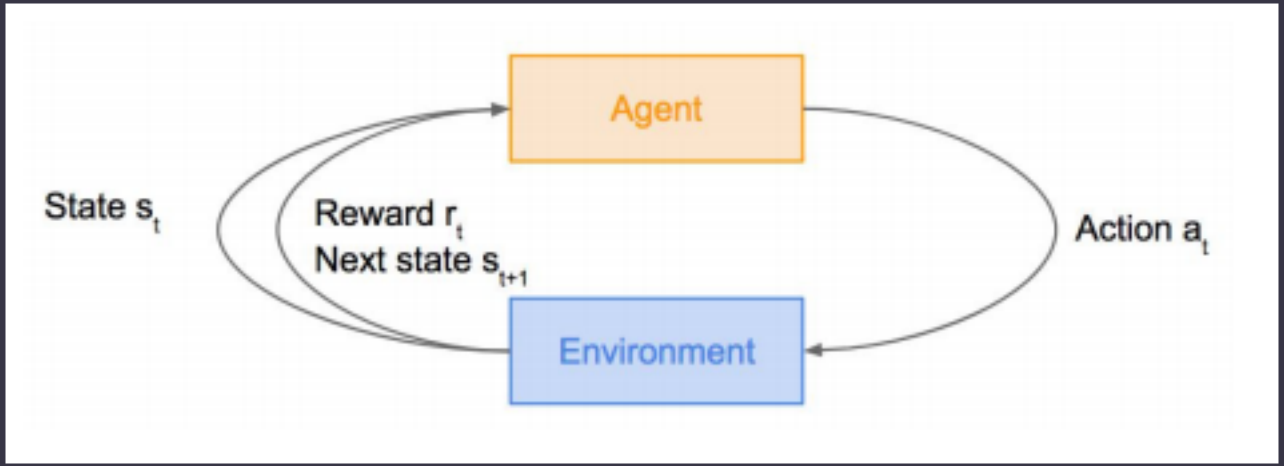
Pong: AI Reinforcement Learning

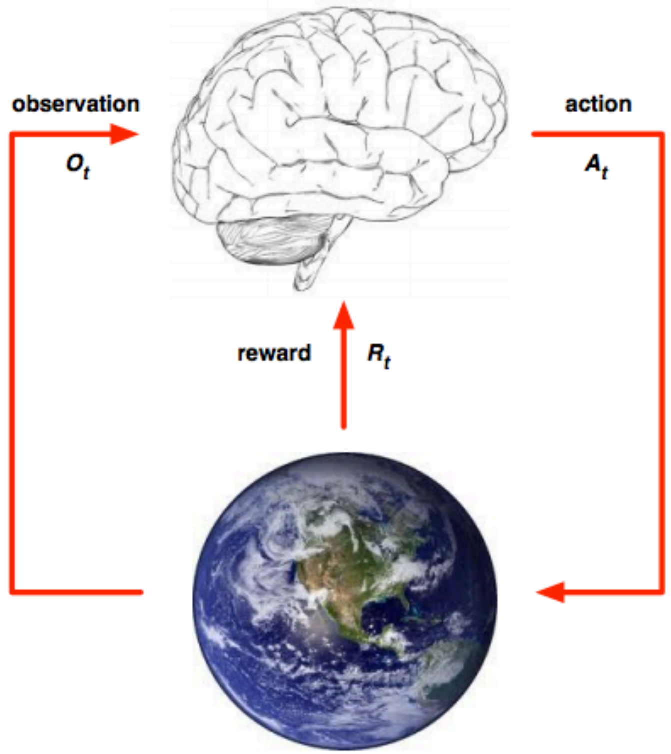
BY JERRY KHONG

Reinforcement Learning

- Process of learning by interacting with an environment
- Rewarded through positive feedback



Reinforcement Learning



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

Source: Silver, University College London. 2015

Reinforcement Learning: Agents

- **Policy**: agent's behavior function (map from state to action)
- **Value function**: how good is each state and/or action

Markov Decision Process

- Mathematical formulation of the RL problem
- **Markov property**: Current state completely characterizes the state of the world

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

Source: Silver, University College London. 2015

Q-Learning (Value Function)

$$U(s) = R(s) + \gamma max_a \sum_{s'} T(s,a,s')U(s')$$

Bellman Equation

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$$

Q-learning function

- Find an optimal action-selection policy
- State-action-value function
- Select action based on argmax of Q function

Policy Gradient

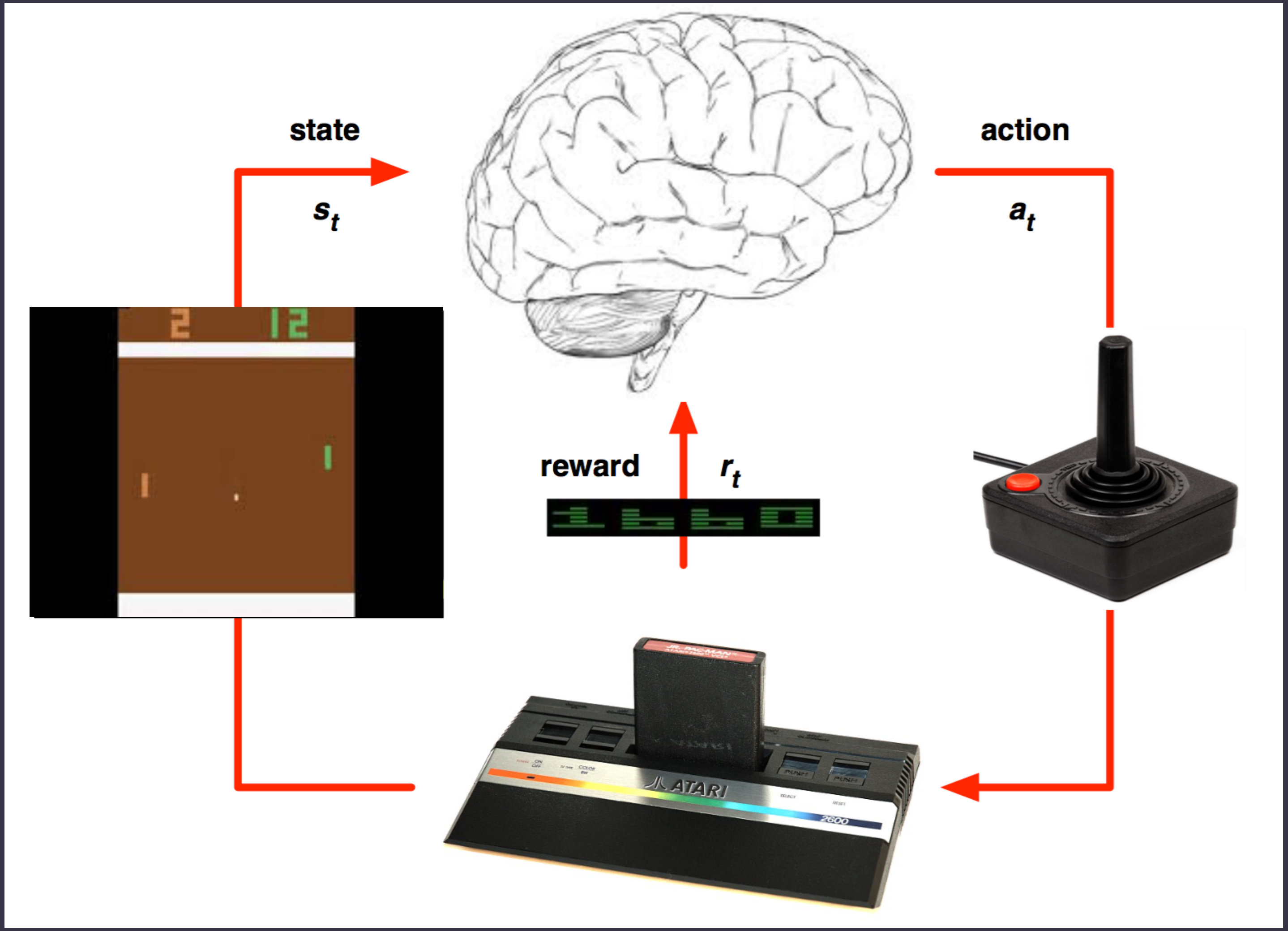
- Maximize E[R | $\pi\theta$]

Intuitions: collect a bunch of trajectories, and ...

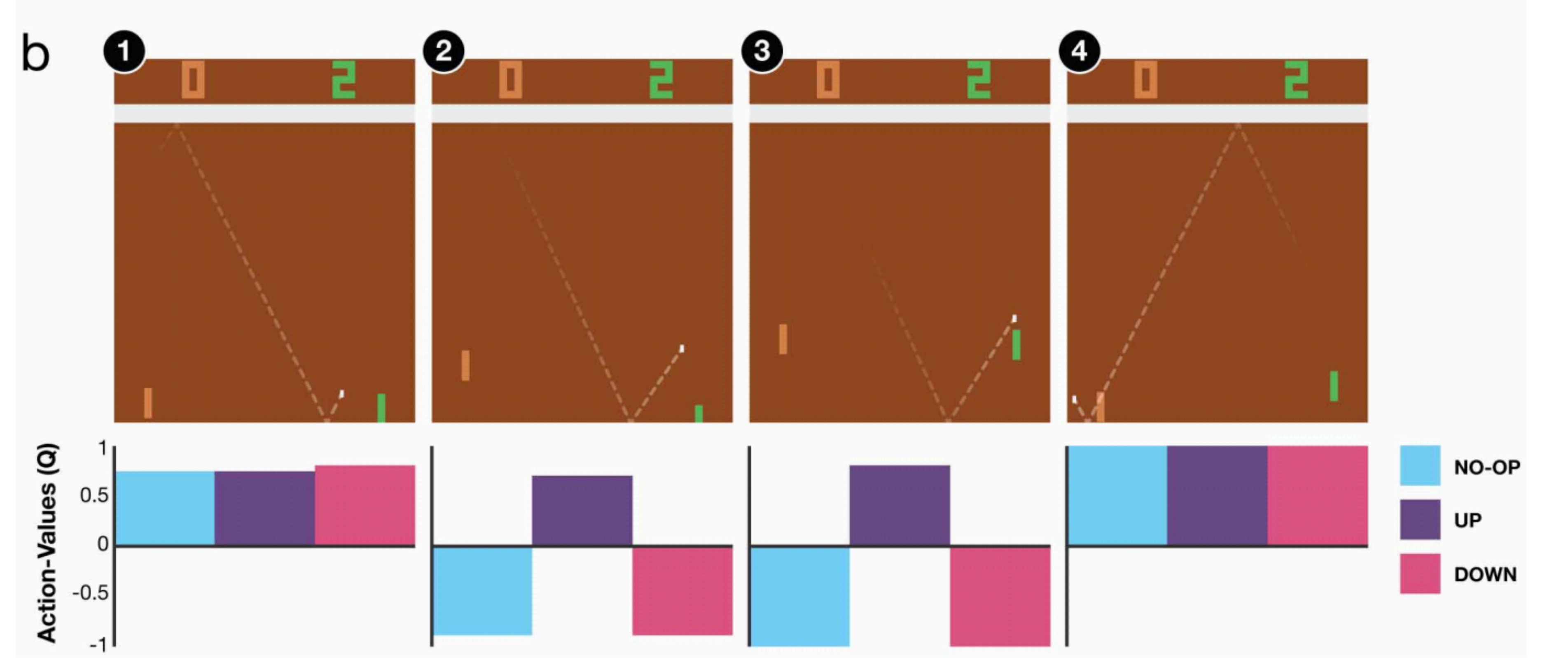
- Make the good trajectories more probable
- Make the good actions more probable
- Push the actions towards good actions

Atari: Pong

- **Objective**: Complete the game with the highest score
- **State**: Raw pixel inputs of the game state
- **Action**: Game controls e.g. Up, Down, No operation
- **Reward**:
    - +1 reward if the ball went past the opponent
    - -1 reward if we missed the ball
    - 0 otherwise



Learned action-value function

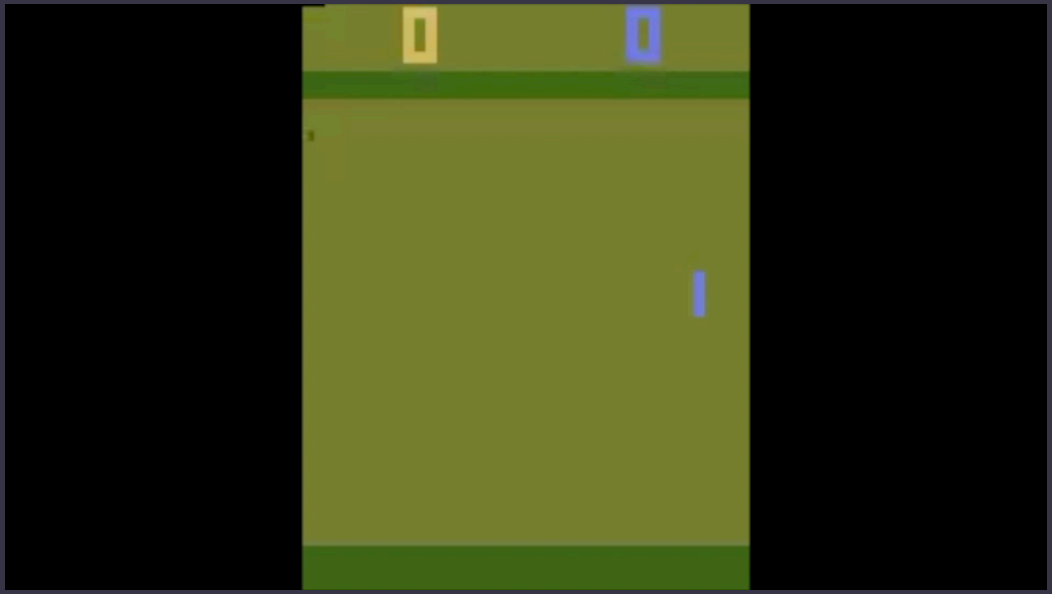


OpenAI Gym

- Non-profit company, founded by Elon Musk
- Focus on creating a positive long-term human impact with AI
- Packaged environment for reinforcement learning
- Only available for Python
- Provides RGB pixels of game screen

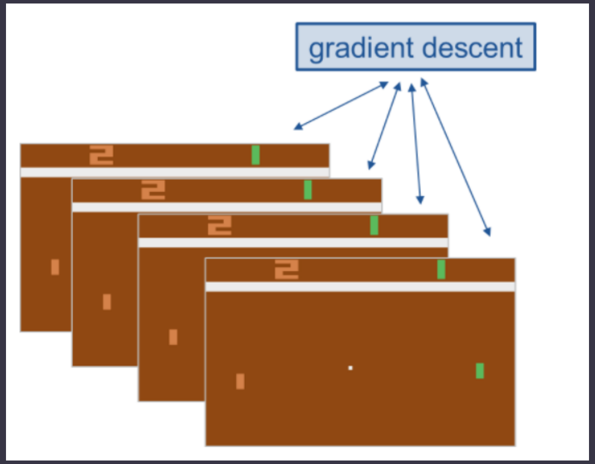
Random Baseline Model

- Random actions
- No learning whatsoever
- Not successful

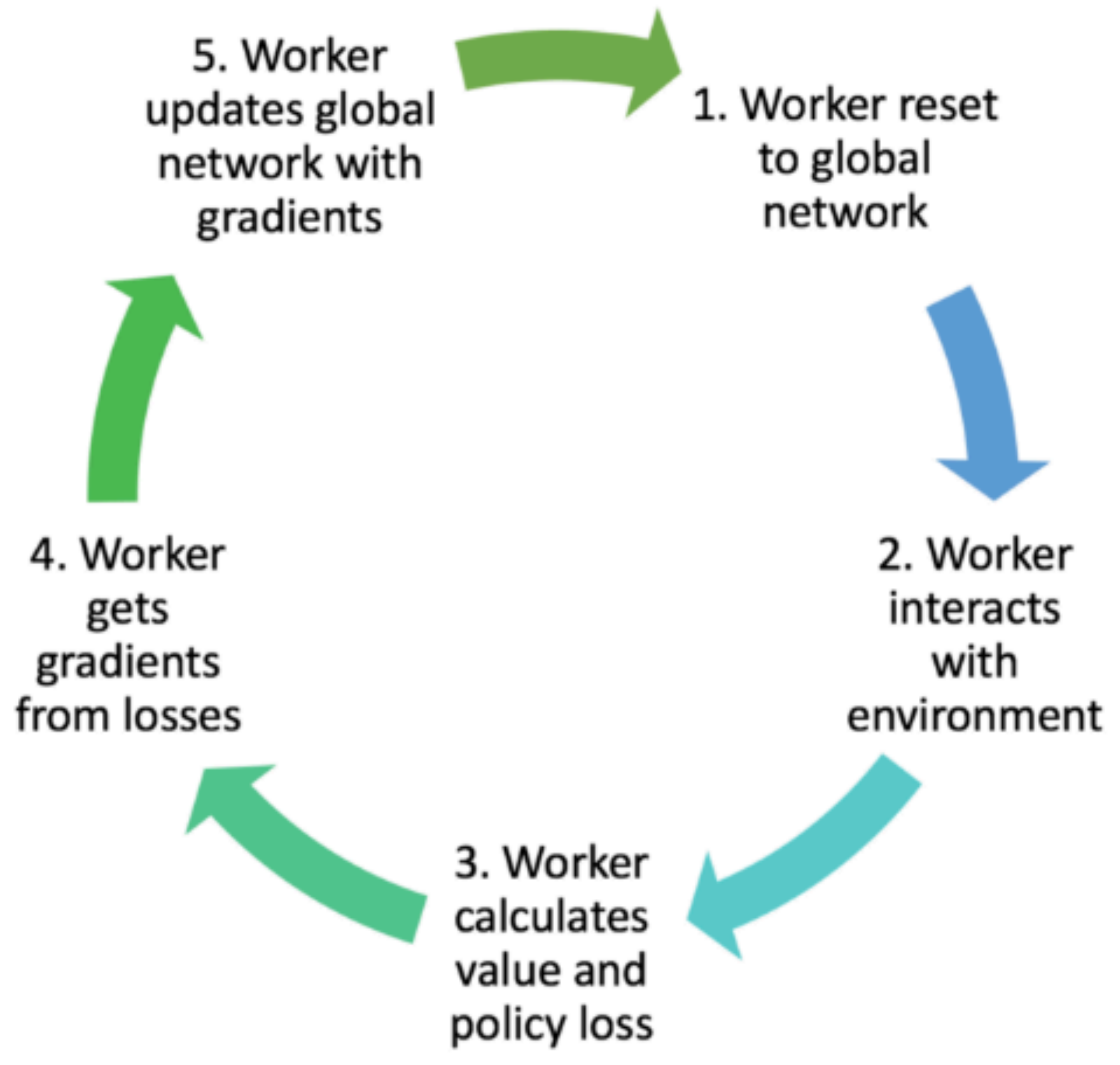


Asynchronous Actor-Critic Agents (A3C)

- Runs multiple Atari environments in parallel, each thread asynchronously updating a global model
- Uses policy gradient and value function
- Training speed increase is roughly linear in number of threads!

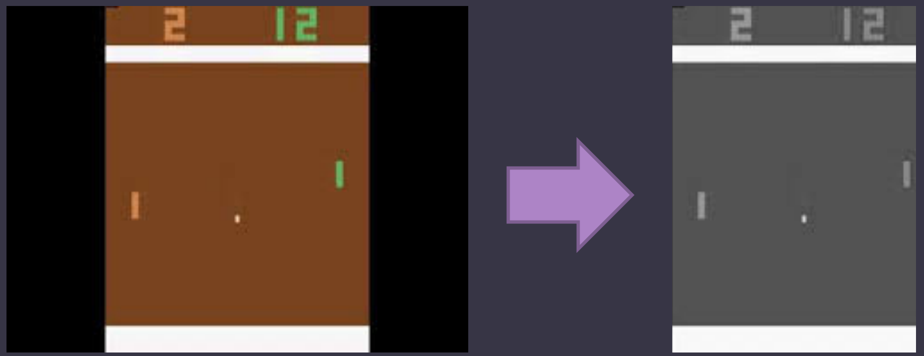


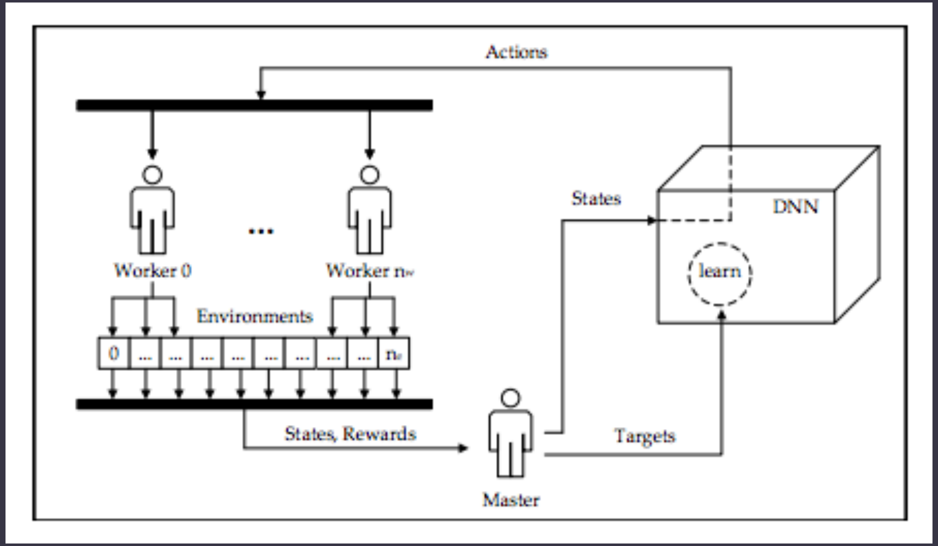
Training workflow



New: Parallel Reinforcement Learning

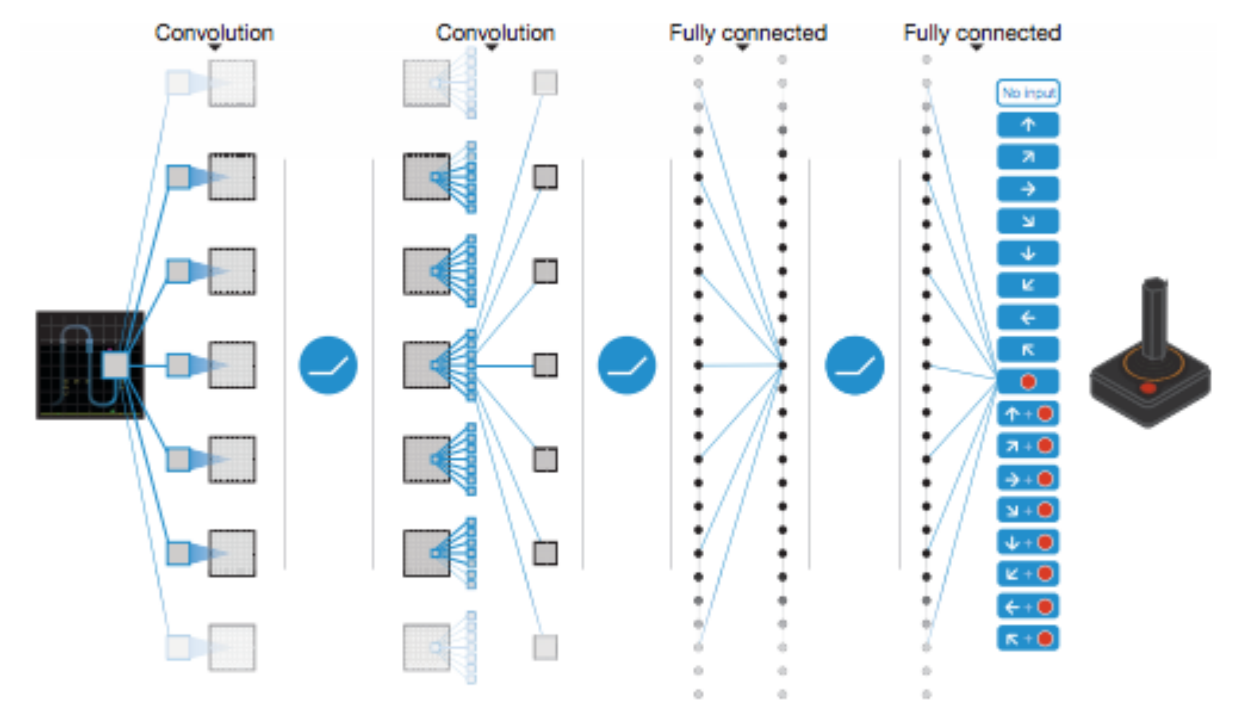
- Preprocessing: RGB → grayscale 210x160 → 84x84
- Set of workers apply all the actions and store observed experiences
- Environment restarts whenever final state is reached
- Significant speed improvements





Framework

- GPU: 1 x NVIDIA Tesla K80
- Ubuntu 16.04
- CUDA 8 + cuDNN5.1*
- VM instance



*No openGL

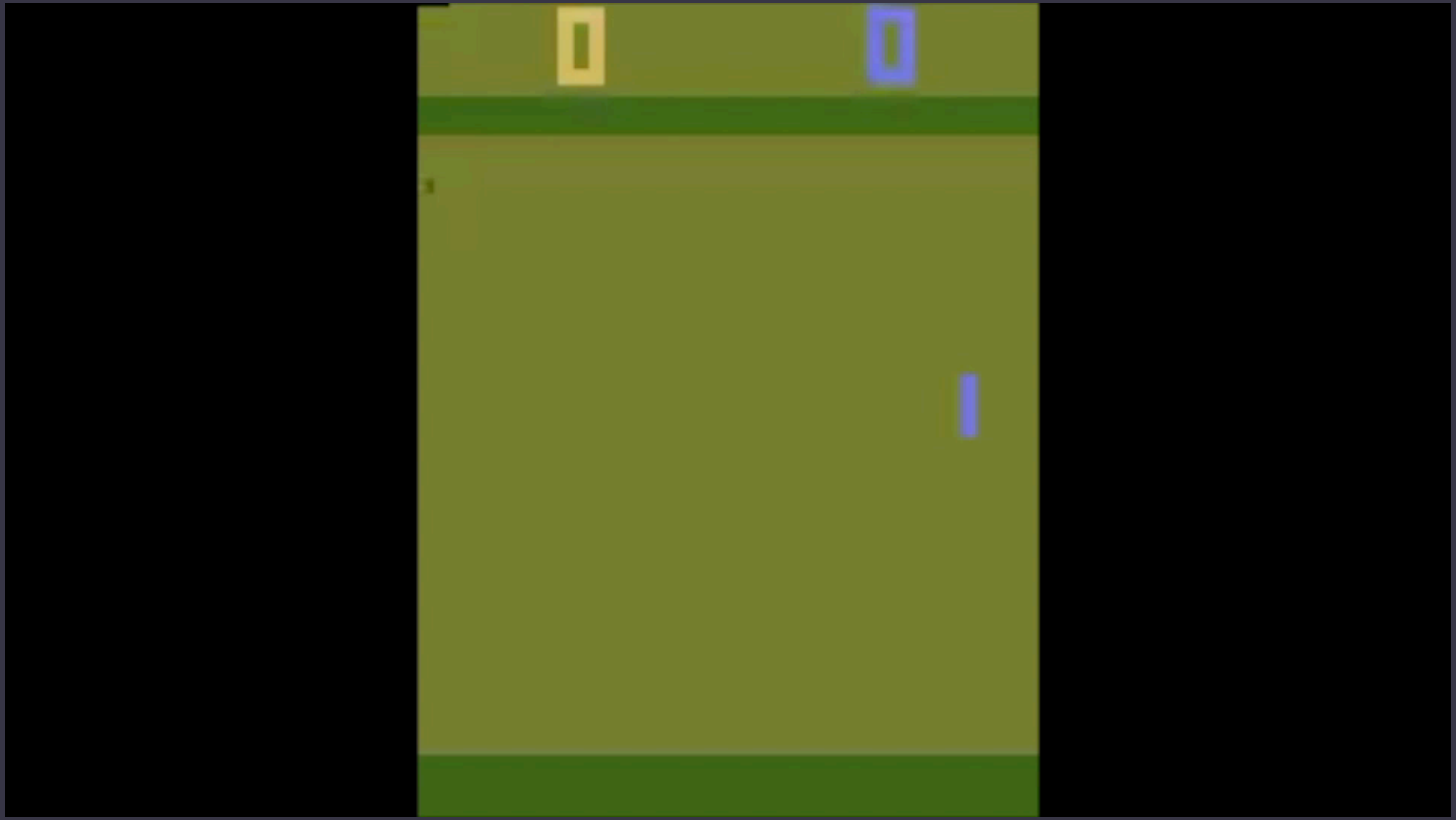
Parallel Advantage Algorithm

Algorithm 1 Parallel advantage actor-critic

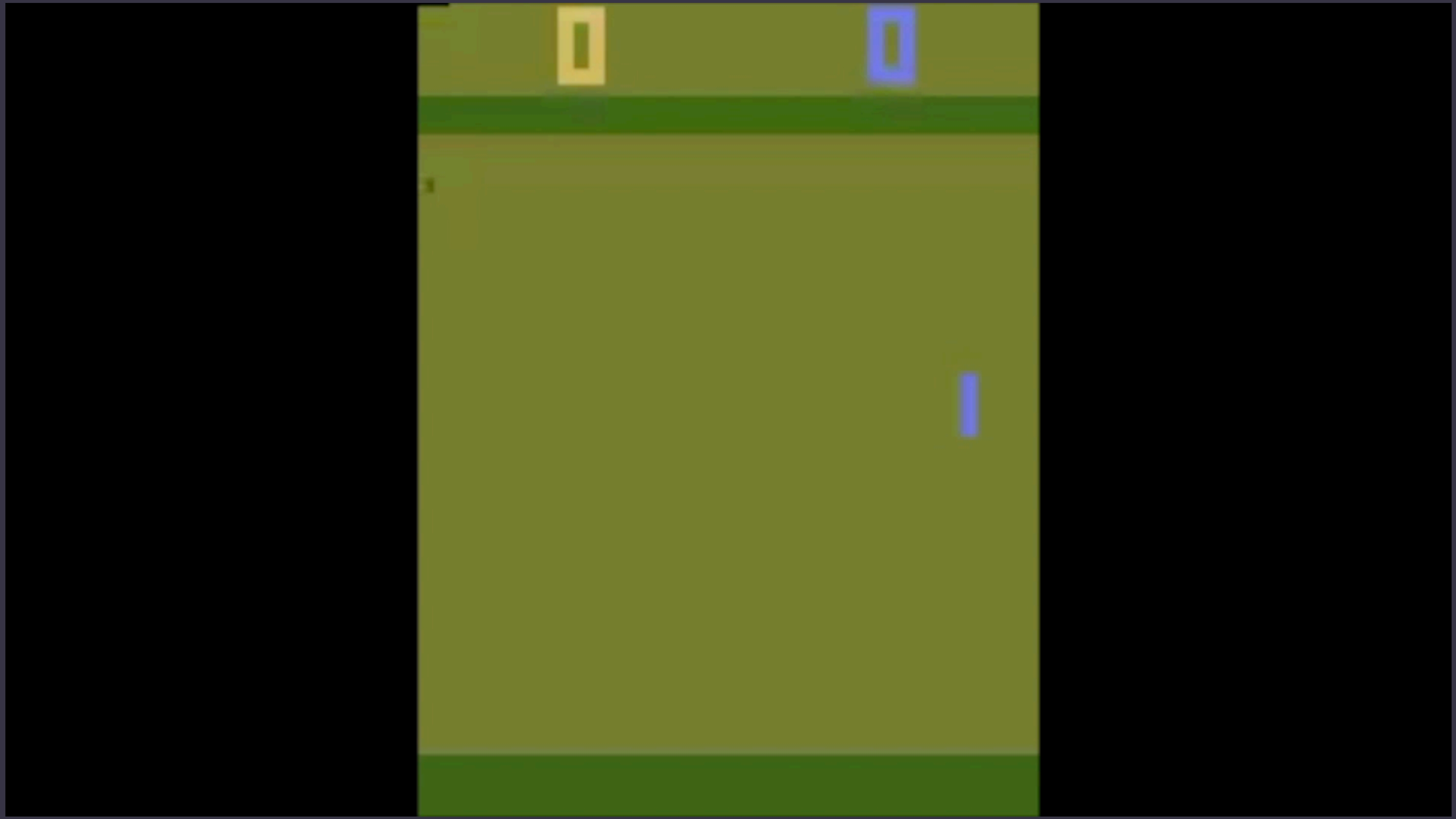
Initialize timestep counter $N = 0$ and network weights θ, θ_v
Instantiate set $\boldsymbol{e}$ of n_e environments
repeat
  for $t = 1$ to t_{max} **do**
    Sample $\boldsymbol{a}_t$ from $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\theta)$
    Calculate $\boldsymbol{v}_t$ from $V(\boldsymbol{s}_t;\theta_v)$
    parallel for $i = 1$ to n_e **do**
      Perform action $a_{t,i}$ in environment e_i
      Observe new state $s_{t+1,i}$ and reward $r_{t+1,i}$
    end parallel for
  end for
  $\boldsymbol{R}_{t_{\max}+1} = \begin{cases} 0 & \text{for terminal } \boldsymbol{s}_t \\ V(s_{t_{\max}+1};\theta) & \text{for non-terminal } \boldsymbol{s}_t \end{cases}$
  for $t = t_{\max}$ down to 1 **do**
    $\boldsymbol{R}_t = \boldsymbol{r}_t + \gamma \boldsymbol{R}_{t+1}$
  end for
  $d\theta = \frac{1}{n_e \cdot t_{max}} \sum_{i=1}^{n_e} \sum_{t=1}^{t_{max}} (R_{t,i} - v_{t,i}) \nabla_\theta \log \pi(a_{t,i}|s_{t,i};\theta) + \beta \nabla_\theta H(\pi(s_{e,t};\theta))$
  $d\theta_v = \frac{1}{n_e \cdot t_{max}} \sum_{i=1}^{n_e} \sum_{t=1}^{t_{max}} \nabla_{\theta_v} (R_{t,i} - V(s_{t,i};\theta_v))^2$
  Update θ using $d\theta$ and θ_v using $d\theta_v$.
  $N \leftarrow N + n_e \cdot t_{\max}$
until $N \geq N_{max}$

Source: Clemente, et al. 2017 Efficient Parallel Methods for Deep Reinforcement Learning.

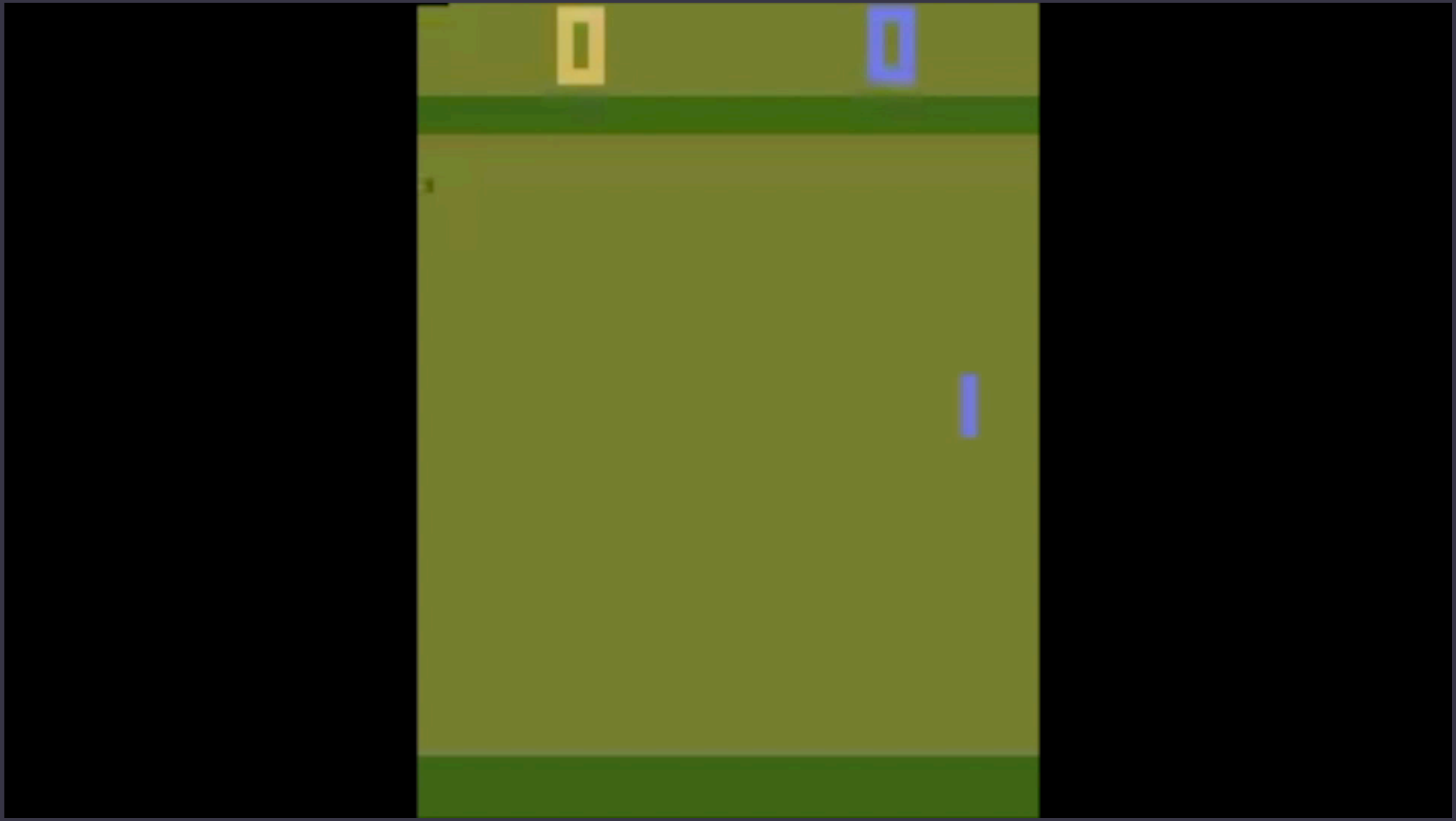
Results after 2 hours of training



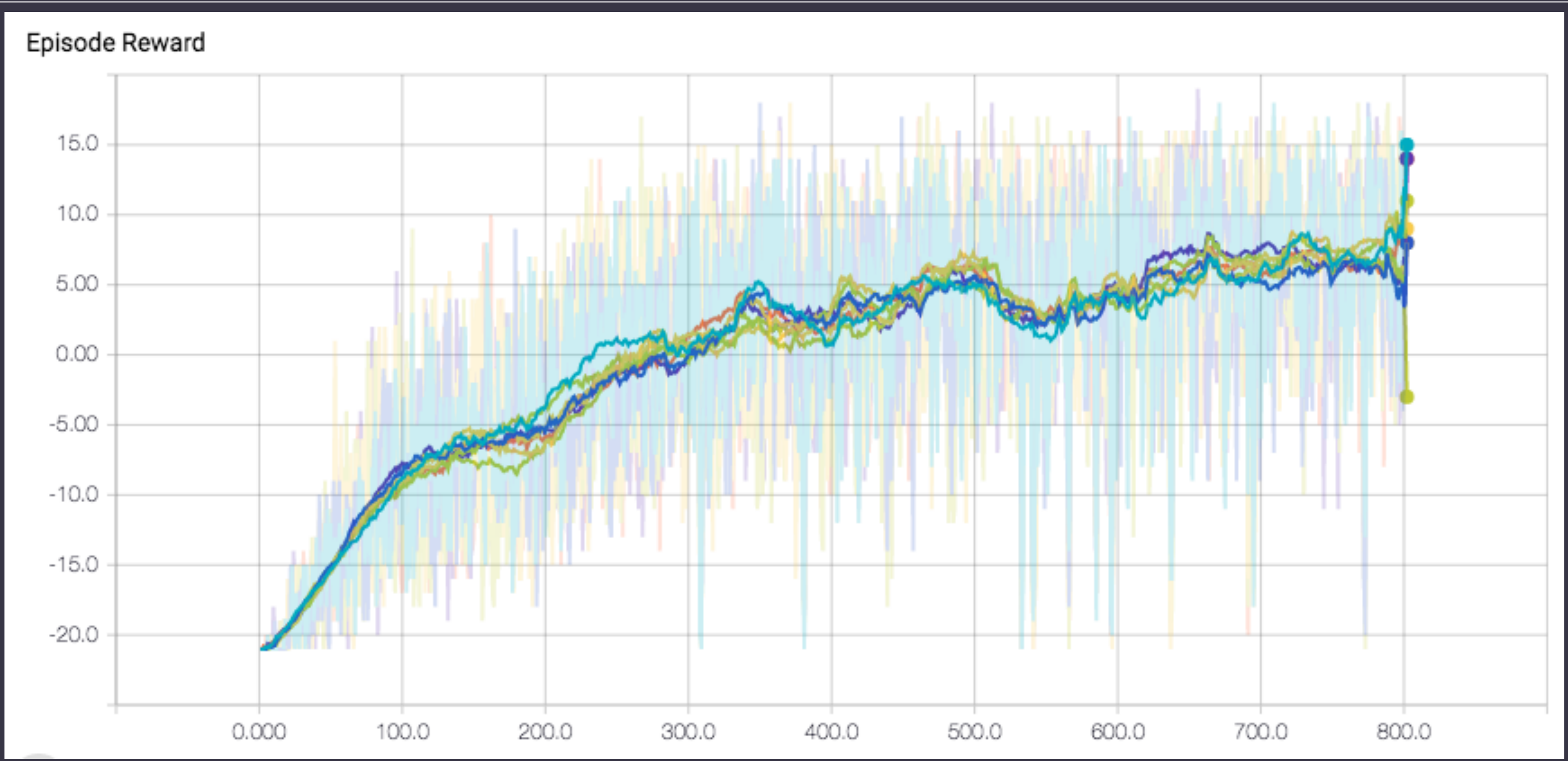
After 12 hours of training…



After 18 hours of training…



Training Episode Reward



Hyperparameters

- epsilon = 1e-7
- decay = 0.99
- lambda = 0.99
- learning rate = 0.001
- entropy = 0.01
- number of environments = 16
- t_max (update period) = 5

Challenges and Potential Risks

- Difficult to set up OpenAI on remote server
- Installing correct drivers with no OpenGL files
- Keras and TF.learn issues, switched to TensorFlow
- Stochastic process → training can be bad

Future Goals

- Tune parameters
- Submit score on OpenAI
- OpenAI Universe: More complex games