Project report  
27-06-2017  
Natural Computing

*Tackling AI and control theory problems using Deep Reinforcement Learning*

* *Eastern Screech Owls*

Jordi Beernink, Roel Bouman and Jeffrey Luppes

Contents

[Introduction 2](#_Toc483403008)

[Deep Reinforcement Learning 2](#_Toc483403009)

[Goal 2](#_Toc483403010)

[Experiment Setup 2](#_Toc483403011)

[A3C-Model 2](#_Toc483403012)

[Proposed model 2](#_Toc483403013)

[Results 3](#_Toc483403014)

[Conclusion 3](#_Toc483403015)

[Discussion 3](#_Toc483403016)

[References 3](#_Toc483403017)

[Appendix A: Screenshots of system behavior 4](#_Toc483403018)

[Appendix B: Algorithm 4](#_Toc483403019)

[Appendix C: Contributions and evaluation of project 4](#_Toc483403020)

# Introduction

In recent years, deep learning has successfully become a staple method in various domains, such as object recognition with visual attention, Computer Aided Diagnosis and Detection in medical imaging (CADe), and many other regression and classification problems. More recently, deep learning has been combined with reinforcement learning resulting in the creation of the new field of Deep Reinforcement Learning [1,2]. This newly developed method combines Q-learning[3], a Reinforcement Learning technique, with Deep Neural Networks, and was trained to play classic Atari 2600 games. This trained Deep Q-network, as the new Reinforcement Learning agent is called, was able to surpass human performance across several of 49 different games. More recent applications involve high-dimensional robot control, solving physics-based control problems, and the playing of the traditional Go game.[4]

On the 27th of April 2016, a large scale platform for Reinforcement Learning called OpenAI Gym[5] was launched. OpenAI gym provides a Python environment for the fair comparison of Reinforcement Learning techniques. The platform supports many classical problems (‘Environments’) from Reinforcement Learning theory, such as balancing a pole atop a mine cart, balancing a pendulum from a random starting position, and driving a mine cart up a hill from a local minimum using momentum. More advanced problems are also presented, such as several of the classic Atari 2600 games, several games from the PyGame platform, and the classic shooter DOOM.

Deep Q-Learning methods, and other Deep Reinforcement Learning techniques, have already proven to be highly applicable to several of the problems represented on the OpenAI Gym platform.

In this project report the goal is (briefly) to test a wide range of approaches on the OpenAI environments. The goal is further explained and formalized in the section dedicated to the goal statement. First though a thorough review of the available literature is given aiming to delve deeper into the available material and to sketch out the related work that has been done by others.

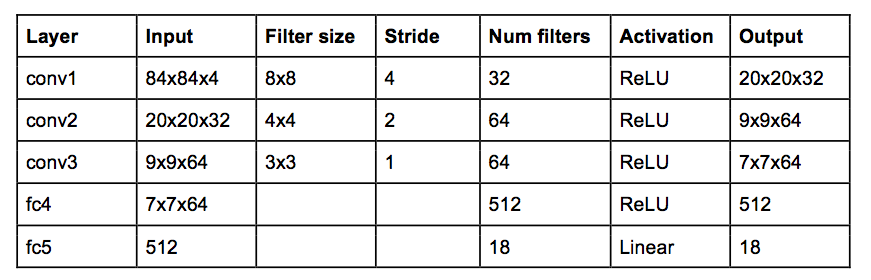
The environment used for testing and evaluating is explained in the section ‘Environment’. Following this train of thought, the models and algorithms are explained in detail. Finally, the results are listed and discussed. The last section of this report is dedicated to making a comprehensive survey of future possible work, as the platform is still far from mature and a lot of possibilities are shining through.. if one would crawl through the woodwork of dependencies.

# Literature Review

\*Roel\*

# Deep Reinforcement Learning

Several methods are applicable to these problems, and most will be incorporated in the Deep Reinforcement Pipeline. Most methods revolve around the neural network architecture, which can differ greatly between problems. The Google subsidiary DeepMind has achieved great performance with their Q-networks[1,2], although some elaboration on their structure is needed. This structure should form an adequate foundation for later exploration for improvements on network structure for more specific problems. To illustrate, the network architecture DeepMind used can be found in table 1.

*Table 1: The Q-network architecture used by DeepMind on the Atari 2600 games.*

It can be seen that the network has several convolutional layers, each of which takes an input of 84x84x4, which consists of four 84x84 grayscale images. These four images are the last four observed images from the game. Using the last four images allows for the abstraction of movement as features. As such, using multiple frames will be thoroughly experimented with in this project.

Of course, parameters such as the filter size, the stride, the number of filters, and the activation function can and should be examined for different problems. In addition, the number of layers might be changed, as adding layers might improve results. It should be noted that although larger networks might improve the performance, the network complexity is severely limited by the available hardware.

In cases like DOOM, the general structure of the network will be evaluated across different challenges, as to assess the general applicability of the trained Deep Q-Network.

The most notable of the Deep Reinforcement techniques to be used will be the well-known Deep Q-Network.[1,2] Deep Q-Networks try to estimate the state and action dependent Q-function using a Neural Network. Using a well-designed network, the long and short terms rewards for a given action sequence can be estimated, which facilitates the quick search for a decent solution, something which is not possible in large scale problems such as Go, Atari 2600 games, and DOOM, or non-deterministic problems. The Deep Q-Network presented by DeepMind also uses experience replay with random mini-batches from replay memory to prevent overfitting and convergence towards a local maximum reward. Another important addition is the exploration algorithm used. Their ε-greedy exploration chooses a random action with probability ε, instead of choosing the optimal action according to the highest Q-value. By decreasing ε from 1 to a lower value, exploration is initially encouraged, but used less and less as more of the training space is explored. Depending on the problem, the value and change rate of ε should be optimized.

In recent years several improvements have been made upon the staple Deep Q-Network technique. One of these is the Double Deep Q-Network, which combines Deep Reinforcement Learning with Double Q-learning[6]. In the Double Q-learning variant, the action selection is decoupled from the evaluation, which leads to less overoptimism due to estimation errors. Double Deep Q Networks will be experimented with, and it remains to be seen whether they significantly increase performs.

The architecture proposed in Double Deep Q-Networks was further extended into an architecture where the state value function and the state-dependent action advantage are completely decoupled.[7]. This procedure leads to the development of a more robust agent, which in turn leads to a better policy evaluation when many actions are similarly valued, which was shown to be a weakness of Deep Q Networks. This new architecture has proven to be effective, and improved upon the Atari 2600 state-of-the-art, and should yield decent results when applied to the OpenAI Gym challenges. It must be noted that the required computation time should be extensively evaluated for such a different architecture, and whether possible increases are worth the performance.

For the development of the architecture, the well-developed Keras library[8] for the Python language can be used. Keras is a library developed for rapid prototyping of Neural Network applications, and it offers an extra layer of abstraction over either the Theano or TensorFlow libraries. Many different Neural Network structures can be tested using the Keras library, without the need to build the architecture from the ground up.

The inclusion of Deep Reinforcement Learning into the process is done using the Keras-rl library[9], which is an extension upon Keras focusing on Reinforcement Learning. It has several Reinforcement Learning techniques available, and can easily be extended. In addition, the Keras-rl library has built-in support for the OpenAI Gym platform, which was elaborated on earlier, and can thus easily be used on various challenges and easy implementation of a reward system.

# Goal

Control theory problems

Solving missions from the OpenAi gym (elements that need to be taken into account)

Focus on specific elements of Q-Learning that can help on the control theory problems

# Environment and evaluation

\*Jordi\* - Dit was het testing procedures/Environment hoofdstuk. Het leek me beter om deze te hernoemen naar dit format.

# Models and Algorithms

\*method.. beschrijf de algoritmen\* [Allen] KIJK EVEN HOEVERRE DIT VERSCHILT VAN ROEL’S THEORIE. WAARSCHIJNLIJK HOEF JE HIER ALLEEN MAAR EVEN DE Q-LEARNING OPZET UIT TE LICHTEN

# Results

\*resultaten\* [Allen]

# Discussion

\*statistische discussion van de resultaten\* [Allen]

//todo: kan pas als alles binnen iss

# Conclusion

//algemeen: todo: [Allen]

Model Limitations

//todo

Method limitation

Many limitations were found in the software used for this project as only limited support is available. For instance, it was not possible to use GPU (e.g. CUDNN) acceleration in any of the systems tried. Another set-up involving Ubuntu found it was impossible to use the Tensorflow library. Some of the libraries used depended on various closed-source libraries, which were not available for 64-bit distributions, and resorting to 32-bit caused many more issues down the line.

Future work

A lot of work is currently being done developing the frameworks used in this report. At the moment OpenAI is largely experimental and with no guarantees given, but a lot of work is being done to mature the platform. As mentioned before Windows is still without any support, but the developers are working on making it more accessible. If this development were to be made it would be huge, as the Gym platform would instantly be accessible to many fledgling data scientists and AI enthusiasts alike. At the moment the software, irregardless of whether development takes place on Ubuntu or Windows, requires the installation of tens of plugins, packages, specialized compilers, modules and code. The installation is far from trivial and realistically, in its current state very much unfit for a classroom setting. If the developers and the community (the vast majority of fixes came from community gists and pull requests) were able to address the many RAM leak issues, dependency mismatches and the overall lack of documentation, this software could see widespread adaption and popularity.

Our own contribution towards this goal is to publish several write-ups regarding the use of Windows and Ubuntu for the platform and how to troubleshoot the sometimes peculiar bugs. The platform looks impressive, but is still very much academic code written by top minds, who- while being the top of their field- are often too busy to refactor and maintain the code base.

We would like to focus on extending our work to the Doom game. A few experimental environments are being trained upon, but the incredibly large action space make it a very challenging environment to tackle with reinforcement learning.

At the moment the platform supports very simple playgrounds: Atari games, the Minecraft game (which is still very experimental), and Doom amongst them. What if we could train an agent in a more advanced open world setting, e.g. letting it discover, in real-time, games such as World of Warcraft? The various classes and limited game resources would provide for many interesting interactions.

Additionally, we plan to contribute to this development if we can with our knowledge and skill-set, by making contributions to the community at large.

# References

1. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.
2. Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).
3. Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8.3-4 (1992): 279-292.
4. Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.
5. Brockman, Greg, et al. "OpenAI gym." *arXiv preprint arXiv:1606.01540* (2016).
6. Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." *AAAI*. 2016.
7. Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581* (2015).
8. Chollet, François. "Keras." (2015).
9. Plappert, Matthias. "keras-rl" (2016).

# Appendix A: Screenshots of system behavior

# Appendix B: Algorithm Code

# Appendix C: Team Contributions and evaluation of project

|  |  |  |  |
| --- | --- | --- | --- |
|  | Roel | Jeffrey | Jordi |
| Organisatie |  |  |  |
| Teamwerk |  |  |  |
|  |  |  |  |
| Project |  |  |  |
| Kennis |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |