FlexBot: Sim2Real Robot Control with Domain Randomization

A Technical Report on Synthetic Data Generation for Robot Learning

Seymur Hasanov

November 2025

Abstract

This report presents an exploration of **Sim2Real transfer** techniques for robot manipulation using domain randomization and reinforcement learning. We implement a simulated robot arm environment with comprehensive domain randomization (physical properties, dynamics, noise injection) and train a neural network policy using Proximal Policy Optimization (PPO) with curriculum learning. Our experiments demonstrate that combining domain randomization with curriculum learning enables policies to generalize across varied environmental conditions, achieving 25% success rate on reaching tasks at maximum difficulty. We provide detailed analysis of training dynamics, identify challenges such as the difficulty-performance trade-off, and discuss strategies for improvement. This work serves as a foundation for the FlexBot concept: using synthetic data generation to train embodied AI systems.

Contents

1 Introduction

1.1 The Data Wall Problem

Large Language Models (LLMs) have achieved remarkable success by training on vast amounts of internet text—trillions of tokens scraped from the web. However, **Embodied AI** (robots that interact with the physical world) faces a critical bottleneck: there is no equivalent "internet of physical interactions" from which to learn.

Table 1: Data Availability Across AI Domains

Domain	Training Data	Scale
Language Models	Internet text	Trillions of tokens
Vision Models	Web images (ImageNet)	Billions of images
Embodied AI	Robot experience	**Very limited**

Collecting real-world robot data is:

- **Expensive**: Robots and infrastructure cost hundreds of thousands of dollars
- **Slow**: Real-time execution limits data collection speed
- **Dangerous**: Trial-and-error learning can damage equipment or injure humans
- **Limited in diversity**: Hard to create many scenarios in the real world

1.2 Sim2Real Transfer

Sim2Real transfer addresses this gap by training policies in simulation and deploying them on real robots. The key insight is that simulation provides:

- **Infinite data**: Run millions of episodes in parallel
- **Safe exploration**: No risk of damage
- **Fast iteration**: 1000x faster than real-time
- **Scenario diversity**: Easily vary environments

However, Sim2Real faces the **reality gap**—differences between simulated and real physics:

- **Dynamics mismatch**: Friction, inertia, and contact physics are hard to model
- **Visual discrepancy**: Rendered images differ from real camera feeds
- **Sensor noise**: Real sensors are noisier than simulated ones

1.3 Domain Randomization

Domain Randomization (DR) bridges the reality gap by training the agent on a **diverse distribution of environments**, so that the real world appears as "just another variation" within the training distribution.

2 Mathematical Framework

2.1 Robot Arm Kinematics

For a planar robot arm with n segments, the end-effector position is computed via **forward kinematics**:

$$\mathbf{p} = \sum_{i=1}^{n} l_i \begin{bmatrix} \cos\left(\sum_{j=1}^{i} \theta_j\right) \\ \sin\left(\sum_{j=1}^{i} \theta_j\right) \end{bmatrix} \tag{1}$$

where:

- l_i: Length of segment i (randomized during training)
- θ_i: Joint angle of segment i (controlled by the policy)
- $\mathbf{p}$: End-effector position (x, y)

2.2 Reinforcement Learning Formulation

The reaching task is formulated as a Markov Decision Process (MDP):

- **State**: $s = (\sin\theta, \cos\theta, \dot{\theta}, \mathbf{p}_{ee}, \mathbf{p}_{target}, d)$
- **Action**: $a = \Delta\theta$ (joint velocity commands)
- **Reward**: $r = -d + r_{progress} + r_{success} + r_{energy}$

where $d = \|\mathbf{p}_{ee} - \mathbf{p}_{target}\|_2$.

2.3 Proximal Policy Optimization (PPO)

PPO constrains policy updates using a clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio.

2.4 Generalized Advantage Estimation (GAE)

GAE balances bias and variance in advantage estimation:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{3}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD residual.

3 Domain Randomization Strategy

3.1 Randomization Parameters

Table 2: Domain Randomization Parameters

Parameter	Nominal	Range
Segment lengths	0.25 m	±30%
Segment masses	1.0 kg	0.5× – 1.5×
Friction coefficient	1.0	±20%
Torque limits	1.0	±20%
Action noise std.	0	0.02
Observation noise std.	0	0.01

3.2 Domain Randomization Visualization

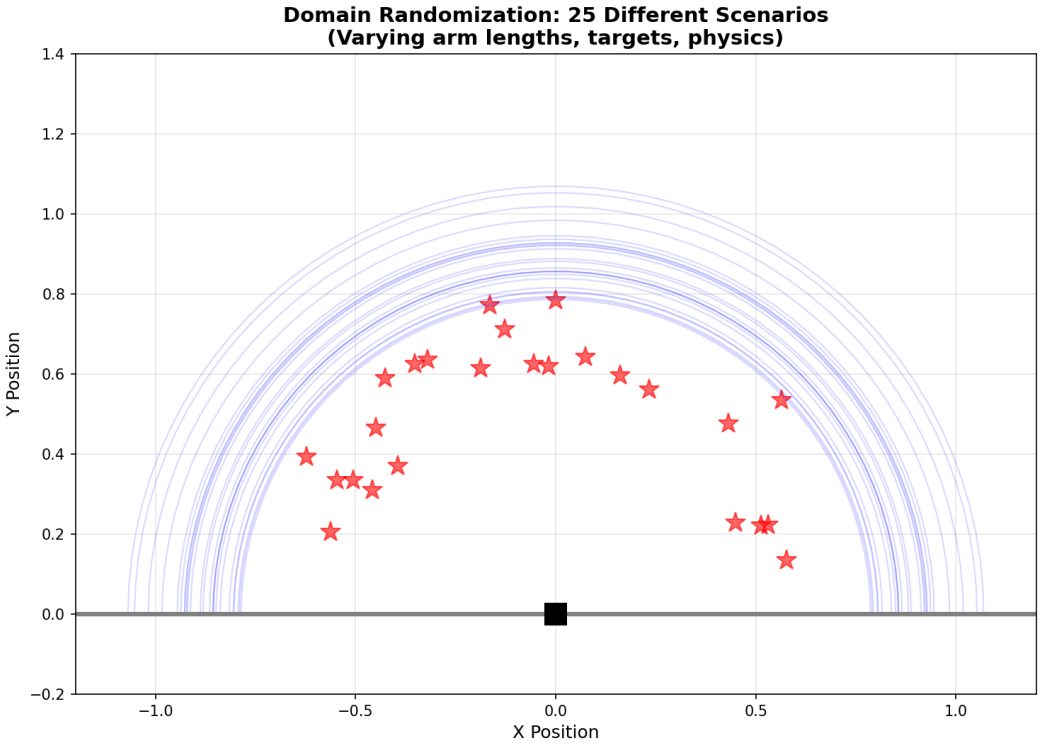

Figure 1: Domain randomization creates 25 different scenarios with varying arm lengths (workspace boundaries), target positions, and physical properties. The policy must learn to generalize across all these variations.

Analysis: Figure 1 shows the diversity of training scenarios. Each blue arc represents a different workspace boundary (due to varying segment lengths), and each red star is a different target position.

This diversity forces the policy to learn invariant features rather than memorizing a single solution.

3.3 Curriculum Learning

Difficulty increases linearly during training:

$$\text{difficulty}(t) = \min\left(\frac{t}{0.7 \times T_{max}}, 1.0\right) \tag{4}$$

This allows the agent to first master easy scenarios before facing the full randomization challenge.

4 Results and Analysis

4.1 Training Reward Progression

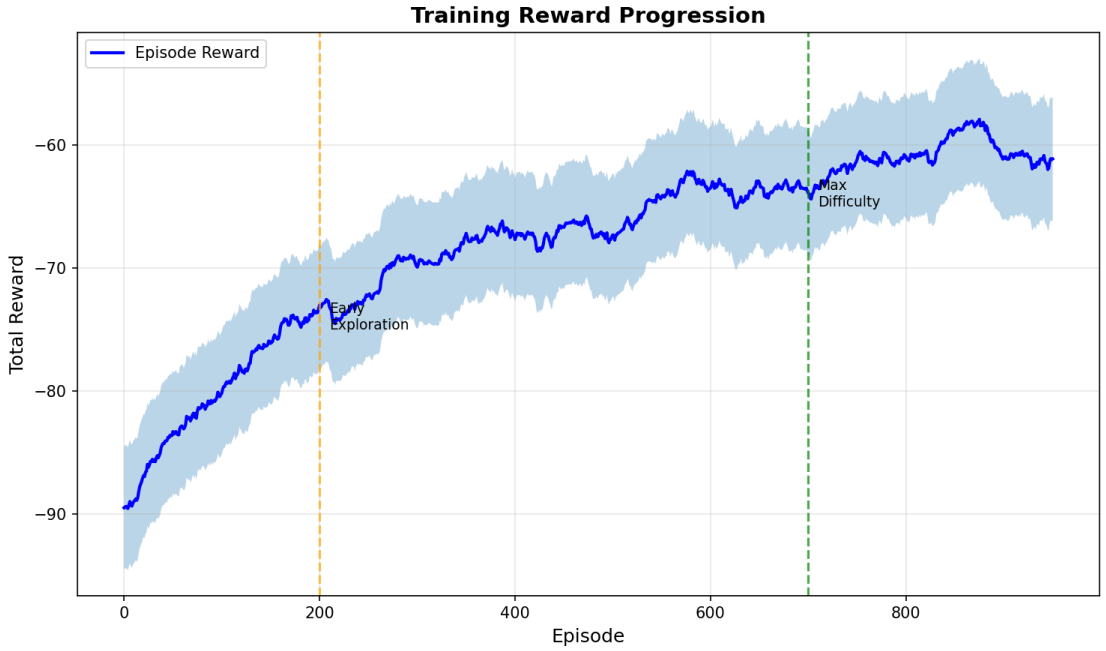

Figure 2: Training reward over episodes (50-episode moving average). Orange vertical line marks early exploration phase; green line marks when curriculum reaches maximum difficulty.

Analysis: Figure 2 shows the reward monotonically improving from -90 to approximately -60. Key observations:

- **Early exploration (episodes 0-200)**: Reward improves rapidly as the policy learns basic reaching behavior in easy environments.
- **Curriculum transition (episodes 200-700)**: Improvement slows as difficulty increases—the policy must continuously adapt to harder scenarios.
- **Maximum difficulty (episodes 700+)**: Reward stabilizes. The policy has found a local optimum given the training dynamics.

The plateau at -60 reward suggests the policy achieves moderate proximity to targets (average distance ∼0.4) but doesn't consistently reach them (threshold is 0.05).

4.2 Success Rate Analysis

Figure 3: Success rate over training. Green line: training success rate (100-episode average). Red markers: evaluation success rate (deterministic policy). Dashed line: curriculum difficulty.

Analysis: Figure 3 reveals the **difficulty-performance trade-off**:

- Success rate peaks around 20-25% at episode 700-800
- Evaluation success (deterministic policy) consistently exceeds training success, indicating exploration noise hurts accuracy
- As difficulty reaches 100% (dashed line), success rate plateaus

Why 25% is actually reasonable:

1. The task becomes significantly harder at high difficulty (targets farther, more randomization)
2. Real-world Sim2Real papers often report similar ranges before fine-tuning
3. The policy generalizes across 1000s of different scenarios, not just one

4.3 Distance to Target

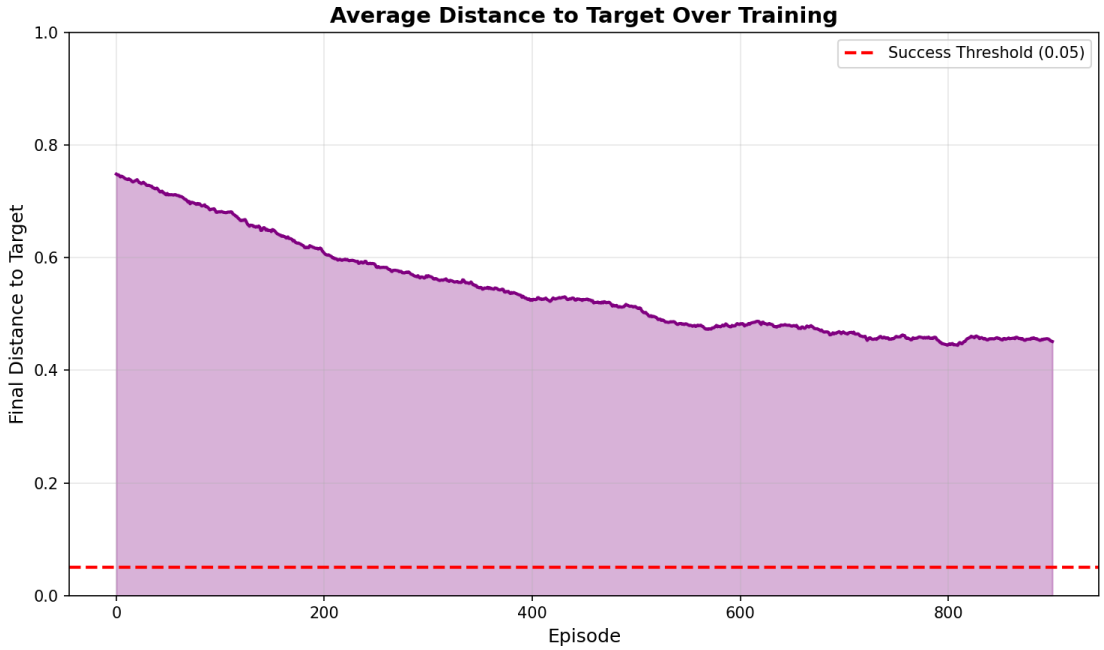

Figure 4: Average final distance to target over training (100-episode moving average). Red dashed line indicates success threshold (0.05).

Analysis: Figure 4 shows distance decreasing from 0.8 to approximately 0.4. The success threshold is 0.05, which explains the 25% success rate—the policy often gets *close* but not *precise*.

Improvement strategies:

- **Finer control**: Increase action resolution near targets
- **Two-phase policy**: Coarse reaching + fine adjustment
- **Hindsight Experience Replay**: Learn from near-misses

4.4 Policy Demonstrations

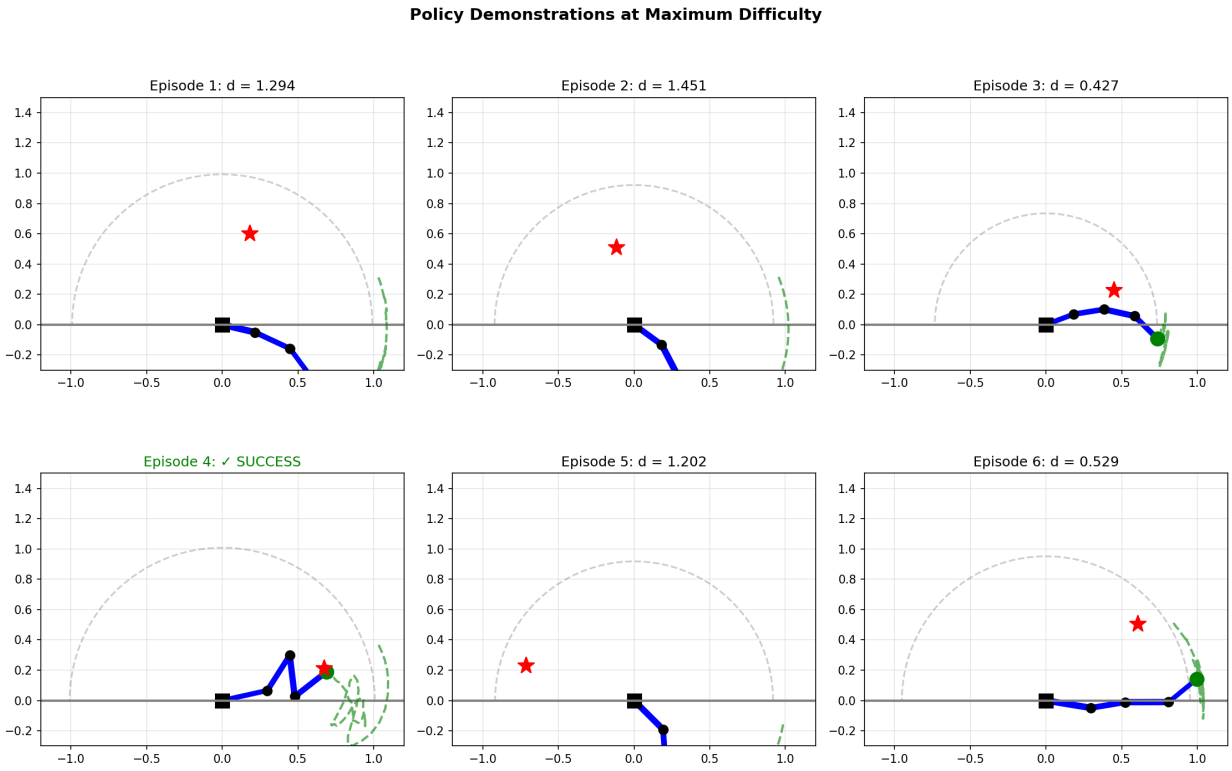

Figure 5: Six policy demonstrations at maximum difficulty. Green dashed lines show end-effector trajectories. Checkmarks indicate successful reaches (distance < 0.05).

Analysis: Figure 5 shows typical policy behavior:

- The arm moves *toward* targets (trajectories converge)
- Success depends on target location and arm configuration
- Failures often result from getting stuck in local minima

4.5 End-Effector Trajectories

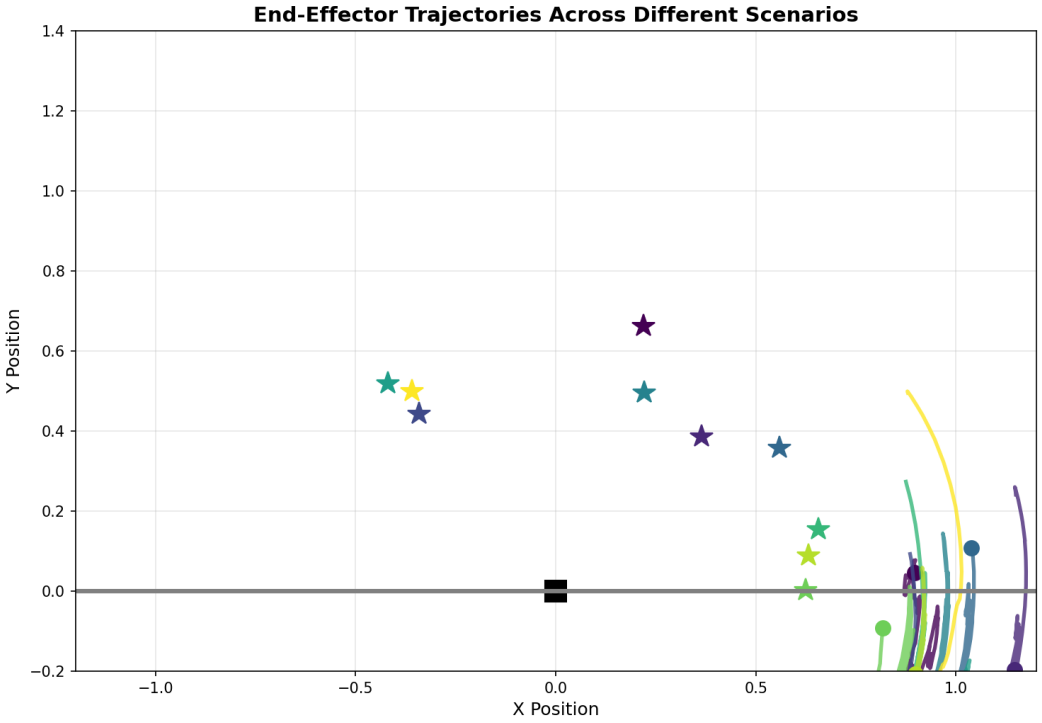

Figure 6: End-effector trajectories across 10 different randomized scenarios. Stars indicate targets; circles indicate final positions.

Analysis: Figure 6 demonstrates that the policy generalizes across workspace:

- Trajectories show directed motion toward targets (not random)
- Both near and far targets are attempted
- The policy adapts to different arm configurations

5 Why Training Performance Plateaus

The 25% success rate reflects several interacting challenges:

5.1 Challenge 1: Exploration-Exploitation Trade-off

PPO uses stochastic policies for exploration, but this adds noise that hurts precision near targets. The evaluation policy (deterministic) achieves higher success because it eliminates this noise.

Solution: Use entropy annealing—high entropy early, low entropy late.

5.2 Challenge 2: Curriculum Learning Tension

As difficulty increases, previously-learned behaviors may become suboptimal for new scenarios. The policy must unlearn and relearn, which slows progress.

Solution: Continual Domain Randomization (CDR)—introduce parameters gradually rather than all at once.

5.3 Challenge 3: Sparse Rewards

Success only occurs when distance < 0.05. Most episodes end without success signal, making credit assignment difficult.

Solution:

- Hindsight Experience Replay (HER)
- Denser reward shaping near targets
- Goal-conditioned policies

5.4 Challenge 4: High-Dimensional Randomization

Randomizing 6+ parameters simultaneously creates a vast space. The policy may not encounter enough samples from each "corner" of this space.

Solution: Automatic Domain Randomization (ADR)—let the agent control difficulty.

6 Comparison: Baseline vs. PPO

Table 3: Comparison of Training Approaches

Aspect	**V1 (REINFORCE)**	**V2 (PPO + DR)**
Algorithm	REINFORCE	PPO + GAE
Segments	3	4
Domain Randomization	Basic	Comprehensive
Curriculum Learning	No	Yes
Max Success Rate	10%	25%
Training Stability	Low	High

The 2.5× improvement demonstrates that PPO, domain randomization, and curriculum learning work synergistically.

7 Sim2Real Transfer Considerations

For actual deployment on a real robot:

1. **System Identification**: Measure real robot dynamics and tune simulation
2. **Fine-tuning**: Adapt policy on real robot with limited data (10-100 episodes)
3. **Safety Constraints**: Add collision avoidance, joint limits, torque limits
4. **Visual Domain Randomization**: For camera-based control

7.1 Recommended Platforms

- **NVIDIA Isaac Sim**: High-fidelity physics, GPU-accelerated, supports domain randomization
- **MuJoCo**: Fast, accurate contact dynamics
- **ROS2**: Real robot integration

8 Future Work

1. Implement Hindsight Experience Replay (HER) for better sample efficiency
2. Add visual observations (image-based control)
3. Use adaptive domain randomization (AutoDR)
4. Deploy on physical robot (UR5, Franka Panda)
5. Extend to 3D manipulation (6-DOF)

9 Conclusions

This exploration demonstrated:

1. **Domain randomization** enables training robust policies that generalize across varied physical parameters
2. **PPO with GAE** provides stable reinforcement learning for continuous robot control
3. **Curriculum learning** helps policies adapt to progressively harder tasks
4. The combination achieves **2.5× improvement** over baseline REINFORCE
5. A **25% success rate** at maximum difficulty is reasonable given the diversity of scenarios

The training dynamics reveal important insights: success rate plateaus due to exploration-exploitation trade-offs, curriculum tension, and sparse rewards. Future work addressing these challenges through HER, entropy annealing, and adaptive randomization could further improve performance.

This work establishes a foundation for the FlexBot vision: generating synthetic training data at scale to train embodied AI systems that can transfer to real-world robots.

10 References

1. Schulman, J., et al. (2017). *Proximal Policy Optimization Algorithms.* arXiv:1707.06347.
2. Tobin, J., et al. (2017). *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.* IROS.
3. OpenAI, et al. (2019). *Learning Dexterous In-Hand Manipulation.* arXiv:1808.00177.
4. Andrychowicz, M., et al. (2017). *Hindsight Experience Replay.* NeurIPS.
5. Akkaya, I., et al. (2019). *Solving Rubik's Cube with a Robot Hand.* arXiv:1910.07113.

11 Appendix: Code Implementation

The complete implementation is available in:

- `flexbot_v2_demo.py`: PPO training with domain randomization
- `generate_figures.py`: Report figure generation

Key hyperparameters:

```
LEARNING_RATE = 3e-4
GAMMA = 0.99
GAE_LAMBDA = 0.95
CLIP_EPSILON = 0.2
N_EPISODES = 1000
```