Project 2: Continuous control using actor-critic reinforcement learning

In this project, agents were trained to solve the Unity Environments Reacher task to track targets with robot end effectors. A deep deterministic policy gradient (DDPG) actor critic approach was used to train 20 agents to continuously control a double-jointed robotic arm with modulating torque at the joints. As this was an actor-critic based approach, actions were selected on-policy from an actor network. The corresponding predicted value was approximated via a critic network, which was used to help speed up and stabilize the actor training. The actor policy network was made up of a linear neural network with 6 layers with the following sizes: 33, 256, 128, 64, 32, and 4. The input layer (33) and output layer (4) sizes correspond to the state and action sizes, respectively. Batch normalization was used as input to each layer to help stabilize training. A rectified linear unit (ReLU) activation function was used between each layer except for the last layer, which incorporated a hyperbolic tangent (TanH) activation function for a continuous joint torque output.

Similarly, the critic value function approximation incorporated a linear neural network with 6 layers with the following sizes: 33, 256, 132, 64, 32, and 1. The input layer (33), third layer (132), and output layer (1) correspond to the state size, action size + 128, and singular value approximation for the input state and action, respectively. Batch normalization was included at the first layer and ReLUs were used as the activation function for each layer of the network except for the last one. At the last layer, a sigmoid was used to help keep the value function approximation between 0 and 1. Including the sigmoid, in addition to clipping the critic gradients, helped keep the critic value approximations stable and train the actor faster. As input to the critic, both the state and action were passed to evaluate the value of the pair.

Two separate neural networks with the same structure for both the actor and critic were used to train the agents, namely a primary network that updated at every step and a target network that updated 15 times after every 20 steps. This provided a fixed value function approximation and target policy to avoid correlation between rewards and weight updates. Additionally, a memory replay buffer of 1,000,000 state, action, reward, next state, and done tuples were saved for all 20 agents. During training, tuples were sampled from the memory buffer at random for an uncorrelated experience learning across all agents.

Over the course of 4 seed iterations, an average of 137 ± 15 episodes were required to achieve an average reward goal of 30 over 100 episodes (Figure 1).

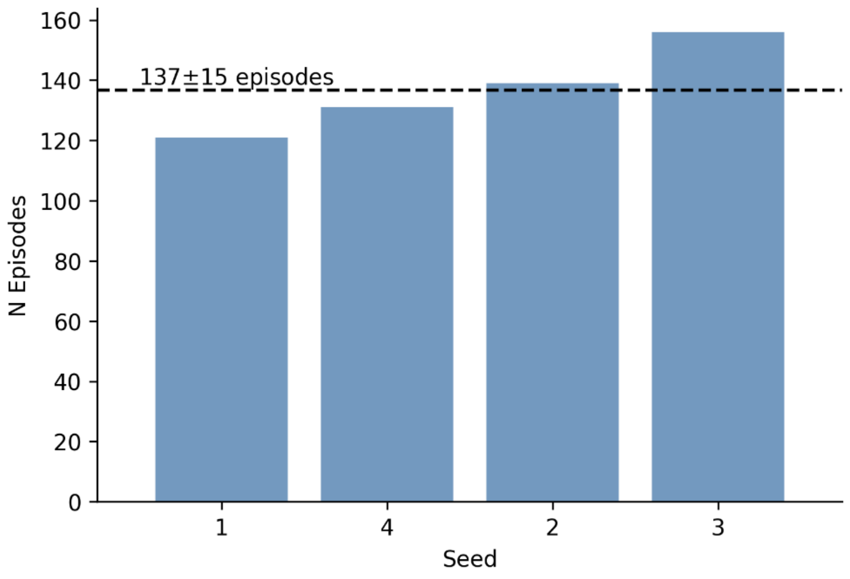

Figure 1: Number of episodes to reach reward goal over 4 seeds

An example of a single seed of 20 agents' training rewards (seed=1) over time is shown in Figure 2. All agents trained relatively quickly and sustained performance once the goal of 30 was achieved.

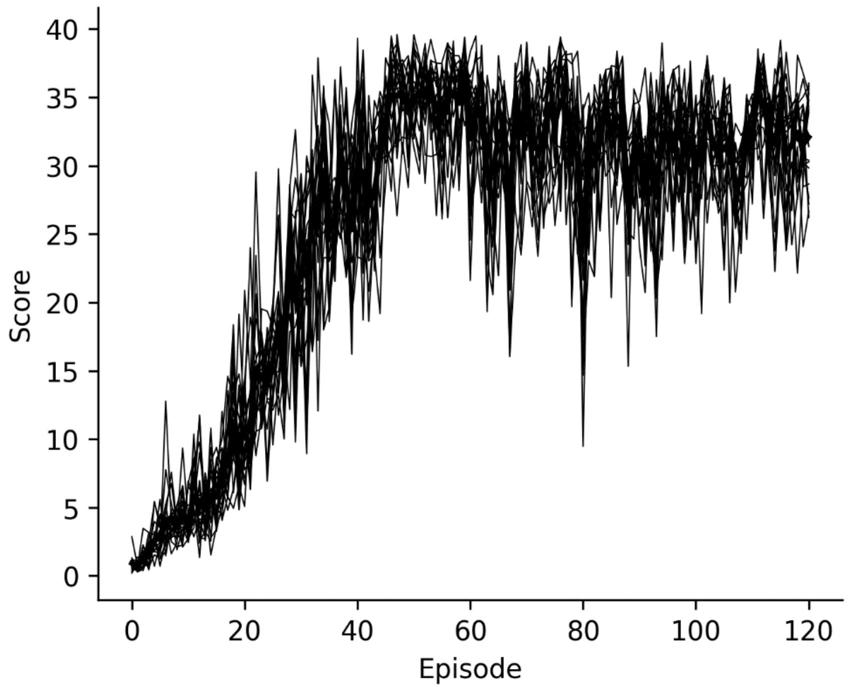

Figure 2: Seed 1 Agent's score throughout training (achieved goal at 121 episodes)

Future ideas would be to consider using prioritized experience replay to help speed up training. Rather than sampling randomly from the buffer, examples which yielded the highest temporal difference (TD) error, indicating greater information could be prioritized. A sampling priority based on TD error magnitude may help the agent’s value function approximation to converge faster. Additionally, alternative policy-based methods or actor critic methods, such as PPO, A3C, or D4PG that distribute the task of gathering experience may help the agents converge faster.