Related Work Survey: Discovering Novel Memory Cell Architectures for Recurrent Neural Networks Using Evolutionary Algorithms

Jared Murphy
Graduate Student
Department of Software Engineering
Rochester Institute of Technology
jmm2188@rit.edu

Travis Desell
Associate Professor
Department of Software Engineering
Rochester Institute of Technology
tjdvse@rit.edu

Introduction

This paper provides a related works survey for the discovery of novel memory cell architectures for recurrent neural networks using evolutionary algorithms. It is the goal of this paper to provide the reader with the necessary background to better understand current and future research on the topic.

Background

Neuroevolution (NE) is an exciting field of research that is gaining interest as gradient-based methods for optimizing neural networks is reaching its potential limits. By emulating biological processes, NE uses algorithms to automate the search for the best performing neural network (NN), given a specific task. This methodology can be used to search the space of possible weight combinations, but it is less costly and even more effective when applied to NN architecture search, especially when it comes to recurrent neural networks (RNN). The number of NE algorithms are many, but this paper will primarily focus on those of the genetic type.

At a high level, A genetic algorithm calculates the fitness of an RNN, and then combines it with other RNN's based off their respective fitness', much like in biological natural selection. If an RNN has relatively low fitness it will be discarded, and if it performs well it will combine with other high performers so that combinations/mutations of only the best performing networks are created. In this way the search space for possible RNN architectures is explored efficiently.

Genetic algorithms often employ what is called genetic programming. This type of programming encodes network architecture which is composed of nodes (sums/activation functions) and edges (multiplies)into a string of bits called genomes. This encoding allows for the efficient combining of two or more RNN's which is called crossover/mutation and many different node and edge level mutations are defined for when RNN's are combined/mutated. It is the overall strategy of this research project to use a genetic algorithm to evolve neural networks given simple primitives as building blocks. These primitives will consist of the simple parts of complex memory cells. By providing parts to a genetic algorithm to use in the search for fitness, it is the hope that new and interesting RNN memory cell structures might emerge.

A memory cell is, at its most basic level, is a collection of connected nodes that feed into a recurrent connection that carries their signal forward through time. Each node within a memory cell can be thought of as a gate, whose weights can be trained to achieve

the optimal amount of signal that is either forgotten or remembered. Because RNN's deal with learning time series data, which is data whose state is temporally connected, this ability to train memory cells to remember/forget past information is extremely useful, especially when predicting the future state of a complex system. This research project seeks to leverage the fact that memory cells are nothing more than constrained layers of partially connected nodes that feed forward and into a recurrent connection that connects the organized unit to itself. While much intention has gone into the design of these memory cells on the part of researchers, there is the distinct possibility that the organizational structure of what defines a memory cell is, in reality, arbitrary. It is with this in mind that the idea of feeding the simple constituent parts of memory cells into an evolutionary algorithm came about. Maybe, with the right building blocks, an evolutionary algorithm could create structures that perform quite well and even generalize. Below is a detailed list of prior research that will help further this investigation.

Related Works

Neuroevolution

Investigating Recurrent Neural Network Memory Structures using Neuro-Evolution [1]

In 2019, the Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm was proposed. EXAMM is an evolutionary algorithm that allows for several complex memory cell structures to be used in network design, such as LSTM, GRU, MGU, UGRNN, and also allows for a simple node that applies an activation function to the sum of its inputs. Many different node, edge, and recurrent edge level mutations are defined for when RNN's are combined/mutated, and weights are initialized randomly from the start.

When EXAMM was used on data obtained by the National General Aviation Flight Information Database (NGAFID) and a coal-fired power plant, an interesting result was obtained. It was found that when hidden node types were restricted to MGU nodes, the results of EXAMM were the overall worst. However, when simple nodes were allowed in combination with MGU nodes, the results were the overall best. In fact, it was found that allowing for simple neurons in addition to complex memory cells helped performance generally with few exceptions. With these results come two observations. The first is that many widely used complex memory cell structures are possibly incomplete and not capable of tracking some key dependencies. The second is that EXAMM has the ability, in most cases, to drastically improve performance when given an RNN's most simple constituent parts as building blocks for designing a network.

Where the researchers potentially fall short is that they did not use the EXAMM algorithm towards its full potential. EXAMM seeks to obviate the need for researcher expertise and heuristics in gross level RNN architectural design, however it is unclear why it allows for this at the level of individual memory cells. If given the most basic constituent parts of memory cells to use as building blocks, EXAMM is likely to evolve novel structures that are specifically adapted toward the task at hand. Through statistical analysis, these structures can then potentially be generalized for use as an "out-of-the-box" memory cell type that is available when there is not enough compute to effectively employ an evolutionary algorithm such as EXAMM.

Designing Neural Networks Through Neuroevolution [2]

This paper is more of an overview of the field of neuroevolution rather than pure research, however it does point to many possible avenues for further exploration within the field. This paper validates the line of reasoning that inspired this current research project, primarily regarding the evolution of memory cells. It is explained that the architecture of the LSTM memory cell has remained relatively unchanged since its inception 25 years ago, however with neuroevolution this is changing. Promising results in several experiments have shown that when given simple building blocks, such as activation functions, along with with memory cells, evolutionary algorithms can evolve superstructures much larger than classic LSTM that perform even better when put into a layered network for natural language processing. This is also confirmed in the previous results of EXAMM that inspired this current research project i.e. the best performing networks were those evolved with both MGU and simple nodes. The reported performance improvement when evolutionary algorithms are applied further points to the need for more study in this area. A diversity of algorithms need to be evaluated in evolving memory cell superstructures. This research project aims to give the evolutionary algorithm EXAMM as much freedom as possible to construct novel RNN architectures, and through a process of ginetic encoding, these structures can be statistically analyzed for recurring patterns.

Evolving Neural Networks Through Augmenting Topologies [3]

The NeuroEvolution of Augmenting Topologies (NEAT) algorithm is one of the most widely known methods for evolving neural network topologies. It can be thought of as a predecessor to EXAMM with several key differences, one of which is that NEAT uses a genetic algorithm to evolve both the structure and the weights of a NN while EXAMM uses a GA to evolve the structure structure, but backpropogation to evolve the weights. The approach of the NEAT algorithm is to begin the evolutionary process with as simple a NN structure as possible so that needless complexity is not introduced. Networks are then allowed to evolve within separate "species" so that promising architectures are not prematurely wiped out. Crossover and mutation rules are specified so that the best performing networks in each species can be effectively combined.

The key findings of the NEAT algorithm were measured through a series of ablations where features of the program were removed and performance was measured. The algorithm was tested on the "double cart-pole" problem with velocities, which is a common benchmark test in reinforcement learning. It was found that, among the ablations, the four that contributed to the greatest weakness were the removal of network growth, the removal of speciation, the introduction of random network structure initialization, and the removal of mating/crossover. These findings provide a much needed benchmark for the most important aspects of an neuroevolutionary algorithm that employes genetic programming to encode the network structure.

Where this study falls short is where EXAMM picks up by employing backpropogation for training weights in combination with building in the potential for massive scaling to millions of compute nodes for the evolutionary process.

From Nodes to Networks: Evolving Recurrent Neural Networks [4]

The approach of these researchers was to create a genetic representation of an LSTM node for recurrent neural networks. It then evolved the node with its constituent parts i.e.

recurrent connections and activation functions using a variation of the NEAT algorithm. It used genetic programming to create crossover and mutation operations.

The significant results were that, in contrast to popular opinion, much improvement can be found by employing evolutionary algorithms to evolve LSTM architecture.

Possible improvements to this work can be found through expanding it to encompass all the constituent parts that make up complex memory cells such as GRU, UGRNN and MGU. Improvements can also be found by using a different evolutionary algorithm such as EXAMM.

Memory Cells

Long Short Term Memory [5]

"Learning to store information over extended time intervals by recurrent backpropagation takes a very long time, mostly because of insufficient, decaying error backflow. We briefly review Hochreiter's (1991) analysis of this problem, then address it by introducing a novel, efficient, gradient- based method called long short-term memory (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete-time steps by enforcing constant error flow through constant error carousels within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is O(1). Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with real-time recurrent learning, back propagation through time, recurrent cascade correlation, Elman nets, and neural sequence chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms."

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [6]

"In this paper we compare different types of recurrent units in recurrent neural networks (RNNs). Especially, we focus on more sophisticated units that implement a gating mechanism, such as a long short-term memory (LSTM) unit and a recently proposed gated recurrent unit (GRU). We evaluate these recurrent units on the tasks of polyphonic music modeling and speech signal modeling. Our experiments revealed that these advanced recurrent units are indeed better than more traditional recurrent units such as tanh units. Also, we found GRU to be comparable to LSTM."

Minimal Gated Unit for Recurrent Neural Networks [7]

"Recurrent neural networks (RNN) have been very successful in handling sequence data. However, understanding RNN and finding the best practices for RNN learning is a difficult task, partly because there are many competing and complex hidden units, such as the long short-term memory (LSTM) and the gated recurrent unit (GRU). We propose a gated unit for RNN, named as minimal gated unit (MGU), since it only contains one gate, which is a minimal design among all gated hidden units. The design of MGU benefits from evaluation results on LSTM and GRU in the literature. Experiments on various sequence data show that MGU has comparable accuracy with GRU, but has a

simpler structure, fewer parameters, and faster training. Hence, MGU is suitable in RNNs applications. Its simple architecture also means that it is easier to evaluate and tune, and in principle it is easier to study MGUs properties theoretically and empirically."

Capacity and Trainability in Neural Networks [8]

"Two potential bottlenecks on the expressiveness of recurrent neural networks (RNNs) are their ability to store information about the task in their parameters, and to store information about the input history in their units. We show experimentally that all common RNN architectures achieve nearly the same per-task and per-unit capacity bounds with careful training, for a variety of tasks and stacking depths. They can store an amount of task information which is linear in the number of parameters, and is approximately 5 bits per parameter. They can additionally store approximately one real number from their input history per hidden unit. We further find that for several tasks it is the per-task parameter capacity bound that determines performance. These results suggest that many previous results comparing RNN architectures are driven primarily by differences in training effectiveness, rather than differences in capacity. Supporting this observation, we compare training difficulty for several architectures, and show that vanilla RNNs are far more difficult to train, yet have slightly higher capacity. Finally, we propose two novel RNN architectures, one of which is easier to train than the LSTM or GRU for deeply stacked architectures."

Learning Simpler Language Models With the Differential State Framework [9]

"Learning useful information across long time lags is a critical and difficult problem for temporal neural models in tasks such as language modeling. Existing architectures that address the issue are often complex and costly to train. The Differential State Framework (DSF) is a simple and high-performing design that unifies previously introduced gated neural models. DSF models maintain longer- term memory by learning to interpolate between a fast-changing data-driven representation and a slowly changing, implicitly stable state. This requires hardly any more parameters than a classical, simple recurrent network. Within the DSF framework, a new architecture is presented, the Delta-RNN. In language modeling at the word and character levels, the Delta-RNN outperforms popular complex architectures, such as the Long Short Term Memory (LSTM) and the Gated Recurrent Unit (GRU), and, when regularized, performs comparably to several state-of-the-art baselines. At the subword level, the Delta-RNN's performance is comparable to that of complex gated architectures."

Conclusion

The proposed work, which uses evolutionary algorithms to discover new memory cell architectures has drawn much inspiration from the sources listed above. This task has been carried out in part already by researchers Rawal and Miikkulainen in evolving LSTM structures, however the proposed work would like to expand their method to discover a more generalized memory cell meta-structure. By providing the constituent parts of several complex memory cell types, and not just LSTM, to an evolutionary algorithm as building blocks, new structures might emerge. Also, using genetic programming to create

a data set of the best performing neural networks to statistically search for patterns within the population is, to our knowledge, not reflected in the literature. In this sense, our endeavor is new.

Using the EXAMM algorithm to search for novel memory cell architectures is the natural next step. EXAMM already expands upon NEAT by including "more advanced node-level mutations, and utilizing Lamarckian weight initialization along with back-propagation to evolve the weights as opposed to a simpler less efficient genetic strategy. Additionally, it has been developed with largescale concurrency in mind, and utilizes an asynchronous steady state approach which has been shown to facilitate scalability to potentially millions of compute nodes" [1]. The proposed work intends to use EXAMM for a novel purpose, thus what is uncovered in the process will be new.

References

1. Ororbia, Alexander, AbdElRahman ElSaid, and Travis Desell. "Investigating recurrent neural network memory structures using neuro-evolution." Proceedings of the genetic and evolutionary computation conference. 2019.

2. Stanley, Kenneth O., et al. "Designing neural networks through neuroevolution." Nature Machine Intelligence 1.1 (2019): 24-35.

3. Stanley, Kenneth O., and Risto Miikkulainen. "Evolving neural networks through augmenting topologies." Evolutionary computation 10.2 (2002): 99-127.

4. Rawal, Aditya, and Risto Miikkulainen. "From nodes to networks: Evolving recurrent neural networks." arXiv preprint arXiv:1803.04439 (2018).

5. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

6. Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

7. Zhou, Guo-Bing, et al. "Minimal gated unit for recurrent neural networks." International Journal of Automation and Computing 13.3 (2016): 226-234.

8. Collins, Jasmine, Jascha Sohl-Dickstein, and David Sussillo. "Capacity and trainability in recurrent neural networks." arXiv preprint arXiv:1611.09913 (2016).

9. Ororbia II, Alexander G., Tomas Mikolov, and David Reitter. "Learning simpler language models with the differential state framework." Neural computation 29.12 (2017): 3327-3352.