EECE5870 Project: Immediate Deliverable Two

Chi Nok Enoch Kan, Zachary Nordgren, Xiang Jin

November 14, 2019

Instructor: Prof. Richard Povinelli

1 Abstract

In this study, we trained a population of agents to play OpenAI's Breakout-V0 using the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. Each agent is trained on random neural network architectures and combinations of activation functions. The elite, top-scoring agents were able to score as high as 19 in a single game with just under 8 hours of training time, whereas our previous single agent trained on dueling-Deep Q Network (dDQN) scored high score of 24 with 24 hours of training. This study demonstrates the robustness and effectiveness of NeuroEvolution, specifically the NEAT algorithm in evolving optimal networks that beat performances of state-of-the-art deep reinforcement learning algorithms such as the dDQN.

2 Introduction

2.1 Evolutionary Algorithms

Evolutionary algorithms is a group of self-organizing and adaptive artificial intelligence algorithms inspired by Darwin's theory of evolution [1], which simulates biological evolution processes and mechanisms. Biological evolution is achieved through reproduction, mutation, competition, and selection. Similarly, evolutionary algorithms mainly solve optimization problems through selection, recombination, and mutation. Evolutionary algorithms are robust methods that can adapt to different optimization problems and in most cases, a satisfactory and optimal solution can be found. Evolutionary algorithms give specific encoding schemes for the entire parameter space of the problem to be solved, rather than directly processing the parameters of the problem. Instead of starting a search from a single initial point, it searches from a set of initial points. There are many kinds of evolutionary algorithms, such as genetic algorithms, genetic programming, evolutionary programming, and evolutionary strategies.

2.2 NeuroEvolution

The idea of NeuroEvolution is simple. NeuroEvolution is an artificial intelligence approach which generates artificial neural networks with evolutionary algorithms [2]. This method is often contrasted with stochastic gradient descent as means of parameter optimization. Recently, researchers

at Uber have shown that basic NeuroEvolution algorithms can often beat the deepest, state-of-the-art stochastic gradient descent algorithms in terms of optimization of model performance [3]. In addition to parameter optimization, many NeuroEvolution algorithms also optimize activation layers and model architectures (i.e. the depth and width of each layer). In NeuroEvolution, network topology is defined as the way neurons are connected in a network. One major issue that conventional NeuroEvolution assumes a fixed topology. When a topology is fixed, NeuroEvolution only changes the weight connections. This assumption is flawed in its nature since it does not allow one to scale a network based on the complexity of the problem. Therefore, there is a need for more flexible methods which allow the evolution of neural network topologies.

2.3 NEAT

NeuroEvolution of Augmenting Topologies (NEAT) is a type of genetic algorithm which directly addresses the previously mentioned deficit of NeuroEvolution [3]. Unlike conventional NeuroEvolution, NEAT evolves the entire topology of the network as well as its activation functions. In NEAT, genomes contain two genes which encode the neurons and their connections respectively. NEAT handles variational operations flawlessly by adding an identifying number to each neuron, which allows the algorithm to keep track of where each neuron comes from. Neurons that are derived from a common ancestor are matched up for crossing over. Genomes are also divided into species in order for mutation to happen. In short, NEAT has the following key steps: direct encoding of neural networks using identifying numbers, performing crossover based on the identifying numbers, performing mutations of neurons and their connections, maintaining inter-specific biodiversity, and minimizing the sizes of the selected networks.

3 Methodology

In this study, we implement the NEAT algorithm in Python to train a population of agents to play OpenAI's Atari Breakout [4]. We chose to use neat-python, an open-source Python library as the basis of our experiments. Neat-python allows easy configuration and implementation of NEAT. The library handles the population, mutation, crossover, and evaluation. There are many configuration options related to the beginning size of the initial network, the types of neurons used, and the evolution parameters. For this project a starting network size of 128 input neurons, 20 hidden neurons, and 4 output neurons are used, with a population size of 100. The activation function chosen for the model is the 'ReLu' function which is defined as max(0,x). The evaluation works by creating a Breakout-ram-v0 environment and using the raw observation string as the input to the network. The output is determined as the index of the largest output. We evaluate each network in the population three times and sum the rewards from each run to use as the overall fitness.

4 Results

The fitness of a random agent on average has a score of zero. Occasionally the random agent will hit the ball and get a point or two but nothing consistent. When our algorithm was allowed to run for 1000 generations the best single run had a score of 15 and the best combined run had a score of 23. This indicates that the model is playing the game better than the random agent. The training time is very long, each generation evaluation consists of over 200,000 frame evaluations and takes

over 7 minutes to run on a mid range CPU. This is still faster than the Dueling Deep Q Networks from our previous project iteration.

5 Conclusion and future work

This stage of the project went very well. The NEAT model was able to beat the baseline and even the dDQN model from the first project iteration. There is still lots of room for improvement. Part of the reason why the model is slow is because the network has to have so many connections with an input size of 128. An idea to reduce the feature space is to train an auto encoder on the input and then train the NEAT model on the latent space. Other future work focuses on hyper parameter optimization, and paralellization.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, NY, USA: Oxford University Press, Inc., 1996.

[2] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *CoRR*, vol. abs/1712.06567, 2017.

[3] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, pp. 99–127, June 2002.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.