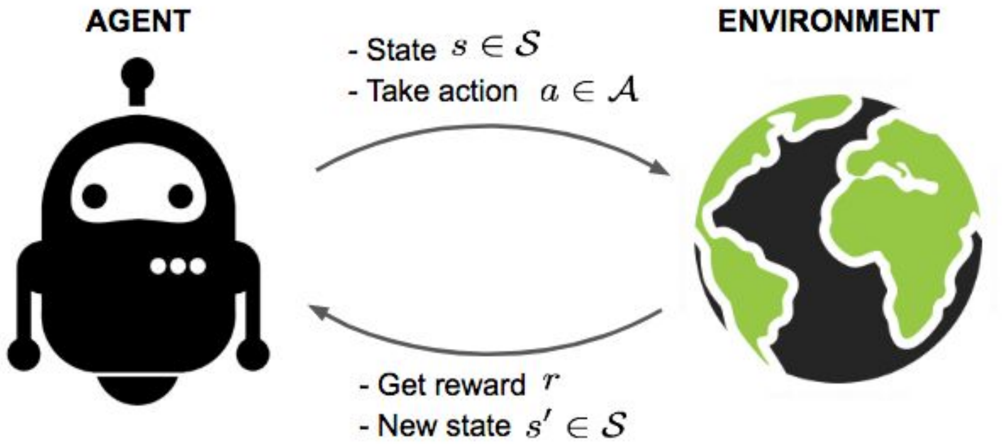
# Ensemble Learning

Teaching Agents to Walk

# Problem Statement



# Problem Statement - Contd.

➔ Discounted returns represent the sum of all rewards '$\mathbf{r_t}$' *ever* obtained by the agent discounted by $\gamma^t \in (0,1)$

$$R_t = \sum_{t=0}^{\infty} \gamma^t r_t$$

➔ Goal of RL is to learn the optimal policy '$\boldsymbol{\pi_\theta}$' which maximizes the expected return:

$$\pi_\theta^* = \max_{\pi_\theta} \mathbb{E}_{\pi_\theta}[R_t]$$

# Background

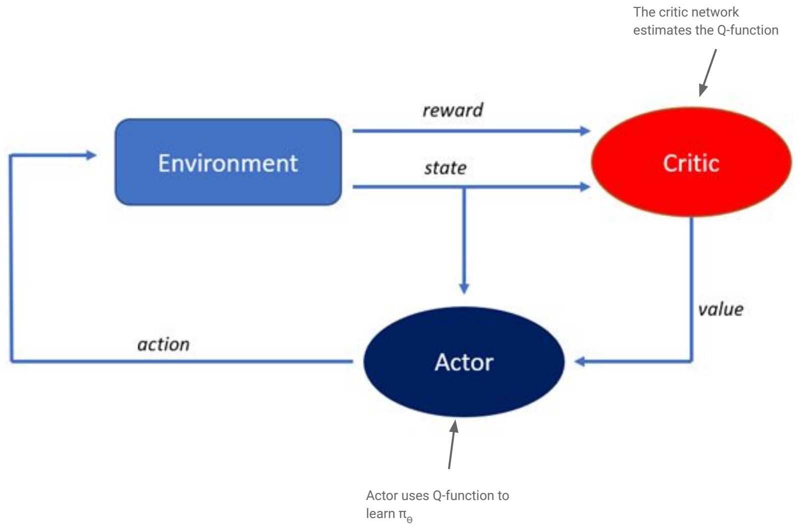
- ➔ Our proposed algorithm builds upon TD3
- ➔ We will go over the necessary information:
    - ◆ Deep Deterministic Policy Gradients (DDPG)
    - ◆ Twin Delayed DDPG (TD3)

# Deep Deterministic Policy Gradients (DDPG)

- Based on the actor-critic framework:
  - ***Actor Network***: Learns the deterministic policy ($\pi_\theta$)
  - ***Critic Network***: Approximates the Q-function

- DDPG is a model-free, off-policy algorithm.

- Makes use of two target networks:
  - Target Actor Network
  - Target Critic Network



# DDPG: Q-Function Estimation

➔ The target is approximated using Temporal Difference in conjunction with the secondary target networks:

Target Network

$$y = r(s, a) + \gamma Q_{\phi'}(s', a')$$

Target

Present rewards

a’ sampled from target actor

➔ Using the target, we minimize the following error:

$$L(\phi') = \mathbb{E}_{s,a,s',a'}\left[(Q_{\phi}(s, a) - y)^2\right]$$

# DDPG: Policy Eval & Replay Buffer

**- Policy Evaluation:**

➔ Since the Q-function is differentiable, we can just perform gradient ascent to find the optimal policy:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p_\pi}\left[\nabla_a Q_\pi(s,a)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s)\right]$$

**- Experience Replay Buffer:**

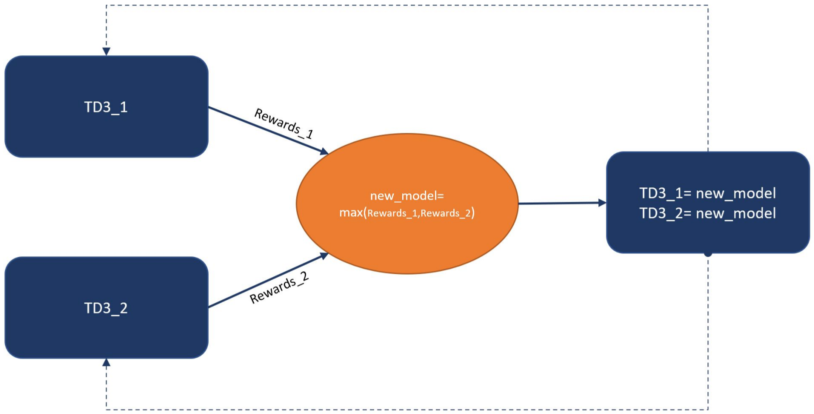
➔ DDPG uses an experience replay buffer to store previous experiences (states, action, reward).
➔ Prevents bias since batches of samples are drawn from it at random.

# Twin Delayed DDPG (TD3)

- Successor to DDPG
- Addresses the overestimation of Q-values faced by DDPG
- Mitigates the problem by three tasks:
  - ***Target Policy Smoothing***: Adds noise to the target actions, **a'**
  - ***Clipped Q-Learning***: Learns two Q-function
  - ***Delayed Policy Updates***: Policy and target policy are updated after every other step

# Multi-TD3

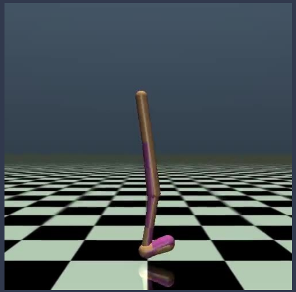
- ➔ Based on the TD3 algorithm
- ➔ Works on the principle of “Ensemble Learning”
- ➔ Performs the following tasks:
  - Two separate instances of TD3 networks are trained
  - Both networks push and sample from the same replay buffer
  - Model with highest rewards serves as the model to be trained in a new episode by the two networks
- ➔ Advantage:
  - Decreases the likelihood of selecting a relatively poor model
  - More experiences are sampled per episode



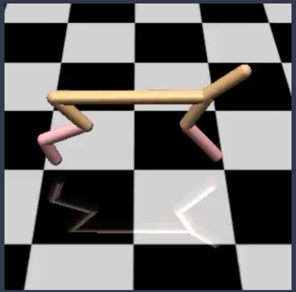

Multi-TD3: Ensemble Learning

# Results

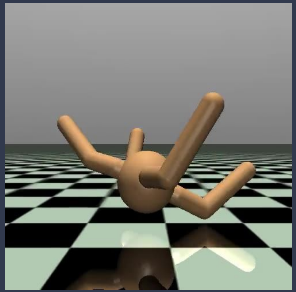
All trainings were conducted using OpenAI's 3 gym environments



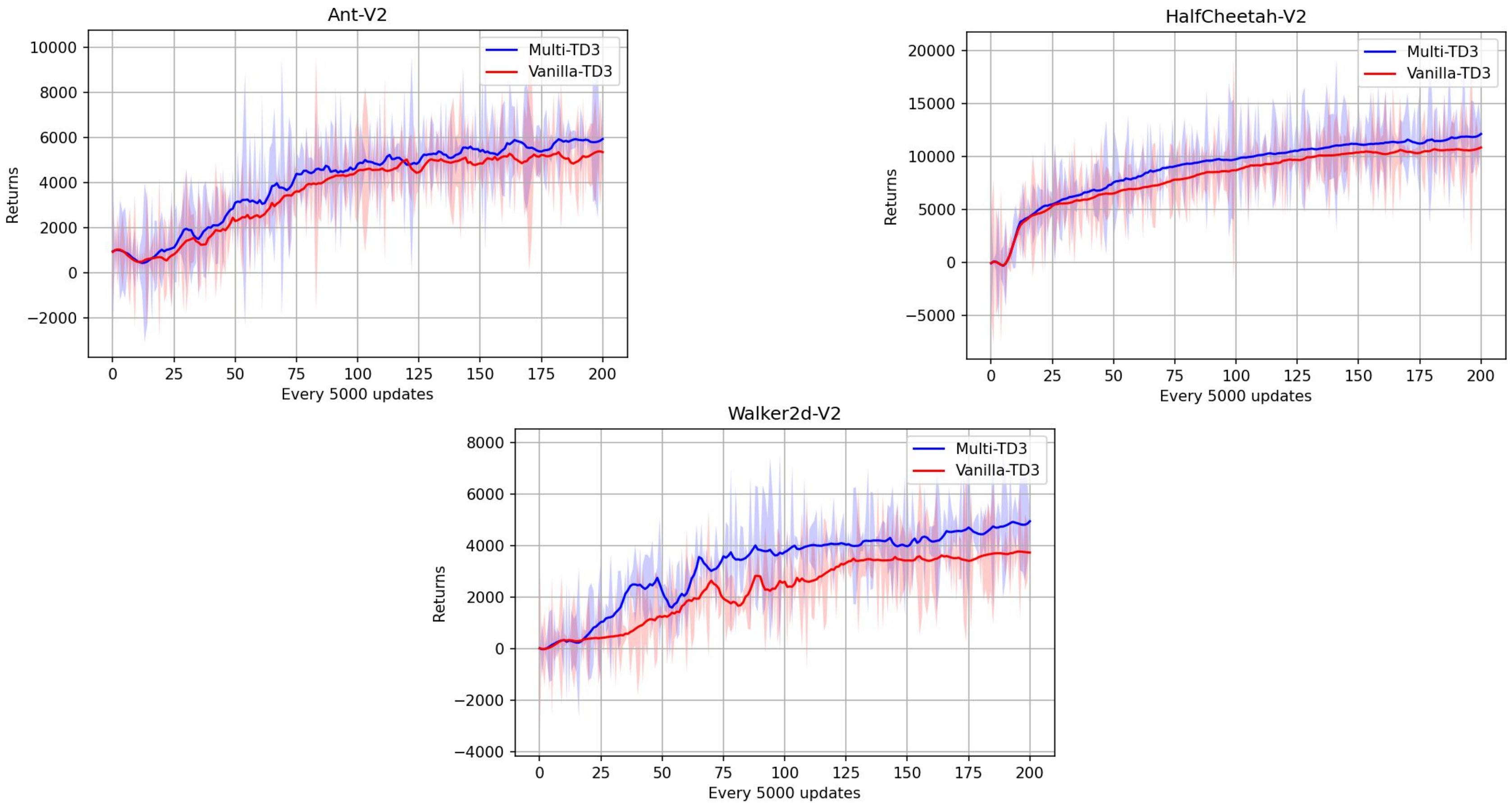
Walker2d-v2



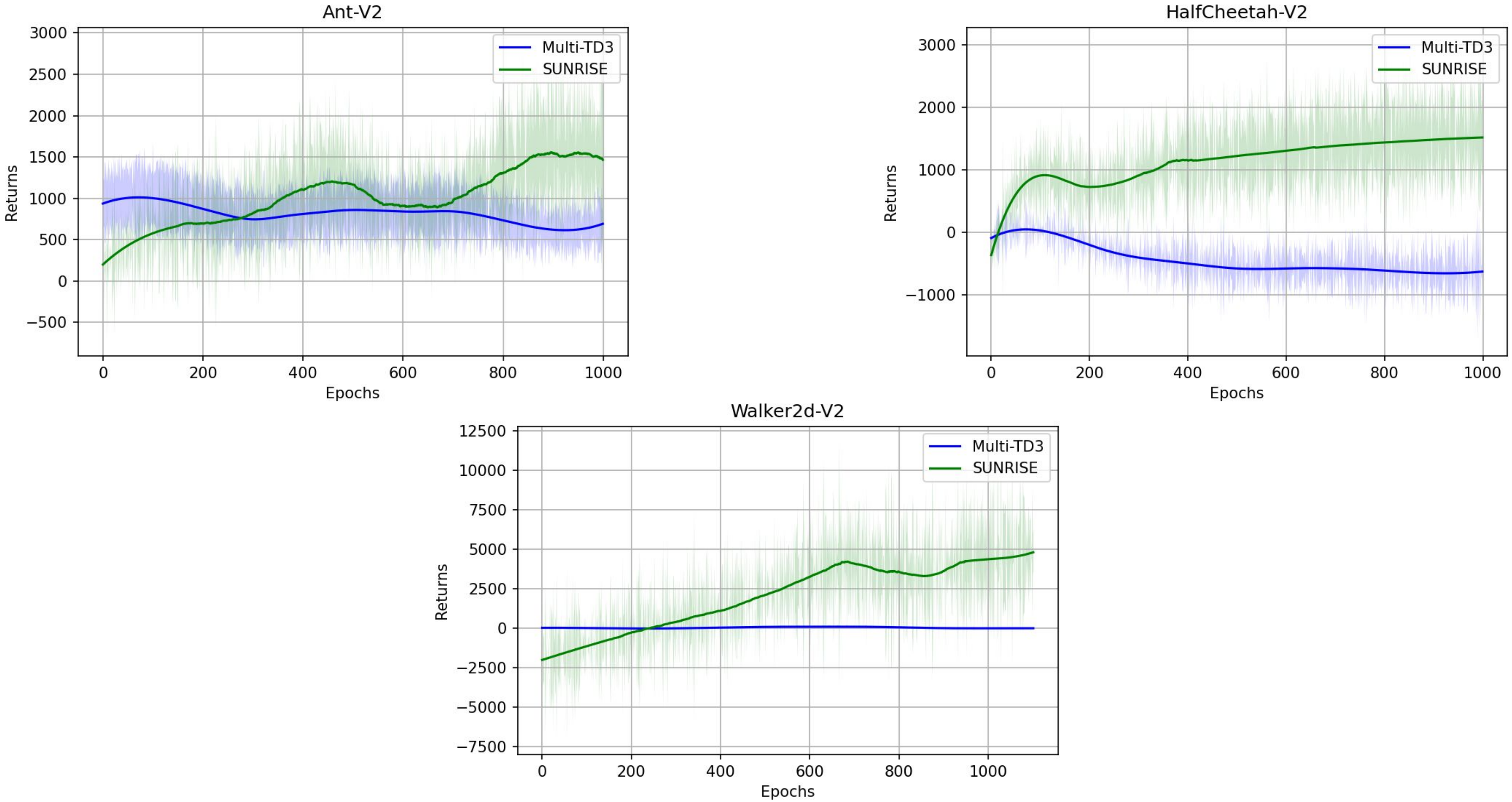
HalfCheetah-v2



Ant-v2




# Results: Multi-TD3 vs TD3




# Results: Multi-TD3 vs SUNRISE

# Discussion

Limitations

- ➔ Not enough time to compare TD3, Multi-TD3, and SUNRISE on the same timescale
- ➔ Because of long training times, no hyperparameter tuning could be done
  - the size of the replay buffer
  - the number of past trajectories sampled

Applications

- ➔ TD3 has found numerous applications in robotics, for example teaching a robotic arm to reach a target; Multi-TD3 should improve it.
- ➔ We already saw the improvement in the learned policy for Walker2d-V2

# Thank you!

# The End