operations. Contrary to the general assumption, tracking the historical origins of genes didn't require much computation in NEAT. This solution helped efficiently overcome competing convention problem.

With the expedition of research on hybrid approaches, augmenting topologies, and modular structures, many other works have been followed. Wang et al. (2002; 2003) took a somewhat interesting approach by combining constructive methods with Genetic Algorithms. In their model, a dynamic *constructive* method is initially adopted to train the neural network, and then GA is used to *prune* the trained network. In this approach, constructing and pruning the network were accomplished by adding or removing nodes and connections. By employing a binary direct encoding, they aimed to ensure simplicity and generality, while facilitating the use of genetic operators such as crossover and mutation. Barrios et al. (2002) described ADANNET (Automatic Design of Artificial Neural Networks by Evolutionary Techniques) synthesizing the structure and accomplishing training. With this work, they introduced a new indirect encoding technique called *basic-architectures codification* and new crossover operators named *Hamming Crossover* and *mathematical morphology crossover*. The same authors further developed GANN (Genetic Algorithm Neural Networks) which used two twin GAs working in parallel to evolve the structure and weights of a neural network (Barrios et al. 2003). Similar to previous work, they employed an indirect encoding scheme based on binary codification on an algebraic structure. The main advantage of their proposed approach is generating regular patterns while protecting the meaningful sub-networks discovered to improve the networks among the population.

A number of researchers have also explored Coevolutionary approaches for finding better architectures (Moriarty and Miikkulainen 1997; Moriarty and Mikkulainen 1996; Opitz and Shavlik 1996; Potter and De Jong 1995). García-Pedrajas et al. (2003) proposed COVNET, a cooperative coevolutionary model for evolving ANNs. As previously explained in Sect. 3. this paradigm is based on the idea of creating several subpopulations of subnetworks evolving in cooperation to solve a problem. The model they developed evolves

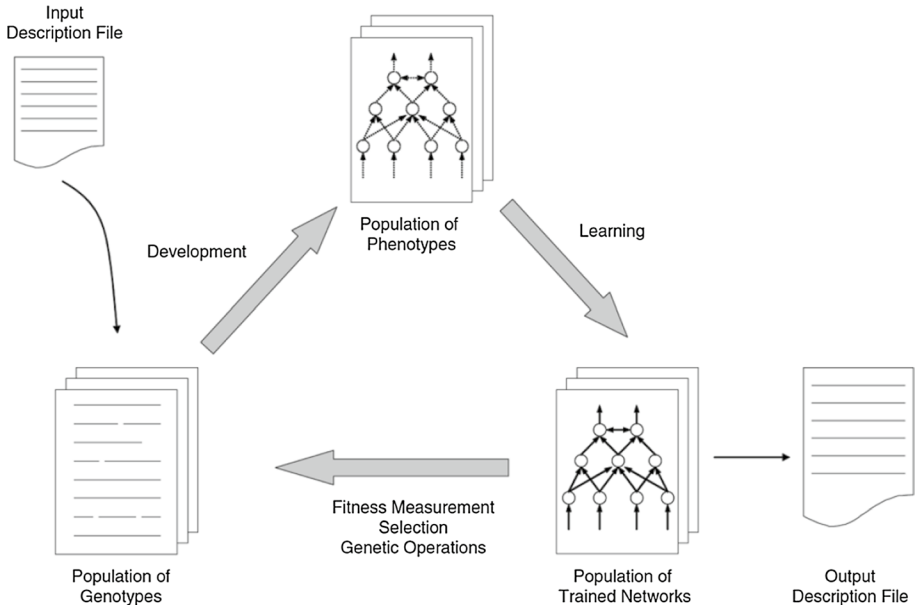


Fig. 34 Descriptive Encoding approach by Jung and Reggia (2004). A human-written specification of neural networks to be evolved using high-level language is fed to the model. A human-readable output description file is obtained after the evolution process

subnetworks rather than complete networks, thus combining them to form groups to build the final network. Later, the authors proposed a new crossover operator to address the permutations problem, which is already known as a competing conventions problem (García-Pedrajas et al. 2006). A similar cooperative coevolutionary strategy was adopted by Reisinger et al. (2004) improving the previously proposed NEAT algorithm as *Modular NEAT*. It is described by the authors as a coevolutionary neuroevolution method which allows reusing fundamental neural substructures for modularity. A population of blueprints specified combinations of modules. As modules were discovered, the evolution and modularization were carried out concurrently. In modular NEAT, instead of duplicating, the genes were reused in different spatial locations in the network. Based on the experimental results, Modular NEAT was able to outperform standard NEAT by obtaining better solution networks as well as finding optimum networks faster. The modularity of networks was a hot topic of research which was also studied by Jung and Reggia (2004; 2006). They introduced a problem-independent approach based on *descriptive encoding* using a high-level language. Unlike conventional evolutionary techniques where all aspects of a network are determined automatically, their model allowed the user to incorporate domain knowledge and restrictions to define search space for better performance and lower computational cost. Similar to the abstraction process in computer programming, they let users specify a problem by writing a simple text file using a high-level language (Fig. 34).

Another enhancement over conventional evolutionary methods was proposed by Leung et al. (2003) with an improved GA model. They described a three-layer neural network with switches introduced to its connections. By using the improved GA, they tuned the structure and parameters. The model they proposed allowed the final architecture to be partially connected after the application of optimization. This helped reduce the computational cost

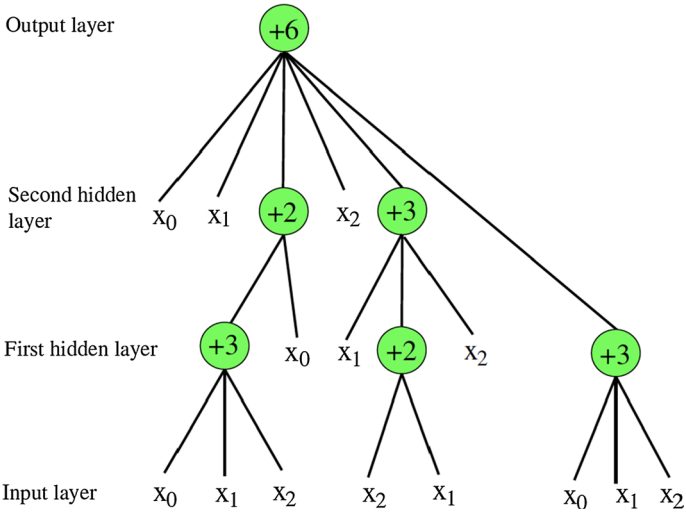


Fig. 35 A Neural Tree representation. In this example, a neural tree is depicted with three instruction sets and three input variables x0, x1, and x2. Redrawn from Chen et al. (2004)

significantly. Chen (2004) et al. proposed an innovative hybrid approach which adopted a variant of Genetic Programming called *modified probabilistic incremental program evolution algorithm* (MPIPE). It eliminates genotype to phenotype, or reverse mapping by using a neural tree representation and determines an optimal structure and parameters of neural trees. In this representation a neural tree can be directly calculated as a flexible neural network, thus helping reduce computational costs to calculate fitness function. In order to overcome the problem of enlarging search space, they restricted architecture of neural trees (Fig. 35).

In order to reduce the computational cost of training every individual network structure during evolution, invasive approaches became more popular among researchers. Tan (2004) proposed a hybrid evolutionary algorithm called GAEPNet to evolve both weights and architecture of ANNs simultaneously. What made GAEPNet different from the other approaches was the way it combined the strengths of GA and EP on a real-valued multi-matrix representation. Considering the challenges of utilizing the crossover operator to evolve ANN architectures, the model he described introduced a linear combination crossover and efficient mutation of EP. The encoding scheme was an innovative approach encompassing four matrices, similar to Miller et al.'s (1989) connectivity matrix. In these matrices, instead of binary values, real values of weights were used. If a connection did not exist between two nodes, the weight was expressed as zero.

With the introduction of NEAT in 2001 by Stanley and Miikkulainen (2001), intensified research on augmenting topologies have been observed. Kassahun and Sommer (2005) proposed EANT (Evolutionary Acquisition of Neural Topologies) to evolve the structure and weights of neural networks. Similar to NEAT, it starts with a minimal architecture and complexifies the model along the evolutionary process. The *exploration* of new structures was initiated in the event that it was impossible to further *exploit* the existing structures. EANT used CMA-ES (Covariance Matrix Adaptation-Evolution Strategy) and introduced relatively compact genetic encoding onto a linear genome. This representation strategy allowed evaluating the network performance without the decoding process. The linear

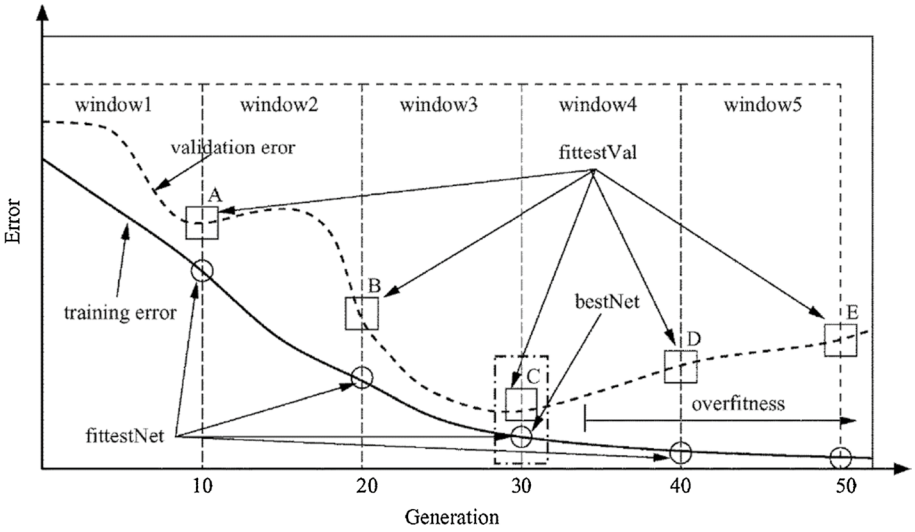


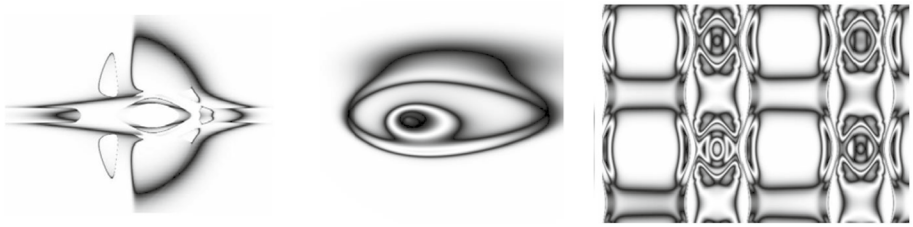
Fig. 36 Sliding Window technique used in Palmes et al. (2005). The boxes show the locations of the validation performance of optimum networks. The stopping criterion compares the trend of these boxes to detect overfitting while marking the network as the fittest (redrawn from Palmes et al. (2005))

genome in EANT had a mechanism similar to a tree-based program encoded by linear programs where terminals were interpreted as network inputs as well as jumper connections and functions as neurons. Later this approach was improved by Siebel and Sommer (2007) as EANT2. Similar to its first version, it used an indirect encoding and evolved neural structures by starting from a minimal model and gradually developing to achieve optimum ANN architectures. The authors stated the main difference of EANT2 as a clear separation of structural exploration and exploitation. They also used CMA-ES in parameter optimization and employed less user-defined parameters. Unlike speciation in NEAT, EANT 2 had an explicit way of preserving diversity.

In another sophisticated approach, Palmes et al. (2005) proposed MGNN (Mutation-based Genetic Neural Network) to address the problem of computer-intensive operations required to train individuals among the population using the gradient-descent-based back-propagation method. MGNN replaced BP by using the mutation strategy of local adaptation inherent in Evolutionary Programming (EP). As an *invasive* approach, the authors adopted an EP-based direct encoding scheme to facilitate flexible and less constraining fitness function which is easier to calculate. The encoding scheme they proposed helped to overcome the challenge of possible occurrences of deceptive mapping, including competing convention problems. Another innovative characteristic of MGNN is the monitoring of overfitting through what the authors called sliding windows which applies a stopping criterion to the algorithm (Fig. 36). The results of the simulations they carried out suggested that overfitting does not necessarily occur only in complex structures. Castellani (2006) proposed ANNE (Artificial Neural Network Evolver) as another invasive approach, which also embedded the evolution of input features concurrent with architecture and weights. He opted to use direct encoding due to the complexity of implementation on his Lamarckian model and aimed to simplify the optimization process to improve performance. Based on his experiments, he claimed that evolutionary feature selection allowed more accurate and consisting learning results compared to widely used PCA (Principal Component Analysis).

Although a multitude of papers investigated the possibility of better optimizing ANNs with evolutionary approaches, few had success to efficiently evaluate the real capability of such combinations. Benardo and Vosniakos (2007) aimed to develop novel criteria which quantify an ANN's performance on both aspects (training and generalization) in addition to its complexity. They proposed a methodology to determine the best architecture by using GAs and defined the best-performing networks based on a set of predetermined criteria. A noninvasive approach by Fiszelew et al. (2007) combined GA and BP to optimize a neural network and utilized direct encoding for chromosome representation. They used GA to define the architecture and BP to train and evaluate the performance. In order to improve results, they applied techniques such as repetition of training, early stopping, and complex regulation. In a comparative study by Rocha et al. (2007) two hybrid combinations of ANNs and evolutionary computation approaches were investigated. In the first method, only the architecture is evolved, while in the second one both architecture and weights are simultaneously optimized by evolutionary computation. Their experiments on various real-world data sets suggested the efficiency of the latter.

The natural way of developing structures has always inspired practitioners and led to more research on defining better encodings and algorithms. Lee et al. (2005) proposed a nature-inspired mechanism for the autonomous design of ANN architectures. They used a developmental model grown from a set of production rules of the L-system represented by the DNA coding. As previously been explained in Sec.4, the L-system they adopted was based on a parallel rewriting mechanism motivated by the growth models of plants. Another motivation from biology is the connectivity patterns in biological brains exhibiting regular repeating motifs. In 2007, Gauci and Stanley (2007; Stanley et al. 2009) aimed to discover such geometric regularities and proposed HyperNEAT (Hypercube-based Neuroevolution of Augmenting Topologies), as a major improvement to NEAT, complex connectivity schemes with evolving ANNs while introducing a generative indirect encoding method called CPPN (Compositional Pattern Producing Networks). Based on the hypothesis that ANN structure should implement a means for evolution to exploit task geometry, this encoding strategy adopts the principle of locality in nature. As being effectively employed by biological brains, relevant operations are performed through local connectivity, since long-distance requires more resources, greater accuracy, and better organization. CPPN was designed to efficiently represent natural regularities and symmetries. As can be seen in Fig. 37, Stanley showed that CPPNs can produce spatial patterns with important geometric motifs such as symmetry, imperfect symmetry, and repetition with variation. The CPPN can encode a high-dimensional



(a) Bilateral Symmetry **(b)** Imperfect Symmetry **(c)** Repetition with variation

Fig. 37 Examples of spatial patterns with important motifs produced by CPPNs (redrawn from Gauci and Stanley (2007)). **a** Bilateral Symmetry, **b** Imperfect Symmetry, **c** Repetition with variation

output space with fewer parameters (Fernando et al. 2016). On a project called Picbreeder (Secretan et al. 2008, 2011), internet users evolve images by selecting which CPPNs to breed. Examples of spatial patterns with important motifs produced by CPPNs are given in Fig. 37. The natural equivalent of these motifs is left–right symmetries in vertebrates, right-handedness, and cortical columns. HyperNEAT proved to be superior to the direct encoding approach by NEAT and was able to generalize significantly better while scaling to a network of over eight million connections.

HyperNEAT showed that a pattern of weights across the connectivity of ANNs can be generated as a function of its geometry. However, it was the user's discretion to place hidden nodes in an infinitely dense geometry. Later Risi et al. (2010) developed an extension to HyperNEAT, namely ES-HyperNEAT (evolvable-substrate) which determines the placement and density of the hidden nodes. The authors' main insight was that there exists, what they call *implicit* clues how to carry this task out to extract useful information in the connectivity scheme. The pursuit of creating the brain-like structure of ANNs continued with Gauci & Stanley's work in 2010 (2010), which argued the effectiveness of creating topographic regularities through HyperNEAT. Their experiments on checkers-playing ANNs revealed that representing evolved ANNs as indirect functions of their geometry which can efficiently exploit geometric regularities creates brain-like structures with better generalization capacity. Another promising study inspired by the natural process of growing nervous system by evolutions is De Campos et al.'s work (2011) which used L-systems as a recipe to develop neurons and GA to evolve the architecture. Similar to augmenting topologies their proposed approach begins the search process with simpler and smaller structures and gradually grows into complex ones. Later De Campos et al. (2015) improved this model with ADEANNs (Artificial Development and Evolution of ANNs) implementing a constructive approach using memory.

A comparative analysis by Drchal and Šnorek (2008) examined tree-based indirect developmental encodings: Gruau's *Cellular Encoding* (1994) and Luke and Spector's *Edge Encoding* (1996). Then, the authors compared their successors: Gene Expression Programming (GEP) and Grammatical Evolution (GE) to optimize trees. They found that GE is superior to GEP in fewer generations needed to optimize the development tree, while GEP is more likely to produce smaller solutions. They also showed that

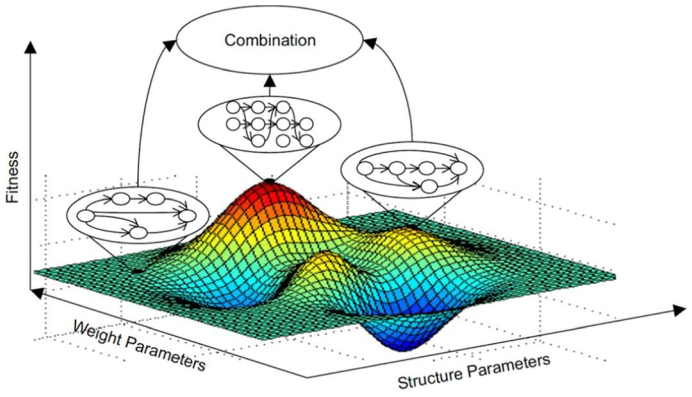


Fig. 38 Identification of peaks in various regions using fitness sharing for speciation (redrawn from Kim and Cho (2008))

Edge Encoding found solutions faster although producing a very large development tree. The authors argued the necessity of a mechanism to prevent such uncontrolled growth on developmental indirect encodings. Tsoulos et al. (2008) proposed a GE approach using GAs while encoding the network architecture and its parameters with a context-free grammar (CFG). Later this work was altered by Soltanian et al. (2013) to evolve only architecture, while weights are optimized by using BP. Another improvement to Tsoulos' original approach was proposed by Ahmadizar et al. (2015) which combined GE and GA to evolve both architecture and weight simultaneously. This work used an adaptive penalty approach to simplify ANNs generated through the evolution process. In 2017, Assunção et al. (2017) proposed DSGE (Dynamic Structured Grammatical Evolution), as a new genotypic representation for structured generic evolution (SGE), to address the shortcomings of grammar-based neuroevolution approaches and evolve networks with more than one hidden layer and multiple outputs. DSGE differs from the previous GE and SGE approaches by growing the genotype as needed (instead of encoding the largest allowed sequence) and removing the necessity of grammar pre-processing to calculate the maximum tree size of each non-terminal symbols. Furthermore, it allows creation of dynamic rules which specify the connection possibility of each neuron. Kim and Cho (2008) adopted a different strategy benefiting from ensembles of GA population. Stating the importance of diversity, they created their speciation-based evolutionary model through fitness sharing (similar to NEAT) and then combined the networks by the behavior knowledge space method. They used average output, Pearson correlation and modified Kullback–Leibler entropy to calculate distance between individuals and aimed to identify peaks in various regions of the search space (Fig. 38).

The main idea in several papers is to remove connection constraints and build a flexible ANN scheme to discover uninhabited structures, such as direct connections from input to output, or parsimonious links with nearby nodes. Rivero et al. (2009) proposed a GP-based approach to automate developing simple ANNs with an independent connectivity scheme. The authors aimed to provide assistance to AI experts to design their models without any prior domain knowledge or requirement to set any parameters. They also set an objective to obtain networks with similar generalization capability with the minimum number of network elements (nodes, connections, etc.). The most recent work, evolving conventional backpropagation neural network was reported by Chen (Yuh Wen) and Shiu (2019)by using a simple GA approach called GABPNN. The authors aimed to compare the performance of their model with state-of-the-art, hand-crafted CNN architectures on the MNIST handwriting recognition dataset. Although performed poorly, the authors showed the significant difference in computational resources and time required to obtain a real-world classifier, since their model was very fast to train and achieved acceptable accuracy.

While most studies are focused on single-objective evolutionary algorithms, more recent work explored the effectiveness of Multi-Objective approaches. Typically, researchers aim to achieve better accuracy while keeping the architecture as simple as possible. Oong and Isa (2011) proposed a multi-objective approach called HEANN (Hybrid Evolutionary Artificial Neural Network) to evolve architecture and weights simultaneously. Their model is based on scalarized multi-objective learning which combines objective into a scalar cost function. It differs from the other evolutionary approaches by providing a balance of global and local search through adaptation of mutation probability and step size of weight perturbation. To be more specific, HEANN reduces the mutation probability over time by using generalization loss (fitness) of individuals among the population, thus creating a gradual shift from global search to local search. In order to alleviate noisy fitness evaluation, they encoded parameters about the architecture and weights in each individual

Table 2 Published papers for evolutionary design of neural networks (Since 2001)

Authors	Year	Target*	Method**	Encoding	Reference
Stanley & Miikkulainen *NEAT*	2001	AW	GA	Direct	(2001)
Wang et al	2002 2003	A	GA	Direct	(2002; 2003)
Barrios et al *ADANNET*	2002	AW	GA	Indirect	(2002)
Barrios et al *GANN*	2003	AW	GA	Indirect	(2003)
García-Pedrajas et al *COVNET*	2003 2006	AW	Cooperative Coevolution	Direct	(2003)
Leung et al	2003	AW	GA	Direct	(2003)
Reisinger et al *Modular NEAT*	2004	AW	GA	Direct	(2004)
Jung & Reggia	2004 2006 2008	A	GP	Tree-based	(2004; 2006; 2008)
Chen et al *MPIPE*	2004	A	GP	Tree-based	(2004)
Tan *GAEPNet*	2004	AW	GA + EP	Direct	(2004)
Kassahun & Sommer *EANT*	2005	AW	CMA-ES	Direct	(2005)
Palmes et al *MGNN*	2005	AW	EP	Direct	(2005)
Lee et al.	2005	A	GA	Indirect	(2005)
García-Pedrajas et al *COVNET*	2006	AW	EP	Direct	(2006)
Castellani *ANNE*	2006	AW	GA	Direct	(2006)
Benardos & Vosniakos	2007	AW	GA	Indirect	(2007)
Fiszelew et al	2007	A	GA	Direct	(2007)
Siebel & Sommer *EANT2*	2007	AW	CMA-ES	Indirect	(2007)
Gauci & Stanley *HyperNEAT*	2007	AW	GA	Indirect	(2007)
Tsoulos et al	2008	AW	GE	Indirect	(2008)
Kim&Cho	2008	A	GA	Direct	(2008)
Rivero et al	2009	A	GP	Direct	(2009)
De Campos et al	2011	A	GA	Indirect	(2011)
OOng & Isa *HEANN*	2011	AW	Multi-Objective (Scalar)	Direct	(2011)
Loghmanian et al	2012	A	Multi-Objective (NSGA-II)	Direct	(2012)
Soltanian et al	2013	A	GE	Indirect	(2013)
Ahmadizar et al *GEGA*	2015	AW	GE-GA	Indirect	(2015)
De Campos et al *ADEANNs*	2015	AW	GA	Indirect	(2015)
Assunção et al *DSGE*	2017	AW	GE	Indirect	(2017)
Chen (Yuh Wen) & Shiu GABPNN	2019	A	GA	Direct	(2019)

Table 2 (continued)

*A: Architecture Optimization, AW: Architecture and Weights Optimization, **GA: Genetic Algorithms, EP: Evolutionary Programming, GP: Genetic Programming, ES: Evolutionary Strategies

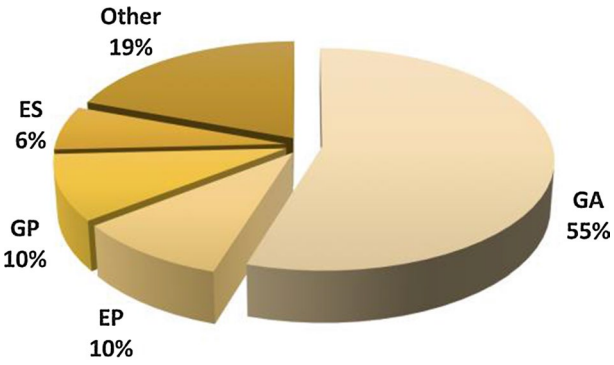


Fig. 39 Distribution Graph for Evolutionary Computation Techniques Adopted (after NEAT)

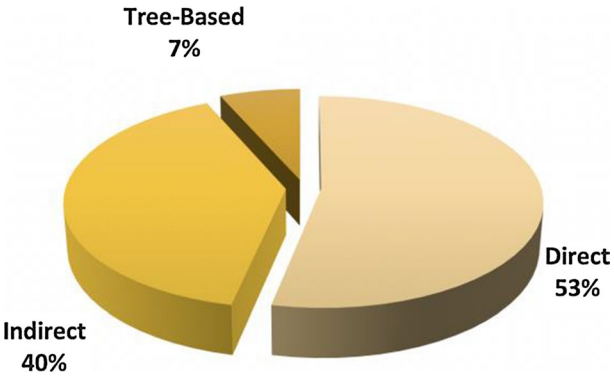


Fig. 40 Distribution Graph for Representation Methods (after NEAT)

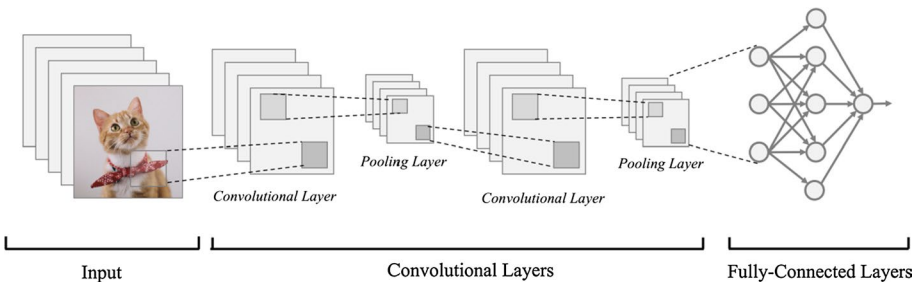


Fig.41 A typical structure of a CNN for an image recognition task

with a direct-encoding approach. Loghmanian et al. (2012) aimed to achieve better accuracy while building a minimum structure with the utilization of multi-objective evolutionary algorithms. They applied NSGA-II, the improved version of the elitist non-dominated sorting genetic algorithm proposed by Deb et al. (2000; 2002), and defined the objective functions as minimizing architecture complexity and mean square error of the test set.

Mason et al. (2017) adopted another enhancement to genetic algorithms and proposed Neuro Differential Evolution (NDE) to optimize both the architecture and weights of a neural network. In their model, architecture is evolved through GA, and weights are evolved by using differential evolution. Similar to NEAT, NDE starts building a network with one neuron and gradually grows it until a feasible solution is found. A list of published papers after NEAT was introduced can be found in Table 2. Distribution Graph for Evolutionary Computation Techniques Adopted in this period is given in Fig. 39 and Distribution Graph for Representation Methods is given in Fig. 40.

5.3 Deep learning era

Inspired by the works on the human visual cortex (Hubel and Wiesel 1962), theoretical concepts of deep neural networks start with Fukushima's *Neocognitron* which takes advantage of the local receptive fields as an input layer and utilize feature detectors (Fukushima 1980). The experiments of Hubel and Wiesel showed that some individual neuronal cells in the brain responded when exposed to vertical edges of a certain orientation (Bhandare and Kaur 2018; Hubel and Wiesel 1968). This idea was developed and improved by LeCun, which will further be named as Convolutional Neural Networks (CNNs, or ConvNets) forming the key architecture of modern deep neural networks (DNNs) (LeCun et al. 1999). The hierarchical structure of multiple layers in CNN provided multiple levels of representation, in which each layer learns a new feature from its preceding layer (Tirumala et al. 2016) (Fig. 41).

Throughout the 1990s and 2000s, further research has been carried out on CNNs by many researchers, and models were successfully applied to real-world tasks such as object recognition, detection, and segmentation. Although innovative and superior in performance, LeCun's model was trained with backpropagation, thus computationally very expensive. Furthermore, due to the excessive number of parameters, the problem of vanishing gradients, where derivatives of gradients in multiple layers get closer to zero, or exploding gradients where derivatives skyrocket to non-computable values have prevented effective use of the method in large-scale problems. The first breakthrough in the training of deep architectures was achieved in 2006 by Hinton with his prominent work proposing an efficient training procedure (Hinton et al. 2006; Hinton and Salakhutdinov 2006). This development attracted the focus of research back into Deep Neural Networks and led to the creation of a very large-scale image dataset called ImageNet. Pioneered by Prof.Fei Fei Li, the dataset contained around 14 million images in more than 20,000 categories, a hierarchically built-in WordNet system. Between 2010 and 2017 the ImageNet project run an annual contest called *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) where researchers compete to correctly classify images and recognize objects using Deep Learning architectures that they developed. The average classification error rate, in the beginning, was around 25%. Then, another breakthrough came in 2012, with contest winner Krizhevsky et al.'s revolutionary architecture called AlexNet (Krizhevsky et al. 2012) which halved the error rate on classification, achieving around a 12-year leap in deep learning research. Thanks to the advancements in the technology of graphic processing units (GPU) and the availability of big data, such deep architectures were able to be trained in a more reasonable time. After AlexNet, many other deep architectures have followed, achieving state-of-the-art classification accuracy, thus surpassing human performance (He et al. 2016; Hu et al. 2018; Huang et al. 2017; Simonyan and Zisserman 2014; Szegedy et al. 2015; Zeiler and Fergus 2014).

5.3.1 Optimization of deep neural networks

Despite their superior performance, designing modern CNNs is a very difficult and tedious task, which requires expert domain knowledge. Therefore, automating this manual process attracted the attention of researchers by shifting the focus from optimizing classical ANNs to designing complex CNN architectures. Unlike previous approaches, this new task requires a vast number of parameters to be optimized such as the number of convolutional layers, number of kernels, activation functions of each layer, regularization options, and general hyperparameters like batch size and learning rate. Studies show that a complex relationship exists between hyperparameters and the performance of different networks. For example, a fine-tuned network's hyperparameters do not have the same effect on another network. Similarly, a good architecture of an image classifier does not necessarily be successful on a different dataset. Due to the necessity of vast computational resources for training, trial and error of designing architectures and selecting hyperparameters for deep CNNs turn out to be inefficient. Thus, domain experts generally use common architectures and default hyperparameters to create their models (Breuel 2015; Young et al. 2015).

With the surge of interest in this field, various optimization methods have been implemented. Based on the papers published so far, these methods divide into two major categories. The first approach applies evolutionary computation such as Genetic Algorithms, Genetic Programming, Evolutionary Strategies, or Hybrid algorithms. The second refers to algorithms based on *Reinforcement Learning* (RL) utilizing the reward-penalty principle (Irwin-Harris et al. 2019; Sun et al. 2019b). The latter approaches such as Neural Architecture Search (NAS) (Zoph and Le 2016), Efficient Architecture Search (EAS) (Cai et al. 2017), Meta-modelling (Meta-QNN) (Baker et al. 2016), and Block Design Method (Block-QNN-S) (Zhong et al. 2017) have demonstrated competitive performance by automatically creating CNN architectures.

5.3.2 Hyperparameter optimization

Designing modern deep architectures are often considered a model selection or hyperparameter optimization problem (Suganuma et al. 2017). Several hyperparameter tuning methods became commonplace such as Grid Search, Random Search (Bergstra and Bengio 2012), Gradient Search (Bengio 2000), or Bayesian Optimization (Snoek et al. 2012). Recently, evolutionary computation is replacing the above methods by providing global search capability and successfully avoiding local minima. Since we have concentrated our focus on evolutionary approaches, a survey of the state-of-the art is presented in this paper.

5.3.3 Preliminary studies

One of the pioneer's works combining evolutionary algorithms and deep neural networks is Verbancsics and Harguess' (2013; 2015) Generative and Developmental System (GDS) approach which utilizes HyperNEAT in deep architectures, training feature extractors for backpropagation learning. They showed that, although HyperNEAT alone has difficulties when classifying images, it is effective at training a feature extractor. The application of HyperNEAT to deep architectures was also investigated by Fernando et al. (2016) who proposed a differentiable version of Compositional Pattern Producing Network, which they called DPPN. The main difference of their model was that the architecture of the network

is evolved through microbial GA but weights are learned. They utilized a Lamarckian algorithm (inheritance of acquired characteristics) which combined evolution and learning, producing DPPNs to reconstruct an image. In another pioneer work, Young et al. (2015) proposed *Multi-node Evolutionary Neural Networks for Deep Learning* (MENNDL) for automating model selection in deep architectures through hyperparameter optimization using GAs. Although it cannot be considered as an architecture builder, their model evolved some of the properties of the architecture, such as kernel size and number of filters for each convolutional layer. Similarly, Loshchilov and Hutter (2016) used CMA-ES to optimize hyperparameters of deep neural networks. Their model supported the evaluation of models in multiple GPUs running in parallel and they compared the proposed approach with state-of-the-art Bayesian optimization algorithms to tune hyperparameters of a CNN on MNIST dataset.

An intensified research on automating the architecture design of CNNs brought better models to the surface. In a prominent work, Xie and Yuille (2017) used GAs to design CNN architectures and called their approach as *Genetic CNN*. They proposed an encoding method to represent network structure in a fixed-length binary string and described the genetic framework to obtain an optimal solution. The encoding scheme allowed to represent various state-of-the-art architectures including chain-shaped networks like AlexNet and VGGNet, multiple-path networks like GoogLeNet, and highway networks like ResNet. They adopted a strategy to run the candidate individual solutions on CIFAR10, which is a small-scale dataset. Then, they conducted large-scale experiments by transferring the architectures they obtained via using GA. In the same year, Mikkulainen et al. (2017) proposed DeepNEAT and CoDeepNEAT (Coevolution DeepNEAT), which follow the same fundamental processes in NEAT, for optimizing deep learning architectures through evolution. It was inspired by Hierarchical SANE (Moriarty and Miikkulainen 1996) and influenced by coevolutionary approaches of ESP (Gomez and Miikkulainen 1999) and CoSyNE (Gomez et al. 2006). Similar to NEAT, their proposed models start with minimal complexity of CNNs and gradually grow through mutation. The main difference of both models from NEAT is that each node in the chromosome represents a CNN layer, instead of a neuron in classical ANNs. The edges in the chromosome indicate how the layers are connected to each other. The authors also included global hyperparameters of the network in the chromosome. In order to evolve repetitive modular structures like GoogLeNet and ResNet, they designed CoDeepNEAT with two populations of modules and blueprints, thus being able to explore diverse and deeper architectures. Another adaptation of NEAT to deep architectures was proposed by Desell (2017b), namely Evolutionary eXploration of Augmenting Convolutional Topologies (EXACT). His method featured large-scale distributed computing and the evolution of convolutional filters. EXACT was run over 4,500 distributed volunteered computers on the *Citizen Science Grid* trained over 120,000 CNNs, achieving 98.32% test accuracy on MNIST dataset. The author was surprised to observe interesting structures obtained by evolution, which bear some similarities to biological neurons. Although not being able to evolve pooling layers and supporting only 2D inputs and filters, his method provided promising results for the automatic design of CNN architectures. Later he improved his algorithm by introducing a new mutation operator and developing an extension with a simplex hyperparameter optimization (SHO), allowing co-evolution of hyperparameters (Desell 2017a). The improved version of EXACT achieved a 99.43% prediction rate on the MNIST dataset with significantly less CNNs evolved.

There have also been several approaches using other forms of evolutionary computation, instead of standard GAs. Suganuma et al. (2017) developed a sophisticated method which utilizes Cartesian Genetic Programming (CGP) to design CNN architectures. This

model represents the CNN structure and connectivity with a direct encoding, allowing variable-length networks and skip connections. Similar to previous approaches, it evaluates some candidate solutions in parallel at each generation to reduce computational resources. Still, being very intensive in terms of computation time, their method provided competitive results compared to the state-of-the-art. Kim et al. (2017) proposed Neuro-Evolution with Multiobjective Optimization (NEMO), an automatic machine learning approach (AutoML) to optimize CNNs for best accuracy and inference speed at the same time by applying multi-objective evolutionary algorithms (MOEAs). The authors implemented the NSGA-II framework, a non-dominated sorting method to rank solutions, and used 60 GPUs to evaluate performance on MNIST and CIFAR-10 image classification datasets. Mitschke et al. (2018) proposed a metaheuristic approach to the automatic generation of CNNs based on gradient evolution. They aimed to maximize accuracy while reducing the resources needed.

In a comprehensive work, Baldominos et al. (2017) aimed to evolve all aspects of network design, including architecture, activation functions, and learning hyperparameters through an innovative encoding scheme. They provided a thorough review of the evolution of CNNs and successfully organized the parameters to be evolved. Their proposal of the organization included three categories, which are *convolutional architecture*, *dense architecture,* and *general hyperparameters*. They designed their framework to be flexible by allowing it to fit any evolutionary computation technique, but they evaluated GA and GE in particular. Due to the necessity of vast computational resources to train each individual, they had to limit possible values for each parameter, as they claimed that small differences do not have a significant impact on network performance. Instead of fully training each individual solution on each generation, they opted to make estimations by training only a reduced sample of data with a smaller number of training epochs. Despite its drawbacks, they were able to reduce the computation time significantly. Al-Hyari and Areibi (2017) proposed a simple yet efficient approach to automate the design of CNNs using GAs. They developed a design exploration framework and achieved promising accuracy on the MNIST dataset.

Many researchers considered the architecture design of deep neural networks as hyperparameter optimization and set a goal to automate the whole process by creating complete frameworks. Bochinsky et al. (2017) proposed an efficient hyperparameter optimization strategy and used evolutionary algorithms by evolving only structure-related parameters, such as layer and kernel sizes. They used committees of multiple CNNs to improve the classification accuracy, where the committee is a set of trained CNNs, and the classification is carried out by fusing CNN scores. They made a comparison on the optimization of independent CNNs and joint optimization for a committee of multiple CNNs. Based on their experiments, they were able to achieve state-of-the-art results on the MNIST dataset. In another competitive study, Kramer (2018) utilized Rechenberg's mutation rate control and a niching mechanism to optimize multiple stacked convolutional layers, called *convolutional highways* (Srivastava et al. 2015), for feature preprocessing. *Rechenberg's rule* is used to adapt mutation rates based on the fitness of the evolutionary algorithm. On the other hand, niching is used to evade trapping into local optima by placing and evaluating offspring in their own niches. In his proposed study, named as $(1+1)$-EA, Kramer obtained significantly different structures from conventional, hand-crafted CNNs and achieved promising results. Later, Prellberg and Kramer (2018) proposed the inheritance of weights over generations through Lamarckian evolution and applied $(1+1)$-EA to CIFAR-10 and CIFAR-100 datasets. They showed that weight inheritance increases data efficiency by 75%. Loussaief and Abdelkerim (2018) proposed Enhanced Elite CNN Model Propagation (Enhanced E-CNN-MP) for designing the optimal structure of CNN through

hyperparameter optimization. They achieved around 90% test accuracy on the stop sign image classification task by using a CNN with a complex architecture obtained by GAs. A similar study was conducted by Bhandare and Kaur (2018), which used GAs to optimize hyperparameters of a CNN benchmarked with handwriting recognition MNIST dataset.

5.3.4 Towards state-of-the-art

The ILSVRC challenge has undoubtedly motivated the deep learning community to develop better architectures. Researchers competed with each other to place their models on state-of-the-art podium and those models helped develop revolutionary technologies such as self-driving cars. As of 2017, automated evolutionary designs started to outperform hand-crafted architectures for image classification tasks. Dufourq and Bassett (2017a) developed an exceptional algorithm called Automated Problem Identification (API) aimed to be the foundation of fully automated machine learning. API utilizes evolutionary deep learning to recognize if a dataset represents a classification or regression problem, achieving an average of 96.3% accuracy. Furthermore, it recommends an architecture and other strategies like loss function to be used. Later, the same authors proposed Evolutionary DEep Networks (EDEN) which achieved state-of-the-art results in three cases among seven image and sentiment classification datasets (Dufourq and Bassett 2017b). EDEN was also the first attempt to apply neuroevolution to building one-dimension CNN for sentiment analysis. The researchers had an eventual goal of evolving deep architectures for a broad range of problems and accomplishing the task on a single GPU, instead of large clusters. To keep their algorithm simple, they interfaced EDEN to Tensorflow (Abadi et al. 2016) (Fig. 42).

Newborn architectures of modern deep learning are surprisingly complex, and it is not an easy task to foresee an effective combination of structures without trial and error. An interesting model was obtained by Assunção et al. (2018) using the combination of Genetic Algorithms and Grammatical Evolution. They proposed Deep Evolutionary Network StructurEd Representation (DENSER) which is an extension of their previous work (DSGE) for searching architectures of conventional ANNs. They encoded a sequence of layers of a CNN in a GA chromosome and used DSGE to evolve the parameters of each layer. By outperforming several previously introduced neuroevolution methods like CoDeepNEAT, the authors were amazed to observe that the best

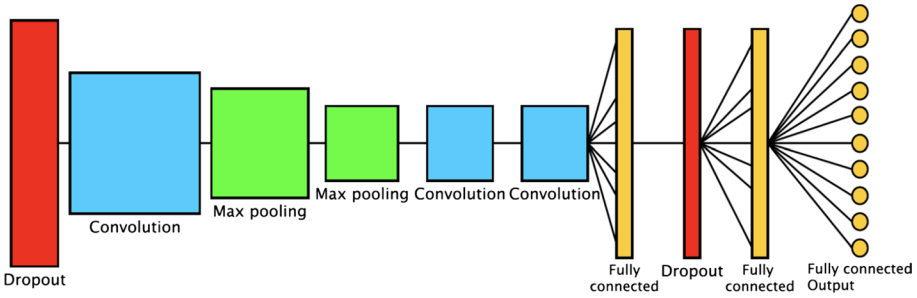


Fig. 42 An example of a CNN architecture generated by API. The number of units and activation functions of the last layer are determined by the algorithm. API also recommended using categorical cross-entropy for loss function (redrawn from Dufourq and Bassett (2017a))

network found by evolution had many dense layers in the end, which would never be thought of by a human designer. They later improved this method and proposed Fast-DENSER++(Assunção et al. 2019a, 2019b), which enabled the training time of candidate solutions to grow gradually as necessary. In this model, initial generations train candidate solutions with fewer epochs, and as generation proceed more training time is allowed to increase accuracy. Such et al. (2018) from Uber AI Labs explored alternative approaches to gradient-based algorithms for training deep neural networks and proposed a GA-based evolutionary method to solve deep RL problems including Atari and humanoid locomotion. The authors were surprised to see that a simple GA-based method proving to be competitive when compared to contemporary gradient-based algorithms like Q-learning, A3C, and ES.

Evolutionary computation methods for designing deep neural network architectures were not limited to standard Genetic Algorithms. Researchers also explored other evolutionary approaches with various encoding schemes and hyperparameter optimization techniques. Wang et al. (2018) implemented a hybrid differential evolution (DE) by introducing a new crossover operator to evolve CNN architectures. In their proposed approach called DECNN, they used an innovative encoding strategy inspired by computer networks, called IP-Based Encoding Strategy (IPES), with an improvement to remove the constraint on the maximum depth of the network, enabling variable-length CNN architectures. In another alternative approach, Zhu (Yiheng) et al. (2018) proposed GP-CNAS, a Genetic Programming framework for Convolutional Neural Architecture Search. Their model is designed to encode CNNs as GP trees where leaf nodes represent residual blocks and non-leaf nodes specify the block assembling procedure.

5.3.5 Latest approaches

By the end of the 2010s, Neuroevolution became one of the most popular topics among the Deep Learning community. Researchers from tech giants and AI Research Groups such as Google Brain Team, OpenAI, Uber Labs, Sentinent Labs, and DeepMind published promising and competitive works attempting to obtain the best deep neural network architectures achieving state-of-the-art or near-state-of-the art results. Real et al. (2017) from Google Brain Team proposed Large-Scale Evolution of Image Classifiers (LEIC), which achieved near-state-of-the-art results on CIFAR-10 and CIFAR-100 image classification datasets. Although computationally intensive, it was one of the first large-scale attempts to apply evolutionary algorithms to optimize the structure of million-parameter-CNNs, using 250 computers. The authors aimed to develop fully trained models and minimize human intervention when designing deep neural networks for generic real-world problems. They invented intuitive mutation operators which were able to navigate large search spaces and slightly modified known EAs while keeping the process as simple as possible. Liu et al. (2018a) from DeepMind developed a hierarchical genetic representation scheme similar to hand-crafted modular design patterns and utilized a simple evolutionary algorithm to discover new architectures. The authors expressed the key idea of this representation as possessing several motifs at different levels of hierarchy, where low-level motifs are used as building blocks during the construction of high-level motifs. They established evolutionary search mechanism by treating the representations as genotypes and the models found by evolutionary algorithm achieved state-of-the art results on CIFAR-10 and near-state-of-the-art results on ImageNet dataset.

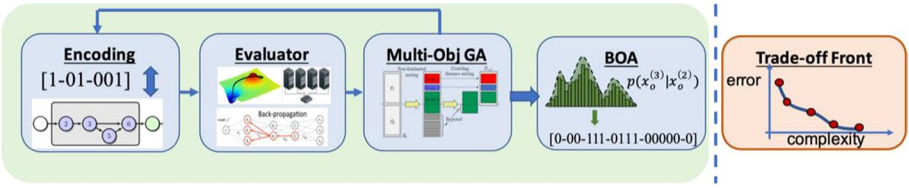


Fig. 43 Proposed stages of NSGA-Net. It represents networks as bit strings and trains with gradient descent. Then, ranking and selection are carried out by NSGA-II. Successful architectures are explored through the exploitation step by utilizing BOA. Finally, a set of networks on the trade-off front are obtained meeting dual objectives of error rate and network complexity. (Redrawn from Lu et al. (2019a)

Due to the necessity of vast computational resources, recent studies aimed to optimize both network performance and computation resources simultaneously. Thus, multi-objective evolutionary approaches have been adopted more often for Neuroevolution. Elsken et al. from DeepAI (2018a; 2019) argued the necessity of vast computational resources for searching CNN architectures and initially aimed to address this issue by proposing Neural Architecture Search by Hill Climbing (NASH) in 2018. They further proposed Lamarckian Evolutionary Algorithm for Multi-Objective Neural Architecture Design (LEMONADE) to optimize multiple objectives including prediction accuracy, inference time, or the number of parameters used. Unlike other multi-objective evolutionary approaches, LEMONADE adopted an interesting approach classifying the objectives as cheap and expensive. For example, they described evaluating the architecture's number of parameters as cheap, while evaluating the prediction performance on validation data as expensive. By prioritizing the cheap objectives on a selected subset of architectures, the authors were able to form the Pareto front with the less computational resource by training the network after probabilities are assigned to architectures found by using the previous step. In a prominent study, Lu et al. (2019a; 2019b) proposed NSGA-Net, a competitive multi-objective evolutionary approach for automatically designing CNN architectures. The authors, including Kalyanmoy Deb, the key researcher and leading scientist of NSGA, aimed to combine multiple objectives of error minimization and reducing computational complexity by measuring FLOPs (Floating-point operations). They utilized NSGA-II (Deb et al. 2000), a non-dominated sorting genetic algorithm which has been effectively applied to many real-world tasks. The proposed study differs from recent evolutionary approaches by applying a crossover operator and employing the Bayesian Optimization Algorithm (BOA) to collect promising solutions in the search history archive (Fig. 43). Based on the experiments, NSGA-Net achieved an error rate on par with other state-of-the-art NAS methods on the CIFAR-10 dataset, while using orders of magnitude less computational resources. The authors further improved this model with NSGANetV2 which adopts a bi-level surrogate model on upper level with architectures and lower level for weights(Lu et al. 2020). Yang et al. (2020) developed CARS, a continuous evolution strategy which initializes a supernet with sufficient cells to accommodate the best architectures found by a non-dominated sorting algorithm (NSGA-III). These cells are continuously updated through the evolution process. Furthermore, they improvised a protection mechanism to avoid the small model trap problem, since small models tend to eliminate large models during the optimization process. They achieved state-of-the-art results on CIFAR-10 and ImageNet.

Contemporary hand-crafted architectures inspired researchers to steer automated evolutionary methods to design models which support similar capabilities. Chen (Zefeng) et al. (2019b) proposed EANN which utilizes the basic building blocks of ResNet for

establishing the basic skeleton and initialization of the evolutionary process. Unlike NEAT, this structure doesn't start with minimal architectures, since block which is not needed can be skipped using the shortcut links. The authors aimed to evolve only the architecture, while weights are trained by using the conventional backpropagation method. In another study, Irwin-Harris et al. (2019) proposed an encoding strategy based on a directed acyclic graph (DAG) representation aiming to apply fewer constraints on the search space and developed an evolutionary method for random generation of CNN architectures. The authors claim to enable arbitrary connection structure and unbounded depth. They adopted the idea of partially training the candidate solutions first and then fully training the best three models obtained by the evolution process. Another inspirational work came from Liu (Peng) et al. (2019; 2018c) who aimed to accelerate the evolution of CNN architectures by using an experience-based greedy exploration strategy and transfer learning. For this goal, they developed an evolutionary framework called *EvoNet* to construct a deep neural network-based medical image denoiser. Similar to previous studies, their model allowed the modular structure of modern hand-crafted architectures, such as ResNet. Achieving state-of-the-art results on image classification tasks, Sun et al. (2019a) evolved CNN architectures by using GA. The proposed algorithm called AE-CNN is based on ResNet and DenseNet blocks, which are key elements of ingenious hand-crafted architectures, surpassing human performance on the ILSVRC challenge. The authors designed an encoding strategy with a variable-length chromosome which can adaptively determine the optimal depth of various CNNs. They also developed a new crossover and a new mutation operator to accomplish the image classification task. The result of the experiments revealed that their proposed method achieved state-of-the-art CIFAR-10 and CIFAR-100 datasets outperforming prominent manual and automatically obtained architectures. The same authors further proposed (2019c) proposed a completely automatic evolutionary approach by using Genetic Algorithms, which they call CNN-GA for designing CNN architectures to handle image classification tasks. They designed an innovative encoding scheme to enable arbitrary depths while incorporating skip connections to allow deeper models. By addressing the incompatibility issue of crossover for variable-length chromosomes, the authors designed a new crossover operator to adapt the individuals for the evolutionary process. They also developed an asynchronous computational component to manage computational resources and a cache component for the acceleration of evaluation for fitness. Later with another paper, they proposed a similar method called EvoCNN which demonstrates significant performance on image classification tasks (Sun et al. 2019b).

By achieving superior performance with reduced error rates, more recent studies aimed to reduce computational costs. Saltori et al. (2019) developed a Regularized Evolutionary Algorithm, which they called EvoA/B, and introduced custom genetic operators to regularize the evolutionary process with the aim of reducing memory footprint and computational resources for a dynamic image classifier. As a modification to Real et al.'s (2019) prominent work, their model brought evolving cell topology with the variable number of hidden nodes, custom crossover, and mutation operators as well as a stagnation avoidance mechanism to offset early convergence. In another successful work, Zhu (Hui) et al. (2019) aimed to reduce computational cost on architecture search and proposed Efficient Evolution of Neural Architecture (EENA) which is inspired by Net2Net by Chen et al. (2015) from Google, accelerating the experimentation process by transferring knowledge from a smaller network to larger models. Based on their experiments, EENA used only 0.65 GPU days to design a network that achieves 2.56% test error on the CIFAR-10 dataset. They were able to transfer the optimum architecture to the CIFAR-100 dataset successfully. Unlike hardware-rich AI labs of tech giants, Lan et al. (2019) implemented NEAT to

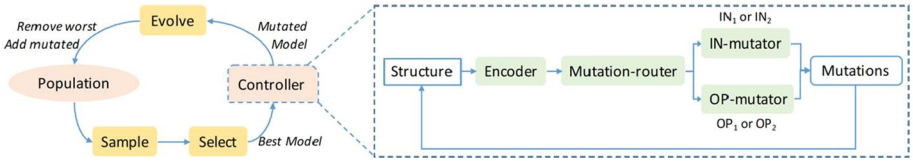


Fig.44 General flowchart for RENAS and the reinforced mutation controller (redrawn from Chen et al. (2019a))

evolve efficient deep neural networks which can be run on Low-Performance Computing Hardware (LPCH) like Raspberry 3 and aimed to achieve at least 95% accuracy on real-time object recognition tasks. NEAT turned out to be useful by reducing the number of parameters from millions to thousands and with the help of an innovative fitness function, the authors were able to achieve their goals. Although most studies preferred to focus on the image classification task, there have also been studies to optimize other deep neural networks for various real-world tasks. Akut and Kulkarni (2019) explored the utilization of CNNs on time series prediction and proposed a GA-based architecture design method in order to challenge state-of-the-art RNN models for this task. Similar to other neuroevolutionary approaches, the authors aimed to optimize hyperparameters and key elements of the CNN structure, such as the number of convolutional layers, the number of fully connected layers, etc. Although not being able to outperform RNN models, they achieved near-optimal results with less computation time. Working on a similar task, Wu et al. (2019) proposed a hybrid ResNet with a GA-based architecture design to optimize and obtain noise-free Time Series Classification (TSC). Their model called GA-ResNet adopted GA to optimize ResNet structure by removing connections between neurons.

Competition between gradient-based and evolution-based approaches paved the way to hybrid approaches with significant success. By combining the advantages to RL and EA, Chen (Yukang) et al. (2019a) proposed Reinforced Evolutionary Neural Architecture Search (RENAS) which utilized a reinforced mutation for learning the effects of small modifications (Fig. 44). The authors applied RENASNet to CIFAR-10 and transferred the obtained architecture to ImageNet which achieved state-of-the-art with 75.7% top-1 accuracy. The innovative solution was further applied to other benchmarks such as semantic segmentation with DeepLabv3 on the PASCAL VOC, and their model outperformed prominent architectures like MobileNet and NASNet. In another hybrid approach, Kobayashi and Nagao (2020) aimed to combine advantages of gradient-based and evolutionary architecture search and achieved competitive performance with state-of-the-art models. Habi and Rafalovich (2019) developed GeneticNAS, a GA-based neural architecture search technique which utilized the search space representing convolutional cells as directed acyclic graphs (DAG). The DAG structure is described by using a fixed-length list of integers. The authors also employed weight sharing as successfully implemented in RL-based NAS methods like ENAS (Pham et al. 2018) and DARTS (Liu et al. 2018b), to reduce computation cost.

Liang et al. (2019) developed Learning Evolutionary AI Framework (LEAF), an AutoML framework for automating architecture design and optimizing hyperparameters. It was designed as an extension to Miikkulainen et al.'s CoDeepNEAT, evolving both hyperparameters and network architecture. LEAF had mainly three components, which are 1) the algorithm layer (using CoDeepNEAT), 2) the system layer and 3) the problem-domain layer (Fig. 45). The advantage of the system layer is the facilitation of training on cloud

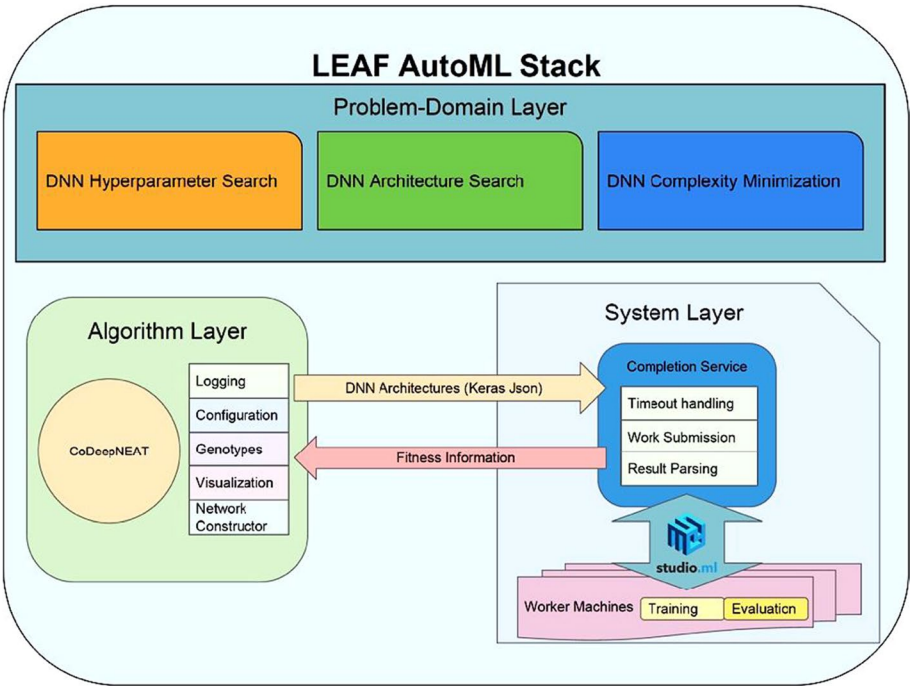


Fig. 45 The three components of LEAF (redrawn from Liang et al.(2019))

services such as Amazon AWS, Microsoft Azure, or Google Cloud. This helps the evaluation of candidate networks in an efficient way. The problem-domain layer carries out three tasks including DNN Complexity Minimization which can extend CoDeepNEAT to multiple objectives. Thus, CoDeepNEAT can maximize performance while minimizing the complexity of networks evolved.

Perhaps the most successful work on our review paper came in 2019 by Real et al. (2019) from Google Brain Team. In their world-famous study titled *"Regularized Evolution for Image Classifier Architecture Search"*, they developed *AmoebaNet-A*, which surpassed hand-crafted network architectures for the first time and set a new state-of-the-art for ImageNet accuracy. The authors defined the key factor of this success as *regularized evolution*, the introduction of age property to favor younger genotypes in tournament selection evolutionary algorithm. By outperforming the best RL-based NAS methods, they proved the effectiveness and speed of evolutionary approaches for discovering optimal CNN architectures automatically. Summary of Evolutionary Approaches for Designing Deep Neural Networks is listed in Table 3.

6 Conclusion and summary

Even after three decades of significant progress, designing an optimal ANN architecture still remains an open problem. With the ultimate goal of achieving fully automated machine learning, this challenging task gained a lot of attention and a wide variety of

Table 3 Summary of evolutionary approaches for designing deep neural networks

Authors	Year	Summary	Reference
Verbancsics & Harguess *GDS*	2013 2015	Used HyperNEAT to train feature extractors	(2013; 2015)
Young et al *MENNDL*	2015	Automation of model selection through optimizing hyperparameters such as kernel size and the number of filters, using GAs	(2015)
Fernando et al *DPPN*	2016	Evolution of the architecture through Lamarckian evolution. Used minimal GA with a small population size	(2016)
Loshchilov & Hutter	2016	Used CMA-ES to tune hyperparameters of CNNs for their model running on 30 GPUs in parallel	(2016)
Xie & Yuille *Genetic CNN*	2017	Used GAs to evolve CNN architectures. Evaluated candidate solutions on small-scale CIFAR10 dataset and transferred the architecture for large-scale experiments	(2017)
Miikkulainen et al *CoDeepNEAT*	2017	Follows similar process with NEAT	(2019)
Desell *EXACT*	2017	Adaptation of NEAT to CNNs, featuring large-scale distributed computing	(2017b)
Desell *EXACT2*	2017	Improved version of EXACT with a new mutation operator and simplex hyperparameter optimization (SHO)	(2017a)
Suganuma et al *CGP-CNN*	2017	Used Cartesian Genetic Programming to design CNN architectures	(2017)
Baldominos	2017	Adopted GA and GE. Made estimations by training only a reduced sample of training data with a small number of training epochs	(2017)
Dufourq & Bassett *API*	2017	Recognize datasets if they are classification or regression problem	(2017a)
Dufourq & Bassett *EDEN*	2017	Interface to Tensorflow to create the architecture of CNN. The method also recommends loss function. Used traditional GA to evolve models	(2017b)
Bochinsky et al *CEA-CNN*	2017	Used joint optimization of a committee of multiple CNNs	(2017)
Real et al *LEIC*	2017	Apply EA on large scale and achieved near-state-of-the-art results for CIFAR-10 and CIFAR-100. They used 250 computers	(2017)
Kim et al *NEMO*	2017	Aimed to achieve the best accuracy and inference speed at the same time. Used 60 GPUs to train the model	(2017)

Table 3 (continued)

Authors	Year	Summary	Reference
Kramer *(1 + 1)-EA*	2018	Used Rechenberg's rule to adapt mutation rate and niching mechanism to avoid local optima	(2018)
Prellberg & Kramer (1 + 1)-EA	2018	Applied inheritance of weights over generations through Lamarckian evolution	(2018)
Loussaief & Abdelkerim *Enhanced-E-CNN-MP*	2018	Evolved CNN architecture through hyperparameter optimization using GAs	(2018)
Bhandare & Kaur	2018	Evolved CNN architecture through hyperparameter optimization using GAs	(2018)
Mitschke et al	2018	Used gradient evolution to design CNN architectures	(2018)
Wang et al *DECNN*	2018	Used hybrid differential evolution (DE) and improved IP-Based encoding strategy to design CNN architectures	(2018)
Such et al	2018	Proposed a GA-based evolutionary method to solve deep RL problems including Atari and Humanoid Locomotion	(2018)
Liu (Hanxiao) et al	2018	Used hierarchical representation scheme and established a simple evolutionary algorithm to evolve CNN architectures	(2018a)
Assunção et al *DENSER*	2018	The encoded sequence of layers of a CNN in a GA chromosome and used DSGE to evolve the parameters of each layer. Evolution recommended many dense layers in the end	(2018)
Zhu et al *GP-CNAS*	2019	Used Genetic Programming to represent and evolve CNN architectures	(2018)
Lu et al *NSGA-Net*	2019	Utilized NSGA-II to create a Pareto-front for error minimization and reducing computational complexity	(2019a; 2019b)
Chen (Zefeng) et al *EANN*	2019	Utilized the basic building block of ResNet to establish the basic skeleton and initialization of the evolutionary process	(2019b)
Assunção et al *Fast-DENSER + +*	2019	Improved version of DENSER, which enables adaptation of training time (number of epochs) throughout generations	(2019b)
Irwin-Harris et al	2019	Evolutionary Random Search by using a Graph-based encoding. Proposed an encoding strategy based on a directed acyclic graph representation	(2019)
Saltori et al *EvoA/EvoB*	2019	Modification to Regularized Evolutionary Algorithm by Real et al. with custom crossover and mutation operators	(2019)

Table 3 (continued)

Authors	Year	Summary	Reference
Liu (Peng) et al *EvoNet*	2019	Developed an evolutionary framework with expedited Genetic Algorithms using an experience-based greedy exploration strategy	(2019)
Sun et al *CNN-GA*	2019	Developed a completely automated evolutionary process for designing CNN architectures	(2019c)
Sun et al *AE-CNN*	2019	Developed an evolutionary approach using GAs, based on ResNet and DenseNet blocks. Achieved state-of-the-art results on CIFAR-10 and CIFAR-100	(2019a)
Sun et al *EvoCNN*	2019	A similar approach with CNN-GA	(2019b)
Zhu (Hui) et al *EENA*	2019	Inspired by Net2Net by Chen et al. accelerating the experimentation process by transferring knowledge from a small network to large models	(2019)
Akut & Kulkarni	2019	Used a GA-based neuroevolutionary approach for time series prediction. Achieved competitive results compared with RNNs	(2019)
Chen (Yukang) et al *RENAS*	2019	Combined the advantages of RL and EA by introducing a reinforced mutation controller for learning the effects of slight modifications	(2019a)
Elsken et al *LEMONADE*	2019	A multi-objective approach classifying the objectives as cheap and expensive. Starts with cheap objectives to reduce computation time	(2019)
Habi & Rafalovich *GeneticNAS*	2019	Inspired by RL-Based ENAS (Pham et al.2018), using the same search space representing convolutional cells as directed acyclic graphs. Utilized weight sharing to reduce costs	(2019)
Lan et al	2019	Implemented NEAT to reduce the number of parameters and developed a fitness function to lower computation costs	(2019)
Liang et al *LEAF*	2019	Complete AutoML framework based on CoDeepNEAT	(2019)
Real et al *AmoebaNet-A*	2019	Sets the new state-of-the-art on ImageNet with regularized evolution, which introduced an age property to favor younger genotypes	(2019)
Wu et al *GA-ResNet*	2019	Combined GA and ResNet to optimize ResNet structure for Time Series Classification (TSC)	(2019)
Yang et al *CARS*	2020	Continuous evolution strategy which initializes a supernet with sufficient cells to accommodate best architectures found by a non-dominated sorting algorithm (NSGA-III)	(2020)

Table 3 (continued)

Authors	Year	Summary	Reference
Kobayashi and Nagao	2020	Proposed a hybrid approach combining the advantages of gradient-based and evolutionary methods	(2020)

optimization approaches have been proposed, among which *Evolutionary Computation* became very popular by demonstrating promising performance. In this review, we narrowed down our focus to evolutionary methods for designing neural network architectures and presented a comprehensive and up-to-date survey covering three decades of research, with a special emphasis on evolutionary computation techniques adopted and various encoding strategies utilized. We investigated the historical progress in three periods based on significant achievements and scientific trends.

The early attempts were mostly concentrated on defining an efficient encoding strategy due to a great influence of representation on the performance of methods adopted. While some researchers like Miller et al. (1989) simply used direct approaches with basic connectivity matrices or concatenation of parameter strings, others believed that biology dictates indirect encoding with specific developmental patterns observed in nature. New insights in neuroscience have led to the introduction of successful indirect representation methods which mimic the biological neural structure of living organisms. Considering the fact that the human genome accommodates only around 30 thousand active genes, this represents a clear indication of a somewhat indirect relationship compared to the existence of about 100 trillion neural connections in the brain. Pioneers such as Mjolsness, Kitano, and Gruau have all proposed indirect representation methods inspired by biological structures.

Although following the same principles of evolution theory, various evolutionary computation techniques have different approaches for solving the task of architecture optimization. For example, Genetic Algorithms were not preferred by many researchers due to their crossover operator, which normally plays a key role in global optimization problems. Instead, Evolutionary Programming was mostly preferred with its diverse selection and mutation procedures. It was believed that the crossover operator in GAs has the potential to deteriorate child solutions and produce invalid or redundant structures, leading to a known phenomenon called *Competing Conventions Problem*. This drawback usually causes longer computation times and low-quality networks.

With the introduction of NEAT, Neuroevolution gained remarkable success and proved the efficiency of evolutionary approaches for designing neural network architectures. Although using a direct encoding approach in their proposed algorithm, authors of NEAT believed that future studies were destined to focus on indirect encoding and suggested researchers explore the mechanism of how the human brain makes it possible. Later, they proposed HyperNEAT with the rationale that the evolution of indirect genotypes demonstrates natural phenomena with geometric regularities. Interesting images can be obtained on *Picbreeder*, an art application based on HyperNEAT's indirect evolutionary approach (Secretan et al. 2008, 2011).

It was inevitable that the research direction of Neuroevolution would be shifted to deep neural models with the recent advances in Deep Learning, high-end GPUs, and innovative network structures. Although hand-crafted architectures like *ResNet* and *DenseNet* achieved groundbreaking performance on image classification tasks, researchers kept searching for an efficient method to automate the discovery of better models. Most recently, *Reinforcement Learning* and *Evolutionary Computation* approach gained remarkable success. The biggest obstacle in front of both approaches was the lack of computation power required to train evolving architectures on each generation. Therefore, initial attempts were far below satisfactory levels in terms of performance and accuracy. With the increased interest in providing AutoML solutions by tech giants, researchers were able to experiment with their ideas on the massive amount of GPUs

and utilize high-end processing power on the cloud, thus bringing state-of-the-art results on CIFAR and ImageNet classification tasks.

Finally, some may ask if it is worth adopting evolutionary algorithms to search for better network architectures by questioning the use of the massive computation power required. The answer lies within the literature, showing how various techniques resulted in promising models, while some approaches yielded far below expectations. Future research will concentrate on fully automated machine learning, with the increased availability of artificial intelligence tools which do not require expert knowledge. Furthermore, smarter algorithms are expected to replace conventional manual and automatic methods which will enable the construction of Artificial Neural Networks architectures in the most efficient way. Together with more data collected during experiments, such as *Autonomous Driving*, Deep Learning approaches will undoubtedly evolve to faster utilities which sufficiently respond to the needs of the industry. It is not yet known if evolutionary algorithms will pave the way for Artificial General Intelligence (AGI), but we already witnessed how evolution is the key to the continuous improvement of biological organisms. Many believe that the future of Artificial Neural Networks will be shaped by the evolution of architectures.

Acknowledgements We would like to express gratitude to our families for their patience and continuous support. We appreciate and express profound thanks to the Library of Selçuk University and publishers worldwide for providing access to the whole database of scientific research on Neuroevolution.

References

Abadi M et al. (2016) Tensorflow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI), pp 265–283

Abdel-Zaher AM, Eldeib AM (2016) Breast cancer classification using deep belief networks. Expert Syst Appl 46:139–144

Ahmadizar F, Soltanian K, AkhlaghianTab F, Tsoulos I (2015) Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. Eng Appl Artif Intell 39:1–13

Aho I, Kemppainen H, Koskimies K, Makinen E, Niemi T (1999) Searching Neural Network Structures with L Systems and Genetic Algorithms. Int J Comp Mat 73:55–75

Akut R, Kulkarni S (2019) Neuroevolution: using genetic algorithm for optimal design of deep learning models. In: IEEE international conference on electrical, computer and communication technologies (ICECCT), IEEE, pp 1–6

Alba E, Aldana J, Troya J (1993a) Genetic algorithms as heuristics for optimizing ANN design. Artificial Neural Nets and Genetic Algorithms. Springer, pp 683–690

Alba E, Aldana J, Troya JM (1993b) Full automatic ANN design: A genetic approach. International Workshop on Artificial Neural Networks. Springer, pp 399–404

Al-Hyari A, Areibi S (2017) Design space exploration of convolutional neural networks based on Evolutionary Algorithms. J ComputVision Imag Syst 3(1)

Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. Am Stat 46:175–185

Anderson CW (1989) Learning to control an inverted pendulum using neural networks. IEEE Control Syst Mag 9:31–37

Angeline PJ, Saunders GM, Pollack JB (1994) An evolutionary algorithm that constructs recurrent neural networks. IEEE Transact Neural Netw 5:54–65

Arifovic J, Gencay R (2001) Using Genetic Algorithms to Select Architecture of a Feedforward Artificial Neural Network Physica a: Statist Mech Appl 289:574–594

Assunçao F, Lourenço N, Machado P, Ribeiro B (2017) Towards the evolution of multi-layered neural networks: a dynamic structured grammatical evolution approach. In: Proceedings of the genetic and evolutionary computation conference, pp 393–400

Assunção F, Lourenço N, Machado P, Ribeiro B (2018) Evolving the topology of large scale deep neural networks. European Conference on Genetic Programming. Springer, pp 19–34
Assunção F, Correia J, Conceição R, Pimenta M, Tomé B, Lourenço N, Machado P (2019a) Automatic design of artificial neural networks for Gamma-Ray detection. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2933947
Assunção F, Lourenço N, Machado P, Ribeiro B (2019b) Fast-DENSER++: Evolving Fully-Trained Deep Artificial Neural Networks arXiv preprint arXiv:190502969
Azzini A, Tettamanzi A (2006) A new genetic approach for neural network design and optimization PhD. University of Milan, Milan
Azzini A, Tettamanzi AG (2011) Evolutionary ANNs: a State of the Art Survey. Intelligenza Artificiale 5:19–35
Bäck T, Hammel U, Schwefel H-P (1997) Evolutionary computation: comments on the history and current state. IEEE Transact Evol Comput 1:3–17
Baker B, Gupta O, Naik N, Raskar R (2016) Designing neural network architectures using reinforcement learning. arXiv preprint, arXiv:161102167
Balakrishnan K, Honavar V (1995) Evolutionary design of neural architectures. A preliminary taxonomy and guide to literature: Tech. Report CS TR95-01 Dep. of Computer Science, Iowa State University, Ames
Baldominos A, Saez Y, Isasi P (2017) Evolutionary convolutional neural networks: An application to handwriting recognition. Neurocomputing 283:38–52
Baldominos A, Saez Y, Isasi P (2020) On the automated, evolutionary design of neural networks: past, present, and future. Neural Comput Appl 1–27
Barrios D, Carrascal A, Manrique D, Rios J (2002) ADANNET: automatic design of artificial neural networks by evolutionary techniques. In: Research and development in intelligent systems XVIII. Springer, London, pp. 67–80
Barrios D, Carrascal A, Manrique D, Rios J (2003) Cooperative binary-real coded genetic algorithms for generating and adapting artificial neural networks. Neural Comput Appl 12:49–60
Barzilai J, Borwein JM (1988) Two-point step size gradient methods. IMA J Num Anal 8:141–148
Bebis G, Georgiopoulos M, Kasparis T (1997) Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. Neurocomputing 17:167–194
Bebis G, Georgiopoulos M (1995) Improving generalization by using genetic algorithms to determine the neural network size. In: Proceedings of South con'95, IEEE, pp 392–397
Benardos P, Vosniakos G-C (2007) Optimizing feedforward artificial neural network architecture. Eng Appl Artif Intell 20:365–382
Bengio Y (2000) Gradient-based optimization of hyperparameters. Neural Comput 12:1889–1900
Bentley PJ, Kumar S (1999) Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In: GECCO, pp 35–43
Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305
Bhandare A, Kaur D (2018) Designing convolutional neural network architecture using genetic algorithms. In: Proceedings on the international conference on artificial intelligence (ICAI), The steering committee of the world congress in computer science, pp 150–156
Blum C, Roli A (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison ACM Computing Surveys (CSUR) 35:268–308
Bochinski E, Senst T, Sikora T (2017) Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. In: IEEE international conference on image processing (ICIP), pp 3924–3928
Boers EJ, Kuiper H (1992) Biological metaphors and the design of modular artificial neural networks. In: Master Thesis. Dept. of computer science and experimental and theoretical psychology at Leiden University
Bongard JC, Pfeifer R (2001) Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In: Proceedings of the genetic and evolutionary computation conference, p 829836
Boozarjomehry R, Svrcek W (2001) Automatic design of neural network structures. Comput Chem Eng 25:1075–1088
Born J, Santibánez-Koref I, Voigt H-M (1994) Designing neural networks by adaptively building blocks in cascades. International Conference on Parallel Problem Solving from Nature. Springer, pp 472–481
Born J, Santibánez-Koref I (1995) Evolutionary structuring of neural networks by solving a binary problem. In: Operations research proceedings 1994. Springer, pp 394–399

Bornholdt S, Graudenz D (1992) General Asymmetric Neural Networks and Structure Design by Genetic Algorithms. Neural Networks 5:327–334
Branke J (1995) Evolutionary algorithms for neural network design and training. In Proceedings of the 1st nordic workshop on genetic algorithms and its applictions
Braun H, Weisbrod J (1993) Evolving neural feedforward networks. Artificial Neural Nets and Genetic Algorithms. Springer, pp 25–32
Breuel TM (2015) The effects of hyperparameters on SGD training of neural networks. arXiv preprint, arXiv:150802788
Cai H, Chen T, Zhang W, Yu Y, Wang J (2017) Efficient architecture search by network transformation. arXiv preprint arXiv:170704873
Campos De LML, Roisenberg M, de Oliveira RCL (2011) Automatic design of neural networks with L-systems and genetic algorithms-A biologically inspired methodology. In: The 2011 international joint conference on neural networks. IEEE, pp 1199–1206
Campos de LML, de Oliveira RCL, Roisenberg M (2015) Evolving artificial neural networks through l-system and evolutionary computation. In: 2015 International Joint Conference on Neural Networks (IJCNN), 2015. IEEE, pp 1–9
Cangelosi A, Elman JL (1995) Gene regulation and biological development in neural networks: an exploratory model. Technical Report CRL-UCSD
Cangelosi A, Parisi D, Nolfi S (1994) Cell division and migration in a 'genotype' for neural networks. Network: Computation in neural systems 5:497–515
Cangelosi A, Nolfi S, Parisi D (2003) Artificial life models of neural development. In: On growth, form and computers, pp 339–352
Cantú-Paz E, Kamath C (2005) An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. IEEE Transact Syst Man Cybernet Part B (cybernetics) 35:915–927
Carpenter G, Grossberg S (1986) Adaptive resonance theory: Stable self-organization of neural recognition codes in response to arbitrary lists of input patterns. In: Proceedings of the eighth annual conference of the cognitive science society. Erlbaum, pp 45–62
Carvalho de A (1997) Evolutionary design of MLP neural network architectures. In: Proceedings 4th Brazilian symposium on neural networks. IEEE, pp 58–65
Castellani M (2006) ANNE-a new algorithm for evolution of artificial neural network classifier systems. In: 2006 IEEE international conference on evolutionary computation. IEEE, pp 3294–3301
Castellani M (2013) Evolutionary generation of neural network classifiers—An empirical comparison. Neurocomputing 99:214–229
Castillo P, Merelo J, Arenas MG, Romero G (2007) Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters. Inf Sci 177:2884–2905
Castillo P, Arenas M, Castillo-Valdivieso J, Merelo J, Prieto A, Romero G (2003) Artificial neural networks design using evolutionary algorithms. In: Advances in Soft Computing. Springer, pp 43–52
Charalambous C (1992) Conjugate gradient algorithm for efficient training of artificial neural networks. IEE Proceedings G (Circuits Devices and Systems) 139:301–310
Chen YW, Shiu JM (2019) Genetic Design of Topology for Neural Network. In: Proceedings of the 11th International Conference on Information Management and Engineering, pp 25–28
Chen Y, Yang B, Dong J (2004) Nonlinear system modelling via optimal design of neural trees. Int J Neural Syst 14:125–137
Chen T, Goodfellow I, Shlens J (2015) Net2net: Accelerating learning via knowledge transfer arXiv preprint arXiv:151105641
Chen Y, Meng G, Zhang Q, Xiang S, Huang C, Mu L, Wang X (2019a) RENAS: Reinforced evolutionary neural architecture search. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4787–4796
Chen Z, Zhou Y, Huang Z (2019b) Auto-creation of effective neural network architecture by evolutionary algorithm and ResNet for image classification. In: 2019 IEEE international conference on systems, man and cybernetics (SMC). IEEE, pp 3895–3900
Chiroma H, Abdulkareem S, Abubakar A, Herawan T (2017) Neural networks optimization through genetic algorithm searches: a review. Appl Math Inf Sci 11:1543–1564
Cho S-B, Shimohara K (1996) Modular neural networks evolved by genetic programming. In: Proceedings of IEEE international conference on evolutionary computation. IEEE, pp 681–684
Cortes C, Vapnik V (1995) Soft Margin Classifiers. Machine Learning 20:273–297
Cramer NL (1985) A representation for the adaptive generation of simple sequential programs. In: Poceedings of an international conference on genetic algorithms and the applications, pp 183–187

Cui Z, Yang C, Sanyal S (2012) Training Artificial Neural Networks Using APPM. Int J Wireless Mobile Comp 5:168–174
Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems 2:303–314
Dasgupta D, McGregor DR (1992) Designing application-specific neural networks using the structured genetic algorithm. In: COGANN-92 international workshop on combinations of genetic algorithms and neural networks. IEEE, pp 87–96
Dawkins R (1986) The Blind Watchmaker. Harlow Logman
Deb K, Agrawal S, Pratap A, Meyarivan TA (2000) Fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. International conference on parallel problem solving from nature. Springer, pp 849–858
Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II IEEE Trans Evol Comp 6:182–197
Dellaert F, Beer RD (1994) Toward an evolvable model of development for autonomous agent synthesis. In: Artificial life IV, proceedings of the fourth international workshop on the synthesis and simulation of living systems. Citeseer, pp 246–257
Dellaert F, Beer RD (1996) A developmental model for the evolution of complete autonomous agents. In: Proceedings of the fourth international conference on simulation of adaptive behavior. MIT Press Cambridge, MA, pp 393–401
Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE, pp 248–255
Dennis J, John E, Moré JJ (1977) Quasi-Newton methods, motivation and theory. SIAM Rev 19:46–89
Desell T (2017a) Developing a volunteer computing project to evolve convolutional neural networks and their hyperparameters. In: 2017 IEEE 13th International Conference on e-Science. IEEE, pp 19–28
Desell T (2017b) Large scale evolution of convolutional neural networks using volunteer computing. In: Proceedings of the genetic and evolutionary computation conference companion, pp 127–128
Dodd N (1990) Optimisation of network structure using genetic techniques. In: 1990 IJCNN international joint conference on neural networks. IEEE, pp 965–970
Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man, Cybernetics Part B (cybernetics) 26:29–41
Drchal J, Šnorek M (2008) Tree-based indirect encodings for evolutionary development of neural networks. International Conference on Artificial Neural Networks. Springer, pp 839–848
Dréo J, Pétrowski A, Siarry P, Taillard E (2006) Metaheuristics for hard optimization: methods and case studies. Springer Science & Business Media,
Dufourq E, Bassett BA (2017a) Automated problem identification: Regression vs classification via evolutionary deep networks. In: Proceedings of the South African institute of computer scientists and information technologists, pp 1–9
Dufourq E, Bassett BA (2017b) Eden: Evolutionary deep networks for efficient machine learning. In: 2017 pattern recognition association of South Africa and robotics and mechatronics (PRASA-RobMech). IEEE, pp 110–115
Eberhart R, Kennedy J (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. Citeseer, pp 1942–1948
Eggenberger P (1997) Creation of neural networks based on developmental and evolutionary principles. In: International conference on artificial neural networks. Springer, pp 337–342
Eggenberger P (2000) Evolving neural network structures using axonal growth mechanisms. In: Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks. IJCNN 2000. Neural computing: New challenges and perspectives for the New Millennium, 2000. IEEE, pp 591–595
Elias JG (1992) Genetic generation of connection patterns for a dynamic artificial neural network. In: COGANN-92: International workshop on combinations of genetic algorithms and neural networks. IEEE, pp 38–54
Elman JL (1990) Finding Structure in Time. Cognitive Sci 14:179–211
Elsken T, Metzen JH, Hutter F (2018a) Multi-objective architecture search for cnns arXiv preprint arXiv:1804090812
Elsken T, Metzen JH, Hutter F (2018b) Neural architecture search: A survey arXiv preprint arXiv:180805377
Elsken T, Metzen JH, Hutter F (2019) Efficient multi-objective neural architecture search via lamarckian evolution arXiv preprint arXiv:180409081
Fahlman SE, Lebiere C (1990) The cascade-correlation learning architecture. In: Advances in neural information processing systems, pp 524–532
Fang J, Xi Y (1997) Neural Network Design Based on Evolutionary Programming. Artificial Intelligence Eng 11:155–161

Feo TA, Resende MG, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Oper Res 42:860–878
Ferdinando Di A, Calabretta R, Parisi D (2001) Evolving modular architectures for neural networks. In: Connectionist models of learning, development and evolution. Springer, pp 253–262
Fernando C et al (2016) Convolution by evolution: differentiable pattern producing networks. In: Proceedings of the genetic and evolutionary computation conference. ACM, pp 109–116
Ferreira C (2006) Gene expression programming: mathematical modeling by an artificial intelligence, vol 21. Springer
Ferreira C (2001) Gene expression programming: a new adaptive algorithm for solving problems arXiv preprint cs/0102027
Fischer MM, Leung Y (1998) A genetic-algorithms based evolutionary computational neural network for modelling spatial interaction dataNeural network for modelling spatial interaction data. Ann Reg Sci 32:437–458
Fiszelew A, Britos P, Ochoa A, Merlino H, Fernández E, García-Martínez R (2007) Finding optimal neural network architecture using genetic algorithms. Advances in computer science and engineering research. Comput Sci 27:15–24
Fletcher R, Reeves CM (1964) Function Minimization by Conjugate Gradients. Comput J 7:149–154
Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evolut Intell 1:47–62
Fogel LJ (1962) Autonomous Automata. Indust Res 4:14–19
Fogel LJ, Owens AJ, Walsh MJ (1966) Intelligent decision making through a simulation of evolution. Behav Sci 11(4):253-272
Fogel L (1964) On the organization of intellect (Ph. D. thesis) University of California, Los Angeles, CA, USA
Fonseca CM, Fleming PJ (1993) Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. In: LCGA. vol July. Citeseer, pp 416–423
Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol cybernetics 36:193–202
Fullmer B, Miikkulainen R (1992) Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In: Toward a practice of autonomous systems: Proceedings of the first european conference on artificial life. MIT Press, pp 255–262
Funahashi K-I (1989) On the Approximate Realization of Continuous Mappings by Neural Networks. Neural Networks 2:183–192
García-Pedrajas N, Ortiz-Boyer D, Hervás-Martínez C (2006) An alternative approach for neural network evolution with a genetic algorithm: crossover by combinatorial optimization. Neural Netw 19:514–528
García-Pedrajas N, Hervás-Martínez C, Muñoz-Pérez J (2003) COVNET: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Trans Neural Networks 14: 575–596
Gauci J, Stanley K (2007) Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the 9th annual conference on genetic and evolutionary computation. ACM, pp 997–1004
Gauci J, Stanley KO (2010) Autonomous evolution of topographic regularities in artificial neural networks. Neural Comput 22:1860–1898
Glover F (1989) Tabu search—part I. ORSA J comp 1:190–206
Glover F (1990) Tabu search—part II. ORSA J comp 2:4–32
Goldberg DE, Holland JH (1988) Genetic Algorithms and Machine Learning. Machine Learning 3:95–99
Gomez F, Schmidhuber J, Miikkulainen R (2006) Efficient non-linear control through neuroevolution. European Conference on Machine Learning. Springer, pp 654–662
Gomez FJ, Miikkulainen R (1999) Solving non-Markovian control tasks with neuroevolution. In: IJCAI, pp 1356–1361
Gonzalez-Seco J (1992) A genetic algorithm as the learning procedure for neural networks. In: IJCNN international joint conference on neural networks. IEEE, pp 835–840
Greenwood GW (1997) Training Partially Recurrent Neural Networks Using Evolutionary Strategies. IEEE Tran Speech Audio Process 5:192–194
Grönroos MA (1998) Evolutionary design of neural networks. In: Master of science thesis in computer science Dept of mathematical sciences University of Turku
Gruau F (1993) Cellular encoding as a graph grammar. In: IEE colloquium on grammatical inference: Theory, applications and alternatives, 1993. IET, pp 17/11–1710

Gruau F (1992) Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In: COGANN-92: International workshop on combinations of genetic algorithms and neural networks. IEEE, pp 55–74
Gruau F, Quatramaran K (1997) Cellular encoding for interactive evolutionary robotics. In: Fourth European conference on artificial life. MIT Press, p 368
Gruau F, Whitley D, Pyeatt L (1996) A comparison between cellular encoding and direct encoding for genetic neural networks. In: Proceedings of the 1st annual conference on genetic programming, 1996. MIT Press, pp 81–89
Gruau F (1994) Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm. PhD Thesis,
Guha A, Harp SA, Samad T (1988) Genetic synthesis of neural networks Honeywell Corporate System Development Division, Tech Rep CSDD-88–14852-CC-l
Gupta JN, Sexton RS (1999) Comparing backpropagation with a genetic algorithm for neural network training. Omega 27:679–684
Habi HV, Rafalovich G (2019) Genetic Network Architecture Search arXiv preprint arXiv:190702871
Hancock PJ (1992a) Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification. In: COGANN-92: International workshop on combinations of genetic algorithms and neural networks. IEEE, pp 108–122
Hancock PJ, Smith LS (1990) GANNET: Genetic design of a neural net for face recognition. In: International conference on parallel problem solving from nature. Springer, pp 292–296
Hancock PJ (1992b) Pruning neural nets by genetic algorithm. In: Artificial neural networks. Elsevier, pp 991–994
Hancock PJ (1993) Coding strategies for genetic algorithms and neural nets. In: PhD Thesis. Centre for cognitive and computational neuroscience, University of Stirling, UK
Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Proceedings of IEEE international conference on evolutionary computation. IEEE, pp 312–317
Hansen N, Ostermeier A (1997) Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The (/I,)-ES. Eufit 97: 650–654
Happel BL, Murre JM (1992) Designing modular network architectures using a genetic algorithm. In: Artificial Neural Networks. Elsevier, pp 1215–1218
Happel BL, Murre JM (1994) Design and Evolution of Modular Neural Network Architectures. Neural Networks 7(6-7):985–1004
Harp SA, Samad T, Guha A (1989) Towards the genetic synthesis of neural network. In: Proceedings of the third international conference on Genetic algorithms, pp 360–369
Harp SA, Samad T, Guha A (1990) Designing application-specific neural networks using the genetic algorithm. In: Advances in neural information processing systems, pp 447–454
Haykin S (1993) Neural networks and Learning Machines. Prentice Hall
He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
Hecht-Nielsen R (1987) Kolmogorov's mapping neural network existence theorem. In: Proceedings of the international conference on neural networks. IEEE Press New York, pp 11–14
Hestenes MR, Stiefel E (1952) Methods of conjugate gradients for solving linear systems vol 49. vol 1. NBS Washington, DC,
Hillis WD (1990) Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D 42:228–234
Hinton GE, Salakhutdinov RR (2006) Reducing the Dimensionality of Data with Neural Networks. Sci 313:504–507
Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:1527–1554
Hintz KJ, Spofford J (1990) Evolving a neural network. In: Proceedings 5th IEEE international symposium on intelligent control. IEEE, pp 479–484
Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Comput 9:1735–1780
Höffgen K-U, Siemon HP, Ultsch A (1990) Genetic improvements of feedforward nets for approximating functions. International Conference on Parallel Problem Solving from Nature. Springer, pp 302–306
Holland J (1975) Adaptation in natural and artificial systems. MIT Press
Hopfield JJ (1982) Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Sciences 79:2554–2558
Horn J, Goldberg DE, Deb K (1994) Implicit niching in a learning classifier system: Nature's way. Evol Comput 2:37–66

Hornik K (1991) Approximation Capabilities of Multilayer Feedforward Networks. Neural Netw 4:251–257
Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141
Huang H-Y (1970) Unified approach to quadratically convergent algorithms for function minimization. J Optim Theory Appl 5:405–423
Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J physiol 160:106
Hubel DH, Wiesel TN (1959) Receptive fields of single neurones in the cat's striate cortex. J Physiol 148:574
Hubel DH, Wiesel TN (1968) Receptive fields and functional architecture of monkey striate cortex. J Physiol 195:215–243
Ilonen J, Kamarainen J-K, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17:93–105
Irwin-Harris W, Sun Y, Xue B, Zhang M A Graph-Based Encoding for Evolutionary Convolutional Neural Network Architecture Design. In: 2019 IEEE Congress on Evolutionary Computation (CEC), 2019. IEEE, pp 546–553
Jacob C, Rehder J (1993) Evolution of neural net architectures by a hierarchical grammar-based genetic system. Artificial Neural Nets and Genetic Algorithms. Springer, pp 72–79
Jakobi N (1995) Harnessing morphogenesis, cognitive science research paper 423. cogs. Tech. rep. University of Sussex,
Jones AJ (1993) Genetic algorithms and their applications to the design of neural networks. Neural Comput Appl 1:32–45
Jong De KA (1975) Analysis of the behavior of a class of genetic adaptive systems. Technical Report No. 185, Department of Computer and Communication Sciences, University of Michigan
Juang CH, Ni S, Lu PC (1999) Training artificial neural networks with the aid of fuzzy sets. Computer-Aided Civil and Infrastructure Engineering 14:407–415
Jung J-Y, Reggia JA (2008) The automated design of artificial neural networks using evolutionary computation. In: Success in evolutionary computation. Springer, pp 19–41
Jung J-Y, Reggia JAA (2004) Descriptive encoding language for evolving modular neural networks. Genetic and evolutionary computation conference. Springer, pp 519–530
Jung J-Y, Reggia JA (2006) Evolutionary design of neural network architectures using a descriptive encoding language. IEEE Transact Evolut Comput 10:676–688
Karaboga D, Akay B Artificial bee colony (ABC) algorithm on training artificial neural networks. In: 2007 IEEE 15th signal processing and communications applications. IEEE, pp 1–4
Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report-TR06, Erciyes University, Faculty of Engineering
Karunanithi N, Das R, Whitley D Genetic cascade learning for neural networks. In: [Proceedings] COGANN-92: International workshop on combinations of genetic algorithms and neural networks. IEEE, pp 134–145
Kassahun Y, Sommer G (2005) Efficient reinforcement learning through evolutionary acquisition of neural topologies. In: ESANN, pp 259–266
Kim K-J, Cho S-B (2008) Evolutionary ensemble of diverse artificial neural networks using speciation. Neurocomputing 71:1604–1618
Kim Y-H, Reddy B, Yun S, Seo C Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. In: ICML 2017 AutoML Workshop, 2017.
Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing Science 220:671–680
Kitano H (1990) Designing neural networks using genetic algorithms with graph generation system. Complex Systems 4:461–476
Knowles J, Corne D The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 1999. IEEE, pp 98–105
Kobayashi M, Nagao T An evolution-based approach for efficient differentiable architecture search. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, 2020. pp 131–132
Koch H (1906) Une Méthode Géométrique Élémentaire Pour L'étude De Certaines Questions De La Théorie Des Courbes Planes Acta Mathematica 30:145–174
Koehn P (1994) Combining genetic algorithms and neural networks: The encoding problem. MSc Thesis. The University of Tennessee, Knoxville, TN

Kohonen T (1989) Self-organizing feature maps. In: Self-organization and associative memory. Springer, pp 119–157
Kohonen T (1995) Learning vector quantization. In: Self-organizing maps. Springer, pp 175–189
Kolmogorov AN (1957) On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In: Doklady Akademii Nauk, vol 5. Russian Academy of Sciences, pp 953–956
Koza JR, Rice JP Genetic generation of both the weights and architecture for a neural network. In: IJCNN-91-seattle international joint conference on neural networks, 1991. IEEE, pp 397–404
Koza JR Hierarchical Genetic Algorithms Operating on Populations of Computer Programs. In: IJCAI, 1989. pp 768–774
Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection vol 1. MIT press,
Koza JR Survey of genetic algorithms and genetic programming. In: Wescon conference record, 1995. WESTERN PERIODICALS COMPANY, pp 589–594
Kramer O (2018) Evolution of convolutional highway networks. International Conference on the Applications of Evolutionary Computation. Springer, pp 395–404
Krizhevsky A, Sutskever I, Hinton GE Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012. pp 1097–1105
Lan G, De Vries L, Wang S Evolving Efficient Deep Neural Networks for Real-time Object Recognition. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019. IEEE, pp 2571–2578
LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1:541–551
LeCun Y, Boser BE, Denker JS, Henderson D, Howard RE, Hubbard WE, Jackel LD Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems, 1990a. pp 396–404
LeCun Y, Denker JS, Solla SA Optimal brain damage. In: Advances in neural information processing systems, 1990b. pp 598–605
LeCun Y, Haffner P, Bottou L, Bengio Y (1999) Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision. Springer, pp 319–345
Lee D-W, Kong SG, Sim K-B Evolvable neural networks based on developmental models for mobile robot navigation. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005, IEEE, pp 337–342
Leung FH-F, Lam H-K, Ling S-H, Tam PK-S (2003) Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Transact Neural Netw 14:79–88
Levine AB, Schlosser C, Grewal J, Coope R, Jones SJ, Yip S (2019) Rise of the machines: advances in deep learning for cancer diagnosis. Trends Cancer 5:157–169
Liang J, Meyerson E, Hodjat B, Fink D, Mutch K, Miikkulainen R (2019) Evolutionary neural automl for deep learning arXiv preprint arXiv:190206827
Lindenmayer A (1971) Developmental systems without cellular interactions, their languages and grammars Journal of Theoretical Biology 30:455–484
Liu Y, Yao X (1996) A Population-Based Learning Algorithm Which Learns Both Architectures and Weights of Neural Networks Chinese Journal of Advanced Software Research 3:54–65
Liu P, Li Y, El, (2018c) Basha MD, Fang R Neural network evolution using expedited genetic algorithm for medical image denoising. International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp 12–20
Liu P, El Basha MD, Li Y, Xiao Y, Sanelli PC, Fang R (2019) Deep evolutionary networks with expedited genetic algorithms for medical image denoising. Med Image Anal 54:306–315
Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2018a) Hierarchical representations for efficient architecture search arXiv preprint arXiv:171100436
Liu H, Simonyan K, Yang Y (2018b) Darts: Differentiable architecture search arXiv preprint arXiv:180609055
Liu Y, Yao X Evolutionary design of artificial neural networks with different nodes. In: Proceedings of IEEE international conference on evolutionary computation, 1996a. IEEE, pp 670–675
Lock D, Giraud-Carrier C (1999) Evolutionary programming of near-optimal neural networks. Artificial Neural Nets and Genetic Algorithms. Springer, pp 302–306
Loghmanian SMR, Jamaluddin H, Ahmad R, Yusof R, Khalid M (2012) Structure optimization of neural network for dynamic system modeling using multi-objective genetic algorithm. Neural Comput Appl 21:1281–1295
Loshchilov I, Hutter F (2016) CMA-ES for hyperparameter optimization of deep neural networks arXiv preprint arXiv:160407269

Lourenço N, Pereira FB, Costa E (2016) Unveiling the properties of structured grammatical evolution. Genet Program Evolvable Mach 17:251–289
Loussaief S, Abdelkrim A (2018) Convolutional Neural Network Hyper-Parameters Optimization Based on Genetic Algorithms. INT J ADV COMPUT SCI APPL 9:252–266
Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W NSGA-Net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, 2019a. ACM, pp 419–427
Lu Z, Whalen I, Dhebar Y, Deb K, Goodman E, Banzhaf W, Boddeti VN (2019b) Multi-Criterion Evolutionary Design of Deep Convolutional Neural Networks arXiv preprint arXiv:191201369
Lu Z, Deb K, Goodman E, Banzhaf W, Boddeti VN (2020) Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search arXiv preprint arXiv:200710396
Luke S, Spector L Evolving graphs and networks with edge encoding: Preliminary report. In: Late breaking papers at the genetic programming 1996 conference, 1996. Citeseer, pp 117–124
Mandelbrot BB (1982) The Fractal Geometry of Nature. Freeman, San Fransisco
Mandischer M (1993) Representation and evolution of neural networks. Artificial Neural Nets and Genetic Algorithms. Springer, pp 643–649
Maniezzo V (1994) Genetic evolution of the topology and weight distribution of neural networks. IEEE Transact Neural Netw 5:39–53
Marshall S, Harrison R (1991) Optimization and training of feedforward neural networks by genetic algorithms. In: 1991 second international conference on artificial neural networks. IET, pp 39–43
Marti L Genetically generated neural networks-I: representational effects. In: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, 1992. IEEE, pp 537–542
Mason K, Duggan J, Howley E Neural network topology and weight optimization through neuro differential evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2017. pp 213–214
McCulloch WS, Pitts W (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin Math Biophysics 5:115–133
McDonnell JR, Waagen DE Evolving neural network architecture. In: Neural and Stochastic Methods in Image and Signal Processing, 1993. International Society for Optics and Photonics, pp 690–701
Merrill JW, Port RF (1991) Fractally Configured Neural Networks. Neural Networks 4:53–60
Michel O, Biondi J (1995) From the chromosome to the neural network. Artificial Neural Nets and Genetic Algorithms. Springer, pp 80–83
Miikkulainen Risto et al. (2017) Evolving Deep Neural Networks ArXiv preprint arXiv:170300548v2
Miikkulainen R et al. (2019) Evolving deep neural networks. In: Artificial Intelligence in the Age of Neural Networks and Brain Computing. Elsevier, pp 293–312
Miller GF, Todd PM, Hegde SU Designing Neural Networks using Genetic Algorithms. In: ICGA, 1989. pp 379–384
Minsky M, Papert S (1969) Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, MA, USA
Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. Adv Eng Software 69:46–61
Mitchell M (1998) An introduction to genetic algorithms. MIT press,
Mitschke N, Heizmann M, Noffz K-H, Wittmann R Gradient based evolution to optimize the structure of convolutional neural networks. In: 2018 25th IEEE International Conference on Image Processing (ICIP), 2018. IEEE, pp 3438–3442
Mizuta S, Sato T, Lao D, Ikeda M, Shimizu T (2001) Structure design of neural networks using genetic algorithms. Complex Systems 13:161–176
Mjolsness E, Sharp DH, Alpert BK (1989) Scaling, machine learning, and genetic neural nets. Adv appl math 10:137–163
Mladenović N, Hansen P (1997) Variable Neighborhood Search. Comp Operations Res 24:1097–1100
Montana DJ, Davis L Training Feedforward Neural Networks Using Genetic Algorithms. In: IJCAI, 1989. pp 762–767
Moon S-W, Kong S-G (2001) Block-based neural networks. IEEE Trans Neural Networks 12:307–317
Moriarty DE, Miikkulainen R (1993) Evolving complex Othello strategies using marker-based genetic encoding of neural networks. Technical Report AI93-206, Department of Computer Sciences, The University of Texas at Austin
Moriarty DE, Miikkulainen R Hierarchical evolution of neural networks. In: IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), 1996. IEEE, pp 428–433
Moriarty DE, Miikkulainen R (1995a) Discovering complex Othello strategies through evolutionary neural networks. Connect Sci 7:195–210

Moriarty DE, Miikkulainen R (1995b) Game Playing Othello Neuro-EVOLUTION Marker-BASED Encoding. Connect Sci 7:195–210
Moriarty DE, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evol Comput 5:373–399
Moriarty DE, Mikkulainen R (1996) Efficient Reinforcement Learning through Symbiotic Evolution. Mach Learn 22:11–32
Nocedal J, Wright S (2006) Numerical optimization. Springer Science & Business Media,
Nolfi S, Parisi D (1997) Neural networks in an artificial life perspective. International Conference on Artificial Neural Networks. Springer, pp 733–737
Nolfi S, Parisi D, Elman JL (1994) Learning and evolution in neural networks. Adapt Behav 3:5–28
Nolfi S, Parisi D (1991) Growing neural networks. The Handbook of Brain Theory and Neural Networks
Noorian F, de Silva AM, Leong PH (2016) Grammatical Evolution: A Tutorial using gramEvol. Massachusetts Institute of Technology
Ojha VK, Abraham A, Snášel V (2017) Metaheuristic design of feedforward neural networks: A review of two decades of research. Eng Appl Artif Intell 60:97–116
Oliker S, Furst M, Maimon O Design architectures and training of neural networks with a distributed genetic algorithm. In: IEEE International Conference on Neural Networks, 1993. IEEE, pp 199–202
Oong TH, Isa NAM (2011) Adaptive evolutionary artificial neural networks for pattern classification. IEEE Trans Neural Networks 22:1823–1836
Opitz DW, Shavlik JW (1999) A genetic algorithm approach for creating neural network ensembles. In: Combining artificial neural nets. Springer, pp 79–99
Opitz DW, Shavlik JW (1996) Actively searching for an effective neural network ensemble. Connect Sci 8:337–354
Opitz DW, Shavlik JW (1997) Connectionist theory refinement: Genetically searching the space of network topologies. J Artificial Intelligence Res 6:177–209
Palmes PP, Hayasaka T, Usui S (2005) Mutation-based genetic neural network. IEEE Trans Neural Networks 16:587–600
Park J, Sandberg IW (1991) Universal Approximation Using Radial-Basis-Function Networks. Neural Comput 3:246–257
Petroski Such F, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J (2018) Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning arXiv preprint arXiv:171206567
Pham H, Guan MY, Zoph B, Le QV, Dean J (2018) Efficient neural architecture search via parameter sharing arXiv preprint arXiv:180203268
Potter MA, De, (1994) Jong KA A cooperative coevolutionary approach to function optimization. International Conference on Parallel Problem Solving from Nature. Springer, pp 249–257
Potter MA, De, (1995) Jong KA Evolving neural networks with collaborative species. Summer Computer Simulation Conference. SOCIETY FOR COMPUTER SIMULATION, ETC, pp 340–345
Prellberg J, Kramer O (2018) Lamarckian evolution of convolutional neural networks. International Conference on Parallel Problem Solving from Nature. Springer, pp 424–435
Pujol JCF, Poli R (1998) Evolving the topology and the weights of neural networks using a dual representation. Appl Intell 8:73–84
Radcliffe NJ (1993) Genetic set recombination and its application to neural network topology optimisation. Neural Comput Appl 1:67–90
Radcliffe NJ (1990) Genetic neural networks on MIMD computers. KB thesis scanning project 2015
Rashed E, El Seoud M Deep learning approach for breast cancer diagnosis. In: Proceedings of the 2019 8th International Conference on Software and Information Engineering, 2019. ACM, pp 243–247
Real E, Aggarwal A, Huang Y, Le QV Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence, 2019. pp 4780–4789
Real E et al. (2017) Large-scale evolution of image classifiers arXiv preprint arXiv:170301041
Rechenberg I (1973) Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart
Reisinger J, Stanley KO, Miikkulainen R (2004) Evolving reusable neural modules. Genetic and Evolutionary Computation Conference. Springer, pp 69–81
Richards N, Moriarty DE, Miikkulainen R (1998) Evolving neural networks to play Go. Appl Intell 8:85–96
Risi S, Lehman J, Stanley KO Evolving the placement and density of neurons in the hyperneat substrate. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation, 2010. pp 563–570

Rivero D, Dorado J, Fernández-Blanco E, Pazos AA (2009) genetic algorithm for ANN design, training and simplification. International Work-Conference on Artificial Neural Networks. Springer, pp 391–398
Robbins G, Plumbley MD, Hughes JC, Fallside F, Prager R (1993) Generation and adaptation of neural networks by evolutionary techniques (GANNET). Neural Comput Appl 1:23–31
Rocha M, Cortez P, Neves J (2007) Evolution of neural networks for classification and regression. Neurocomputing 70:2809–2816
Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65:386
Roy A, Dutta D, Choudhury K (2013) Training artificial neural network using particle swarm optimization algorithm. Int J Adv Res Comput Science Softw Eng 3(3)
Rudnick M (1990) A bibliography of the intersection of genetic search and artificial neural networks. (Tech. Rep. No. CS/E 90-001). Beaverton: Oregon Graduate Center, Department of Computer Science and Engineering
Rudolph S (1995) Eine Methodik zur systematischen Bewertung von Konstruktionen. PhD thesis, Universität Stuttgart
Rudolph S On a genetic algorithm for the selection of optimally generalizing neural network topologies. In: Proceedings of the 2nd International Conference on Adaptive Computing in Engineering Design and Control, 1996. Citeseer, pp 79–86
Rumelhart D, Hinton G, Williams R (1986) Learning internal representation by error backpropagation, parallel distributed processing: explorations microstructure of cognition. MIT Press, Cambridge
Ryan C, Collins JJ, Neill MO (1998) Grammatical evolution: Evolving programs for an arbitrary language. European Conference on Genetic Programming. Springer, pp 83–96
Saltori C, Roy S, Sebe N, Iacca G (2019) Regularized Evolutionary Algorithm for Dynamic Neural Topology Search. International Conference on Image Analysis and Processing. Springer, pp 219–230
Sałustowicz R (1995) A genetic algorithm for the topological optimization of neural networks. PhD Thesis, Technische Universitat Berlin
Schaffer JD, Whitley D, Eshelman LJ Combinations of genetic algorithms and neural networks: A survey of the state of the art. In: Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on, 1992. IEEE, pp 1–37
Schaffer JD Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the first international conference on genetic algorithms and their applications, 1985, Lawrence Erlbaum Associates. Inc.
Schaffer JD (1986) Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms. PhD Thesis, Vanderbilt University
Schiffmann W, Joost M, Werner R (1990) Performance evaluation of evolutionarily created neural network topologies. International Conference on Parallel Problem Solving from Nature. Springer, pp 274–283
Schiffmann W, Joost M, Werner R (1993) Application of genetic algorithms to the construction of topologies for multilayer perceptrons. Artificial Neural Nets and Genetic Algorithms. Springer, pp 675–682
Schiffmann W, Joost M, Werner R (1992) Synthesis and performance analysis of multilayer neural network architectures Technical Report, University of Koblenz 16
Schwefel H-P (1977) Evolutionsstrategien für die numerische optimierung. In: Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Springer, pp 123–176
Secretan J, Beato N, D Ambrosio DB, Rodriguez A, Campbell A, Stanley KO Picbreeder: evolving pictures collaboratively online. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2008. pp 1759–1768
Secretan J, Beato N, D'Ambrosio DB, Rodriguez A, Campbell A, Folsom-Kovarik JT, Stanley KO (2011) Picbreeder: A case study in collaborative evolutionary exploration of design space Evolutionary computation 19:373–403
Sexton RS, Gupta JN (2000) Comparative Evaluation of Genetic Algorithm and Backpropagation for Training. Neural Networks Information Sciences 129:45–59
Sexton RS, Dorsey RE, Johnson JD (1999) Beyond Backpropagation: Using Simulated Annealing for Training Neural Networks. J Organizational End User Comput (JOEUC) 11:3–10
Siddiqi AA, Lucas SM A comparison of matrix rewriting versus direct encoding for evolving neural networks. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), 1998. IEEE, pp 392–397
Siebel NT, Sommer G (2007) Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems 4:171–183
Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition arXiv preprint arXiv:14091556

Smith RE, Forrest S, Perelson AS (1993) Searching for diverse, cooperative populations with genetic algorithms. Evolutionary computation 1:127–149
Snoek J, Larochelle H, Adams RP Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, 2012. pp 2951–2959
Soltanian K, Tab FA, Zar FA, Tsoulos I Artificial neural networks generation using grammatical evolution. In: 21st Iranian Conference on Electrical Engineering (ICEE), 2013. IEEE, pp 1–5
Spears WM, De, (1993) Jong KA, Bäck T, Fogel DB, De Garis H An overview of evolutionary computation. European Conference on Machine Learning. Springer, pp 442–459
Spielberg S, Tulsyan A, Lawrence NP, Loewen PD, Bhushan Gopaluni R (2019) Toward self-driving processes: A deep reinforcement learning approach to control AIChE Journal 65: e16689
Sprinkhuizen-kuyper IG, Boers EJ, Happel BL, Sprinhuizen-Kuyper IG, Kuiper H Designing modular artificial neural networks. In: Proceedings of computing Science in the Netherlands CSN'93, 1993. Citeseer,
Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2:221–248
Srivastava RK, Greff K, Schmidhuber J (2015) Highway networks arXiv preprint arXiv:150500387
Stanley KO, Miikkulainen R (2001) Evolving neural networks through augmenting topologies. Evol Comput 10:99–127
Stanley KO, Miikkulainen R (2003) A Taxonomy for Artificial Embryogeny. Artificial Life 9:93–130
Stanley KO, D'Ambrosio DB, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15:185–212
Stanley KO, Clune J, Lehman J, Miikkulainen R (2019) Designing neural networks through neuroevolution Nature. Machine Intelligence 1:24–35
Stanley KO, Miikkulainen R Efficient evolution of neural network topologies. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), 2002. IEEE, pp 1757–1762
Stanley KO (2004) Efficient evolution of neural networks through complexification.
Stepniewski SW, Keane AJ (1996) Topology design of feedforward neural networks by genetic algorithms. International Conference on Parallel Problem Solving from Nature. Springer, pp 771–780
Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces Journal of global optimization 11:341–359
Suganuma M, Shirakawa S, Nagao T A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the Genetic and Evolutionary Computation Conference, 2017. ACM, pp 497–504
Sun Y, Xue B, Zhang M, Yen GG (2019a) Automatically evolving cnn architectures based on blocks arXiv preprint arXiv:181011875
Sun Y, Xue B, Zhang M, Yen GG (2019b) Evolving deep convolutional neural networks for image classification IEEE Transactions on Evolutionary Computation
Sun Y, Xue B, Zhang M, Yen GG, Lv J (2019c) Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification IEEE Transactions on Cybernetics
Szegedy C et al. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. pp 1–9
Tan Z-H (2004) Hybrid evolutionary approach for designing neural networks for classification. Electron Lett 40:955–957
Tang K, Chan C, Man K, Kwong S Genetic structure for NN topology and weights optimization. In: First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. IET, pp 250–255
Thorburn WM (1918) The Myth of Occam's Razor Mind 27:345–353
Tirumala SS, Ali S, Ramesh CP Evolving deep neural networks: A new prospect. In: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016. IEEE, pp 69–74
Tsoulos I, Gavrilis D, Glavas E (2008) Neural network construction and training using grammatical evolution. Neurocomputing 72:269–277
Verbancsics P, Harguess J (2013) Generative neuroevolution for deep learning arXiv preprint arXiv:13125355
Verbancsics P, Harguess J Image classification using generative neuro evolution for deep learning. In: 2015 IEEE winter conference on applications of computer vision, 2015. IEEE, pp 488–493
Voigt H-M, Born J, Santibánez-Koref I Evolutionary structuring of artificial neural networks. In: University Berlin, Bionics, 1993. Citeseer,

Vonk E, Jain L, Johnson R Using genetic algorithms with grammar encoding to generate neural networks. In: Proceedings of ICNN'95-International Conference on Neural Networks, 1995a. IEEE, pp 1928–1931

Vonk E, Jain LC, Veelenturf L, Hibbs R Integrating evolutionary computation with neural networks. In: Proceedings Electronic Technology Directions to the Year 2000, 1995b. IEEE, pp 137–143

Vonk E, Jain LC, Veelenturf L, Johnson R Automatic generation of a neural network architecture using evolutionary computation. In: Proceedings Electronic Technology Directions to the Year 2000, 1995c. IEEE, pp 144–149

Wang B, Sun Y, Xue B, Zhang MA (2018) hybrid differential evolution approach to designing deep convolutional neural networks for image classification. Australasian Joint Conference on Artificial Intelligence. Springer, pp 237–250

Wang W, Lu W, Leung AY, Lo S-M, Xu Z, Wang X Optimal feed-forward neural networks based on the combination of constructing and pruning by genetic algorithms. In: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), 2002. IEEE, pp 636–641

Wang W, Lu W, Wang X, Leung AY Developing optimal feed-forward neural networks using a constructive dynamic training method and pruning with a genetic algorithm. In: 7th International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, AICivil-Comp 2003, 2003. Civil-Comp Press

Watkins CJ, Dayan P (1992) Q-Learning. Machine Learning 8:279–292

Weiß G (1994a) Neural networks and evolutionary computation. part I. Hybrid approaches in artificial intelligence. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence. IEEE, pp 268–272

Weiß G (1994b) Neural networks and evolutionary computation. part II: Hybrid approaches in the neurosciences. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence. IEEE, pp 273–277

Weiß G (1993) Towards the synthesis of neural and evolutionary learning. In: Progress in neural networks, vol 5. Ablex Publishing Corporation, pp 145–176

Werbos PJ (1974) Beyond Regression: New tools for prediction and Analysis in the Behavioral Sciences. Harvard University

White D, Ligomenides P (1993) GANNet: A genetic algorithm for optimizing topology and weights in neural network design. International Workshop on Artificial Neural Networks. Springer, pp 322–327

Whitley D (1995) Genetic Algorithms and Neural Networks Genetic Algorithms. Eng Comput Sci 3:203–216

Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: Optimizing connections and connectivity Parallel computing 14:347-361

Whitley D, Gordon VS, Mathias K (1994) Lamarckian evolution, the Baldwin effect and function optimization. International Conference on Parallel Problem Solving from Nature. Springer, pp 5–15

Whitley LD, Gruau F, Pyeatt LD Cellular Encoding Applied to Neurocontrol. In: ICGA, 1995. Citeseer, pp 460–467

Wiegand RP (2003) An analysis of cooperative coevolutionary algorithms. George Mason University

Wilson SW (1989) Perception redux: Emergence of structure Physica D. Nonlinear Phenomena 42:249–256

Wistuba M, Rawat A, Pedapati T (2019) A survey on neural architecture search arXiv preprint arXiv:190501392

Wong F, Goh G (1994) Genetically optimized neural networks. Report, NIBS Pte Ltd

Wu J, Zhang Z, Ji Y, Li S, Lin L A ResNet with GA-based Structure Optimization for Robust Time Series Classification. In: 2019 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE), 2019. IEEE, pp 69–74

Xie L, Yuille A Genetic cnn. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. pp 1379–1388

Yang X-S (2009) Firefly algorithms for multimodal optimization. International symposium on stochastic algorithms. Springer, pp 169–178

Yang X-S, Deb S Cuckoo search via Lévy flights. In: 2009 World congress on nature & biologically inspired computing (NaBIC), 2009. IEEE, pp 210–214

Yang Z et al. Cars: Continuous evolution for efficient neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. pp 1829–1838

Yao X (1993) Evolutionary artificial neural networks. Int J Neural Syst 4:203–222

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87:1423–1447

Yao X, Liu Y (1997) A New Evolutionary System for Evolving Artificial Neural Networks. IEEE Trans Neural Networks 8:694–713

Yao X, Liu Y (1998) Towards designing artificial neural networks by evolution. Appl Math Comput 91:83–90

Yao X, Liu Y Evolving artificial neural networks for medical applications. In: Proceedings of the First Korea-Australia Joint Workshop on Evolutionary Computation, 1995. Citeseer, pp 1–16

Yao X, Liu Y Ensemble structure of evolutionary artificial neural networks. In: Proceedings of IEEE international conference on evolutionary computation, 1996. IEEE, pp 659–664

Yen GG, Lu H Hierarchical genetic algorithm based neural network design. In: 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, 2000. IEEE, pp 168–175

Yen G, Lu H (2002) Hierarchical genetic algorithm for near-optimal feedforward neural network design. Int J Neural Syst 12:31–43

Young SR, Rose DC, Karnowski TP, Lim S-H, Patton RM Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, 2015. ACM, p 4

Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. European conference on computer vision. Springer, pp 818–833

Zhang B-T, Muhlenbein H (1993) Evolving optimal neural networks using genetic algorithms with Occam's razor. Complex Systems 7:199–220

Zhang B-T, Ohm P, Mühlenbein H (1997) Evolutionary induction of sparse neural trees. Evol Comput 5:213–236

Zhang B-T, Mühlenbein H Genetic programming of minimal neural nets using Occam's razor. In: Proceedings of the 5th international conference on genetic algorithms ICGA'93, 1993. Citeseer

Zhang J, Xing L A survey of multiobjective evolutionary algorithms. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2017. IEEE, pp 93–100

Zhong Z, Yan J, Liu C-L (2017) Practical network blocks design with q-learning arXiv preprint arXiv:170805552 1:5

Zhou A, Qu B-Y, Li H, Zhao S-Z, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm Evol Comput 1:32–49

Zhu Y, Yao Y, Wu Z, Chen Y, Li G, Hu H, Xu Y (2018) GP-CNAS: Convolutional Neural Network Architecture Search with Genetic Programming arXiv preprint arXiv:181207611

Zhu H, An Z, Yang C, Xu K, Zhao E, Xu Y EENA: Efficient Evolution of Neural Architecture. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report 103

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE trans Evol Comput 3:257–271

Zoph B, Le QV (2016) Neural architecture search with reinforcement learning arXiv preprint arXiv:161101578