Neuro-Evolution and Transfer Learning on OpenAIs Shadow Hand Environments

Submitted May 2022, for the fulfillment of
the conditions of the module: **Designing Intelligent Agents.**

Thomas Cotter
20160230

School of Computer Science
University of Nottingham

Abstract

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Neuroevolution is a practice where Deep Neural Networks (DNNs) are optimized by evolutionary algorithms, rather by the usual gradient-based learning algorithms. A paper by Such et al (2017, [1]), states that Genetic Algorithms (GAs) are a competitive alternative for training deep neural networks for reinforcement learning. Neuroevolution can be used to train the weights of a DNN. This can be tested on OpenAIs robotics environment called ShadowHands [2]. These are a set of simulated robotics environments to test reinforcement learning algorithms. There are 4 environments available for use, these can be seen in Table 1.1.

Enviroment Name	Description
HandManipulateBlock	The ShadowHand must try to rotate a block into the desired position.
HandManipulateEgg	The ShadowHand must try to rotate an egg into the desired position.
HandManipulatePen	The ShadowHand must try to orient a pen into the desired position.
HandReach	The ShadowHand must try to reach a desired position with its fingers.

Table 1.1: The OpenAI ShadowHand Environments

Furthermore, these enviroments are very similar in terms of the goal that each ShadowHand has, therefore a question arises that if we train a DNN on one of these environments, how will that DNN perform on a different ShadowHand environment? These virtual ShadowHands are based on real-life ShadowHand robots. If the hands can transfer learning across multiple different tasks, this will greatly reduce the training time required for the hands to perform different tasks, which in turn will make them more useful robots.

1.1 Aim of the Project

The aim of this project is to answer the question: **Can we transfer the learning of Deep Neural Networks trained with neuroevolution techniques on one environment to another similar environment?**. This aim can broken down into some goals which are detailed here:

- Train a DNN using a genetic algorithm to consistently solve the HandManipulateBlock task.
- Use this DNN on each of the 3 other ShadowHand enviroments.
- Measure how well the DNN performed using different loss metrics.

For this project, I will be using Python to use the OpenAI enviroments, and TensorFlow to implement the DNN.

Chapter 2

Related Work

2.1 Neuro-Evolution

Agents in reinforcement learning (RL) tasks need to transform high-dimensional sensory inputs from the environment into an action that they should take. The most logical way to do this is with deep neural networks (DNNs) due to the number of parameters that can be trained to produce an output. In recent years, hardware has improved drastically allowing for the weights of a neural network to be tuned with a genetic algorithm (GA). GAs were initially proposed by John Holland in a paper called Adaptation in Natural and Artificial Systems (1975, [3]). It is a search heuristic that is inspired by Charles Darwin's theory of natural selection. They involve 3 main processes: selection, crossover and mutation. Selection is the process of choosing the fittest (most suited to solve the task) individuals in a population and let them pass their genes to the next population. Crossover is the act of combining parent individuals into children. Mutation is when certain genes in certain offspring are mutated before being added to the next generation. These 3 processes simulate the process of natural selection. The act of using GAs to train deep neural networks is called neuroevolution.

There has been a lot of work on neuroevolution in the machine learning field. Such et al demonstrated that the weights of a DNN can be evolved with a GA and it performs well on reinforcement learning tasks (2017, [1]). In this paper, the deep neural network had over four million parameters. These results suggest that following the gradient is not always the best choice for optimizing performance in reinforcement learning tasks. To evovle the neural network truncation selection was used, in which the top T individuals become the parents of the next

generation. Gaussian mutation was also used, in which Gaussian noise was added to the parameter. Crossover was not included for simplicity in this case. With a population size of 1,000 the GA quickly converged on high performing individuals. Gomez et al (2006, [4]) also looked into the uses of neuroevolution in reinforcement learning tasks, in which the weights of the neural network were evolved to solve the Cart-Pole balancing task. Whilst this is a simple RL task, the neural network performed well. In a paper by Pawelczyk et al, titled Genetically Trained Deep Neural Networks (2018, [5]), is it stated that the DNNs trained with GAs consistenly outperformed those trained with a classical method within the same time budget. This was tested on the computer vision task of classifying the MNIST dataset, and the models weights were updated via a GA. This suggests that genetic algorithms perform better than following the gradient in certain image recognition tasks too.

2.2 Transfer Learning

Transfer Learning is an important process in all aspects of machine learning. It is commonly used in deep convolution neural networks (CNN) which are used for computer vision tasks. The weights in the first few layers a CNN trained for one task can be reused for a second task. The CNN can be used to recognise high level features such as edges in images in the second task without the need to be trained again. Transfer learning can also be used in RL, Lazaric (2013, [6]) showed that whenever the tasks are similar, the transferred knowledge can be used by a learning algorithm to solve the target task and significantly improve it's performance.

Chapter 3

Design

OpenAI Gym is a toolkit for reinforcement learning research. It includes a growing collection of benchmark problems that expose a common interface (2016, [7]). OpenAI Gym focuses on the episodic setting of reinforcement learning, where the agent's experience is broken down into a series of episodes. In each episode the agent's initial state is randomly sampled from a distribution, and the interaction proceeds until the environment reaches a terminal state. The robotics environments that are used in this project are defined using MuJoCo, a python library for advanced physisc simulation [8]. OpenAI Gym does not include a built-in agent, so this will have to be built from scratch.

The agents used to solve this task will be deep neural networks. These will be built using TensorFlow, and end-to-end open source platform for machine learning. The agents will contain initially contain 3 Dense layers, which could be altered depending on the computing power required to train this many weights. The input to the agents will be the current state of the environment and the output will be an action to take. In doing this, the agents should be able to take actions that take them towards the goal state.

The selection, crossover and mutation processes will also have to be designed. For selection, roulette wheel selection will be use, as well as elitism. Roulette wheel - also known as fitness proportionate - selection is when the higher performing agents have a greater chance to be selected but each agent always has a chance to be selected. The reason for allowing some weaker individuals to pass their genes onto the next generation is that such individuals may have some weights that could be useful in certain scenarios. Elitism is where the top N individuals in terms of fitness will be automatically passed to the next generation. Different values of N will be

experimented with to determine the most optimal value. For crossover, uniform crossover will be used. This can be performed in two ways:

- Choose each individual weight from either parent for the child.
- Choose each entire layer's weight from either parent for the child.

Due to the time constraints in this project, crossover will be performed with the entire layers weight but with more time more complex ways of crossover would be researched. Mutation is used to maintain and introduce diversity into the population. The idea would be to select a random weight, and perform a mutation operation to this weight. The weight could be replaced a random value (within certain constraints), the weight could be changed by some percentage or the sign of the weight could be changed. These 3 will be experimented with in order to find the most optimal process for mutated the weights.

The population should converge on weights that solve the HandManipulateBlock tasks consistently. Then, these networks can be taken and tested on the other 3 tasks. In order to test the transfer learning, we would need to see how many steps it takes the networks to solve each task or if they reach a certain number of steps without solving the task at all. What also can be tested is if we continue evolving these pre-trained neural networks for a N generations (where N is a very small number), do they reach the goal in a fewer number of steps than untrained neural networks trained for N generations.

Chapter 4

Implementation

Chapter 5

Evaluation

Chapter 6

Summary and Reflections

Bibliography

[1] 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," 2017.

[2] OpenAI, "Ingredients for robotics research." `https://openai.com/blog/ingredients-for-robotics-research/`. [Online; Accessed 19-04-2022].

[3] J. Holland, "Adaptation in natural and artificial systems," 1975.

[4] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Efficient non-linear control through neuroevolution," pp. 654–662, 2006.

[5] K. Pawelczyk, M. Kawulok, and J. Nalepa, "Genetically-trained deep neural networks," 2018.

[6] A. Lazaric, "Transfer in reinforcement learning: a framework and a survey," 2013.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.

[8] DeepMind, "Mujoco." `https://mujoco.org/`. [Online; Accessed 20-04-22].