**AI Project Final Report**

**Literature Search:**

For this project, the group has decided to focus on implementing reinforcement learning within Unity’s [1] 3D simulation environment. This game development platform was selected because of its available content for implementing custom reinforcement learning algorithms using the ML-Agents package [2]. Additionally, Unity has been compared to other simulation platforms [4] and was found to be an effective open-source tool for developing AI agents. There are several example environments [3] that demonstrate the wide range of reinforcement learning applications that can be constructed.

Having decided on a particular software and selecting the ML-agents Unity asset package as a primary resource, a decision was made about whether to create a reinforcement learning application for a real-world task or for a video game AI. The group preferred the former, and additional research was conducted to identify an area serving this purpose. This was found to be robotics, an application-heavy field that is quickly incorporating reinforcement learning to automate robotic tasks across many industries [5]. Sharing a mutual interest in drone technology, the group members have chosen to adapt a similar variation of the work done in [6] within Unity.

**Problem Statement:** *Clearly state the research problem that is being addressed.*

Similar to the objective described in [6], this project aims to control a quadrotor drone for stabilization and navigation by using reinforcement learning. This will be developed entirely in simulation, however in the final presentation an outline of how this can be applied to a physical system will be included.

**Methods**: *Conduct a quick literature search and list a few methods that will be explored. This does not have to be the final list or a detailed review.*

The example environments in [3] utilize many default reinforcement learning algorithms, including proximal policy optimization (PPO), soft actor-critic (SAC), and Deep Q-learning (DQL). In most modern implementations, a neural network is incorporated to help approximate a reward function that will build an effective policy pi(A|S), as shown in the following figure.

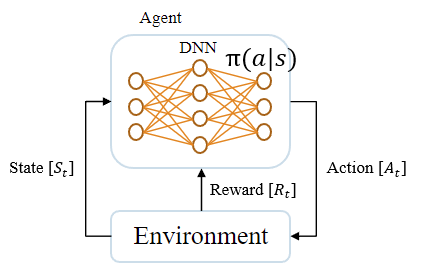


Figure: Deep Neural Networks in Reinforcement Learning

These methods will be tested with the selected application and compared based on their ability to consistently produce a reliable and effective policy after training. Additionally, some parameters will be modified and studied for the best overall method, including the learning rate and target network update rate, as shown in the figure below.

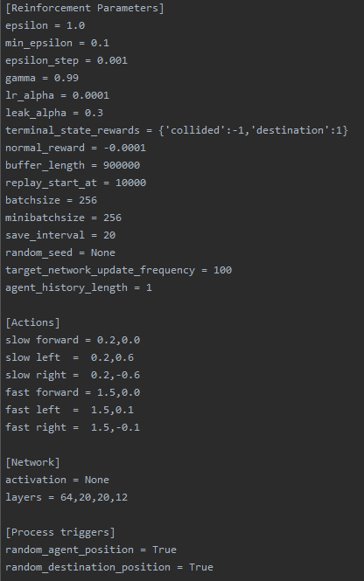


Figure: ML-Agents Reinforcement Learning Configuration Parameters

**Experimental Setup:** *Describe the experimental setup by listing which metrics and datasets will be used for evaluation.*

First and foremost, a custom simulation training environment will be constructed in Unity for implementing reinforcement learning with the chosen application. After configuring the GameObject blocks and necessary components to train a network throughout the learning process, each of the algorithms stated above will be applied to the simulation. Each method will be compared based on the speed of learning an approximated optimal policy as defined by observing the average reward across all episodes, the overall performance of the policy after a fixed number of episodes, and the average number of samples needed for training during each episode. For the particular task being studied in this project, these metrics have been identified in [7], a review of deep reinforcement learning for drone applications.

*Tutorials & references:*

[Training a Virtual Drone Using Machine Learning](https://www.youtube.com/watch?v=6LxjUvXOo74) [11]

<https://unitylist.com/p/1252/AI-Drone-Unity-Simulation> [13]

*ML-agents version required: 1.0.8*

*ML-agents extension package installation instructions:*

[*https://github.com/Unity-Technologies/ml-agents/blob/release\_8/docs/Installation.md#install-the-comunityml-agents-unity-package*](https://github.com/Unity-Technologies/ml-agents/blob/release_8/docs/Installation.md#install-the-comunityml-agents-unity-package)

[*https://forum.unity.com/threads/unity-does-not-recognise-mlagents-namespace.947286/*](https://forum.unity.com/threads/unity-does-not-recognise-mlagents-namespace.947286/)

*Drone asset package [10]:*

Download and import the following asset package in the Unity Package Manager:

<https://assetstore.unity.com/packages/tools/physics/free-pack-117641#content>

*Making a custom learning environment with ml-agents [8]:*

Three steps are involved in creating a new training environment in Unity with ml-agents:

1. Create an environment where the agent will interact with its surroundings.
   1. First, a ground plane is added to the empty scene at position [0, 0, 0] under a new folder named ‘TrainingArea’.
   2. Second, a target cube named ‘Target’ is added at position [3, 0.5, 3].
2. Implement the custom agent subclasses, where code definitions specify the agent’s observations, action selection, and reward function to be used.
3. Add the agent’s subclasses to the GameObject representing the agent model in simulation.

During each episode of training, if the agent (drone) achieves its goal (reaching a destination cube), falls off the map / crashes, or reaches the time limit, the episode terminates and the goal is relocated to a new random position. The scene is then randomized to promote learning in a variety of conditions.

A reference to the Rigidbody component of the agent is needed to reset the agent’s velocity and apply actions (forces) to its actuators. There are therefore 4 continuous actions, a force applied to each thruster of the quadcopter. The observations made by the agent’s sensors are sent to the ‘Brain’, analogous to a trained policy, where decisions (actions) are made based on the observations. This sensor data is used as input to a neural network as a feature vector. In this implementation we follow an approach similar to the one taken in [6], where the state observations include the agent’s displacement relative to the target cube, the agent’s velocity, and its orientation.

The agent model needs the following essential components (scripts):

1. <Custom> Agent:
   1. Target: Target (Transform)
   2. Force Multiplier: 10
2. Behavior Parameters:
   1. Behavior Name: <Custom>Agent
   2. Vector Observation:
      1. Space Size: 10
      2. Stacked Vectors: in scenarios where a vector of observations need to be remembered or compared over time, stacking the vectors provides the agent with memory without using an RNN.
   3. Actions:
      1. Continuous Actions: 4
      2. Discrete Branches: 0
3. Decision Requester:
   1. Decision Period: 1

*Running the simulator:*

1. Change directories into the project package containing the modified ml-agents package and activate the virtual python environment (3.7).
2. To start training from the beginning, within the directory containing ml-agents:

$ mlagents-learn ml-agents/config/ppo/DroneEnv\_1.yaml --run-id=drone\_test

1. To monitor the training process, use Tensorboard to visualize the cumulative reward and value estimates:

$ tensorboard --logdir summaries

*Trainer configuration:*

[*https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md*](https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md)

Summary frequency [10000]: number of experiences to be collected before generating training statistics (viewed in Tensorboard).

Time horizon [128]: how many steps to collect per-agent before adding it to the experience buffer. Shorter -> more biased, less varied value estimate. More ideal for very large episodes, or if there are frequent rewards within an episode. Should be large enough to capture the important behaviors within an action sequence.

Learning rate [0.0001]: initial learning rate (strength) for gradient descent updates. Should be decreased if training is unstable and the reward does not consistently increase.

Batch size (Continuous PPO: [1024]): number of experiences in each gradient descent iteration. Should always be a smaller multiple of the buffer size. Should be larger for continuous actions.

Buffer size [10240]: for PPO, the number of experiences to collect before updating the policy.

Learning Rate Schedule [PPO: linear]: how learning rate changes over time. Learning converges more stably when decaying until max\_steps for PPO.

Hidden units [128]: how many units in each fully connected layer. Should be larger when the actions taken have a complex interaction between the observations.

Number of layers [2]: number of hidden layers present after the observation input or CNN encoding of a visual observation. Fewer layers can be used for simpler problems to train fast & efficiently.

Normalize [true]: whether normalization is applied to the vector observation inputs. Can be helpful for complex continuous control problems.

*PPO-specific configurations:*

Beta [5.0e-3]: strength of entropy regularization, making the policy more random to ensure exploration. Should slowly decrease alongside an increasing reward. Increase if entropy drops too quickly.

Epsilon [0.2]: the acceptable threshold of divergence between old and new policies during updates which affects how rapidly the policy can evolve throughout training. Smaller values result in more stable updates, but slower training.

Lambda [0.95]: the regularization parameter for calculating generalized advantage estimate (GAE). Corresponds to how much the agent relies on the current value estimate when calculating an update. Lower values rely more on the current value estimate (can be high bias), higher values rely more on actual rewards received (can be high variance).

Number of epochs [3]: the number of passes to make through the experience buffer when optimizing with gradient descent. Larger values used when the batch size is larger. Decreasing provides more stable updates, with a slower learning process.

*Memory-enhanced Agents using Recurrent Neural Networks:*

Memory size [64-128]: size of the memory kept by the agent. Must be a multiple of 2 and scale with the amount of information the agent is expected to remember to successfully complete the task.

Sequence length [64]: how long the sequences of experiences must be while training. Larger values take longer to train, but more memory is kept. NOTE: LSTM does not work well with continuous actions.

**(Optional) New Approach**: *If you are planning to come up with a novel idea, provide a rough outline of the research approach.*

If the task can be accomplished successfully using the standard reinforcement learning methods above, and if time permits, a modification / extension of the best method may be applied.

**GITHUB:** *Set up a GitHub repository where all the implemented methods will be hosted. Add me to the repo -* [*https://github.com/kevinpdesai*](https://github.com/kevinpdesai)

**References**

[1] https://docs.unity3d.com/Manual/UnityManual.html

[2] https://github.com/Unity-Technologies/ml-agents

[3] https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Learning-Environment-Examples.md

[4] Juliani, A., Berges, V. P., Teng, E., Cohen, A., Harper, J., Elion, C., ... & Lange, D. (2018). Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.

[5] Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238-1274.

[6] Hwangbo, J., Sa, I., Siegwart, R., & Hutter, M. (2017). Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, *2*(4), 2096-2103.

[7] Azar, A. T., Koubaa, A., Ali Mohamed, N., Ibrahim, H. A., Ibrahim, Z. F., Kazim, M., ... & Casalino, G. (2021). Drone deep reinforcement learning: A review. *Electronics*, *10*(9), 999.

[8] <https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Learning-Environment-Create-New.md>

[9] Juliani, A., Berges, V., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D. (2020). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809.02627. <https://github.com/Unity-Technologies/ml-agents>.

[10] <https://www.reddit.com/r/Unity3D/comments/e60jts/100_physics_based_drone_pack_free_on_the_unity/>

[11] https://www.youtube.com/watch?v=6LxjUvXOo74

[12] https://github.com/dracolytch/ML-Simplest-Scenario

[13] https://unitylist.com/p/1252/AI-Drone-Unity-Simulation

[14]