Reacher Continuous Control Report

Learning Algorithm

The learning algorithm is a multi-agent extension of the **Deep Deterministic Policy Gradient (DDPG)** algorithm.

While inspired by D4PG in terms of distributed learning structure, it does not include D4PG-specific components such as distributional critics, N-step returns, or prioritized experience replay. Instead, it enables distributed exploration by running 20 agents in parallel, all contributing to a shared experience buffer for off-policy learning.

The implementation follows an actor-critic architecture suitable for continuous control tasks. Both the actor and critic networks are shared among all agents. The actor network maps each agent's individual state to a continuous action, while the critic network evaluates the Q-value for a given state-action pair. Separate target networks are maintained for both the actor and the critic to stabilize learning.

Exploration is driven by adding Ornstein-Uhlenbeck (OU) noise to the actions. This process introduces temporally correlated perturbations, which are suitable for smooth action spaces. Experiences from all agents are stored in a shared replay buffer in the form (state, action, reward, next_state, done).

Instead of learning after every step, learning occurs intermittently. After enough experiences have been collected, multiple learning iterations are performed. During each iteration, a batch of experiences is sampled from the buffer. The critic network is updated to minimize the mean squared error between predicted and target Q-values. The actor network is updated using the deterministic policy gradient, maximizing the critic's evaluation of predicted actions. Soft updates are applied to the target networks to slowly track the local networks and ensure stability.

Model Architecture: **Critic Network**

Critic Network is a fully-connected feedforward neural network with the following layers:

1. Layer 1: Linear(state_size, 256), followed by **Leaky ReLU**.
2. Layer 2: Linear(256+action_size, 256), followed by **Leaky ReLU**.
3. Layer 3: Linear(256, 128), followed by **Leaky ReLU**.
4. Layer 4: Linear(128, 1), no activation (outputs Q-value).

Model Architecture: **Actor Network**

Actor Network is a fully-connected feedforward neural network with the following layers:

1. Layer 1: Linear(state_size, 256), followed by ReLU.

2. Layer 2: Linear(256, action_size), followed by Tanh activation. Then the values are clipped so that the actions have a value in the range [-1, 1]

In version 2 of the Reacher environment, the state size is of length 33 per agent (i.e. 660 in total) while the action size is 4 continuous actions for each agent (i.e. 80 in total).

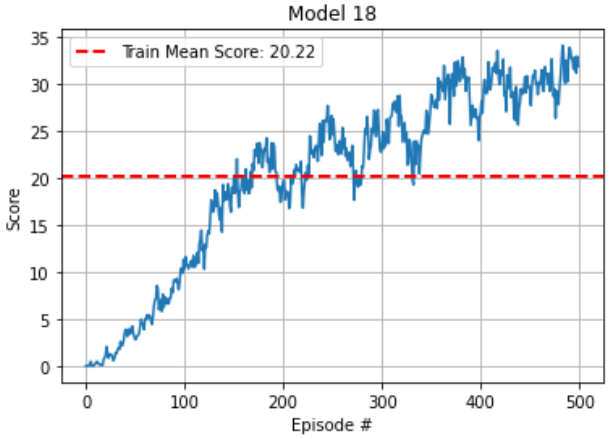
Chosen Hyperparameters

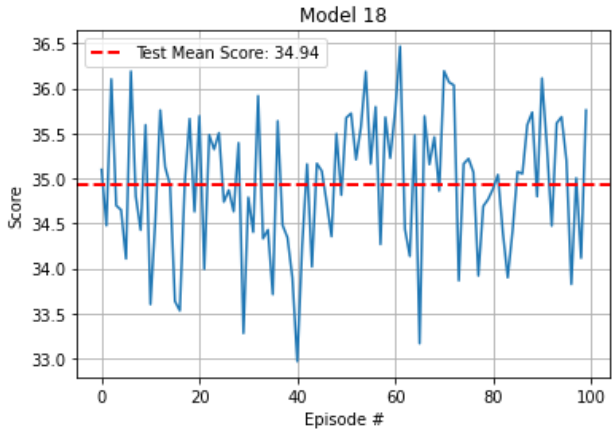
After training 3 models with different hyperparameters, the hyperparameters in the table below achieved the best results:

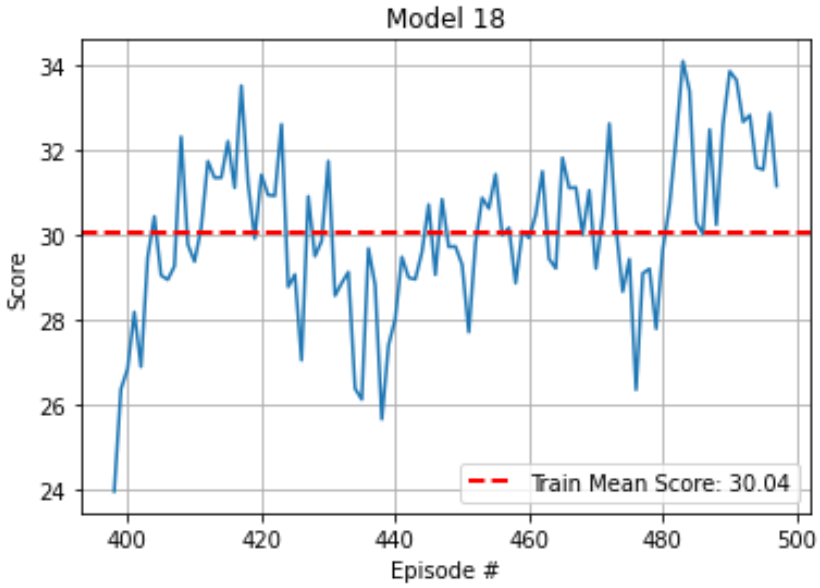
Hyperparameter	Value	Description
n_episodes	500	Number of training episodes
max_t	1000	Max steps per episode
BUFFER_SIZE	1,000,000	Size of replay memory
BATCH_SIZE	128	Mini-batch size for training
GAMMA	0.99	Discount factor for future rewards
TAU	0.001	Soft update interpolation parameter
LR_ACTOR	0.0001	Learning rate for Adam optimizer for the Actor Network
LR_CRITIC	0.001	Learning rate for Adam optimizer for the Critic Network
LEARN_EVERY	20	How often (in steps) the agent should update its networks
LEARN_ITERATIONS	20	How many training iterations are performed each time learning is triggered

Best Model Results

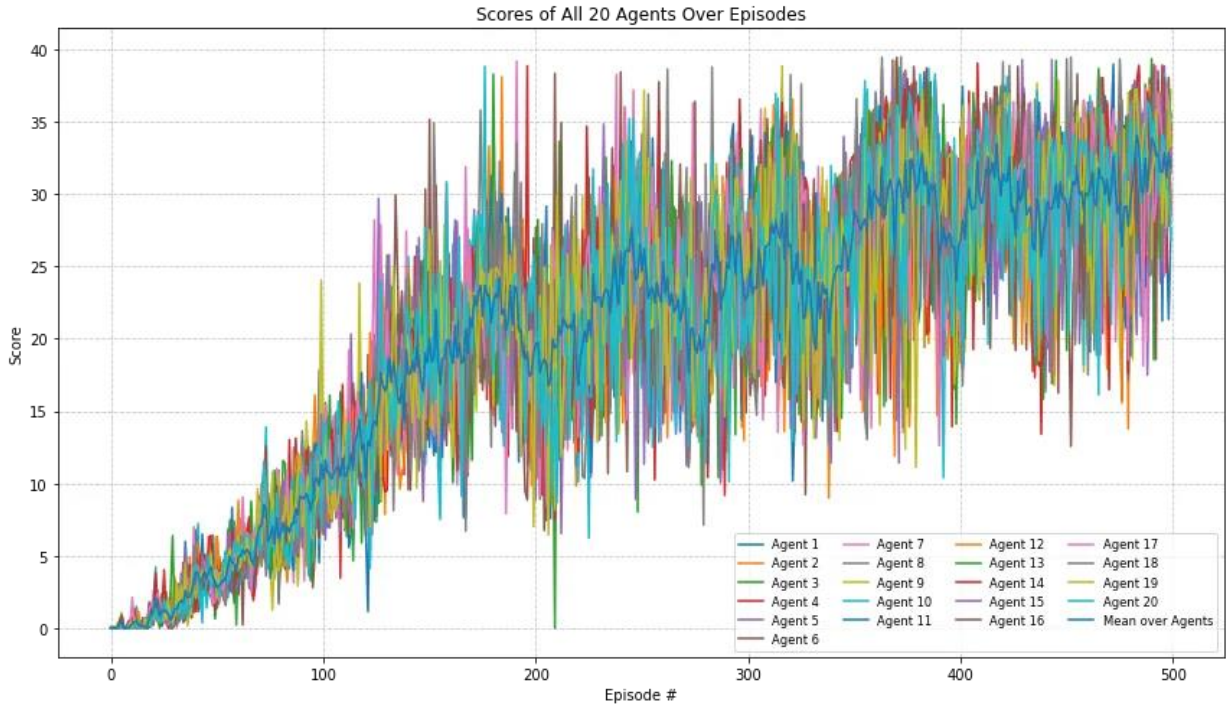
The model was trained and tested on version 2 of the Reacher environment. The following tables show the mean reward score in the best model for the training and the testing respectively. The environment was solved at episode 398 as can be seen in the zoomed in view of the training episodes.

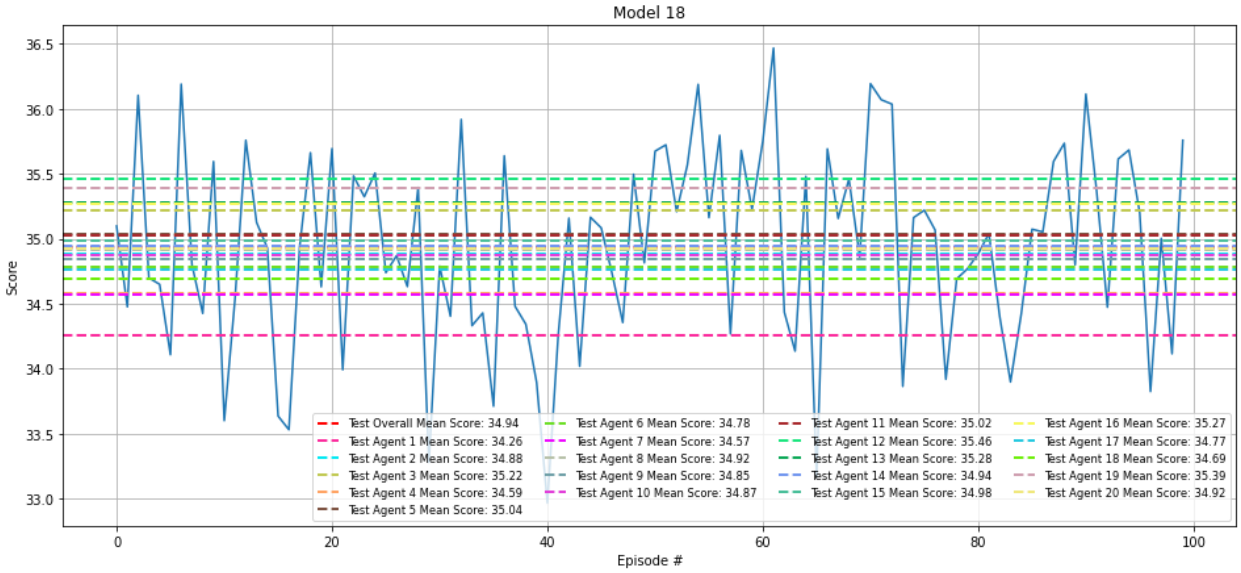






The graph below is the in-depth figure of the training scores on all 20 agents over the 500 training episodes. The last graph shows the overall mean score in the testing and the mean score of each agent over all the 100 test episodes. It is clear that all of the 20 agents surpass the 30.0 score needed to solve the environment.





Ideas for Future Work

- Distributional Critics (True D4PG): Extend the current implementation to include Multi-critics and N-step output for a more enhanced training.
- Prioritized Experience Replay: Instead of uniform sampling, use prioritized sampling based on TD error to improve sample efficiency.
- Exploration Strategies & Hyperparameter Tuning: Explore other forms of noise (e.g., Gaussian, parameter noise), and experiment with different values for TAU, LR_ACTOR, and LR_CRITIC to further improve learning stability and convergence speed.