Comparing Reinforcement Learning Techniques on carRacing-v0 Environment

Julius Stener & Victoria Valverde
Stanford University

Motivation

- The combination of Deep Learning and Reinforcement Learning have been successful at solving complicated riddles and extracting non-linear features from high dimensional media such as images[1]
- The novelty of the field and our interest in deep learning and reinforcement learning drove us to choose this project

Problem Definition

- Compare how three continuous action space RL algorithms -- **PPO** (Proximal Policy Optimization), **DDPG** (Deep Deterministic Policy Gradient), and **TD3** (Twin Delayed DDPG) - perform the task of teaching a race car to drive around a track
- Tune hyperparameters to maximize performance

Approach: Algorithm Description

PPO[3]: on-policy, e-greedy and Gaussian noise exploration method[3]

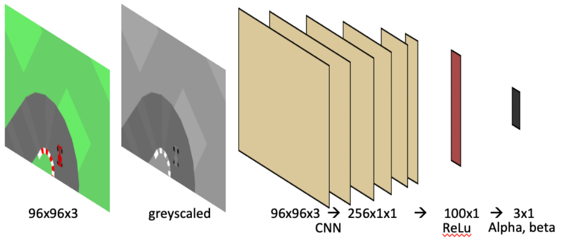
- Pros: uses SGD to optimize parameters, parallelizable[3]
- Cons: does not account for how exploration should be performed, L can still change significantly between updates

DDPG[3]: off-policy, Ornstein–Uhlenbeck process exploration method[3]

- Pros: Q-function used to learn the value of actions (not states)
- Cons: policy is only as good as the Q-function is accurate, learned Q-function commonly overestimates Q-values[4]

TD3[4]: improved version of DDPG

- Pros: improved through clipped double learning, delayed policy updates and target policy smoothing
- Cons: very compute intensive



Illustrative state processing from pixel image to linear vector with CNN

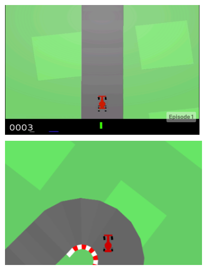
Approach: Model Description

CarRacing-v0 environment[2]:

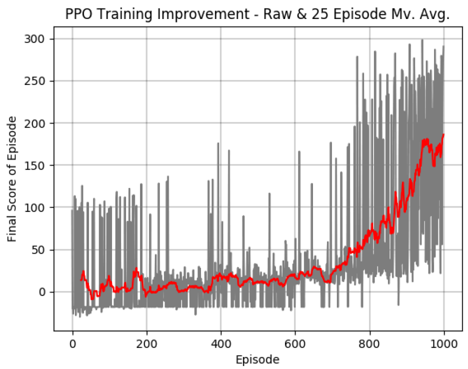
- Game engine from OpenAI gym, part of the continuous control tasks in the Box2D simulator environments

We used a Markov Decision Process (**MDP**) to model our task:

- **Observation Space:** set of 96x96x3 vectors encoding the image on the right
- **Action Space:** set of 3D vectors of the form [s,t,b] where each letter specifies the steering angle (s), throttle (t) and brake (b)
- **Reward:** -0.1 for every frame and +1000/N for every track tile visited, where N is the total number of tiles in the track



Results: Algorithm Performance & Noise



PPO

- Trained E=1000 episodes, N=1000 parallel actors each collecting a default of T=8 timesteps of data. Each episode constructs surrogate loss on the NT timesteps of data and optimize using SGD
- Showed **considerable improvement** after 700 episodes, increasing at an average of 0.3 points per episode during the last 300 iterations
- Considerable amount of **noise** (specially at start) due to the nature of PPO and L being able to change significantly between updates

Results: Algorithm Performance & Noise

DDPG & TD3

- Trained for E=1000 episodes, T=1000 timesteps each choosing actions with exploration noise of 0.5, and batch size of N=100 from the replay buffer to update the policy
- Implementation was unable to learn, hence score stayed around -17.9 but we can still observe a considerable amount of noise

Error Analysis

- Unable to make the race car move due to an unintended mismatch in the dimensionality of our NN's output and that of the action space
- Suspect needs tuning of hyperparameters as well as in more iterations to train in order to see a significant improvement

Challenges & Future Directions

- ***Further tuning parameters**:* would potentially improve performance
- ***Running on GPU**: would allow us to further tune and run tests more efficiently, which was the main challenge in out project*
- ***Discretizing action space***: would allow us to make the search and optimization faster/simpler
- ***Removing the bottom panel:*** (containing information such as lateral g-force, reward indicator and velocity) does not add meaningful information about current state. Decreasing the number pixels to convolve through has the potential to increase accuracy and speed
- ***Setting a maximum on the rewards:*** would stop the model from incentivizing high speeds that make car loose control in tight curves

[1] Dwibedi, Vemula (link), [2] OpenAI Gym, CarRacing-v0 (link), [3] Berseth (link), [4] Spinning Up (link), [5] Schulman et al. "PPO", [6] Lillicrap et al. "DDPG", [7] Fujimoto et al. "TD3".