Neuroevolutive Strategies for Topology and Weights Adaptation of Artificial Neural Networks

L. F. Muniz[1], C. N. Lintzmayer[1], C. Jutten[2], D. G. Fantinato[3]

[1] Center for Mathematics, Computing and Cognition, Federal University of ABC, Brazil
l.muniz@aluno.ufabc.edu.br, carla.negri@ufabc.edu.br
[2] University Grenoble-Alpes, France
christian.jutten@grenoble-inp.fr
[3] School of Electrical and Computer Engineering, University of Campinas, Brazil
denisf@unicamp.br

Abstract. Among the methods for training Multilayer Perceptron networks, backpropagation is one of the most used ones on problems of supervised learning. However, it presents some limitations, such as local convergence and the a priori choice of the network topology. Another possible approach for training is to use Genetic Algorithms to optimize the weights and topology of networks, which is known as neuroevolution. In this work, we compare the efficiency of training and defining topology with a modified neuroevolution approach using two different metaheuristics with backpropagation on 5 classification problems. The network's efficiency is assessed through Mutual Information and Information plane. We concluded that neuroevolution found simpler topologies, while backpropagation showed higher efficiency at updating the weights.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning

Keywords: Artificial Neural Networks, Genetic Algorithms, Neural Evolution

1. INTRODUCTION

In recent years, artificial neural networks (ANN) have arisen as a useful machine learning model to solve a great range of real-world problems [Bishop 2006]. One of the most used types of ANN on supervised learning problems is the MultiLayer Perceptron (MLP), which is usually trained by using the error gradient with respect to the parameters that are calculated through the backpropagation algorithm. To update the weights one may use the gradient information in different manners, such as the stochastic gradient descent and the Adaptive Moment Estimation (ADAM) methods [Goodfellow et al. 2016]. Even being an efficient way of updating the several parameters present within an MLP, training algorithms based on backpropagation can have some well-known limitations in the training process. Among those limitations, we highlight the vanishing gradient problem, the high convergence to local minima, and the occurrence of overfitting. Another important fact for successful network training is the adequate definition of network topology, since too many parameters may cause overfitting, while too few could cause underfitting [Goodfellow et al. 2016].

Some techniques were developed to overcome the limitations related to backpropagation [Bishop 2006; Haykin 2009], but the topology definition is mainly done by testing a few architectures, which may lead to a poor/suboptimal choice among the wide range of possible structures. However, from reinforcement learning, there is a different training approach named neuroevolution [Stanley and Miikkulainen 2001], which trains a network using Genetic Algorithms (GA) [Boussaïd et al. 2013] and

This research was funded by São Paulo Research Foundation, grant nº 2020/16456-7 and 2020/10014-2. C. N. Lintzmayer was partially supported by CNPq (Proc. 312026/2021-8 and 428385/2018-4).

may provide a way to overcome the aforementioned limitations, allowing both weight adaptation and topology definition during training, by means of a gradual increment of complexity. In this work, we explore the idea of modifying a neuroevolution algorithm to train an MLP on a set of 5 supervised learning problems. Using the information plane [Shwartz-Ziv and Tishby 2017], the neuroevolution approach was then compared to the backpropagation algorithm in order to verify its capability of overcoming the limitations of the latter.

The rest of this paper is organized as follows. Section 2 briefly explains neuroevolution and the training algorithm proposed in this work. Section 3 introduces the mutual information measure and how it is used to construct the information plane. Section 4 presents and analyses the results obtained in the simulations. Ultimately, Section 5 conveys the conclusion obtained from the results.

2. NEUROEVOLUTION AND MNEAT

Since its creation, Genetic Algorithms (GA) have shown to be a useful optimization method for solving complex optimization problems [Boussaïd et al. 2013]. The GA is a population-based metaheuristic inspired by Darwin's evolution theory. It performs a search for a good solution within a search space by applying small modifications in the solution and selecting the best ones throughout several generations, using the genetic operators of mutation, crossover, and selection. Modified versions of GA can be made by changing the genetic operators, e.g., the Biased Random Key Genetic Algorithm (BRKGA), which has a genetic operator that pushes the search process to the fittest solutions [Martí et al. 2018].

Due to its capacity of optimizing solutions, GA can be used to train an ANN, which is named neuroevolution. One of the most used neuroevolution algorithms is the NeuroEvolution of Augmenting Topologies (NEAT), which is capable of optimizing the network's weights and topology at the same time [Stanley and Miikkulainen 2001; Stanley et al. 2019]. It uses a genetic encoding that is capable of representing precisely an artificial neural network, and each gene has an innovation number that provides a historical mapping among individuals (solutions) within a population. This historical mapping allows individuals to be inserted into a species by comparing their evolutionary history, which allows better search space exploration since species are capable of protecting structural innovation and maintaining diversity throughout generations. The algorithm begins with a population of individuals with a minimum topology, which is gradually increased by structural mutations that add new weights and neurons. Starting with a minimal topology, NEAT reduces the initial search space, optimizing the search process more efficiently than other neuroevolution algorithms.

Even though NEAT was originally built to solve reinforcement learning problems, it has interesting features that could also be used to solve supervised learning problems, as we intend to show. In order to use those features, in this work we created a modified version of NEAT that was applied to train an MLP network on supervised learning problems, named Modified NeuroEvolution of Augmenting Topologies (MNEAT). In addition, we considered two different metaheuristics for MNEAT. The first one is MNEAT-GA, which has genetic operators inspired by GA, and the second one is MNEAT-BRKGA, whose genetic operators are inspired by BRKGA.

One of the main differences between MNEAT and NEAT is the genetic encoding used to represent a solution. In MNEAT, the solution's genome is composed of two genes, the layer gene and the neuron gene, that will form solutions that represent an MLP. Differently from NEAT, the connection weights associated with a certain neuron are stored inside the neuron gene, in a way that each neuron will keep its connection weights. The neuron gene also stores the enable bit, which indicates whether a neuron and its associated connections shall be present in the network or not. Similarly to NEAT, the crossover is done by matching the innovation number stored in the neuron genes of each individual. This new genetic encoding was created to generate a more compact encoding that facilitates crossover. Another difference from NEAT is that MNEAT has three types of mutation. The *add layer mutation*

will add a new layer between two hidden layers that has a number of neurons equal to the average number of neurons of the backward and forward layers. The *add neuron mutation* will add a new neuron in one of the hidden layers that was selected randomly. The *enable bit mutation* will randomly select a neuron gene and change its enable bit. The quality of the solutions within the population was calculated using the error rate on the training set $e_t(i)$ for the i-th solution candidate. Then the fitness function is

$$f(i) = \frac{2}{e_t(i) + 1} - 1\,, \tag{1}$$

with $f(\cdot)$ being maximized in the evolution process. Notice that it will have its maximum value when $e_t = 0$ and minimum when $e_t = 1$, therefore $0 \leq f(i) \leq 1$.

In the same way as NEAT, the evolution process of MNEAT is defined by a set of parameters. Parameter p_m defines the probability of a mutation occurring in a certain solution that is passed through the mutation operator. When a solution suffers mutation, the probability of structural mutation, p_{ms}, defines whether the mutation will add new structures or only change the weight of a certain neuron. In case it suffers a structural mutation, the parameters p_{man}, p_{mal}, and p_{meb} will define the probabilities of adding a new node, adding a new layer, or changing the enable bit, respectively. Otherwise, if it suffers a weight mutation, then a random neuron will be selected and the parameter p_{mw} will define the probability of it having one of its weights randomly changed, while p_{mb} will define the probability of changing its bias weight. Parameters κ_p and κ_b define how much the weight mutation changes a connection weight and a bias weight, respectively. To increase topology diversity within species, MNEAT also allows interspecies crossover, which is controlled by the parameter p_{ic}, which defines the probability of it occurring. The species uses a distance function that measures how different a certain solution is from the species representative, which is defined randomly. A solution will belong to a certain species if its distance is below the compatibility threshold δ_t. Additionally, MNEAT-BRKGA has parameters p_e and ρ, which define the percentage of best solutions of a species that will be put into the elite group and the probability of a new solution inheriting the genes from the elite parent, respectively.

Regarding the application of genetic operators, MNEAT-GA applies operators likewise the original NEAT, i.e., a certain group with the best solutions is selected as the parents group, which will pass through crossover and generate an offspring group. This group also passes through mutation, with the population of the next generation being composed of the parents group plus the offspring group (see Fig. 1). In the MNEAT-BRKGA, a certain percentage of the best solutions is selected as the elite group, and the offspring group will be generated by the crossover operator, which constructs a new solution using one parent from the elite and another from the non-elite group, with the new solutions passing through the mutation operator (see Fig. 2). The mutation operator will be simply a process that generates random solutions, creating the mutant group. In the end, the population of the next generation will be composed of the elite group plus the offspring and the mutant group. MNEAT's mutation operator is also different from the original NEAT since the amount of random change applied to a certain weight will vary accordingly to the fitness value, in such a way that the higher the fitness, the smaller the change will be.

The performance of MNEAT (and backpropagation) can be assessed, as usual, through accuracy. However, this performance measure solely focuses on the quality of the output, without considering the topology and efficiency of the ANN layers. In that sense, as a complementary perspective, we also consider the evaluation of the ANN solution through the Information Plane, as described next.

3. MUTUAL INFORMATION AND INFORMATION PLANE

In supervised learning problems, the weights of a network will be adjusted in order to allow the model to get meaningful information from the input data about the expected results. Hence, an efficient

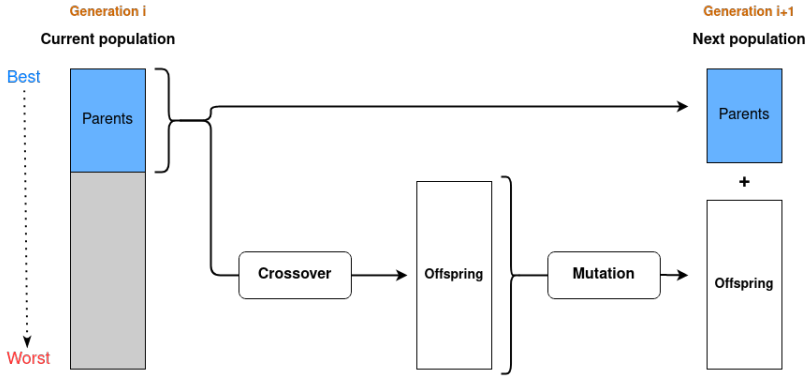


Fig. 1: Application of genetic operators in MNEAT-GA.

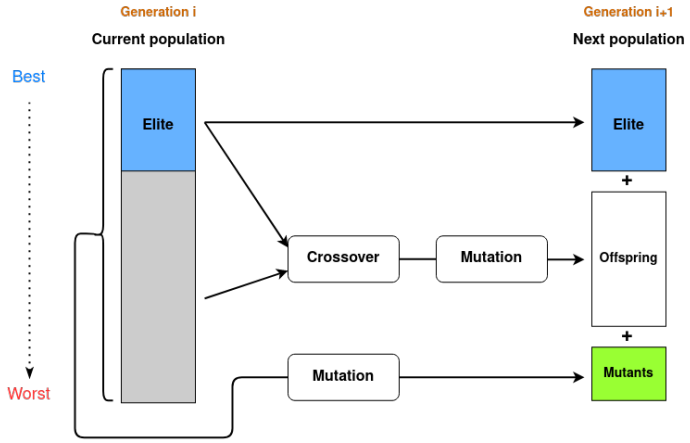


Fig. 2: Application of genetic operators in MNEAT-BRKGA.

network will be one that can get the maximum amount of information about the output from the input data.

In Information Theory, the mutual information measure, denoted as $I(X, Y)$, is capable of evaluating how much information a random variable X carries about another random variable Y. It is defined as [Cover and Thomas 1991]

$$I(X,Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right), \tag{2}$$

where $p(x, y)$ is the joint distribution of X and Y (with values over the space $\mathcal{X} \times \mathcal{Y}$), and $p(x)$ and $p(y)$ are the marginal distributions. Thus, this measure can be used to calculate the efficiency of each network's layer by calculating the amount of information that each layer has about the output. Let us assume X is the random variable associated with the input data, Y is the random variable associated with the output data, and Z_i is the random variable that represents layer i, for $1 < i \leq t$. Then, measuring $I(X, Z_i)$ evaluates the amount of information layer i has about the input data, while $I(Z_i, Y)$ evaluates the information layer i has about the output data. During training, an efficient layer will decrease $I(X, Z_i)$ while increasing $I(Z_i, Y)$, therefore, the former indicates the compression ratio while the latter indicates prediction quality. The efficiency of the entire network can be visualized by using the Information Plane, which allows one to see the evolution of information in each layer throughout the training. In the information plane, each layer i will be placed at the coordinate $(I(X, Z_i), I(Z_i, Y))$ and neighboring layers will be connected by a line [Shwartz-Ziv and Tishby 2017].

The movement of the layers in the information plane can be analyzed to verify if a training method has the capacity of setting efficient layers through training.

The pair $(I(X, Z_i), I(Z_i, Y))$ requires the knowledge of the adjacent distributions, but they are unknown. However, they can be estimated using a number of methods, such as Parzen window [Parzen 1962]. In this work, we used histograms, since it presents reduced computational cost.

4. SIMULATIONS AND RESULTS

In order to compare the training capability of the training methods, each one of them was applied to 5 datasets of classification problems selected from public databases, with each method being simulated 10 times. All the methods used/generated an MLP that has the hyperbolic tangent function as activation in the hidden layer and softmax as activation in the output layer. The training process took 1000 epochs in backpropagation and 1000 generations in MNEAT-GA and MNEAT-BRKGA, with the same MNEAT's parameters set being used on all simulations.

Table I: Characteristics of the datasets used in this work.

Dataset	Number of samples	Number of features	Number of classes
1	200	5	5
2	3000	13	2
3	6497	11	7
4	1885	12	7
5	297	13	5

The number of solutions in the population was set to 150. The parameters p_m, p_{ms}, p_{man}, p_{mal}, p_{meb}, p_{mw}, p_{mb}, and p_{ic} were set as 0.2, 0.15, 0.7, 0.2, 0.15, 0.9, 0.5, and 0.001, respectively. The mutation powers κ_p and κ_b were set to 0.5. In the MNEAT-BRKGA, parameter p_e was set as 20%, while ρ was 0.85. For the MNEAT-GA, the compatibility threshold δ_t was 5, while for MNEAT-BRKGA it was 6. The average error rate and its standard deviation obtained on each training and test set are shown in Tables II and III, respectively, being "BP" an acronym that stands for backpropagation. The datasets' characteristics are described in Table I. Concerning the results' topology, in the backpropagation it was set manually by testing different topologies, while in the neuroevolution methods it was set automatically by the evolution approach.

Table II: Average error rate of 10 simulations on the training set. "BP" stands for backpropagation.

	BP	MNEAT-GA	MNEAT-BRKGA
Dataset 1	0.0000 ± 0.0000	0.1482 ± 0.0802	0.0304 ± 0.0113
Dataset 2	0.1457 ± 0.0219	0.2224 ± 0.0196	0.2178 ± 0.0196
Dataset 3	0.6511 ± 0.0663	0.6175 ± 0.0203	0.6154 ± 0.0292
Dataset 4	0.7815 ± 0.0399	0.7071 ± 0.0208	0.6634 ± 0.0140
Dataset 5	0.2030 ± 0.0392	0.4896 ± 0.0201	0.4192 ± 0.0253

Table III: Average error rate of 10 simulations on the test set. "BP" stands for backpropagation.

	BP	MNEAT-GA	MNEAT-BRKGA
Dataset 1	0.0050 ± 0.0158	0.3475 ± 0.1293	0.1725 ± 0.0583
Dataset 2	0.2135 ± 0.0231	0.2490 ± 0.0337	0.2368 ± 0.0290
Dataset 3	0.7238 ± 0.1539	0.8404 ± 0.0509	0.8193 ± 0.0595
Dataset 4	0.7966 ± 0.1089	0.8822 ± 0.0439	0.8891 ± 0.0523
Dataset 5	0.5317 ± 0.0564	0.5333 ± 0.0491	0.5200 ± 0.0576

By analyzing the results, we can observe that the neuroevolution-based algorithms generated good results on datasets 1 and 2, but the results on datasets 3, 4, and 5 were not as good. However,

it is important to notice that the results got by MNEAT-GA and MNEAT-BRKA were similar to backpropagation, which shows that the neuroevolution approach has the capability of training the networks. Furthermore, the backpropagation approach usually obtained smaller error rates, mainly on the test set, showing that this approach was more precise at adapting the weights with the aim of getting a network with good generalization. Table IV shows the average number of weights (connection and bias weights) of the networks trained in each training method. Comparing the number of weights of the networks obtained in each method, it is noticeable that, in general, the neuroevolution methods obtained smaller topologies than backpropagation, which indicates that these methods were more efficient in finding simpler topologies. In addition, the topologies found by MNEAT-GA were smaller than those found by MNEAT-BRKGA, but the latter obtained a smaller error rate than the former, indicating that the networks found by MNEAT-GA suffered some underfitting while the networks found by MNEAT-BRKGA had better generalization capacity.

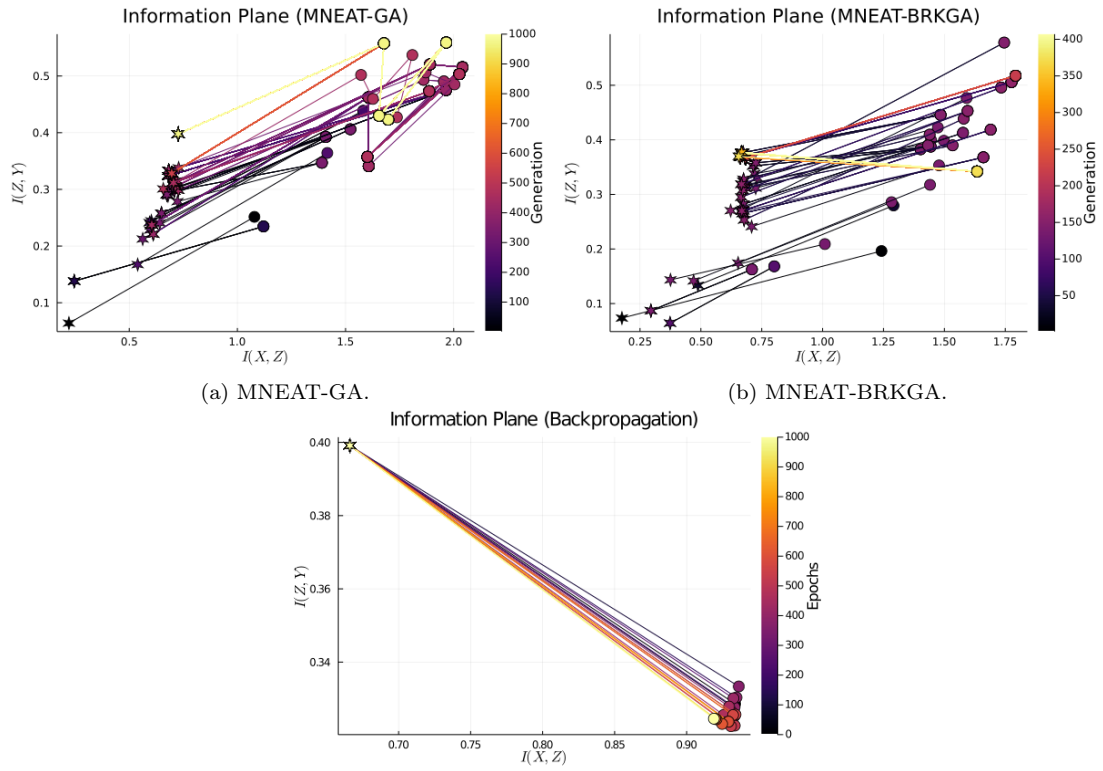
Table IV: Average number of weights of the networks trained in each method. "BP" stands for backpropagation.

Dataset	**BP**	**MNEAT-GA**	**MNEAT-BRKGA**
1	60.0	49.3	112.7
2	290.0	46.8	131.7
3	311.0	75.7	139.3
4	343.0	80.6	183.0
5	176.0	63.7	173.2

To assess the efficiency of the trained networks, the mutual information associated with the layers of the network was plotted on the information plane throughout training, as illustrated in Fig. 3. In this figure, the star marker represents the output layer and the dot markers represent the hidden layers (the input layer is not represented in the plane), and an ANN is represented by a star mark linked with other dot marks. The color in the plane represents the epoch/generation when the network was plotted in the plane: the black network represents the first epoch/generation while the yellow one represents the last one. It is important to mention that in Figs. 3a and 3b the position of the best network was plotted at each generation, while in Fig. 3c the average position of 10 networks was plotted at each 50 epochs. Since in Figu. 3c the network is plotted after the first 50 epochs, we can observe that the position of the output swiftly converges to a certain spot and remains there for the rest of the training, while in Figs. 3a and 3b the output layer position increases slowly. Examining the information plane plots, we can observe that the amount of information in the output layer was similar in all methods, but backpropagation showed better information compression of input data at hidden layers (Fig. 3c) since the amount of input information in the hidden layers was usually smaller than in the neuroevolution. These results show that optimization of weights and topology by neuroevolution is a feasible way of training ANNs and obtaining simple topologies, but this method lacks an efficient mechanism of adapting weights since it is done in a randomized way.

One possible explanation for the low performance of the neuroevolution algorithms in some datasets is the definition of the MNEAT's parameters, since all datasets used the same values for the parameters. Searching for the best value of parameters on each dataset could have generated better results. Another limitation of neuroevolution is that the genetic encoding used to represent the networks tends to generate a lot of species in the same generation and, because of that, the species have few solutions that cannot optimize properly due to lack of diversity within it. It is noteworthy to mention the absence of meaningful stopping criteria for the neurevolution which is capable of stopping the execution when the solutions stop evolving. Currently, the execution stops when a certain number of generations was executed or when a solution with a fitness value above 0.95 is found.

One possible way to overcome the limitations mentioned before is by changing the process of training. As mentioned earlier, the neuroevolution approach is able to define good small topologies while backpropagation is better at updating weights. Thus, a new process of training could be composed of two parts: the first one would be responsible for obtaining a partially trained solution with a



(a) MNEAT-GA.

(b) MNEAT-BRKGA.

(c) *Backpropagation*.

Fig. 3: Evolution of the information plane on dataset 1.

good topology by using neuroevolution, while the second one would update the weights of the found topology by using a gradient-based training algorithm, e.g., backpropagation with stochastic gradient descent. Another strategy to improve the efficiency of the network's layers, mainly its compression capacity, would be to create a fitness function that takes into account some layer efficiency criterion, e.g., *Information Bottleneck* [Shwartz-Ziv and Tishby 2017].

5. CONCLUSION

In this work, we proposed the use of a neuroevolution-based approach in the training of MLPs on supervised learning problems through the adaptation of the NEAT algorithm, usually used on reinforcement learning problems. Moreover, we also investigated the definition of simple and efficient network topologies through an incremental building process used in the GA and BRKGA evolution approaches. After performing simulations using the backpropagation, MNEAT-GA, and MNEAT-BRKGA methods, we observed that the neuroevolution methods (MNEAT-GA and MNEAT-BRKGA) and backpropagation obtained similar performance on all 5 selected datasets. However, backpropagation showed a better capacity for adequately adapting the network's weights, therefore achieving the lowest error rates. By comparing the topologies acquired in each method, it was noticed that MNEAT-BRKGA found the best architectures since it generated the smallest networks with performance similar to the ones trained by backpropagation. Thus, the metaheuristics showed to be efficient at searching for small topologies, however, backpropagation had a better capability of updating weights. The results indicate that a better training method could be achieved by combining MNEAT and backpropagation. This analysis, however, should encompass the use of a more expressive number of datasets. In this sense, for future works, we intend to increase the experimental part by testing on more datasets and

comparing it with other Neural Architecture Search methods [Elsken et al. 2018], which could provide a better understanding on how efficient the proposed evolutionary approach is. Also, in order to improve MNEAT's weight adaptation, we intend to investigate the addition of a weight tuning step in MNEAT based on backpropagation.

REFERENCES

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Boussaïd, I., Lepagnot, J., and Siarry, P. A survey on optimization metaheuristics. *Information sciences* vol. 237, pp. 82–117, 2013.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley-Interscience, 1991.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey, 2018.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

Haykin, S. *Neural Networks and Learning Machines*. Number v. 10 in Neural networks and learning machines. Prentice Hall, 2009.

Martí, R., Pardalos, P. M., and Resende, M. G. C., editors. *Handbook of Heuristics*. Springer International Publishing, 2018.

Parzen, E. On estimation of a probability density function and mode. *The annals of mathematical statistics* 33 (3): 1065–1076, 1962.

Shwartz-Ziv, R. and Tishby, N. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1 (1): 24–35, 2019.

Stanley, K. O. and Miikkulainen, R. Evolving neural networks through augmenting topologies. Tech. Rep. AI-01-290, Department of Computer Sciences, The University of Texas at Austin, 2001.