Cooperation and Competition based Mechanisms for Multi-Agent Self Play and Transfer Learning.

Mid Sem Evaluation

—

Shiv Kumar, 2016UIT2563

Required Background

- Markov Games
- Value Based Methods
- DQN
- Double DQN
- Policy Gradient Based Methods
- Actor Critic Methods
- Deep Deterministic Policy Gradients
- Proximal Policy Gradients with General Advantage Estimation

Problem Statement

- Understanding the mechanisms of cooperation and competition in multi agent settings in Deep Reinforcement Learning.

- Studying how to transfer knowledge from one domain and environment to another rather than having to learn in each environment from scratch every time using better generalisation techniques such as environment randomisation, opponent sampling (in policy gradient methods) and common regularisation techniques.

- Making the training of Deep Neural Networks in Reinforcement Learning more time and space efficient using techniques such as pruning and quantisation.

Problem Statement

- Gaining better insight into the workings of state of the art Reinforcement Algorithms like Proximal Policy Optimisation using General Advantage Estimation (PPO viz GAE), Double Deep Q Network, Rainbow Deep Q Network, Deep Deterministic Policy Gradient (DDPG) and Multi Agent Deep Deterministic Policy Gradient (MADDPG), so as to understand how activation functions and standard regularisation techniques affect their generalisation mechanisms so as to combine all favourable conditions to supplement a strong learning environment.

- Development of open sourced environments with Openai Gym support to augment the research in Deep Reinforcement Learning created using game development frameworks in Python like PyGame.

Motivation

Till now the neural nets employed in Deep RL are known to be not so deep. This is because of the over fitting seen in Deep RL wherein agents over fit over the underlying environment and hence face difficulty in generalising to other similar environments which humans do with ease. This study deals with how this overfitting can be reduced allowing for deeper neural networks to be incorporated into the general paradigm of Deep RL. Furthermore this project deals with how knowledge gained while learning in one environment can be used to interact efficiently in other environments, which can allow for more practical and efficient use of Deep RL. Furthermore the training of Deep RL networks are known to be very slow. This study also tries to briefly understand as to how the training can be made more efficient in general through techniques like quantisation and pruning.

Some Important Applications

1. Traffic Management System
2. Swarm of Autonomous Drones for Safer Delivery and Emergency Assistance Systems.
3. Safe Movement for Self Driving Cars
4. To compare the techniques used by RL agents and real humans for a given circumstance or environment.

Methodology

Development of game environments in PyGame to support multi agent behaviour.

Development of Openai gym wrappers for the developed games to ease interfacing with algorithms.

Baseline Implementation of Double DQN, DDPG for single agent, DDPG for symmetrical multi agents, PPO with GAE, and MADDPG for unsymmetrical multi agent environments.

Methodology

Training the agents using the implemented algorithms

Transfer learning on other environments

Environment Randomisation and Ensemble Policies

ELU vs ReLU + Batch Normalisation

Quantisation and other efficiency aspects of the training and inference phase

Development of game environments in PyGame to support multi agent behavior and Development of Openai gym wrappers for the developed games to ease interfacing with algorithms

The prime objective of the project is to be able to develop environments suitable and appropriate to the study of Multi agent reinforcement learning. These include games like the snakes game, robber-police game, the proximity tracking game, among many others. These games will be implemented in a python framework called Pygame.

The developed games will then be wrapped in appropriate Openai environments. This brings in a level of abstraction which allows the programmer to focus on one aspect of the process at a time. The Openai environment allows a perfect simulation of an environment for the agent and hence makes implementation and debugging of the codes much easier. It separates the implementation of algorithms and environments which is much needed for effective development of the codes related to the research.

Baseline Implementation of Double DQN, DDPG for single agent, DDPG for symmetrical multi agents, PPO with GAE, and MADDPG for unsymmetrical multi agent environments to achieve benchmark results on standard environments and training the agents using the implemented algorithms.

It is imperative to check the correctness of the set of algorithms on standard benchmarks. Once the standard results are met and requisite experiments are done for the environments, they will be ported to our environments to check the behaviours of agents under different circumstances ranging from being alone to having many different agents in the same environment.

The algorithms are benchmarked as per the following environments :

1. Double DQN : Navigation (Unity) (Completed)
2. Distributed DDPG : Continuous Control (Unity) (Completed)
3. Symmetrical DDPG : Tennis (Unity) (Completed)
4. PPO with GAE : Dexterity (MuJoCo) (Under process)
5. MADDPG : Football (Unity) (Under process)

Transfer learning on other environments, Environment Randomisation and Ensemble Policies, and ELU vs ReLU

Transfer Learning is the main premise of this project. Transfer Learning deals with being able to use the knowledge gained in one domain to other domains. To enable efficient transfer learning techniques like environment randomisation and ensemble policies will be employed. A novel technique proposed in this project is randomised environment sampling. A study of the effects of Exponential Linear Unit and Rectified Linear Unit will be done throughout the project as has been done for the implementations done so far.

Quantisation and other efficiency aspects of the training and inference phase

The training and inference of Deep RL models is known to be extremely slow and extremely heavy once the models are trained. There is inherent latency due to the deep neural network based architecture and then come the RL based factors. Quantisation and Pruning can be used to handle the neural network side of things. The effects of these techniques will be checked on neural networks employed in Deep Reinforcement Learning.

Work done till now

- Gained understanding of the field of Reinforcement Learning and went through multiple standard papers in the related literature. These papers include the DQN paper on Atari Games, the Rainbow DQN, the Proximal Policy Optimisation paper by Openai, the Actor Critic Paper, the Deep Deterministic Policy Gradient Paper, the Multi Agent Deep Deterministic Policy Gradient Paper. Frameworks like Openai Gym, pyTorch and Tensorflow were also learnt in the process.

- Learnt PyGame and implemented basic snake game. Wrapped versions of the same are in the process of implementation.

- Implemented DQN, Double DQN, DDPG on single agent environments, DDPG on symmetrical multi agent environments and achieved the required benchmarks scores on standard environments from NVIDIA.

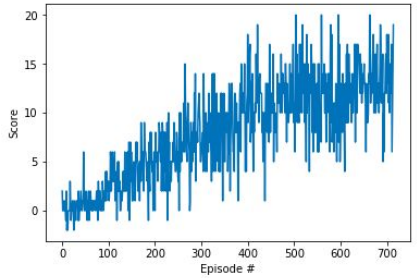
- Studied and compared the effects of ELU and pair of ReLU + Batch Normalisation on the implemented baselines of the project.

Work done till now

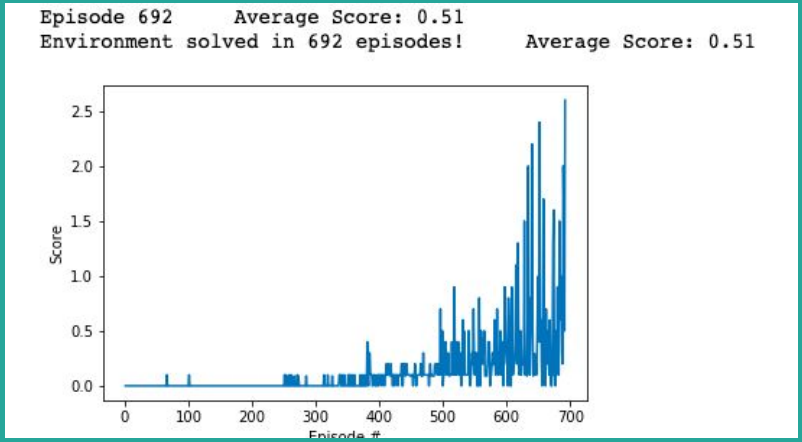
- Regularisation techniques like CutOut and Shake Shake were studied, understood and implemented in Deep Neural Networks.
- Post Training Quantisation and Quant Aware Quantisation were implemented for Deep Neural Networks in PyTorch.
- Other Neural Network Efficiency Improvement Techniques like Depthwise Convolution were studied and will be used for more efficient representation learning in the networks of the project.
- A technique to improve the robustness of the learning process was also formalised, which is being called random environment sampling.

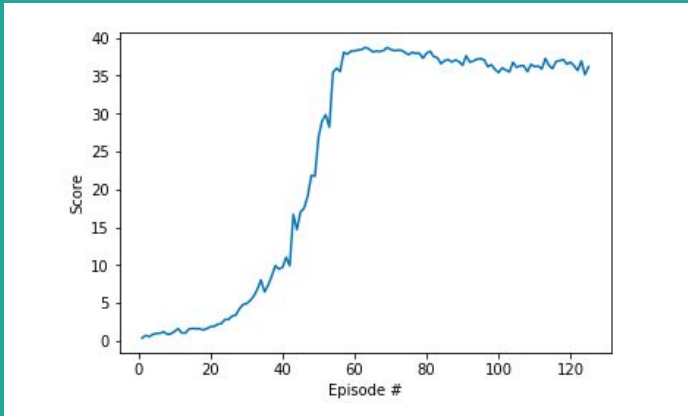
Related Screenshots and Graphs

```
Episode 100     Average Score: 0.78
Episode 200     Average Score: 3.73
Episode 300     Average Score: 6.39
Episode 400     Average Score: 8.38
Episode 500     Average Score: 11.18
Episode 600     Average Score: 11.67
Episode 700     Average Score: 12.53
Episode 715     Average Score: 13.04
Environment solved in 715 episodes!     Average Score: 13.04
```



Related Screenshots and Graphs



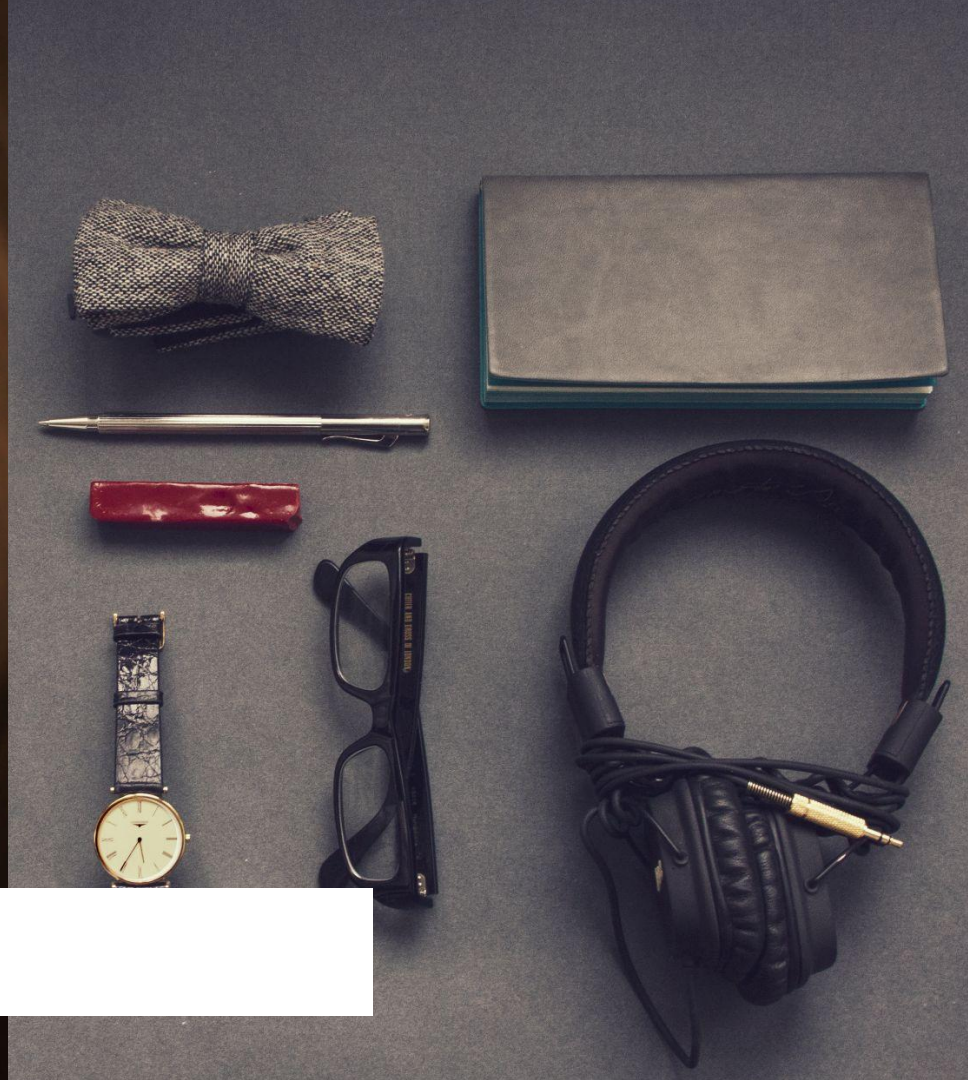


Random Environment Sampling

A technique called Random Environment Sampling is proposed in this project wherein, the agent does not learn in one single environment at a time, rather the environment with which the agent interacts with will be picked at random from the ones being offered. This along with other randomised environment parameters, and randomised opponent sampling methods should give better results in being able to generalise to newer and different environments using the knowledge gained in one environment to do better interaction in another environment. Factors like stability in learning, and introduction of different hyper parameters for these randomisation will have to worked on carefully.

Signing off

The work done so far has been very promising and enjoyable. At the start, I had absolutely zero knowledge of the field of Reinforcement Learning, and the journey of the past two months has been brilliant. A key element of research is the uncertainty whether something will work or not, and that uncertainty is considerably high in the field of Reinforcement Learning. Keeping that in mind, I hope that some strides can be made in the field of Reinforcement Learning through this small effort.



Thankyou. :)