Teaching a Digital Humanoid to Walk Using Reinforcement Learning in the Unity Engine

Sam Edwards

Abstract

This paper documents a framework for training a bipedal humanoid using the Unity physics engine and the ML-Agents Toolkit. The central thesis is that stable locomotion relies heavily on a carefully engineered, multi-component reward function rather than solely on algorithmic selection. By leveraging a continuous feedback loop of real-time observations and rewards, the system allows the agent to iteratively refine its policy. This dynamic pipeline enables the emergence of a coherent walking gait through trial-and-error learning.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

The core objective of this project is to develop and document a process for training an intelligent agent within the Unity game engine [6]. By utilizing the powerful physics simulator in Unity and the ML-Agents Toolkit, we can create a virtual environment where an agent learns through thousands of trial-and-error cycles. These cycles are run at an accelerated speed (e.g., 20x) to facilitate rapid data collection and learning.

II. Foundations: Background and Core Concepts

A. Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning in which an autonomous agent learns to make optimal decisions to achieve a specific goal. Unlike other machine learning paradigms, the agent is not given explicit instructions or labeled data. Instead, it acquires a "policy"—a strategy for choosing actions—by interacting with an environment [2], [3].

The RL framework consists of a few key components (see Fig. 1):

- **Agent:** The learner and decision-maker (in this case, the bipedal humanoid).
- **Environment:** The world the agent interacts with (the Unity scene).
- **State:** A snapshot of the environment and the agent at a particular moment (e.g., joint positions, velocity).
- **Action:** A decision made by the agent to interact with the environment (e.g., applying torque to a joint).
- **Reward:** A feedback signal from the environment that tells the agent how good or bad its last action was. The agent's sole objective is to maximize the cumulative reward over time.

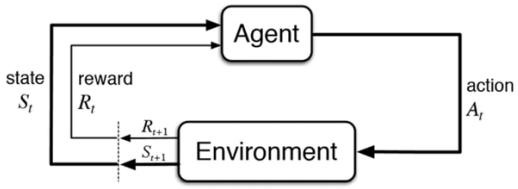

Fig. 1: A standard diagram for a reinforcement learning algorithm loop in which an agent is rewarded or penalized for the action it takes in its environment. The agent takes action A_t in state S_t, and the environment returns a new state S_{t+1} and a reward R_{t+1}.

B. Neural Network as Policy

In modern deep reinforcement learning, the policy of the agent is represented by a neural network. This network functions as the "brain" of the agent, mapping the high-dimensional state (observations) to the best possible actions. For example, it takes inputs like joint rotations and velocities and outputs the optimal torques to apply to each joint. This policy network is trained using the feedback from the reward function. The reward signal is used to perform backpropagation, updating the network's weights and biases to favor actions that lead to higher cumulative rewards.

C. PPO Algorithm

To train the agent, this project employs Proximal Policy Optimization (PPO) [8], the default algorithm provided within the ML-Agents Toolkit. PPO was selected for this task over other policy gradient methods due to its combination of sample efficiency, ease of implementation, and robust stability in high-dimensional continuous control tasks like bipedal locomotion.

PPO builds on an "Actor-Critic" architecture. This structure involves two distinct neural networks that work in tandem:

- **The Actor (Policy):** This network takes the agent's state (observation) and decides on an action to take (e.g., which torques to apply). It is the policy itself.
- **The Critic (Value Function):** This network observes the agent's state and estimates the expected future cumulative reward from that state. Its job is not to act, but to evaluate how good the current state is.

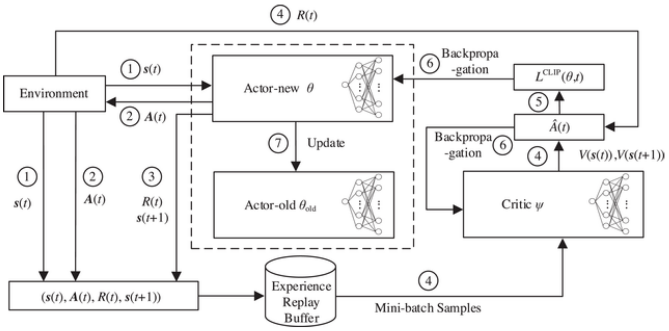

Fig. 2: A diagram illustrating the PPO Actor-Critic architecture. The Actor network selects an action, while the Critic network evaluates the state. The resulting experience is used to update both networks.

The core challenge PPO addresses is preventing destructively large policy updates during training [9]. It accomplishes this by using the Critic's evaluations to calculate an "Advantage" (how much better or worse the Actor's action was than the Critic's expectation). This advantage is then used to update the Actor's policy, but only within a "clipped" surrogate objective function. This clipping mechanism, the key innovation of PPO, ensures that the change between the old policy and the new policy remains small, preventing the agent from "forgetting" what it has learned and ensuring a stable, iterative learning process. This is a more robust and simpler-to-implement approach than its predecessor, Trust Region Policy Optimization (TRPO) [7].

D. Reward Function Design

The design of the reward function is the most critical element, as it implicitly defines the goal of the task and breaks down the end goal into smaller, simpler objectives. Instead of a single, sparse reward for "walking," we use a multi-component, or composite, reward function (see Table I) that is calculated at each step to guide the agent toward the desired emergent behavior.

This function is a carefully tuned combination of positive incentives and negative penalties. The primary positive drivers

TABLE I: Reward Function Components and Goals

Component	Goal	Value Type
Stand Tall	Encourages lifting the torso. Continuous curve favors "trying" over lying flat.	Continuous $[0, 1]$
Move Forward	Direct reward for velocity in the target direction.	Continuous $[-1, 1]$
Alternating Gait	Returns true only if exactly one foot is grounded. Punishes hopping.	Triggered Bonus
Existence	Small survival bonus to counter the urge to terminate early if it perceives it cannot win.	Constant
Critical Failure	Immediate fail state for collapsing or falling below minimum height.	Terminal Penalty

are centered on balance and locomotion. To encourage balance, the agent receives a constant small reward for "staying upright," which is calculated by measuring the dot product of the head's "up" vector against the world's "up" vector. This is supplemented by a reward for "standing tall," which provides a small bonus based on the agent's height, incentivizing it to keep its posture erect.

To encourage locomotion, a constant reward is awarded when the agent's root velocity is moving in the target's forward direction. This component guides the agent to explore movement, not just static balancing.

These rewards are balanced by penalties designed to promote efficiency and punish failure. To encourage "smooth movements," a penalty is applied that is proportional to the square of the actions (torques) applied. This heavily penalizes jerky, erratic, high-force movements and promotes an energy-efficient gait.

Finally, a large terminal penalty is applied for catastrophic failure. If the agent falls, holds an improper stance for too long, or fails to move forward, the episode is terminated, and a significant negative reward is applied. This creates a strong incentive to avoid these failure states and, in combination with the positive rewards, guides the policy toward a stable and efficient walking gait [10], [11].

III. Methodology: Environment and System Design

The success of the training process is highly dependent on a well-defined environment and a physically plausible agent.

- **Training Environment:** The initial training environment will be a simple, flat plane. This minimalistic design is intentional, as it isolates the learning problem to the core challenges of balance and locomotion, removing confounding variables like complex terrain.
- **Agent Definition:** The agent will be a standard bipedal humanoid model (Fig. 3). It will be articulated by 13 primary rigid bodies (representing the torso, pelvis, head, thighs, shins, and feet) connected by Configurable Joints. These joints, particularly at the hips, knees, and ankles, will be constrained to mimic human biomechanics and degrees of freedom.
- **Observation Space:** The agent's policy will be informed by a list of normalized floating-point numbers. This set of observations is crucial for the agent to perceive its state relative to the environment. The observations will include:

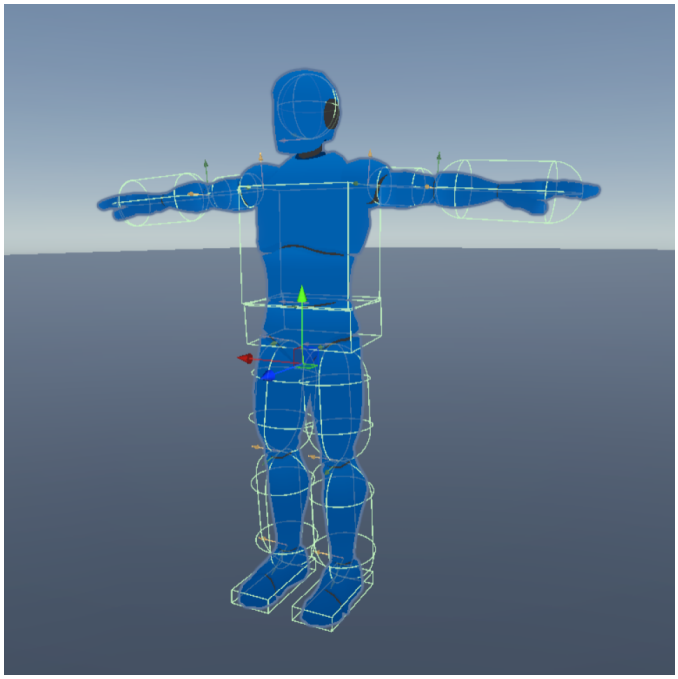

Fig. 3: The bipedal humanoid model in Unity is shown here. The visual highlights the component Rigidbodies, Configurable Joints, and hitboxes that enable articulation.

  - Root Body State: The position, rotation, velocity, and angular velocity of the torso.
  - Limb States: The rotation, velocity, and angular velocity for each of the other body parts.
  - Joint States: The angular position and rotational velocity of each configurable joint.
  - Ground Truth: A boolean value for each foot indicating if it is in contact with the ground.
  - Target Direction: A normalized vector pointing in the desired direction of travel.
- **Action Space:** The agent will operate in a continuous action space. It will control its body by applying torque to its joints. The action vector will consist of continuous values, each normalized to the range [-1, 1], corresponding to the target rotational "drive" for the hip, knee, and ankle joints.

IV. Explorations: Prototype Implementation and Results

A. Prototype Setup

The experimental prototype utilized the bipedal agent configuration defined in Section III within a controlled training environment. To maximize sample efficiency, the training process utilized parallelization, running 64 concurrent agent instances simultaneously within the same Unity scene.

The agent operated within a continuous action space of 10 dimensions, corresponding to the torque forces applied to the hip, knee, and ankle joints. This was coupled with an observation space of size 65, which provided the neural network with a dense vector of proprioceptive data, including

joint angular velocities, torso orientation, and center-of-mass velocity.

The time scale of the simulation was accelerated to 20 times normal speed ($20\times$), allowing for rapid data collection. The model was trained for a total duration of approximately 1 million iterations. This setup allowed the PPO algorithm to gather a diverse set of experiences across the multiple instances, averaging the gradient updates to ensure stable learning.

B. Training Results

The training metrics indicated a successful, albeit imperfect, result. The agent began the training session with a mean cumulative reward of 0, indicating immediate failure or random noise. Over the course of the 1 million iterations, the learning curve demonstrated a consistent upward trend. Qualitatively, this corresponded to a distinct split in behavioral success: while the agent successfully learned to maintain a standing posture, it failed to achieve a natural walking gait. Instead, forward locomotion resulted in suboptimal policies characterized by short hops, shuffling, or other unconventional methods of displacement.

By the conclusion of the training sessions, the policy stabilized, with agents achieving cumulative rewards fluctuating between 50 and 100 points per episode. This variance in the final reward range is attributed to the stochastic nature of the survival times; while most agents could sustain balance, the duration they remained upright varied before the eventual termination of the episode.

C. Emergent Behaviors

The resulting policy exhibited distinct emergent behaviors that satisfied the mathematical requirements of the reward function but deviated from naturalistic human motion. Three primary behavioral patterns were observed:

1) **Posterior Counter-Balancing:** The agent adopted a strategy of leaning its torso significantly backward. This "zombie-like" posture appeared to be a mechanism to counterbalance the center of mass, preventing forward falls.
2) **Ankle Torque Exploitation:** The agent learned to apply abnormal amounts of force to the ankle joints. This allowed the model to maintain a standing position while leaning forward at biomechanically unreasonable angles, effectively exploiting the physics engine's allowance for high torque to defy gravity.
3) **Stability via Crouching:** Ultimately, the agent converged on a crouching gait. By lowering its center of mass, the agent found it easier to align its body vertically—maximizing the "staying upright" reward component—without the instability inherent in fully extending the knee joints.
4) **Propulsion via Shuffling:** Forward locomotion was not achieved through a continuous gait cycle but rather through short, disjointed hops and shuffling motions. This strategy minimized the duration of single-leg support phases, allowing the agent to move forwards while reducing the risk of falling.

D. Interpretation

The results highlight a classic reinforcement learning challenge: specification gaming. The agent successfully maximized the reward signal (staying off the ground and moving forward) but did so by finding local optima—crouching and extreme leaning—that were not intended by the design.

The interpretation of these results suggests that the current reward function provides a solid foundation for balance but lacks sufficient constraints on posture. To achieve a more natural gait, future training runs must introduce stricter penalties for deviation from a vertical root rotation and potentially limit the maximum torque available to the ankles. However, the underlying logic of the current reward structure should be retained, as it successfully taught the agent the fundamental physics of preventing a fall.

V. Futures: Evidence-Based Short-Term Predictions

A. Future Work

To bridge the gap between static stability and dynamic locomotion, future iterations of this research will prioritize simplification of the 3D model and reward function regularization. Specifically, a shift to a humanoid rig with fewer degrees of freedom or strictly limited joint ranges to increase inherent mechanical stability.

Concurrently, the reward function will be simplified to reduce the agent's reliance on complex shaping terms. By reducing the noise introduced by an overly complex physical model, the objective is to successfully train an agent that can transition smoothly from a standing state to a continuous, naturalistic walking gait without resorting to the shuffling behaviors observed.

B. Justifications

These proposed changes are directly justified by the suboptimal policies observed in the training results. The emergence of "zombie-like" leaning and "ankle torque exploitation" suggests that the current model's complexity overwhelmed the learning algorithm, trapping the agent in local optima where staying upright took precedence over forward momentum.

- **Complexity:** The inability to achieve a standard gait indicates that the current rig possesses a high-dimensional action space that is difficult to explore efficiently. Simplifying the physical components will reduce the search space, allowing the agent to discover fundamental locomotion patterns more rapidly.
- **Reward Signal Clarity:** The agent's tendency to "shuffle" rather than walk implies that the reward function, when applied to a complex rig, created sparse positive feedback for walking. A more stable rig allows for a cleaner reward signal, ensuring that forward locomotion is penalized less heavily by immediate instability.

VI. Conclusion

This project implemented a reinforcement learning framework for training a digital bipedal humanoid using the Unity

engine and the ML-Agents Toolkit. A successful pipeline was established for agent observations, actions, and reward signal processing, resulting in an agent capable of mastering static equilibrium. However, the emergence of idiosyncratic behaviors, specifically the "shuffling" and "zombie-like" postures, demonstrated the challenges of high-dimensional control and the sensitivity of reward shaping.

While the agent did not achieve a naturalistic human gait, the emergent strategies provided critical insights into the relationship between model and action space complexity and training stability. This work serves as a foundational step, identifying the necessary simplifications required to create more intelligent, adaptive, and believable digital characters.

APPENDIX

The development of this bipedal locomotion agent relied heavily on technical competencies established during previous Video Game Design and Development projects within the Unity engine. These earlier projects provided the essential scaffolding for the current simulation, specifically regarding the manipulation of Rigidbody physics, collider interactions, and the C# scripting API. Prior experience with the Unity Editor allowed for the rapid construction of the training environment and the configuration of the bipedal rig's joint constraints.

The conceptual framework for the agent's learning process was influenced by a *AI Self-Driving Car Agent Design* project. That coursework introduced the fundamental loops of agent-environment interaction, sensor processing, and autonomous decision-making. While the self-driving car relied on ray-casting and navigation meshes for steering behaviors, this project expanded those concepts into the domain of continuous control. Similarly, a *Machine Learning Music Classification* project provided critical experience in analyzing high-dimensional data and interpreting training metrics. The ability to read loss curves and recognize overfitting—skills honed while classifying audio features—was instrumental in monitoring the PPO algorithm's performance and recognizing when the agent was exploiting the reward function rather than learning the intended task.

This project highlighted the connection between deterministic software engineering and probabilistic machine learning. Unlike previous game design projects where behavior was explicitly hard-coded, this work required relinquishing control to the stochastic nature of Reinforcement Learning. It showcased the complexity of reward shaping, demonstrating that defining *what* an agent should do is often harder than programming *how* to do it. By forcing the convergence of game physics and neural network training, this project provided a practical demonstration of the challenges inherent in high-dimensional continuous action spaces.

REFERENCES

[1] Al Warehouse. 2023. "AI Learns to Walk (Deep Reinforcement Learning)." YouTube, April 23, 2023. https://www.youtube.com/watch?v=L_4BPjLBF4E

[2] AltexSoft Editorial Team. 2019. "Reinforcement Learning Explained: Overview, Comparisons and Applications in Business." AltexSoft, January 21, 2019. https://www.altexsoft.com/blog/reinforcement-learning-explained-overview-comparisons-and-applications-in-business/

[3] Gonkee. 2024. "The FASTEST Introduction to Reinforcement Learning on the Internet." YouTube, December 23, 2024. https://www.youtube.com/watch?v=VnpRp7ZglfA

[4] Graphics in 5 Minutes. 2023. "Reinforcement Learning from Scratch." YouTube. August 14, 2023. https://www.youtube.com/watch?v=vXtfdGphr3c

[5] Huang, Tao, Zhibo Chen, and Jianyuan Wang. 2022. "Learning Humanoid Standing-up Control across Diverse Postures." *arXiv*, October 25, 2022. https://arxiv.org/abs/2210.14158

[6] Juliani, Arthur, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. 2020. "Unity: A general platform for intelligent agents." *arXiv preprint arXiv:1809.02627*. https://arxiv.org/pdf/1809.02627.pdf

[7] Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. "Trust Region Policy Optimization." *arXiv*, February 17, 2015. https://arxiv.org/abs/1502.05477

[8] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017). https://arxiv.org/abs/1707.06347

[9] Wang, Y., He, H., and Tan, X. "Trust Region-Guided Proximal Policy Optimization." *arXiv preprint arXiv:1901.03030* (2019). https://arxiv.org/abs/1901.03030

[10] Song, S., et al. "Emergence of natural and robust bipedal walking by learning from biologically plausible objectives." *Nature Communications Biology* (2024). https://pmc.ncbi.nlm.nih.gov/articles/PMC12002607/

[11] Yoshida, E., et al. "Reinforcement Learning of Bipedal Walking Using a Simple Reference Motion." *Applied Sciences* 14, no. 5 (2024). https://www.mdpi.com/2076-3417/14/5/1803