Reinforcement Learning
Final Project
Progress Report

Jan-Henrik Lambrechts & Lawrence Kurowski

30 May 2020

1. Proposal Abstract

It is well-known that state-of-the-art Deep Reinforcement Learning (DRL) suffers from a reproducability crisis [1, 6, 7]. Recent work has found DRL methods to be brittle to random seeds [6], optimizers [7] and codebase-level optimizations [1], making them notoriously hard to benchmark. While the sensitivity of DRL algorithms regarding optimizers and random seeds do make practical DRL more cumbersome, it does not nearly pose the same danger to reproducability as codebase-level optimizations. More specifically, Engstrom *et al.* found that little 'tricks' such as gradient clipping, reward normalization and state normalization - that are not mentioned in the paper - can lead to strikingly different reward schemes and even change the inner workings of DRL algorithms [1]. While the aforementioned observation instilled a lot of interest, reviewers of the work are concerned about the scope and scale of experimental results [8]. In this project we examine the supposedly impactful implementation augmentations suggested in [1] to the Proximal Policy Optimization algorithm[2] applied to train an agent playing the game of football [3]. We are convinced that extending the body of experimental work regarding this issue in both scale and scope will contribute to meaningfully to the scientific knowledge regarding this issue and the DRL community as a whole, especially given the weight of these claims.

2. Methods

As a starting point we select the PPO (proximal policy optimization) model created by OpenAI[2]. PPO is a RL model widely used due to its relative simplicity and efficiency compared to other approaches [2]. While the PPO framework is widely used, it is also notoriously difficult to train, especially in an environment as complex as Google Football. Engstrom et. al. [1] analyzed how fine-tuning the PPO code can aid making this framework more efficient to train. They tested their approach on a number of tasks, including continuous-space tasks such as humanoid running, as well as discrete-space tasks such as the game of Atari.

In this work, we will consider the considerably more challenging and relatively new environment Google Football, with two teams of 11 players, where the PPO optimizes a single agent (a single player). The environment is difficult to train due to action space size as well as multi-player complexity.

The baseline paper [1] provides examples trained on 3 random seeds which had been previously pointed out to be a potential weak point of this work.[1] We will thus aim to train the model for 10 seeds on a selection of code augmentations proposed in this paper.

2.a. Environment introduction

We will implement the model on the Google Football environment [3]. The environment contains a number of different gameplay scenarios, such as “empty goal”, “3 vs 1 with keeper” etc. which simulate possible scenarios playing out during a game of football. In this paper we will focus our attention on the “11 vs 11” scenario which simulates the entire game.

The state space in this game corresponds to a tuple representing positions on the field of all the players, as well as that f the ball. The “goal” results in score +1.

The action space corresponds to agents’ movements (“up”, “right”, “up-right”, etc.)

2.b. Model summary

We will base our PPO implementation on OpenAI’s PPO2 baseline [4].

Proximal Policy Optimization replaces second-order optimization with first-order gradient descent-style optimization algorithm by optimizing a surrogate objective function which then defines the bounds for the actual objective.

The implementation “trick” examined in [1] concerns implementation parameter tuning independent of the algorithm itself. The implementation details examined include e.g. reward normalization, Adam annealing and network initialization. The paper also examine the effect of clipping effect (clipping refers to clipping of the probability ratio in the objective expectation that we are optimizing at the heart of the PPO algorithm.)

2.c. A note on results presentation format

In our work we follow the approach in [1]. We consider several code-level implementation details of the PPO algorithm, and run the algorithm on the gfootball environment recording rewards. We plot the rewards with the optimization trick turned on / off and compare the proportion of the times we obtained the given reward, which we refer to as CDF following notation in [1]. For reference, see figure **??** taken from [1]. In this work we will follow similar presentation style.

[1]See the reviewers’ comments available at `https://openreview.net/forum?id=r1etN1rtPB`

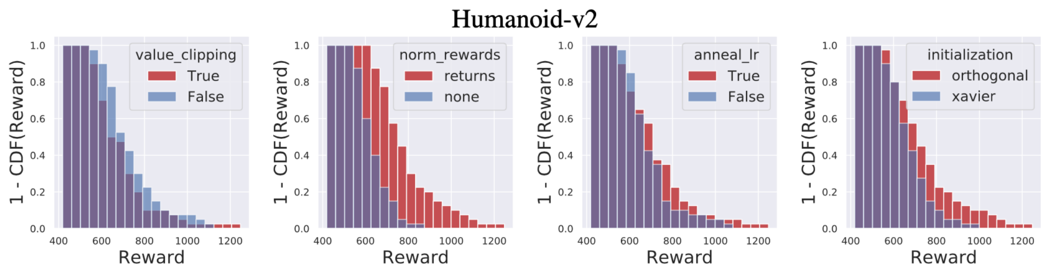

Figure 1: Example of the results of the study in [1]. The horizontal axis is the reward obtained, and the vertical is the cumulative proportion of time when the give reward was obtained at testing. The histograms are partitioned for when the given “trick” is turned on / off.

3. Initial results

3.a. Hyperparameter setup

Hereafter, unless stated otherwise we will keep all hyperparameters of the model as proposed in Google’s repository [5] and OpenAI’s repository [4]. We assume these values have been picked to optimize the algorithm’s performance for the `gfootball` environment.

3.b. Value clipping

We examine the impact of value function clipping on the training performance, as measure by cumulative rewards.

[ ]

References

[1] Engstrom L., et.al., “Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO” (2019) ICLR

[2] Schulman et.al., “Proximal Policy Optimization Algorithms” (2017), arXiv:1707.06347

[3] Kurach et.al., “Introducing Google Research Football: A Novel Reinforcement Learning Environment” (2019) Google AI Blog, available at `ai.googleblog.com/2019/06/introducing-google-research-football.html` (accessed 2020-5-20)

[4] PPO2 baseline by OpenAI, repository available at: `github.com/openai/baselines/blob/master/baselines/ppo2`

[5] `gfootball` baseline repository by Google, available at: `github.com/google-research/football/blob/master/gfootball/examples`

[6] Henderson et.al., ’Deep Reinforcement Learning that Matters’ (2018) AAAI

[7] Henderson et.al., ’Where Did My Optimum Go?: An Empirical Analysis of Gradient Descent Optimization in Policy Gradient Methods’ (EWRL14)

[8] ICLR Review Committee, 'Review: Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO' https://openreview.net/forum?id=r1etN1rtPB

A. Hyperparameter setup

The hyperparameter values presented in table 1 are based on [5] and [4].

timestep per iteration	2×10^6
number of environment steps / epoch	128
minibatches / epoch	8
LR	0.00008
entropy coeff.	0.01
discount factor	0.993
clip range	0.27
max. gradient norm (clipping)	0.5
GAE discount*	0.95

Table 1: Hyperparameters used in this paper, as per [5] and * [4]