Scalable Neuroevolution of Ensemble Learners

Marcel Merten[*]
University of Bremen
Bremen, Germany
mar_mer@uni-bremen.de

Rune Krauss[*]
University of Bremen
Bremen, Germany
krauss@uni-bremen.de

Rolf Drechsler
University of Bremen / DFKI
Bremen, Germany
drechsler@uni-bremen.de

ABSTRACT

In recent years, machine learning has become increasingly important in daily life. One of the most popular machine learning models used in many applications is an *Artificial Neural Network* (ANN). While in applications such as automatic speech recognition there is sufficient knowledge about the expected behavior for each input to use supervised learning, other applications like robot control define only an overall target so that the expected output for a given input can be ambiguous, making supervised learning inapplicable. Therefore, *Topology and Weight Evolving ANN* (TWEANN) has been developed in the past to evolve ANN topologies and connection weights. However, challenges of TWEANN are the design of genetic recombination and the exploration of huge search spaces for suitable solutions induced in particular by large-scale problems which can lead to impractical runtimes.

To address the aforementioned issues, this paper proposes a novel evolutionary framework to evolve ensemble learners as specialized ANNs in a divide-and-conquer manner. Specifically, genetic recombination is heavily orchestrated for ANN combinations to effectively find suitable solutions in the search space restricted by ensemble methods. Experiments on various benchmark problems for solving control tasks show that the proposed framework clearly outperforms existing TWEANN algorithms.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic algorithms**; **Ensemble methods**;
• **Mathematics of computing** → *Graph theory*;
• **Applied computing** → Computer games.

KEYWORDS

Neuroevolution, neural networks, genetic algorithms, machine learning, games

ACM Reference Format:
Marcel Merten, Rune Krauss, and Rolf Drechsler. 2023. Scalable Neuroevolution of Ensemble Learners. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590711

[*]Both authors contributed equally to this research.

1 INTRODUCTION

In the course of technological progress, big data is nowadays processed in many computer systems [18]. Useful applications are, e. g., automatic speech recognition and robot control. Due to the increased complexity, interest in machine learning has been growing in recent decades in numerous areas such as the video game industry and medicine [11], making it an essential part of the software design process for many applications [17].

In order to meet user requirements, there is a need to continuously improve algorithms and models in machine learning. A suitable model to realize artificial intelligences for big data problems is a deep *Artificial Neural Network* (ANN) [2]. For example, ANNs can be predetermined for automatic speech recognition by using input neurons to receive acoustic observations and pass their activations forward to output neurons for classifying words on a dictionary. By adjusting connection weights in order to minimize a loss function using gradient descent algorithms such as backpropagation, acoustic observations can be mapped to expected words, with the goal of being able to generalize after supervised learning [12].

Unlike supervised learning, unambiguous outputs do not usually exist for robot control tasks. During *Reinforcement Learning* (RL) using algorithms like deep Q-learning, a (software) robot is trained by receiving feedback from value functions on its actions made by an ANN to improve the behavior in an unknown environment using backpropagation [9]. Another approach is taken by *Neuroevolution* (NE) that evolves ANNs gradient-free using *Genetic Algorithms* (GAs), which are a class of evolutionary algorithms [7]. Compared to Q-learning, direct exploration for solutions in the search space is possible without indirect inference from value functions, and problems with hard-to-compute or without gradients can be handled effectively [15]. Studies have shown that NE is more efficient than RL algorithms like Q-learning for various problems such as robot control [15, 16].

Subsequently, improvements have been researched, especially in terms of NE algorithms. These include, e. g., conventional NE to evolve weights for a fixed ANN topology, novelty search [6] to explore many solutions in the search space without genetic recombination, and *Topology and Weight Evolving ANN* (TWEANN) [13] that evolves both the topology of the ANN and its weights.

Although TWEANN has demonstrated that it can efficiently solve various control problems, there are still challenges to overcome: design of genetic recombination and the lack of scalability, i. e. for large-scale problems an impractical amount of learning time must be spent to find suitable solutions in a huge search space [3].

To address these issues, in this work we present a novel evolutionary framework for the evolution of ensemble learners as specialized ANNs in a *Divide and Conquer* (D&C) manner. While genetic recombination is heavily orchestrated for ANN combinations to effectively find suitable solutions in the search space, *Ensemble Learning* (EL) methods restrict this space. Experiments on benchmark problems for solving control tasks demonstrate that our proposed framework significantly improves the generalization performance of evolved ANNs compared to related TWEANN algorithms, and there is a speedup by a factor of about 10 on average.

In summary, the main contributions are described as follows:

- Recombination for ANN combinations;
- EL methods for search space restriction;
- Framework evaluation and experimental comparison with related TWEANN algorithms on benchmark problems.

This paper is organized as follows: At first, we give an overview of related TWEANN algorithms in Section 2. Section 3 presents the proposed approaches for the evolution of ensemble learners. In Section 4, the experimental setup is shown and results are discussed. Finally, Section 5 concludes the paper.

2 RELATED WORK

The efficiency of TWEANN depends to a large extent on two factors [3]: *design of recombination* and *search space size*. In order to optimize these essential factors, based on ideas in [14], a lot of research has been conducted in TWEANN in the last decades. A comprehensive survey is available in [8]. In the following, we briefly describe TWEANN algorithms that have achieved promising results using related approaches.

Turbo NeuroEvolution of Augmenting Topologies (TNEAT) [5] is an extension of NEAT [14] with D&C methods. It pursues the idea of using multiple populations to evolve ANN combinations, where each population provides an ANN as part of the combination. Generally, the populations are trained using an approach named *Interleaving*, i. e. every ANN of each population is evaluated in combination with every ANN of the other populations. Depending on the problem, there is another approach called *Relay* that divides input data into sequences of equal length, using a different population for each sequence. If a population does not find a solution, another population takes control to continue the search.

Artificial Life Form (ALF) [4] has the following advantages compared with NEAT: speciation via semantic similarity, dynamic populations, and use of fitness-based genetic operators. First, the semantic ANN behavior is used for species classification to improve the search for suitable solutions. Second, the population is dynamically adjusted depending on deleted species to be able to leave unsuitable search regions faster. Third, fitness is used in genetic operations to increase the probability of optimization success.

Although these TWEANN algorithms have demonstrated to be able to successfully solve RL problems, they come with some drawbacks. TNEAT's Interleaving needs exponential time because it is always necessary to compute fitness over entire population combinations. This limitation also makes other D&C methods, based on EL [10], impractical. TNEAT's Relay fails in non-deterministic environments and tends to overfit. Considering ALF, it can be noticed that no D&C methods are integrated to effectively address large-scale problems. Hence, the main goal of this work is to overcome these limitations:

(1) Exponential time for the evolution of ANN combinations;
(2) Failure of learning in non-deterministic environments;
(3) Poor generalization performance due to overfitting.

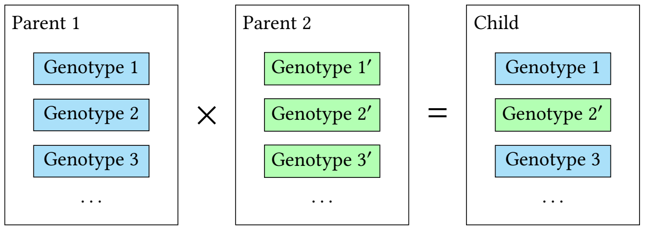


Figure 1: Basic principle of genetic recombination for simultaneous evolution of ANN combinations

3 EVOLUTION OF ENSEMBLE LEARNERS

In this section, we present our novel approaches for the evolution of ensemble learners to improve performance in solving large-scale problems in detail. To this end, Section 3.1 describes a recombination operator that effectively enables ANN combinations for finding solutions in the search space restricted by EL methods, which are then explained in Section 3.2.

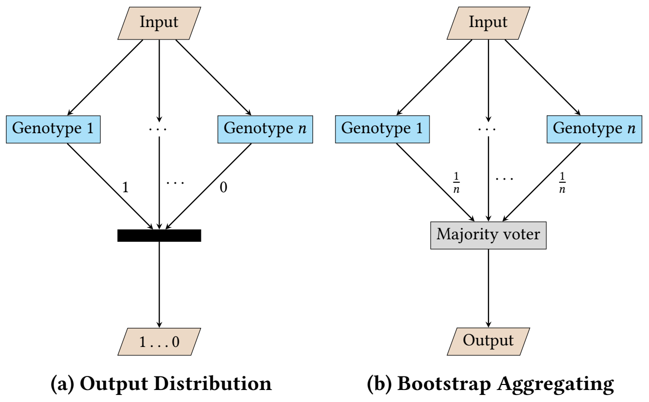
3.1 Recombination for ANN Combinations

Related TWEANN algorithms generally start with initializing a population of random candidate solutions (individuals) within species to solve a problem formulated as a search in a multidimensional parameter space, where each individual represents a point in the search space [8]. A potential solution (individual's phenotype) is encoded into a genotype that corresponds to a single ANN. During generations, the search space is explored for a suitable individual by modifying ANNs of fitter parents using genetic operators: While mutations cause random perturbations to ANNs, genetic recombination mixes ANN topologies in the hope of breeding fitter offsprings. In order to provide ANN combinations for specialized exploration, TNEAT's Interleaving [5] instantiates multiple populations but this leads to significant complications during the learning process as explained in the last section.

Thus, considering the factor *design of recombination*, limitation (1) listed in Section 2 is addressed by introducing an adapted recombination operator that can simultaneously evolve specialized ANN combinations, which is a prerequisite of applying EL methods. The basic principle is shown in Figure 1, where each individual contains a generic vector with multiple genotypes (here: ANNs). Genotypes are compared fitness-based as in [4], i. e. with a higher probability a genotype of the fitter parent is inherited by the child. Assuming that the genotypes of both parents are assigned to the same subproblems, the child can inherit one genotype for each subproblem. As a result, this new operator enhances the capabilities of TWEANN to recombine an ensemble of learners with superior performance compared to both parents.

3.2 EL Methods for Search Space Restriction

The goal of TWEANN is exploring the search space for the best individual, i. e. until the predetermined fitness threshold is reached. The corresponding fitness is generally computed by transforming an ANN into a phenotype using a decoding function and evaluating it for the whole problem using a fitness function with forward passes to compute the loss [7]. In order to be able to evaluate subproblems



(a) Output Distribution **(b) Bootstrap Aggregating**

Figure 2: Concepts of EL methods for ANN combinations

and therefore to restrict the search space, an efficient way must be found to assign them. TNEAT's Relay described in Section 2 bypasses the poor time complexity of Interleaving by dividing the whole problem into consecutive sequences that are solved sequentially, where a specific population is responsible for one subproblem represented by a sequence. Nevertheless, Relay relies on the deterministic behavior of the respective problem. Once a population has finished learning, the evolved ANN can no longer be updated, i. e. evolved ANNs cannot handle random events. Due to the sequential distribution, it also becomes difficult to find patterns during learning that are useful after the training process.

Therefore, considering the factor *search space size*, limitation (2) and (3) listed in Section 2 are addressed by using EL concepts for evaluation of subproblems shown in Figure 2. One concept for dividing a given problem into multiple genotypes in a D&C manner is to predetermine several distinct subproblems. A developed EL method that follows this concept is called *Output Distribution* and is illustrated in Figure 2a. With known n subproblems, each of the n genotypes can be assigned one, where this method uses specialized ANNs that refer to the number of output neurons. The concept of another EL method named *Bootstrap Aggregating* is shown in Figure 2b. Compared to Output Distribution, no distinct subproblems need to be identified before learning. Instead, multiple weak learners are evolved that collaboratively predict for each ANN input. No distribution is required since each ensemble learner is contributing equally to the output given an arbitrary input. The advantage of this method is the majority vote, yielding a more fault-tolerant output behavior of the individual. In particular, faulty outputs of single ANNs in the ensemble can be compensated by the other ANNs. As a result, such concepts have excellent generalization properties compared to single learners [10].

4 EXPERIMENTAL RESULTS

This section summarizes the experiments conducted to empirically analyze the proposed approaches from the last section and demonstrates their benefits. To this end, Section 4.1 discusses the system specification and benchmark problems used for the following performance evaluations. Section 4.2 presents the impact of our approaches by comparing them against related TWEANN algorithms to test their suitability for solving large-scale RL problems.

4.1 Experimental Setup

In order to evaluate the proposed approaches, they were implemented in ALF [4] using C++20 as it has outperformed NEAT [14] on several control tasks. Thus, ALF has been extended and is now considered as an evolutionary framework because of its generic genotype vector described in the last section. However, without using the novel approaches, in this work it is simply called *ALF* for better distinction. For performance evaluation, the framework is compared with related TWEANN algorithms discussed in Section 2: TNEAT's Interleaving, TNEAT's Relay, ALF, and Relay of ALF implemented for better comparability.

Three experiments were conducted because of the limitations identified. Due to representative purposes, *Double Pole Balancing Without Velocities* (DPNV) [14], Snake [19], and *Super Mario Bros.* (SMB) [4] were used as benchmarks for robot control. To allow a fair comparison, the same settings were used in all experiments; they were not tuned for any particular problem. The population size was set to 200 based on analyses presented in [1]. The size of ANN combinations was adjusted to the number of populations in [5] and therefore set to 5. The probability of reproducing an offspring using recombination was 70 %, which is common for GAs [14]. Algorithm-specific parameters corresponded to the default settings.

To demonstrate the scalability of the framework against limitation (1) listed in Section 2, the well-known DPNV problem was set up as in [14] and compared to TNEAT's Interleaving.[1] Framework's Output Distribution was not used here because there is only one output neuron. The criteria for success on this task was keeping both poles balanced for several time steps.

To show the framework's learning ability in non-deterministic environments against limitation (2), Snake was set up similarly to [19] and compared to Relay. This problem is suitable as apples appear randomly on the board. It was measured how much time the snake needs to eat 20 apples on average during 10 games.

To compare generalization performance with Relay in terms of limitation (3), SMB was configured as in [4]. Level 2-4 was used as the reference level for learning, measuring the time in minutes to finish the level at x-coordinate 2,167. Then, levels 1-4 and 3-4 were used as validation levels for testing to measure the generalization performance in x-coordinates, i. e. the position reached by Mario using the trained ANNs from the reference level.

All evaluations were carried out on a Fedora 37 machine with an Intel Xeon E5-2680 v1 CPU with 3.5 GHz and 32 GB of main memory. For each benchmark problem, 10 runs were performed and the average was calculated. The *Time Out* (TO) was set to 24 h.

4.2 Performance Evaluation

The results of the DPNV experiment can be seen in Figure 3 and demonstrate that framework's Bootstrap Aggregating increases the scalability of ALF and outperforms TNEAT's Interleaving as multiple weak learners are trained more efficiently. Considering the runtimes for all time steps, Bootstrap Aggregating solves DPNV faster than Interleaving by a factor of about 10.

The results of the Snake experiment (Table 1) show that Relay fails to solve this problem. The reason is that ANNs learned based on sequences do not provide reliable output for inference results

[1]Due to its high complexity, this D&C method is used only for this experiment.

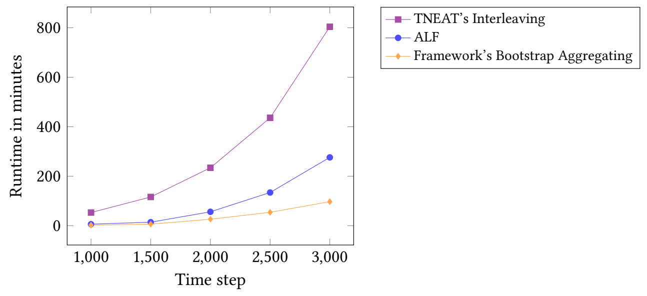


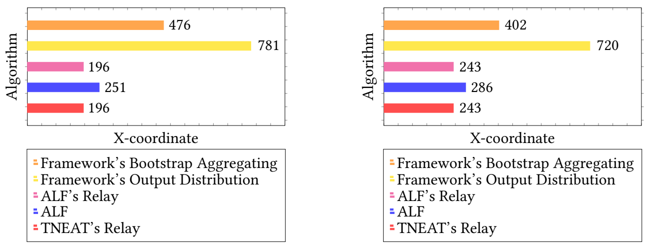
Figure 3: Results of the DPNV experiment

Table 1: Results of the Snake experiment

Algorithm	Learning time in minutes
TNEAT's Relay	TO
ALF	45
ALF's Relay	TO
Framework's Output Distribution	13
Framework's Bootstrap Aggregating	36

Table 2: Learning results of the SMB experiment

Algorithm	Learning time in minutes
TNEAT's Relay	547
ALF	986
ALF's Relay	521
Framework's Output Distribution	629
Framework's Bootstrap Aggregating	803



(a) Level 1-4 **(b) Level 3-4**

Figure 4: Test results of the SMB experiment

in non-deterministic environments like Snake. The framework, especially Output Distribution that increases the performance by a factor of about 3 compared to ALF, solves Snake the fastest due to simultaneous evolution. In this context, Bootstrap Aggregating needs more learning time because of the higher fault tolerance.

While the learning results for the SMB experiment are visualized in Table 2, the test results can be seen in Figure 4: level 1-4 (Figure 4a) and level 3-4 (Figure 4b). Although the framework needs more learning time for level 2-4 compared to Relay due to the deterministic behavior of this problem, it can generalize significantly better. Compared to, e. g., ALF, the best performance achieved by framework's Output Distribution increases by a factor of about 3 w. r. t. the validation levels. Since ANNs are usually used to achieve high generalization performance, this is a successful compromise.

5 CONCLUSION

In this paper, we presented a novel evolutionary framework that can evolve ensemble learners as specialized ANNs in a D&C manner. To this end, genetic recombination for ANN combinations was heavily orchestrated to effectively find individuals in the search space restricted by EL methods. Experimental results on large-scale benchmark problems for robot control confirmed that the implemented framework clearly outperforms related TWEANN algorithms. It significantly improves the generalization performance of evolved ANNs and there is a speedup by a factor of about 10 on average.

Future work will focus on a deeper analysis of the genetic mutation operator to avoid connection weights without impact on the output neurons. Additionally, further genotypes will be explored to evaluate different types of classifiers.

REFERENCES

[1] S. Chen, J. Montgomery, and A. Bolufé-Röhler. 2015. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence* 42, 3 (2015), 514–526. https://doi.org/10.1007/s10489-014-0613-2

[2] K. Du and M. Swamy. 2020. *Neural Networks and Statistical Learning*. Springer.

[3] E. Galván and P. Mooney. 2021. Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges. *IEEE Transactions on Artificial Intelligence* 2, 6 (2021), 476–493. https://doi.org/10.1109/TAI.2021.3067574

[4] R. Krauss, M. Merten, M. Bockholt, and R. Drechsler. 2021. ALF: A Fitness-Based Artificial Life Form for Evolving Large-Scale Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 225–226. https://doi.org/10.1145/3449726.3459545

[5] R. Krauss, M. Merten, M. Bockholt, S. Froehlich, and R. Drechsler. 2020. Efficient Machine Learning through Evolving Combined Deep Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 215–216. https://doi.org/10.1145/3377929.3390055

[6] L. Le Goff, E. Hart, A. Coninx, and S. Doncieux. 2020. On Pros and Cons of Evolving Topologies with Novelty Search. In *The Conference on Artificial Life*. MIT Press, 423–431. https://doi.org/10.1162/isal_a_00291

[7] R. Miikkulainen and K. Stanley. 2009. Evolving Neural Networks. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*. ACM, 2977–3014. https://doi.org/10.1145/1570256.1570410

[8] S. Mirjalili. 2019. *Evolutionary Algorithms and Neural Networks*. Springer Cham.

[9] S. Mousavi, M. Schukat, and E. Howley. 2018. Deep Reinforcement Learning: An Overview. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys)*. Springer International Publishing, 426–440. https://doi.org/10.1007/978-3-319-56991-8_32

[10] D. Opitz and R. Maclin. 1997. An empirical evaluation of bagging and boosting for artificial neural networks. In *Proceedings of International Conference on Neural Networks*. IEEE, 1401–1405. https://doi.org/10.1109/ICNN.1997.613999

[11] A. Pothen. 2022. *Artificial Intelligence and its Increasing Importance*. L'Ordine Nuovo, 74–81.

[12] C. Sekhar and P. Meghana. 2020. A Study on Backpropagation in Artificial Neural Networks. *Asia-Pacific Journal of Neural Networks and Its Applications* 4, 1 (2020), 21–28. https://doi.org/10.21742/AJNNIA.2020.4.1.03

[13] K. Stanley and R. Miikkulainen. 2002. Efficient Evolution of Neural Network Topologies. In *Proceedings of the Congress on Evolutionary Computation*. IEEE, 1757–1762. https://doi.org/10.1109/CEC.2002.1004508

[14] K. Stanley and R. Miikkulainen. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10, 2 (2002), 99–127. https://doi.org/10.1162/106365602320169811

[15] K. Stanley and R. Miikkulainen. 2004. Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research* 21, 1 (2004), 63–100. https://doi.org/10.5555/1622467.1622471

[16] F. Such, V. Madhavan, E. Conti, J. Lehman, K. Stanley, and J. Clune. 2017. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. arXiv:1712.06567 [cs.LG]

[17] Z. Wan, X. Xia, D. Lo, and G. Murphy. 2021. How does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering* 47, 9 (2021), 1857–1871. https://doi.org/10.1109/TSE.2019.2937083

[18] M. Yang. 2017. Research Progress on Data Analysis in Big Data Technology. In *International Conference on Computer Engineering, Information Science & Application Technology*. Atlantis Press, 851–854. https://doi.org/10.2991/iccia-17.2017.153

[19] J. Yeh, P. Su, S. Huang, and T. Chiang. 2016. Snake game AI: Movement rating functions and evolutionary algorithm-based optimization. In *Conference on TAAI*. IEEE, 256–261. https://doi.org/10.1109/TAAI.2016.7880166