Tensors in the Petri Dish: Evolutionary Approaches to Designing and Training Neural Networks

Josh Beal
Department of Computer Science
Stanford University
Stanford, CA 94305
joshbeal@stanford.edu

Morgan Bryant
Department of Computer Science
Stanford University
Stanford, CA 94305
mrbryant@stanford.edu

Abstract

Nature has discovered how to evolve intelligent systems, but our current deep learning methods rely on a great deal of manual configuration and human oversight. Neuroevolutionary methods offer a promising alternative to standard methods for designing and training neural networks. This literature review places the problem in context and surveys the existing work on neuroevolution in deep learning, much of which has been published very recently. This work also offers commentary on future directions for research, from the perspective of students in computer science with knowledge of the problems and methods of theoretical neuroscience.

1 Introduction

Applying deep learning methods to real-world problems involves numerous design challenges in choosing network architecture, tuning hyperparameters, configuring weight initialization, etc. While these decisions are often made through "intelligent design", where a deep learning engineer produces a solution based on their own knowledge of the field, it seems intuitive that better decisions can be made through biologically-inspired, evolutionary approaches.

Leveraging advances in computing power, recent work has found success with neuroevolutionary approaches in both designing and training deep neural network models. Evolutionary techniques have a rich history and offer unique advantages on reinforcement learning tasks and the design and training of convolutional neural networks, recurrent neural networks, higher-order neural networks (such as pi-sigma networks), and other types of deep learning models.

Our goal in this literature review is to give background on neural network training and design, provide a framework for thinking about evolutionary approaches, and then survey the specific existing applications of neuroevolution in deep learning. Finally we will discuss future directions for research, including model compression and defensive mechanisms to adversarial attacks.

2 Training Neural Networks

Artificial Neural Networks are capable tools for solving a variety of problems for numerous reasons, chief among which are the high information capacity and computational inexpense at test time. However, networks must be trained before they can be used, and before they can be trained, their 'topology' must be decided: that is, layer structure, data representation, and other factors collectively called hyperparameters.

The training of neural networks has received significant attention [53, 18] and effective methods are rather well understood. However, hyperparameter design is much more hazy and their tuning is usually done manually [51]. Evolutionary algorithms, however, are capable of automating *both*

Term Paper for APPPHYS 293: Theoretical Neuroscience (Spring 2017) at Stanford University.

aspects [58], and for this reason, their study is worthwhile as one of the first potentially viable methods for doing so.

Hyperparameter optimization. For practical networks, there are often many parameters that can be tuned, such as learning rate schedules, network structure choices, or update mechanism in the simplest of neural networks. In more complicated networks such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), there are often orders more parameters that can be considered. Often only several of the tunable parameters make critical differences. This landscape results in the challenging task of identifying good values of numerous nonlinearly-interacting variables. Furthermore, this problem is especially monumental in neural network research because it takes such a significant amount of computation to train a single network that usual hyperparameter identification strategies are often infeasible. [47]

One parameter space that is particularly difficult to search is in determining an effective network structure. In feedforward networks, this reduces to determining number of layers, layer sizes, and activation choices; however, most real-world deep learning applications use networks at least as complicated as CNNs or RNNs, which have significantly more hyperparameters. In CNNs, additional selection choices include a options for filters and pooling/convolutional/dense layer patterns; for vanilla RNNs, any layer can be made recurrent, each of which has its own unique layer back to itself.

However, much of the recent success in modern network research has come from even more complicated networks. Some examples are:

- The LSTM [35]: this type of network replaces the standard recurrent cell with two layers for the input and the recurrent state to a new 'black box' containing four recurrent cells on the input and the previous state representing the input, output, forget, and state.
- DenseCap solution to image captioning [42]: this network finds a solution to the problem of jointly recognizing a visual image and responding with a semantic description of the image. It does so by first passing the image input to 18 convolutional and pooling layers, fed to a series of layers that localize and warp features, fed finally to an LSTM for language generation. That is, this network has all the parameters of CNNs, RNNs, and more as tunable parameters.
- The Neural Turing Machine (NTM) [23]: this network defines a cell that has a functional memory module with a read-write controller accessible to a feedforward or LSTM network. The original paper identifies novel tunable hyperparameters of memory size, number of read and write heads, and maximum location shift.

Each of these models more than twelve parameters to tune, which already is too many to do brute searches on in most contexts.

Besides the engineer's problem of tuning hyperparameters for a given type of network, the research must also determine *which* type of network structure to use, a problem whose optimum is even more unclear. There have been several attempts at determining reliable, effective ways to identify good hyperparameters, but not many. In the clearest example of recent work on this, Jozefowicz et al. [44] attempted to search for architectures that would outperform GRUs and LSTMs. While they were able to identify numerous comparable structures and several better structures, their relatively naive search strategy involved primarily random alterations to baseline GRU structures searched via beams, which only allowed them to find qualitatively similar architectures, even when they used their full extent of computational resources. These researchers made the point that in order to identify more diverse networks using their strategy, much more computational resources would be necessary.

These realities have left practical hyperparameter tuning as a task to be done manually by experts or guided manually using iteration using large computation at their disposal. Even at this level, hyperparameter search is still coarsely planned. In these scenarios, there are some rules-of-thumb for manual hyperparameter search, most as simple as a grid search [51] or random (e.g. Monte Carlo search) or quasi-random (eg, Latin hypercube) search [5], although the authors Bergstra and Bengio [5] acknowledge that random search (or any other strategy they proposed) is unable to reliably find optimal parameters in some kinds of networks.

Unfortunately, the current search methods as described in Bergstra and Bengio [5] work well primarily only in low-dimensional spaces on real-valued or categorical parameters [38]. Recently a family of model-based strategies have been proposed called *Sequential Model-Based Optimization* or SMBO.

These meta-algorithms attempt to automatically choose a set of hyperparameters, or algorithm, that will perform well, and they do so by iterating between fitting a model and choosing good next hyperparameters to explore. Broadly, SMBO methods define schemes for identifying best-parameters for and deciding next parameter configurations to explore. These methods attempt to model the hyperparameter space and are ready to be extended as model-free or as Bayesian processes [38]. Sequential algorithms can achieve success in parameterizing deep belief networks [6], but some work has shown that random search is often competitive with it, indicating a vacancy for improvement.

Sequential optimizers, however, maintain only approximately one 'best-so-far' estimate and are only able to supplant that champion with a single better parameterization. Additionally, they are primarily effective at searching in a compact hyperparameter space and have not been demonstrated to be effective at designing algorithmic architectures to our knowledge. On the contrary, Evolutionary Algorithms, competitors to SMBO methods, can fundamentally reconcile and combine the capabilities of multiple successful parameter settings. In the remainder of this study, we analyze evolutionary approaches as another viable alternative to initializing networks.

3 Evolutionary Algorithms

There are four major paradigms of evolutionary algorithms, which are helpful to understand in comparing different approaches to evolving neural networks. **Genetic algorithms (GA)** are the most popular type, and are based on a genotype/phenotype model inspired by biology. The genetic encoding typically occurs in a binary space, and the algorithm allows the models to evolve over time with the goal of increased performance. Within the context of deep learning, the phenotype of a neural network model is the specific architecture, weights, hyperparameters, and so on. **Evolution strategies (ES)** differ from GA in that the genotype space is real-valued, and the algorithm is controlled by mutation parameters that generate new candidates. **Evolutionary programming** allows each parent to produce an offspring uniformly, but exerts selection pressure on the offspring, and keeps the structure of the program fixed while the numerical parameters are allowed to vary. **Genetic programming** is similar to GA, but the evolution occurs on programs, which have an enormously large state space. [100]

4 Neuroevolution

Neuroevolutionary methods encompass a wide variety of biologically-inspired approaches to intelligent system design. Based on the notion of fitness from natural selection, one can evolve models with the goal of performing better on a given learning task. The design space of these methods includes the candidate generation / selection and the fitness function / approximation. In many deep learning scenarios, the algorithm is tuning weights via stochastic gradient descent, which narrowly limits the system adaptations that are possible. It is up to the research scientist to specify the topology, figure out the hyperparameters, and tune performance. The goal of neuroevolutionary methods in this context of deep learning model development are to outperform the research scientist through iteration and adaptation. These methods have been popular in the field of reinforcement learning, and have historically found success on tasks that lack a gradient, are partially observable, or have a large space of states and actions for the learning algorithm to explore in order to find a good solution. [54]

The genetic encoding of neural networks is an open research problem, and increasingly sophisticated types of encodings will be needed as the field progresses. The simplest type of encoding, known as conventional neuroevolution (CNE), works by storing all of the weights of a given architecture in sequence. This is a type of direct encoding, with other variants encoding component-level features and topological information. Indirect encodings map to the biological notion of DNA, yielding more compact and modular solutions than can be achieved through direct encoding. Indirect encodings fall into two major categories – grammar-driven and cell chemistry-driven – and can be analyzed along five dimensions in the taxonomy proposed by Stanley and Miikkulainen. These dimension are cell fate, targeting, heterochrony, canalization, and complexification, which give a logical system for analyzing the properties of these "embryogenic" systems for indirect encoding. [90]

Before proceeding into modern work, we thought it would be helpful to briefly survey some of the major examples of neuroevolution methods that are commonly referenced in the literature and helped to inspire some of the approaches that we will discuss later on in this survey.

Angeline et al. [1] introduced GNARL, an evolutionary approach to constructing RNNs. Direct encoding was used in this work, and evolutionary programming was determined to be better for this problem than genetic algorithms. They found that more interesting topologies and architectures arose when the model design was not directly managed by a human programmer.

Yao and Liu [105] also used direct encoding and evolutionary programming in their work on EPNet. By evolving weights and architecture in parallel, through effective mutation operators, models were developed with strong performance and compactness properties.

Gruau et al. [28] introduced an approach called Cellular Encoding that provided a viable alternative to direct encoding for an evolutionary approach to the pole-balancing problem. Their approach encoded both the weights and the architecture, along with syntactic constraints on the possible architectures for the problem. A fitness function was provided that enables the neural network to learn without the velocity provided as input for the pole-balancing problem.

Stanley and Miikkulainen [89] introduced the Neuroevolution of Augmenting Topologies (NEAT) method, which used genetic algorithms to achieve top performance on the pole-balancing benchmark. Their method focused on the intermediate structures rather than just the final product. This work is one of the most-cited in the evolutionary algorithms literature, and has inspired many variants.

In later work, Stanley et al. [91] extended the NEAT approach with Hypercube-based Neuroevolution of Augmenting Topologies (HyperNEAT). HyperNEAT used an indirect encoding scheme called connective Compositional Pattern Producing Networks (connective CPPNs), which improve upon previous work by the capacity to generate regular patterns of connections, much like in the brain.

Siebel and Sommer [86] suggested the Evolutionary Acquisition of Neural Topologies, Version 2 (EANT2) approach, which maintains a clearer separation of structural adaption and parameter optimization as compared to prior work. They use mutation operators for generating new structures, and leverage Covariance Matrix Adaptation Evolution Strategy (CMA-ES), [31] a derandomized variant of evolution strategies, for the parameter optimization.

Gomez et al. [20] proposed Cooperative Synapse Neuroevolution (CoSyNE), which extends the idea of creating modular structures through evolution to creating modular sets of weights that evolve together. This strategy resulted in improved performance on the pole-balancing problem.

We will now survey the modern usage of evolutionary techniques in recent work, spanning convolutional neural networks (CNNs), recurrent neural networks (RNNs), reinforcement learning (RL), and higher-order networks (HONs). While techniques can be relevant to multiple categories, the nature of the network often has an influence on the evolutionary approach used in the work.

5 Evolving Neural Architectures

5.1 Convolutional Neural Networks

Convolutional neural networks have seen remarkable success on a wide variety of computer vision tasks over the past few years, starting with the success of AlexNet on the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Researchers have proposed many improvements to this original architecture, which have increased the number of parameters and possible design choices. In general, increasing depth (VGGNet), developing smart modules (GoogleNet), and optimizing gradient flow through skip connections (Residual Networks) have been winning strategies. However, these models are only the tip of the iceberg, and the reasons for their success, particularly in the case of ResNets, is not well understood. Given the Cambrian explosion of new model architectures in the literature, and the high level of technical expertise necessary to propose innovative architectures, evolutionary methods show great promise in this domain. These methods generate interesting structural improvements and display advantages on weight / hyperparameter selection for training the convolutional neural networks. We discuss both specific techniques and large-scale experiments that have been developed and performed very recently, and hope to give a sense of the fresh enthusiasm for these approaches to deep learning model creation.

Loshchilov and Hutter [55] proposed using the CMA-ES technique for hyperparameter optimization of a CNN trained on the MNIST dataset, commonly used as benchmark for image classification. The CMA-ES method was mentioned previously as a technique for parameter optimization in the EANT2 strategy. They compared this method to GP-based Bayesian optimization as implemented by the

Spearmint package, and trained on 30 GPUs (an indication of the horsepower required for this kind of work). The CMA-ES method consistently achieved better validation accuracy on MNIST. In addition, this method parallelizes very naturally, making it a good fit for distributed training scenarios.

Fernando et al. [13] evolved the CPPN architecture mentioned earlier to support gradient descent, resulting in Differentiable Pattern Producing Networks (DPPNs). These DPPNs could replace CPPNs in a HyperNeat-like framework, but the authors found that a Lamarckian evolution based approach performed better than this approach or Darwinian/Baldwinian evolution. In the end, an approximately convolutional neural network was discovered through evolution rather than by design.

Miikkulainen et al. [60] published a landmark work this year on evolving deep neural networks using an improved version of their past work, namely DeepNEAT and CoDeepNEAT. As opposed to NEAT, the DeepNEAT method encodes the properties of layers rather than neurons, in addition to the hyperparameters of the training. CoDeepNEAT extends this with a coevolution method that facilitates the generation of blueprints that mimic the role of human-designed modules such as the Inception module in generating networks based on deep repetition. Experiments were run on the CIFAR-10 dataset, achieving a comparable error rate and rapid training time.

Real et al. [74] worked on the application of evolutionary techniques to the creation of image classifiers, with a focus on making the process require as little configuration as possible and function well in a distributed setting. Their work is compared to other work from Google, which used reinforcement learning to hone the classifiers. [108] They develop an independent approach from NEAT in this work and base the reproduction mechanism on the notion of tournament selection. Models were evaluated on the CIFAR-10 and CIFAR-100 datasets, and the final models achieved performance comparable to that of the reinforcement learning approach. [19] Future work for this approach includes reducing the computational cost of the evolutionary experiments, which was on the order of 10^{20} FLOPs for their work, which spanned five experiments.

Desell [10] worked on a similar approach that used a volunteer computing project rather than Google's compute infrastructure. Their algorithm, Evolutionary Exploration of Augmenting Convolutional Topologies (EXACT), is more similar to NEAT, and achieved good results on MNIST given the simple backpropagation strategy. The evolved CNNs are remarkably different from human-designed CNNs, and display some of the trademarks of biological evolution, such as vestigial features. Some networks yield many disconnected nodes, and some of the node patterns resemble the input/ouput patterns of neurons found in the cerebral cortex.

Mundt et al. [62] discuss the challenges of reaching optimal representational capacity when designing and training neural networks and propose a metric for evaluating the importance of each feature in the network. This gives an approach for pruning, or, on the other hand, bottom-up network creation. The latter approach yields novel architectures that have reduced size or improved accuracy compared to standard architectures for convolutional neural networks.

Suganuma et al. [92] use a Cartesian genetic programming approach to develop convolutional neural network architectures. This work differs from other works in that it uses direct encoding for the representation, which allows for more flexibility in design. The search space is reduced through highly functional modules, such as convolutional blocks and tensor concatenation. This differs from other approaches such as Real et al. [74] which hoped to avoid high-level unit specification. Fitness was evaluated on the CIFAR-10 classification task. Since many of the hyperparameters were already specified, this work had lower computational cost than other approaches.

5.2 Recurrent Neural Networks

Recurrent neural networks, or networks with cyclic connections and reusable units that are characterized by their suitability for sequential tasks, are widespread in modern deep learning. Broadly, evolutionary techniques have been recently demonstrated to indeed improve recurrent network weight training and architecture design in a number of contexts. However, evolved RNNs have yet to find a commercial or wide-scale application [3]. Here we outline the work that has been done so far to warrant and advance RNN evolution – most of which is rather recent.

There has been significant interest in determining effective RNN cell variations. Numerous studies [33, 25, 51, 44, 11] have taken a look at the most successful recurrent cell today, the LSTM, and analyzed whether or not its expert-chosen features were in fact ideal. All were able to find variants

that are at least comparable in terms of success, but none were able to explore drastically different structures. The research groups acknowledged that standard search methods for comparing LSTM cell to alternatives were restricted by the enormous space of possible cell structures and were unable to confidently express a method for choosing an effective cell type without a computationally expensive search.

The RNN cell choice in the previous paragraph is an example of the extent of choice for a network's topology. In the following paragraph, even more variation is presented within the realm of just recurrent networks. Furthermore, it is often unclear what choices will be effective: while recurrent networks are often chosen for sequential problems and convolutional networks are chosen for image-based problems, the recent Convolutional Sequence to Sequence Learning network [15] counterintuitively found success when they applied a fully *convolutional* architecture to a classically sequential machine translation challenge. Such a poorly understood landscape of effective network structures indicates that automatic methods for deciding network structure would be essentially useful.

Beyond relatively 'simple' RNN cell modifications such as LSTMs or GRUs [35, 9], Olah and Carter [66] have identified more than a dozen recently successful 'augmented RNNs' that contain attentional components, differentiable data structures, or other significant parts not typical to recurrent neural networks [45, 23, 106, 50, 43, 65, 75, 24]. They also note that any or all of the augmentations can hypothetically be combined – a consequence of which is a combinatorial explosion of possible topologies; the models they mention in the study are merely 'points in a broader space'. So, not only are evolutionary methods a potentially helpful way to speed up novel design, but they may become essential as automated ways to select components from the fast-growing, diverse, and highly effective possible RNN augmentations. Indeed, Greve et. al. [26] have demonstrated that an evolutionary version of the Neural Turing Machine (NTM) augmented-RNN architecture [23], evolved using NEAT, exhibits more generalization and has a significantly smaller architecture than NTM, and is able to solve unprecedented tasks. Evolved NTMs have also been demonstrated as capable of performing one-shot reinforcement learning without catastrophic interference [57]. Memory networks [101], or networks augmented with a persistent read-write memory module, are a recent introduction to the modern neural network repertoire that are quickly being addressed as a target for demonstrating advanced learning techniques such as one-shot learning [81], meta learning [80], and low-resource challenges [72]. In a paper of the same flavor [48], researchers designed a new GRU cell with memory but then subjected it to neuroevolution with NEAT. These researchers demonstrate that when a problem has rearrangeable components or a naturally decompositional structure, applying evolutionary techniques can be simple, can make the network effective and competitive, and can fit naturally as a standard neural network training technique (i.e., as a preprocessing strategy as in [48]). Also, by training their networks exclusively with neuroevolution, not backpropogation, they demonstrate that neuroevolution can be fully a viable learning rule.

Olah and Carter [66] also mention that attentional and memory mechanisms are inefficient, since they make their approximate selections by considering *all* selections (through a softmax function). Methods for hard-selecting attentional subjects, such as an evolutionary prior selection, would be an enormous computational improvement.

Among the first to apply neuroevolution to successfully improve on the LSTM cell are Bayer et al. [3], who used NEAT. Since then, Miikkulainen et. al. [60] have identified that NEAT-like evolutionary algorithms applied to recurrent networks can be advantageous both in constructing novel recurrent cells related to LSTMs as in [3, 44] as well as in deciding how to arrange static LSTM-like cells [60]. They further demonstrate that much of the current success in recent LSTM research due to successful architectures can be improved on using NEAT cousin, CoDeepNEAT. In particular, CoDeepNEAT is demonstrated to improve upon the hand-designed current best solutions to Image Captioning problems [60, 46].

5.3 Reinforcement Learning

Reinforcement learning is a set of techniques for learning decisions in a state-action space with rewards. It is fitting that evolutionary techniques originally found success in this field, and are now making a comeback through comparisons to reinforcement learning in popular work published by OpenAI. These techniques find particular relevance here due to the large space for optimization, or in cases where there has been a lack of success in applying gradient-based methods for learning.

For a good overview of older work on reinforcement and neuroevolution, we recommend that the reader consult Whiteson [102], which covers this work in much greater depth. In particular, we would like to highlight Heidrich-Meisner and Igel [34], which explored these types of strategies in 2009. Their work used CMA-ES for policy learning on the pole-balancing problem.

The topic of reinforcement learning has rich intersections with convolutional neural networks and recurrent neural networks. One example of this is Koutník et al. [49], which developed an unsupervised learning approach based on CNNs for a racing car simulation problem. Another example is the use of the previously discussed "Evolving Neural Turing Machines" approach in Lüders et al. [57] for one-shot learning in a reinforcement learning context.

In 2017, OpenAI published their argument that evolution strategies (ES) could serve as a scalable alternative to reinforcement learning. In this case, scalable means parallelizable, and comparable results were achieved on the MuJoCo and Atari problems in reinforcement learning. [79]

Fernando et al. [14] propose PathNet, a modular neural network architecture that helps in learning how to reuse parts of neural networks for other problems, in a step toward general artificial intelligence. Pathways undergo tournament selection, and transfer learning is used to speed up the evolution process. This work demonstrates interesting results on reinforcement learning benchmarks, and the authors intend for PathNet to mimic the function of basal ganglia in the human brain.

Veniat and Denoyer [96] introduced the idea of Budgeted Super Networks that bound inference time for applications that require running models on mobile devices. Reinforcement learning methods are used to optimize for the budget of time efficiency. While evolutionary methods are not used in this work, the idea of using computational budget as a fitness objective would be useful in pursuing similar research that leverages evolutionary techniques for goals beyond just improved accuracy.

Zimmer and Doncieux [107] proposed a way to kickstart the reinforcement learning process with evolutionary methods. Much like with transfer learning, it is often better to start from a good state rather than a random state. This method was applied to several robotics problems using Q-learning.

5.4 Higher-Order Networks

A higher-order network is a network that has layers whose inputs interact beyond just a linear weighted sum. An example second-order network layer may output weights that are the cross product of the input vector with itself. These were first identified as Sigma-Pi networks [78], called so because they take sums over monomials over elements of the input vector, and they have since been used but often in different names: isolated LSTM cells are third-order in the previous state and second-order in the input, and GRU cells are second-order in both input and state. In fact, it has been demonstrated that second-order plain recurrent cells comparable to the LSTM precisely learn finite state automata [17, 68, 22]. Feedforward neural networks are inherently first-order: there are only linear operations on the inputs, and the elementwise nonlinearities do not increase this. High-order artificial networks have been praised for their flexibility and speed at solving non-linear problems [63]. Plain softmax-attentional modulation of weights, such as the read operation or the write operation in Neural Turing Machines [23], or equivalently, gating units [40], define a second-order quadratic interaction; Olah and Carter [66] identify numerous such successful modern attentional networks (NTMs specifically) that are fundamentally higher-order applications. The ILSVRC 2014 champion, GoogLeNet [94], sources much of its success to the repeated second-order module; the ILSVRC 2015 champion, ResNet [32], is fundamentally just a deep series of second-order layers. In a biological system, N competitively-inhibiting neurons form an N-order network. Higher-order networks capture a broad class of interactions, and though they may be a strange way of describing some kinds of networks, they are pervasive in modern neural network research and application. Higher-order networks have been identified and studied in theoretical [39, 82, 83], neuroscientific [7, 27], and computational [67, 99] contexts. Overall, this vein of exploration is rather untouched.

Backpropagated gradient descent, the standard tool for training networks, works decreasingly well as the order of the function increases due to a prevalence of local minima that inhibit the effectiveness of local optimization [39]. Instead, global optimization methods, especially particle swarm optimization, genetic algorithms, and architecture pruning, are viable and suggested alternatives [63, 64]. Since numerous modern successful architectures, such as the LSTM or networks with attentional components, incorporate higher order components, it is fitting to discuss the application of evolutionary approaches as improved optimization methods over gradient descent. Orthogonally, if a researcher

were to decide to use gradient descent, Nayak et al. [64] notes that higher-order networks can alleviate the diminishing learning signal problem [4].

In addition to the utility of high-order layers, cognitive neuroscientific work has supported the idea that biological neurons can and do modulate each other, and that such capabilities may even be critical to synaptic learning [88]. Since many advances in neural network have found initial inspiration in biological analogues [76, 36, 53], it is sensible that the idea of neuromodulation may be useful in future work.

Initial work on high-order networks. There have been several high-order models intentionally designed as such. Functional link networks [16, 37, 109], pi-sigma networks [84], sigma-pi networks [78], ridge polynomial networks [85], and second-order networks [61] have all been demonstrated to have some success. Rovithakis et al. [77] and Sierra et al. [87] have applied specifically evolutionary algorithms to train functional link networks with mild success [70, 71, 69].

Some other work [97, 98] has demonstrated that architecture optimization is an effective way to make high-order networks effective and practical, and related work [103, 98] has demonstrated that genetic algorithms are an effective method for conducting this architecture selection. In one case [63], genetic algorithms were the most effective at learning a high-order network structure, due in part to the infinite number of high-order network layers.

In Miikkulainen et al. [60], a second-order network is discovered as the best model for CIFAR-10 using CoDeepNEAT. It identified the utility of identifying and repeating a fixed-structured module, as in [32, 94]. Examples such as these suggest further analysis on the order of a network may be worthwhile.

Recent advanced work additionally credits work in high-order networks as worthwhile. Reed and de Freitas [75] credit past work on high-order and second-order networks as important for their own work in using part of a network to modulate another part [78, 93].

Progress has been made in reconciling evolutionary and gradient-based techniques as well: Epitropakis et al. [12] discuss an experiment in which the most successful method for training Pi-Sigma networks was hybrid evolutionary and differentiable.

5.5 Other Networks

Researchers have used evolutionary principles in designing 'meta networks', networks that design target networks for specific tasks. The HyperNetwork and the HyperLSTM [29] are networks evolved using HyperNEAT [91] to choose initial weights for a task convolutional or LSTM network respectively, and they are competitive with state-of-the-art models while using fewer parameters.

Jain et. al. [41] have shown that evolutionary techniques can also improve competitive networks, a prospect that is appealing to the very new, promising Generative Adversarial Networks [52, 21].

Loshchilov et al. [56] presented the Limited-Memory Matrix Adaptation Evolution Strategy (LM-MA-ES) and applied this method to generating adversarial inputs to neural networks.

Gruau [27] presented a method for encoding and creating modular neural networks using genetic algorithms.

Baluja [2] presented an early solution to autonomous driving that applied evolutionary techniques to designing a network. In the study, they found that evolutionary approaches were more effective also at training the weights than backpropagation. It cited the fact that evolution approaches optimized globally and thus avoided settling at local minima, an issue common to backpropagation, though backpropagation learned more quickly.

Cangelosi et al. [8] performed preliminary work on the dynamics of evolutionary learning applied to neural networks; results indicate that on simple evolutionary processes, early decisions can persist well into later models.

6 Discussion & Conclusion

Recent advances in the power and availability of deep learning hardware help to explain the resurgence in popularity of these computationally-expensive methods for neural network training and design.

As cloud providers offer more GPU support, and the cost of computation decreases, one would expect these methods to become more widely used in research. New platforms are emerging that will increase access to distributed training. In particular, Sentient Technologies worked with researchers to leverage their DarkCycle platform for idle computational resources. [60] Additionally, the Golem Project is building a decentralized platform for utilizing global computational resources, powered by smart contracts, the Ethereum blockchain, [104] and their own network token. [95]

While most work has focused on accuracy of the resulting model as the measure of fitness, new research could explore alternative measures of fitness that would help solve other problems in deep learning beyond just maximizing performance on a given learning task.

Several papers mentioned how neuroevolution resulted in smaller models. These techniques could explicitly use parameter count and inference time as a fitness measure, yielding new results for the field of model compression. For this direction of research, the space of possible mutations could be expanded to include compression strategies such as pruning, quantization, and Huffman coding. [30]

As another example, neuroevolution could result in more robust defenses against adversarial attacks in machine learning. Just as the body evolves new defenses to harmful bacteria, neural networks could evolve robustness to adversaries, in a defense through diversity approach. [59]

It would also be interesting to study the expressivity of neural networks over the course of evolution by these methods, using the notions developed by Raghu et al. [73] in their work that centered on trajectory length. Exploring this direction could help yield a better understanding of which evolutionary techniques work best for neuroevolution and how best to parameterize these techniques.

References

[1] Peter J Angeline, Gregory M Saunders, and Jordan B Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE transactions on Neural Networks*, 5(1):54–65, 1994.

[2] Shumeet Baluja. Evolution of an artificial neural network based autonomous land vehicle controller. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(3):450–463, 1996.

[3] Justin Bayer, Daan Wierstra, Julian Togelius, and Jürgen Schmidhuber. Evolving memory cell structures for sequence learning. *Artificial Neural Networks–ICANN 2009*, pages 755–764, 2009.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[6] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf`.

[7] E. Blanzieri, F. Grandi, and D. Maio. High-order behaviour in learning gate networks with lateral inhibition. *Biological Cybernetics*, 74(1):73–83, 1996. ISSN 1432-0770. doi: 10.1007/BF00199139. URL `http://dx.doi.org/10.1007/BF00199139`.

[8] Angelo Cangelosi, Domenico Parisi, and Stefano Nolfi. Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*, 5(4):497–515, 1994. doi: 10.1088/0954-898X_5_4_005. URL `http://www.tandfonline.com/doi/abs/10.1088/0954-898X_5_4_005`.

[9] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL `http://arxiv.org/abs/1409.1259`.

[10] Travis Desell. Large scale evolution of convolutional neural networks using volunteer computing. *arXiv preprint arXiv:1703.05422*, 2017.

[11] Rahul Dey and Fathi M. Salem. Gate-variants of gated recurrent unit (GRU) neural networks. *CoRR*, abs/1701.05923, 2017. URL `http://arxiv.org/abs/1701.05923`.

[12] MG Epitropakis, VP Plagianakos, and MN Vrahatis. Higher-order neural networks training using differential evolution. In *International Conference of Numerical Analysis and Applied Mathematics, Wiley-VCH, Crete, Greece*, pages 376–379, 2006.

[13] Chrisantha Fernando, Dylan Banarse, Malcolm Reynolds, Frederic Besse, David Pfau, Max Jaderberg, Marc Lanctot, and Daan Wierstra. Convolution by evolution: Differentiable pattern producing networks. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 109–116. ACM, 2016.

[14] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[15] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017. URL `http://arxiv.org/abs/1705.03122`.

[16] Joydeep Ghosh and Yoan Shin. Efficient higher-order neural networks for classification and function approximation. *International Journal of Neural Systems*, 3(04):323–350, 1992.

[17] C Lee Giles, Clifford B Miller, Dong Chen, Hsing-Hen Chen, Guo-Zheng Sun, and Yee-Chun Lee. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405, 1992.

[18] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[19] David E Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[20] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Efficient non-linear control through neuroevolution. In *European Conference on Machine Learning*, pages 654–662. Springer, 2006.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[22] Mark W Goudreau, C Lee Giles, Srimat T Chakradhar, and D Chen. First-order versus second-order single-layer recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(3):511–513, 1994.

[23] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL `http://arxiv.org/abs/1410.5401`.

[24] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[25] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 2016.

[26] Rasmus Boll Greve, Emil Juul Jacobsen, and Sebastian Risi. Evolving neural turing machines for reward-based learning. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 117–124. ACM, 2016.

[27] Frederic Gruau. Genetic synthesis of modular neural networks. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 318–325, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann. URL `http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/icga93_gruau.pdf`.

[28] Frédéric Gruau, Darrell Whitley, and Larry Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the 1st annual conference on genetic programming*, pages 81–89. MIT Press, 1996.

[29] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. *CoRR*, abs/1609.09106, 2016. URL `http://arxiv.org/abs/1609.09106`.

[30] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[31] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[33] Joel Heck and Fathi M. Salem. Simplified minimal gated unit variations for recurrent neural networks. *CoRR*, abs/1701.03452, 2017. URL `http://arxiv.org/abs/1701.03452`.

[34] Verena Heidrich-Meisner and Christian Igel. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 64(4):152–168, 2009.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[36] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[37] A Hussain, JJ Soraghan, and TS Durbani. A new neural network for nonlinear time-series modelling. *NeuroVest Journal*, pages 16–26, 1997.

[38] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[39] Adiel Ismail. *Training and optimization of product unit neural networks*. PhD thesis, 2005.

[40] R Jacobs and M Jordan. A competitive modular connectionist architecture. *Network*, 2:Y2, 1990.

[41] Ashish Jain, Anand Subramoney, and Risto Miikulainen. Task decomposition with neuroevolution in extended predator-prey domain. *Artificial Life*, 13:341–348, 2012.

[42] Justin Johnson, Andrej Karpathy, and Fei-Fei Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015. URL `http://arxiv.org/abs/1511.07571`.

[43] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems*, pages 190–198, 2015.

[44] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

[45] Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *CoRR*, abs/1511.08228, 2015. URL `http://arxiv.org/abs/1511.08228`.

[46] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[47] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. CS 231N: Convolutional neural networks for visual recognition. `http://cs231n.github.io/`, 2017. Accessed: 2017-6-3.

[48] Shauharda Khadka, Jen Jen Chung, and Kagan Tumer. Evolving memory-augmented neural architecture for deep memory problems. In *Proceedings of the Genetic and Evolutionary Computation Conference 2017*, 2017. To appear.

[49] Jan Koutník, Jürgen Schmidhuber, and Faustino Gomez. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 541–548. ACM, 2014.

[50] Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural random-access machines. *CoRR*, abs/1511.06392, 2015. URL `http://arxiv.org/abs/1511.06392`.

[51] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[52] Yann LeCun. Predictive learning. URL `https://drive.google.com/file/d/0BxKBnD5y2M8NREZod0tVdW5FLTQ/view`. NIPS 2016 Keynote Presentation, 2016.

[53] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] J. Lehman and R. Miikkulainen. Neuroevolution. *Scholarpedia*, 8(6):30977, 2013. doi: 10.4249/scholarpedia.30977. revision #133684.

[55] Ilya Loshchilov and Frank Hutter. Cma-es for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

[56] Ilya Loshchilov, Tobias Glasmachers, and Hans-Georg Beyer. Limited-memory matrix adaptation for large scale black-box optimization. *arXiv preprint arXiv:1705.06693*, 2017.

[57] Benno Lüders, Mikkel Schläger, and Sebastian Risi. Continual learning through evolvable neural turing machines. In *NIPS 2016 Workshop on Continual Learning and Deep Networks (CLDL 2016)*, 2016.

[58] V. Maniezzo. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, Jan 1994. ISSN 1045-9227. doi: 10.1109/72.265959.

[59] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. *arXiv preprint arXiv:1705.09064*, 2017.

[60] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. Evolving deep neural networks. *arXiv preprint arXiv:1703.00548*, 2017.

[61] Srdjan Milenkovic, Zoran Obradovic, and Vanco Litovski. Annealing based dynamic learning in second-order neural networks. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 458–463. IEEE, 1996.

[62] Martin Mundt, Tobias Weis, Kishore Konda, and Visvanathan Ramesh. Building effective deep neural network architectures one feature at a time. *arXiv preprint arXiv:1705.06778*, 2017.

[63] Janmenjoy Nayak, Bighnaraj Naik, and Himansu Sekhar Behera. Solving nonlinear classification problems with black hole optimisation and higher order jordan pi-sigma neural network: a novel approach. *International Journal of Computational Systems Engineering*, 2(4):236–251, 2016.

[64] Janmenjoy Nayak, Bighnaraj Naik, and HS Behera. A novel nature inspired firefly algorithm with higher order neural network: Performance analysis. *Engineering Science and Technology, an International Journal*, 19(1):197–211, 2016.

[65] Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *CoRR*, abs/1511.04834, 2015. URL `http://arxiv.org/abs/1511.04834`.

[66] Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 1(9):e1, 2016.

[67] Christian W Omlin. Training second-order recurrent neural networks using hints. 2014.

[68] Christian W Omlin and C Lee Giles. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM (JACM)*, 43(6):937–972, 1996.

[69] Yoh-Han Pao, Stephen M Phillips, and Dejan J Sobajic. Neural-net computing and the intelligent control of systems. *International Journal of Control*, 56(2):263–289, 1992.

[70] Jagdish C Patra and Ranendra N Pal. A functional link artificial neural network for adaptive channel equalization. *Signal Processing*, 43(2):181–195, 1995.

[71] Jagdish Chandra Patra, Ranendal N Pal, BN Chatterji, and Ganapati Panda. Identification of nonlinear dynamic systems using functional link artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(2):254–262, 1999.

[72] Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*, pages 3621–3629, 2016.

[73] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

[74] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

[75] Scott E. Reed and Nando de Freitas. Neural programmer-interpreters. *CoRR*, abs/1511.06279, 2015. URL `http://arxiv.org/abs/1511.06279`.

[76] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[77] George A Rovithakis, M Maniadakis, and M Zervakis. A hybrid neural network/genetic algorithm approach to optimizing feature extraction for signal classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):695–703, 2004.

[78] David E Rumelhart, James L McClelland, PDP Research Group, et al. Parallel distributed processing, vol. 1&2. *Cambridge, MA: The MIT Press*, 1986.

[79] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[80] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[81] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. One-shot learning with memory-augmented neural networks. *CoRR*, abs/1605.06065, 2016. URL `http://arxiv.org/abs/1605.06065`.

[82] Michael Schmitt. Vc dimension bounds for higher-order neurons. 1999.

[83] Michael Schmitt. Vc dimension bounds for product unit networks. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 4, pages 165–170. IEEE, 2000.

[84] Yoan Shin and Joydeep Ghosh. The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 1, pages 13–18. IEEE, 1991.

[85] Yoan Shin and Joydeep Ghosh. Ridge polynomial networks. *IEEE Transactions on neural networks*, 6(3): 610–622, 1995.

[86] Nils T Siebel and Gerald Sommer. Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, 4(3):171–183, 2007.

[87] Alejandro Sierra, Jose A. Macias, and F Corbacho. Evolution of functional link networks. *IEEE Transactions on Evolutionary Computation*, 5(1):54–65, 2001.

[88] Andrea Soltoggio, John A Bullinaria, Claudio Mattiussi, Peter Dürr, and Dario Floreano. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th International Conference on Artificial Life (Alife XI)*, number LIS-CONF-2008-012, pages 569–576. MIT Press, 2008.

[89] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[90] Kenneth O Stanley and Risto Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2): 93–130, 2003.

[91] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[92] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. *arXiv preprint arXiv:1704.00764*, 2017.

[93] Ilya Sutskever and Geoffrey E Hinton. Using matrices to model symbolic relationship. In *Advances in Neural Information Processing Systems*, pages 1593–1600, 2009.

[94] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[95] Golem Team. The golem project: Crowdfunding whitepaper. `https://golem.network/doc/Golemwhitepaper.pdf`. Accessed: 2017-06-06.

[96] Tom Veniat and Ludovic Denoyer. Learning time-efficient deep architectures with budgeted super networks. *arXiv preprint arXiv:1706.00046*, 2017.

[97] Li Wangchao, Wang Yongbin, Li Wenjing, Zhang Jie, and Jinyan Li. Sparselized higher-order neural network and its pruning algorithm. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 1, pages 359–362 vol.1, May 1998. doi: 10.1109/IJCNN.1998.682292.

[98] Michael J Watts and Nikola K Kasabov. Genetic algorithms for the design of fuzzy neural networks. In *ICONIP*, pages 793–796, 1998.

[99] Cornelius Weber and Stefan Wermter. A self-organizing map of sigma–pi units. *Neurocomputing*, 70(13): 2552–2560, 2007.

[100] Karsten Weicker. *Evolutionary algorithms and dynamic optimization problems*. Der Andere Verlag Berlin, 2003.

[101] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014. URL `http://arxiv.org/abs/1410.3916`.

[102] Shimon Whiteson. Evolutionary computation for reinforcement learning. In *Reinforcement Learning*, pages 325–355. Springer, 2012.

[103] L Darrell Whitley and Christopher Bogart. *Evolution of Connectivity: Pruning Neural Networks Using Genetic Algorithms*. Colorado State University, Department of Computer Science, 1989.

[104] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[105] Xin Yao and Yong Liu. A new evolutionary system for evolving artificial neural networks. *IEEE transactions on neural networks*, 8(3):694–713, 1997.

[106] Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines. *CoRR*, abs/1505.00521, 2015. URL `http://arxiv.org/abs/1505.00521`.

[107] Matthieu Zimmer and Stephane Doncieux. Bootstrapping q-learning for robotics from neuro-evolution results. *IEEE Transactions on Cognitive and Developmental Systems*, 2017.

[108] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[109] Jacek M Zurada. *Introduction to artificial neural systems*, volume 8. West St. Paul, 1992.