Atari, 1976
Nolan Bushnell
Steve Wozniak
Steve Bristow
42 TTL

breakout

Deepmind, 2013
Playing Atari with Deep Reinforcement Learning
CNN + FC
publication

Abraham Alcaina

Omar Brid

Bernat Martinez

Atari Breakout

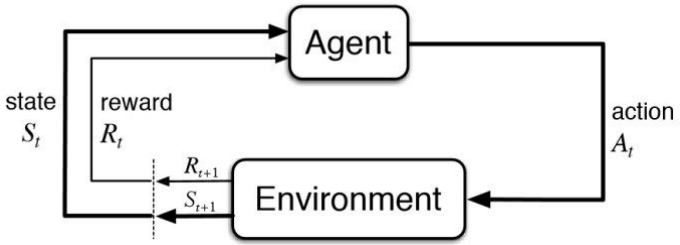
repository

https://github.com/DaniFojo/aidl-2020-spring-team-dani

Motivation of the project

What is Reinforcement Learning?

In Reinforcement Learning we have an agent in an unknown environment that can obtain some rewards by interacting with it. The agent ought to take actions so as to maximize cumulative rewards.



Why Reinforcement Learning?

Increasingly used in the industry

Rapidly growing learning resources

We believe it has a bright future

What did we achieve?

Vanilla Policy Gradient

Advantage Actor Critic

Policy Proximal Optimization

The Atari Challenge

Why Atari and Breakout?

It’s complexity was a reasonable challenge

Was part of Deepmind breakthrough with RL

Sparked Wozniak and Jobs onto grander things

Playing Atari with Deep Reinforcement Learning, *Deepmind, 2013*

Milestones

Goals presented in the Critical Review

RL algorithms

- ✓ Vanilla Policy Gradients
- ✓ Advantage Actor Critic (A2C)

Implement PPO

- ✓ Comparably to state-of-the-art approaches in Reinforcement Learning

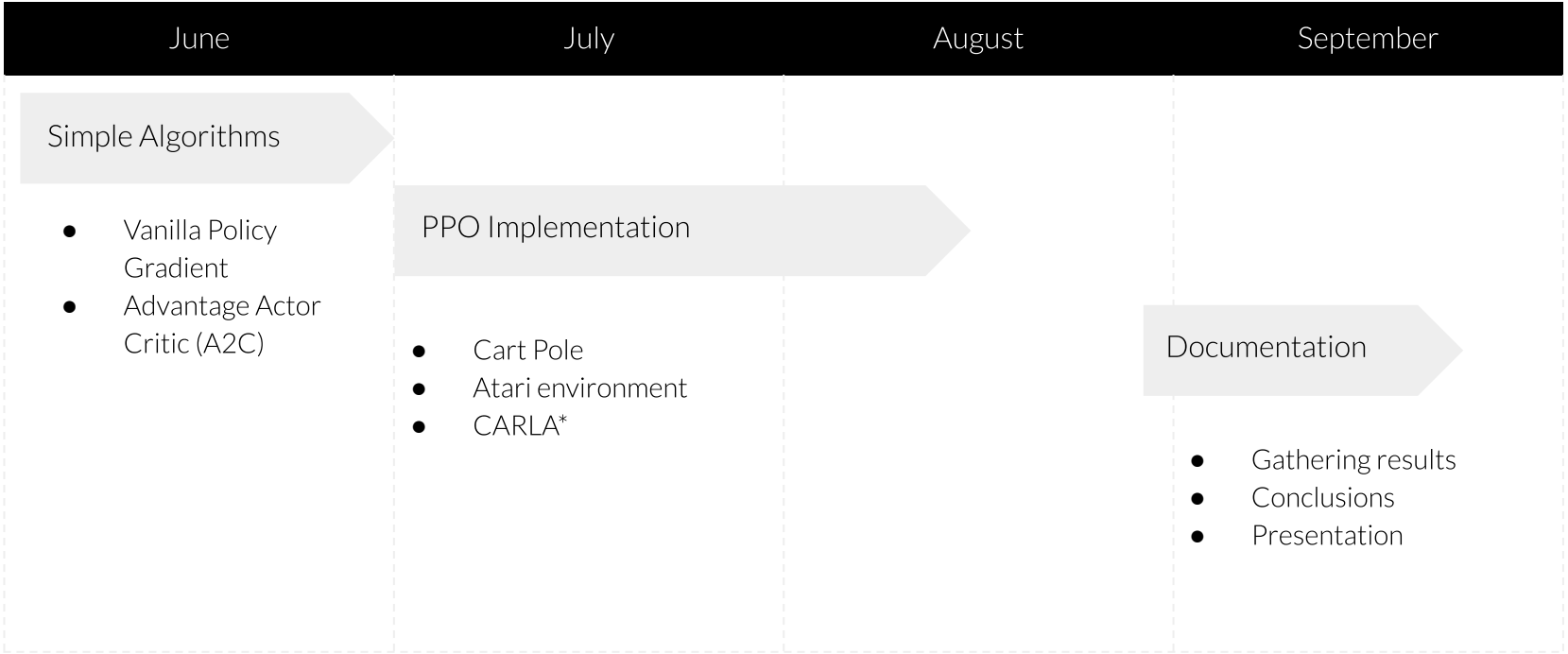
Solve environments

Use PPO to solve some of the following environments:

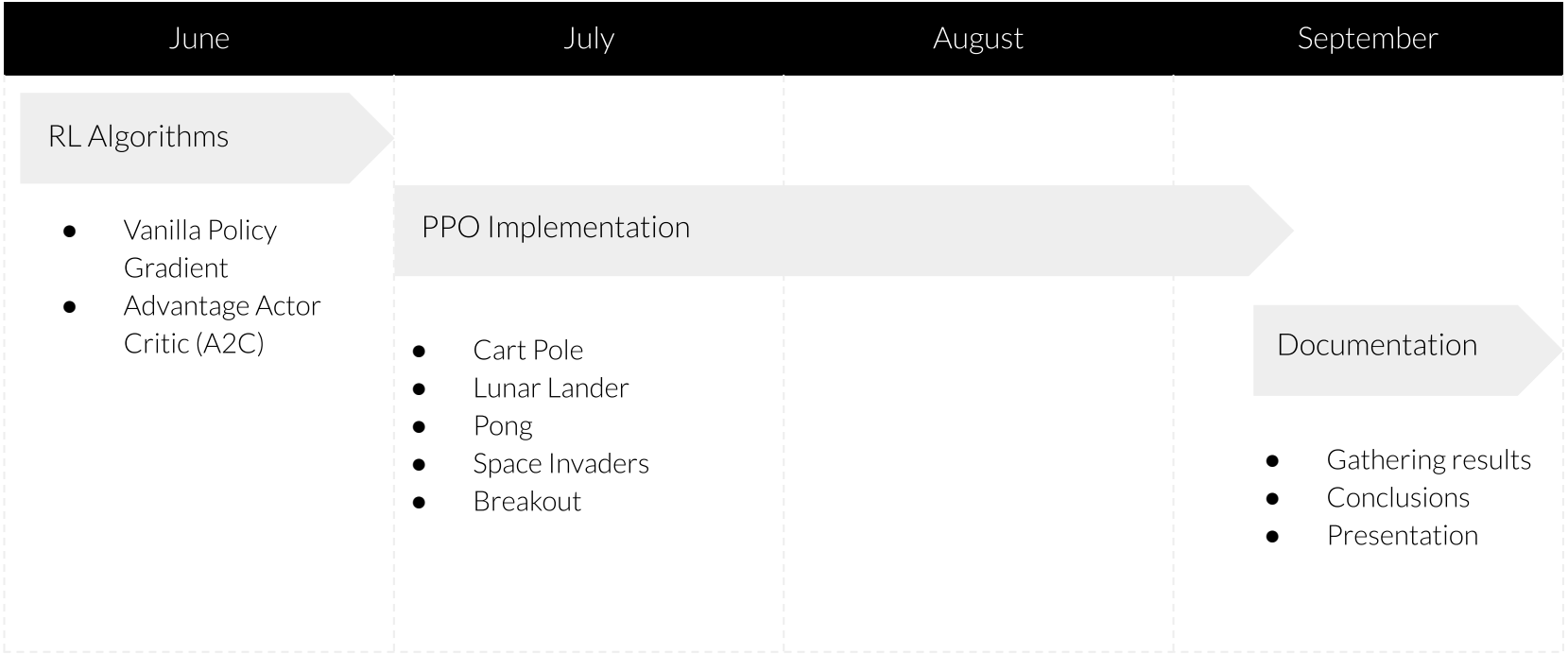
- ✓ OpenAI gym
- ✓ Some Atari game
- ✗ CARLA*

Project plan

Project Plan from the Critical Review



Real Project Plan Execution



Our proposed solution

The Tools

PyTorch

OpenAI Gym

Tensorboard

GPU, Quadro T1000

Vanilla Policy Gradient

Key Idea

probabilities of actions

that lead to higher return

probabilities of actions

that lead to lower return

Exploration

probabilities of actions

that lead to higher return

VPG explores by sampling actions according to the latest version of its stochastic policy.

Over the course of training, the policy typically becomes progressively less random, as the update rule encourages it to exploit rewards that it has already found.

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right]$$

Where A is the advantage function for the current policy, π is the policy, 'a' is the action and 's' is the state

Cart Pole

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track.

The system is controlled by applying a force of +1 or -1 to the cart.

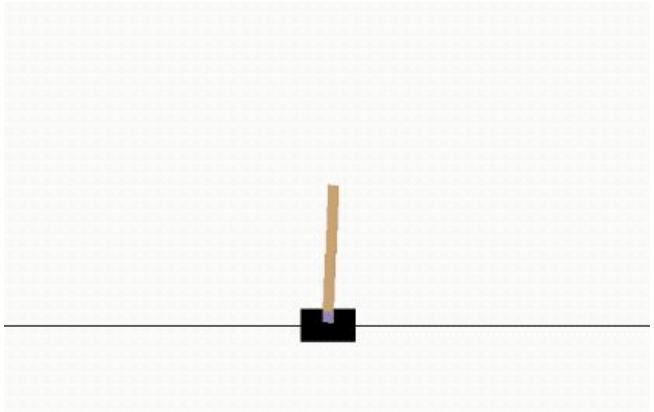
The pendulum starts upright, and the goal is to prevent it from falling over.

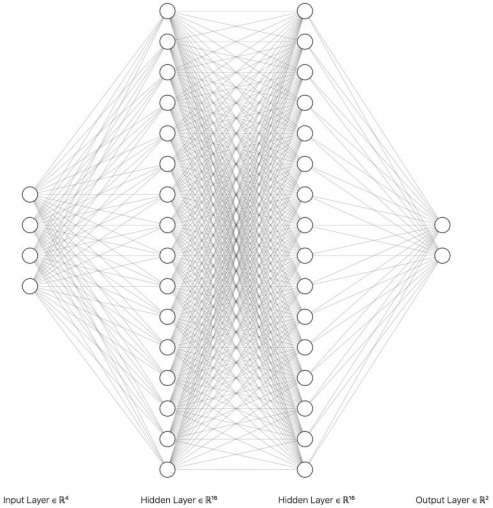
A reward of +1 is provided for every timestep that the pole remains upright.

The episode ends when the pole is more than 15 degrees from vertical,
or the cart moves more than 2.4 units from the center.

Vanilla Policy Gradient

CartPole-v1





Advantage Actor Critic

Actor Critic uses two neural networks.

The actor is is a policy function that controls how our agent acts.

The critic is a value function that measures how good these actions are.

Both Actor and Critic run in parallel.

As we have two models that need to be trained, we have two set of weights.

VPG: monte-carlo (take random samples). This introduces high variability in probabilities and cumulative rewards. This leads to noisy gradients, and cause unstable learning.

In Advantage Actor Critic (A2C) we introduce an advantage function, which will tell us the improvement compared to the average the action taken at that state is.

The Advantage functions is as follows: **$A\pi(s,a)=Q\pi(s,a)-V\pi(s)$**

One way to reduce variance and increase stability is subtracting the cumulative reward by a baseline

Recall the new update equation, replacing the discounted cumulative award from vanilla policy gradients with the Advantage function:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right]$$

On each learning step, we update both the Actor parameter and the Critic parameter.

Policy Proximal Optimization

In many Policy Gradient Methods, policy updates are unstable because of larger step size, which leads to bad policy updates and when this new bad policy is used for learning, it leads to further worse policy.

Many learning methods learn from current experience and discard the experiences after gradient updates.

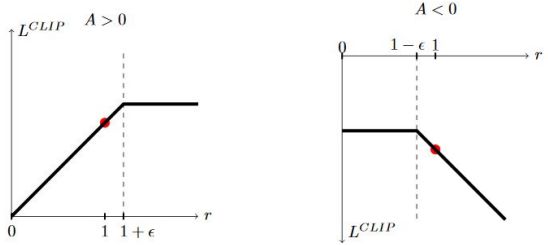
PPO overcomes those issues.

Instead of the log of current policy, PPO takes the ratio of current policy and old policy.

$$\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right].$$

Clipping the ratio and take the minimum of the two (clipped and unclipped).

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$



Data ingestion

Previous models used the RAM version of AI Gym which provided a featured representation of the game state.

As we are using the raw pixels input with PPO, we do some pre-processing:

Images are transformed to grayscale, so we only use 1 channel instead of 3 (RGB) in the CNN

Images are cropped (remove top score) and resized to 50% size,
which is enough for the Atari games and will reduce computational needs.

Images are stacked in blocks of 4 (C3D) so we can get more context (like the direction and speed of the ball).

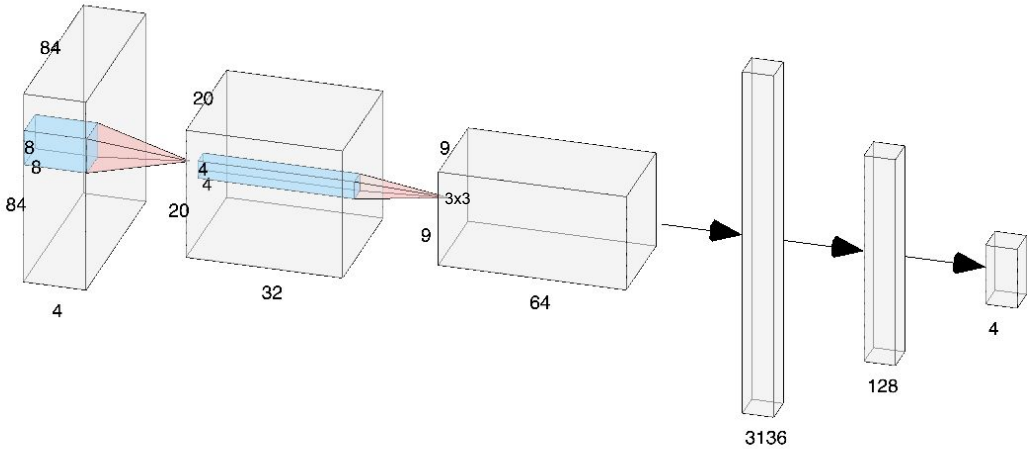
Training helpers

We used a multi-worker approach, so we could have multiple threads running on multiple cores training the model.

Most games were trained with 12 workers.

Models

PPO for image architecture



Policy Proximal Optimization with A2C

We implemented PPO with Advantage Actor Critic using :

Computed Value function:

$$V'(S_t) = R_{t+1} + \gamma V(S_{t+1})$$

Generalized Advantage Estimation (GAE):

$$\hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n-1} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_\phi^\pi(\mathbf{s}_t) + \gamma^n \hat{V}_\phi^\pi(\mathbf{s}_{t+n})$$

Results

The goal of this project is to solve Atari games from OpenAI Gym applying RL.

We first implemented VPG, then A2C and finally PPO in order to understand how RL algorithms work.

All the documentation, evidences, experiments and instructions to run them can be found here:

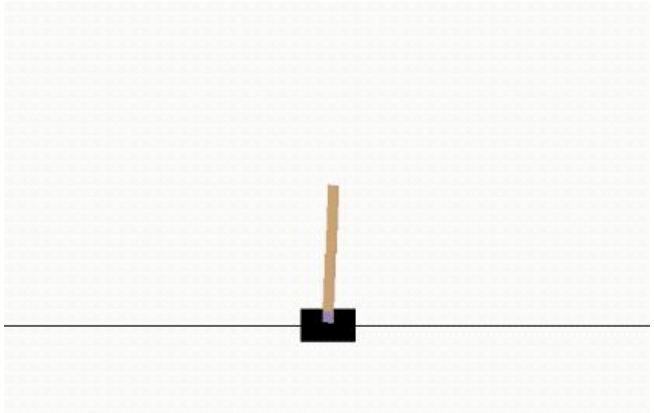
https://github.com/DaniFojo/aidl-2020-spring-team-dani

Experiments

Policy Proximal Optimization - RAM

Cart Pole

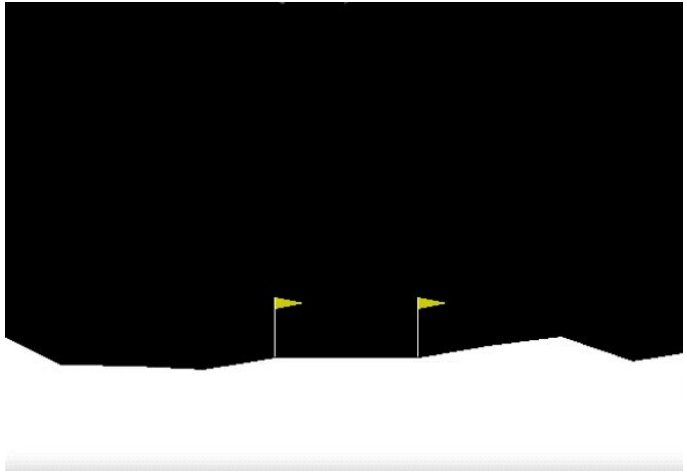
Implementation

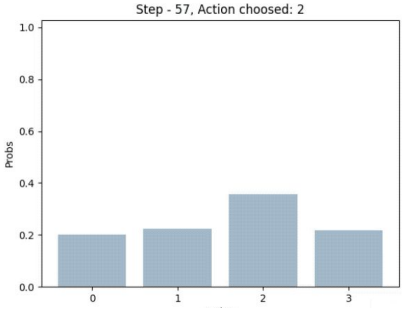


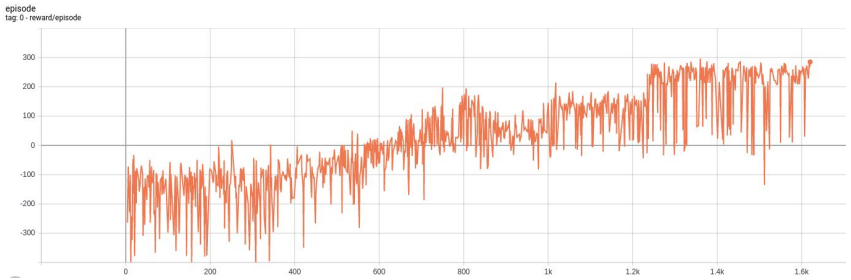
Proximal Policy Optimization - RAM

Lunar Lander

Implementation



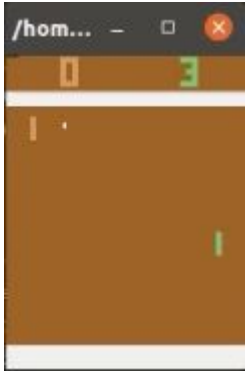




Proximal Policy Optimization - Frames

Pong

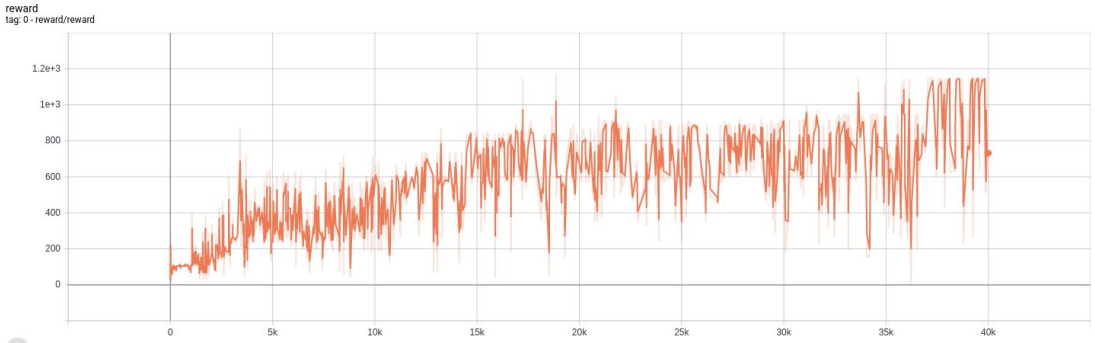
Implementation



Proximal Policy Optimization - Frames

Space Invaders

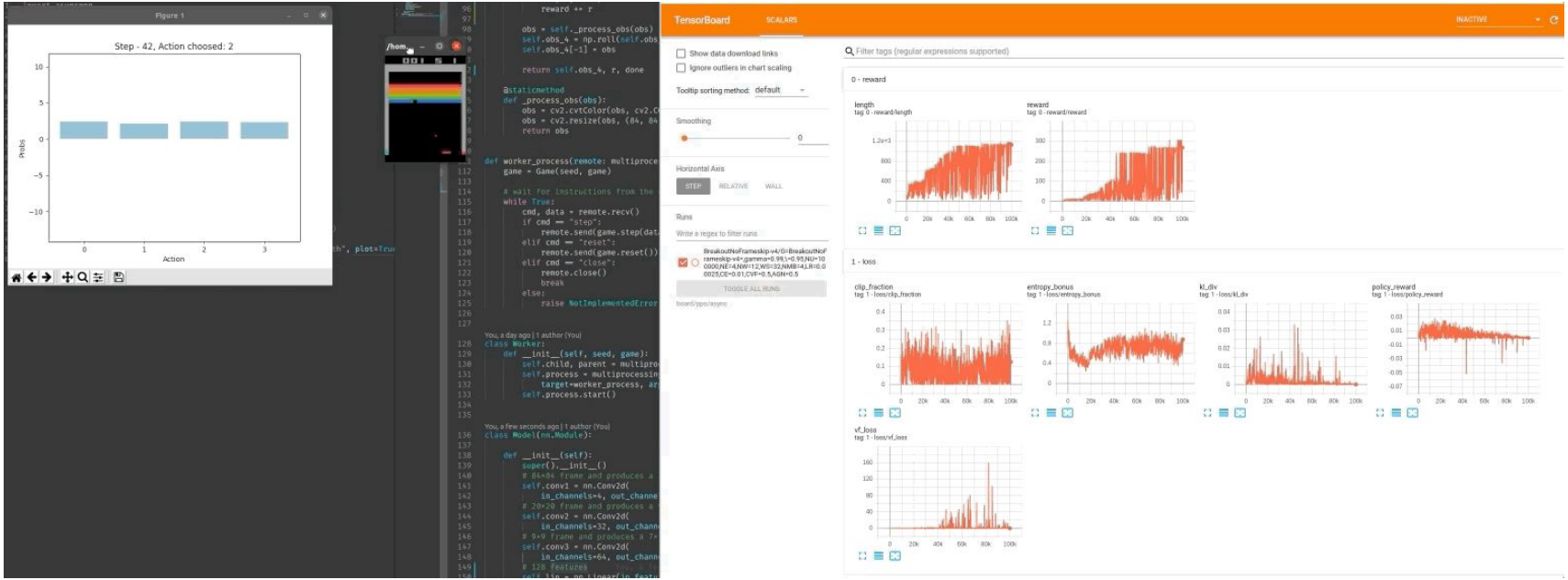
Implementation



Proximal Policy Optimization - Frames

Breakout

Implementation

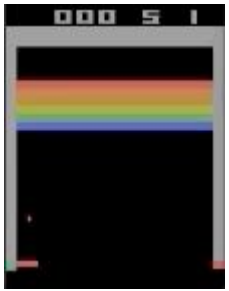


Results

Proximal Policy Optimization - Frames

Breakout

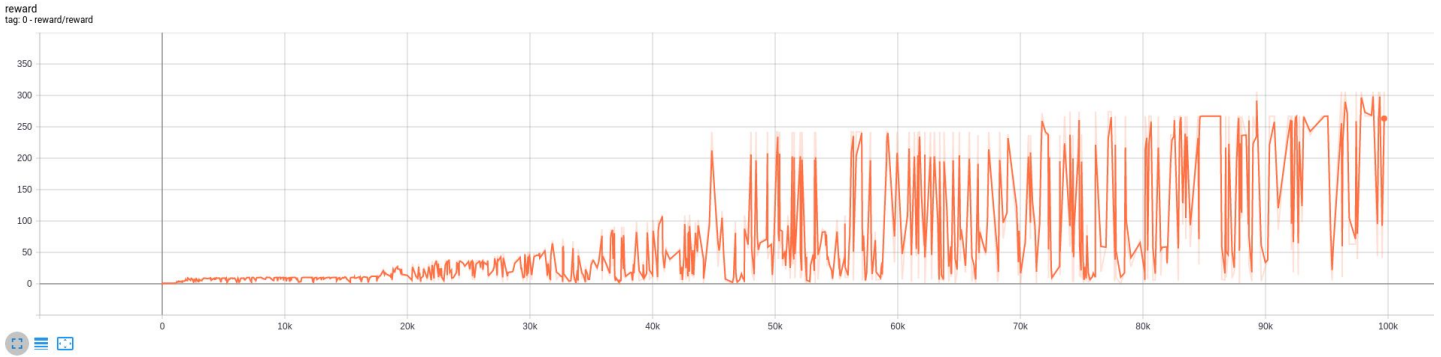
Implementation



1k

100k

300k



Results

Conclusion

One of the first issues we faced is the hardware limitation.

It took a lot of time to train the models, and the batch size was also limited to the hardware compute power.

We have observed that in PPO with A2C the models were training successfully in simple environments, but when complexity was increased [Atari] we needed more episodes and the model was unable to solve the environments.

We have seen that it is very important to monitor the performance of the models with tools like TensorBoard.

Both monitoring and testing different techniques led us to the final algorithm, with which we would be able to train a model that was able to solve the Atari environments.

It has been very challenging, specially with PPO.

There are many factors to take into account: hyperparameters, architecture, data pre-processing, training length...

Deep Learning has a huge community behind it, which makes it easier to find help and resources.

However, some DL challenges can lead you to face very specific problems which might be cryptic and harder to solve.

We believe, right now, this is especially true for RL.

Early in the project the team reduced in size from 4 to 3

Bibliography

Learning from the memory of Atari 2600
https://arxiv.org/pdf/1605.01335.pdf

High-dimensional Continuous Control Using Generalized Advantage Estimation
https://arxiv.org/pdf/1506.02438.pdf

What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study
https://arxiv.org/pdf/2006.05990.pdf

Stabilizing Transformers For Reinforcement Learning
https://arxiv.org/pdf/1910.06764.pdf

RL — Proximal Policy Optimization (PPO) Explained | by Jonathan Hui
https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

A (Long) Peek into Reinforcement Learning
https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Proximal Policy Optimization Algorithms
https://arxiv.org/pdf/1707.06347.pdf

Proximal Policy Optimization
https://openai.com/blog/openai-baselines-ppo/

Proximal Policy Optimization (PPO)
https://towardsdatascience.com/proximal-policy-optimization-ppo-with-tensorflow-2-x-89c9430ecc26

RL — Proximal Policy Optimization (PPO) Explained
https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

Understanding Actor Critic Methods and A2C
https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f

Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning)
https://towardsdatascience.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288

Docs » Proximal Policy Optimization
https://spinningup.openai.com/en/latest/algorithms/ppo.html#documentation-pytorch-version

RL — Policy Gradient Explained
https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146

RL — Policy Gradients Explained (Part 2)
https://medium.com/@jonathan_hui/rl-policy-gradients-explained-advanced-topic-20c2b81a9a8b

Asynchronous Methods for Deep Reinforcement Learning
https://arxiv.org/pdf/1602.01783.pdf

Reinforcement learning algorithms with Generalized Advantage Estimation
https://github.com/bsivanantham/GAE

Proximal Policy Optimization Tutorial (Part 1/2: Actor-Critic Method)
https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9afffbf6

Proximal Policy Optimization Tutorial (Part 2/2: GAE and PPO loss)
https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-2-2-gae-and-ppo-loss-fe1b3c5549e8

Applications of Reinforcement Learning in Real World
https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12

Human-level control through deep reinforcement learning
https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

Playing Atari with Deep Reinforcement Learning
https://deepmind.com/research/publications/playing-atari-deep-reinforcement-learning

UC Berkeley Reward-Free RL Beats SOTA Reward-Based RL
https://syncedreview-com.cdn.ampproject.org/v/s/syncedreview.com/2020/09/21/uc-berkeley-reward-free-rl-beats-sota-reward-based-rl/amp/?usqp=mq331AQFKAGwASA%3D&amp_js_v=0.1#referrer=https%3A%2F%2Fwww.google.com&amp_tf=De%20%251%24s&ampshare=https%3A%2F%2Fsyncedreview.com%2F2020%2F09%2F21%2Fuc-berkeley-reward-free-rl-beats-sota-reward-based-rl%2F

From 0 to 200 - lessons learned from solving Atari Breakout with Reinforcement Learning
http://blog.jzhanson.com/blog/rl/project/2018/05/28/breakout.html

ACCELERATED METHODS FOR DEEP REINFORCEMENT LEARNING
https://arxiv.org/pdf/1803.02811.pdf

Questions?