Investigating Reward Shaping Strategies for Improving PPO on Continuous Control Reacher-v5

Dinh Hung Nguyen (Hung)
Khoury College of Computer Science
Northeastern University
`nguyen.hun@northeastern.edu`

Abstract

I investigate how potential-based reward shaping strategies affect the learning efficiency and final performance of Proximal Policy Optimization (PPO) on the Reacher-v5 continuous control task. Several reward shaping variants based on tip-target distance, including both constant and time-decayed shaping terms, were implemented and evaluated against a PPO baseline without any shaping. Experiments show that reward shaping consistently improves success rates, with decayed shaping schedules achieving the best results; specifically, the decay300K variant trained for 1 million (1M) steps achieved a success rate of approximately 26%, outperforming constant-shaping variants. Correlation analysis revealed that higher success rates were more closely associated with smoother and more stable learning curves than with higher mean episode returns. These findings suggest that appropriately phased-out reward shaping can enhance both sample efficiency and policy robustness in reinforcement learning tasks.

1 Introduction

Proximal Policy Optimization (PPO) is a widely used reinforcement learning algorithm valued for its stability and simplicity in continuous control tasks (Schulman et al., 2017). However, recent evaluations on standard benchmarks reveal that PPO struggles on certain environments such as Reacher, often failing to reach target positions even after extended training (Wan et al., 2025). The poor performance of PPO on Reacher has been attributed to several factors. First, as an on-policy algorithm, PPO lacks the ability to reuse past experiences efficiently, unlike off-policy methods such as Twin Delayed DDPG (TD3) and Soft Actor-Critic (SAC), which leverage replay buffers. Secondly, PPO's simpler exploration strategy is less effective in environments with dynamic or continuing goals, where the task resets internally without external resets. Finally, PPO's sensitivity to reward structure changes can lead to unstable learning dynamics when the reward landscape is poorly shaped.

Improving PPO's performance on Reacher remains an open problem. One promising direction is reward shaping, which modifies the reward signal to guide learning without altering the optimal policy. In particular, Potential-Based Reward Shaping (PBRS) (Ng et al., 1999) provides theoretical guarantees that the optimal policy is preserved under certain shaping functions. Prior studies suggest that applying carefully designed shaping terms can accelerate learning, especially in continuous control environments (Devlin & Kudenko, 2012).

In this project, I investigate whether simple potential-based shaping strategies can improve PPO's sample efficiency and success rate on Reacher-v5. Specifically, I implement three shaping methods-L2 shaping, L2-squared shaping, and time-decayed shaping. I also systematically evaluate their impact against a PPO baseline.

My code was implemented from scratch using PyTorch, Gymnasium (MuJoCo), and the standard data science toolkit (numpy, pandas, etc) with the exception of the `Monitor` wrapper class imported from stable-baselines3 [4] to track episode rewards and lengths. For information on the Demo, please visit the github README file.

2 Related Work

Potential-Based Reward Shaping (PBRS) offers a principled way to modify rewards without changing the optimal policy (Ng et al., 1999). Later theoretical work establishes deeper connections between reward shaping and value initialization. Wiewiora (2003) showed that PBRS is mathematically equivalent to initializing Q-values differently, meaning that shaping does not inject new optimal policies but rather influences the agent's learning trajectory without altering its asymptotic behavior. Extensions to deep reinforcement learning, such as in multi-agent systems (Devlin & Kudenko, 2012) and continuous domains (Brys et al., 2015), have shown that shaping can accelerate learning when appropriately applied.

In continuous control tasks, reward shaping has been used to improve exploration and sample efficiency, particularly in sparse-reward environments such as the MuJoCo simulations (Duan et al., 2016). These findings suggest that shaping is especially valuable when the default rewards are poorly informative during early training.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a popular on-policy method for continuous control, but has been observed to struggle in Reacher-v5, failing to consistently reach targets (Wan et al., 2025). This limitation motivates exploring reward shaping as a lightweight intervention to improve PPO's sample efficiency and robustness without altering its core algorithm. Our work builds on this foundation by systematically evaluating several potential-based shaping strategies to improve PPO's performance on Reacher-v5.

3 Methods

3.1 Environment

The Reacher-v5 environment from the Gymnasium library simulates a two-degree-of-freedom robotic arm tasked with reaching a target position. The observation space is a 10-dimensional vector composed of cosine and sine of the joint angles, the x, y position of the target, joint angular velocities, and the relative vector from the tip to the target. The action space is a two-dimensional vector representing continuous torques applied to the joints, each bounded in $[-1, 1]$.

The reward at each timestep is computed as:

$$r = -\|\text{target} - \text{tip}\|_2 - 0.1\|a\|^2$$

where $\|\text{target} - \text{tip}\|_2$ is the Euclidean distance between the tip and the target, and $\|a\|^2$ is the squared magnitude of the action vector. **Episodes** last for a maximum of 50 environment steps. For the purpose of this project, **iterations** refer to $\left\lceil \frac{\text{\# time steps}}{\text{\# rollout len}} \right\rceil$. For our experiments, "# rollout len" is 2048.

3.2 PPO Baseline

PPO agent is implemented in-house for full flexibility to integrate and customize reward-shaping modification during training based on the PPO algorithm proposed by Schulman et al. (2017). At each iteration, PPO optimizes the following objective:

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

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a clipping hyperparameter. Generalized Advantage Estimation (GAE) is used to compute $\hat{A}_t$ to reduce the variance of the advantage estimates.

The PPO baseline uses a two-layer multilayer perceptron (MLP) with 64 hidden units per layer and Tanh activations. The policy outputs two continuous torques controlling the 2-DoF arm, modeled as independent Gaussian distributions with learnable log standard deviations.

The training setup follows standard PPO hyperparameters: - Learning rate: 3×10^{-4} - Discount factor (γ): 0.99 - GAE lambda (λ): 0.95 - Clip range (ϵ): 0.2 - Batch size: 64 - Optimizer: Adam - Number of environment steps per iteration: 2048 - Number of policy updates per iteration: 10 epochs - Number of timesteps: 500K and 1 million time steps

3.3 Reward Shaping strategies

Reward shaping modifies the original reward function by adding an auxiliary term that encourages desirable behaviors. In Potential-Based Reward Shaping (PBRS), the shaped reward is defined as:

$$r'(s, a, s') = r(s, a, s') + \gamma\Phi(s') - \Phi(s)$$

where $r(s, a, s')$ is the environment reward, γ is the discount factor, and $\Phi(s)$ is a potential function mapping states to scalar values. Andrew Ng et al. (1999) showed that shaping rewards in this manner preserves the set of optimal policies.

I investigate two shaping functions based on the distance between the robot's tip and the target:

L2 Shaping:

$$\Phi(s) = -\|\text{tip} - \text{target}\|_2$$

L2-Squared Shaping:

$$\Phi(s) = -\|\text{tip} - \text{target}\|_2^2$$

The L2 shaping term penalizes proportional to the Euclidean distance, while the L2-squared variant imposes a stronger penalty on larger deviations. Both are designed to encourage faster tip-target alignment during training by providing smoother reward gradients compared to the sparse environment reward.

3.4 Time-Decayed Shaping

While fixed shaping functions can accelerate learning initially, they may interfere with fine-tuning later in training. To address this, I use a time-decayed shaping variant where the influence of the shaping term diminishes over time. This approach aims to provide strong guidance during initial exploration while allowing the policy to fine-tune directly toward the environment's original reward in later stages. I'm using **L2 reward shaping** as the **default** shaping function, unless specified otherwise with something like "l2sq" for L2-squared shaping.

Decayed shaping reward:

$$r'(s, a, s', t) = r(s, a, s') + \alpha(t) \left(\gamma\Phi(s') - \Phi(s)\right)$$

where the decay factor $\alpha(t)$ decreases linearly with timestep t:

$$\alpha(t) = \max\left(0, 1 - \frac{t}{T}\right)$$

Here, T is a user-defined decay horizon (e.g. the experiments used 200K, 300K, 500K steps). Early in training, $\alpha(t) \approx 1$ retains strong shaping, while later in training, $\alpha(t)$ gradually reduces the shaping contribution to zero.

3.5 Training details

Training is conducted with 500,000 and 1 million environment steps per run, equivalent to approximately 10,000 and 20,000 PPO iterations respectively. I use "episode" to describe the 50-timestep session of Reacher. Thus, with a rollout (simulation for training) length of 2048 steps there are respectively 245 and 489 data points collectedn for average rollout episode returns.

Training is conducted as described in Section 3.2, with each session capped at 1 million environment steps. During training, I log the unshaped environment episode returns to monitor learning progress.

3.6 Evaluation metrics

For evaluation, I compute success rates, average episode returns, and visualize policies through rollout GIFs generated.

The GIFs will be included in the presentation and publish on Github, which is listed in Appendix A.

Performance is evaluated based on two primary metrics:

Success Rate: Defined as the fraction of evaluation episodes in which the tip reaches within 0.02 meters of the target at any timestep. Success rates are computed by averaging over 1000 evaluation episodes.

Mean Episode Return: The average cumulative reward per episode, computed using only the environment's original reward (without shaped terms) over the course of the training session.

4 Experiments

4.1 Guiding questions

I present these guiding questions as a way to focus the readers on the goal of this project:

- Does PBRS accelerate PPO learning on Reacher-v5?
- Which $\Phi(s)$ shaping function produces better learning dynamics: L2 or L2-squared? Does stronger penalties for larger deviations improve convergence?
- Does a decaying shaping term further enhance long-term policy performance?
- Can shaping help PPO's poor performance compared to baseline?

4.2 Training curves

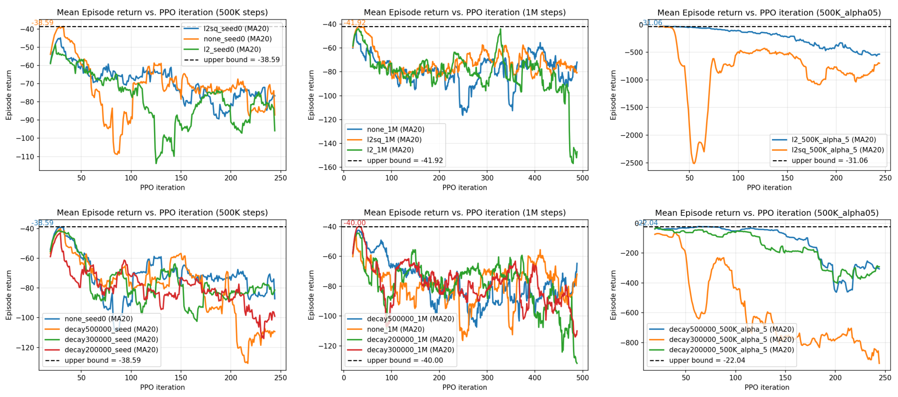


Figure 1: Training curves (mean episode return over iterations). Top row: L2 and L2-squared shaping variants. Bottom row: decay variants. Each column correspond to training configurations (500K steps, 1M steps, 500K steps with $\alpha = 0.5$. Baseline "none" (no shaping) is included for reference. Upper-bound is the maximum point. The number indicates the y-value, and the color indicates the curve that achieved it.

Across all methods, I observe that training curves typically peak early, between 25 to 50 PPO iterations, before descending to lower returns. None of the methods show sustained improvement beyond their initial peak, and long-term stability remains a challenge. This effect may arise from overfitting to the initial shaped rewards, reduced exploration once a suboptimal policy is found, and persistent shaping terms that bias learning away from the true environment reward. Additionally, the on-policy nature of PPO can amplify small instabilities in the policy over time, further contributing to performance decline during late training.

Several variants exhibit erratic behavior after reaching their peak returns, with varying degrees of instability. This suggests that while reward shaping may influence early learning dynamics, it does not consistently prevent late-stage degradation of performance.

4.3 Success rates

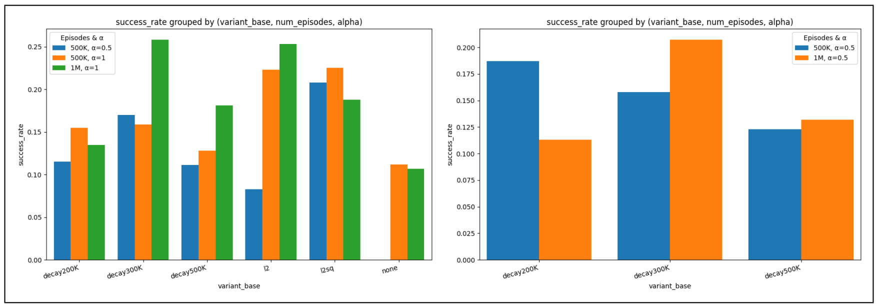


Figure 2: Success rate by reward shaping variant, by training episodes and α. The left bar plot's decay-variants use L2 by default. The right bar plot compares the decay-variants with L2-squared and with lowered $\alpha = 0.5$ for 500K and 1M steps training.

The baseline PPO agent ("none") achieves low success rate of approximately 10%, with minimal improvement after more training. In contrast, reward shaping variants (L2, L2-squared, and decay300K with 1M steps) consistently outperform the baseline. In particular, decay300K variant achieves a success rate above 15% after 500K steps and at 25.8% after 1M steps, the highest among all methods tested.

Training for a longer horizon improves success rates for some methods, such as decay300K, but does not uniformly benefit all variants. Additionally, tweaking the starting $\alpha(0) = 1$ to $\alpha(0) = 0.5$ generally decrease overall performance in some cases, suggesting that stronger initial shaping signals are advantageous for exploration. More work should be done on this angle to conclude a definitive pattern.

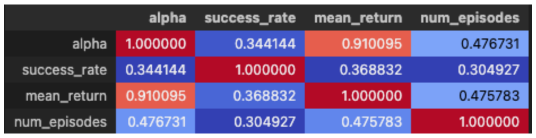
4.4 Final metrics table

run	variant_base	num_episodes	alpha	success_rate	mean_return
decay300000_1M	decay300K	1M	1	0.258	-67.56
l2_1M	l2	1M	1	0.253	-81.81
l2sq_seed0	l2sq	500K	1	0.225	-56.34
l2_seed0	l2	500K	1	0.223	-121.18
l2sq_500K_alpha_5	l2sq	500K	0.5	0.208	-809.02
l2sq_1M	l2sq	1M	1	0.188	-67.92

Table 1: Top 6 performers sorted by `success_rate` On eval run of 1000 episodes. For the full experiment results please visit Table 2 under Appendix B.

The highest success rate is achieved by decay300K variant after 1M steps (26%) followed closely by L2 shaping variants. Top 5 are all above 20%. Notably, some agents trained for only 500K steps, such as the L2-squared variant, still achieve competitive success rates around 22–23%, suggesting that reward shaping can improve efficiency even with limited training. Furthermore, I observe that high success rates do not always correspond to high mean episode returns. For example, the L2-squared variant trained with an initial $\alpha(0) = 0.5$ achieves a reasonable success rate despite a *very* poor mean return.

To better quantify these relationships, I computed a Pearson correlation matrix among success rate, mean return, shaping strength α, and training episodes. The results show that mean return and α exhibit a strong positive correlation ($r \approx 0.91$), while mean return and success rate have only a moderate correlation ($r \approx 0.37$). Similarly, success rate correlates weakly with training duration ($r \approx 0.30$).



	alpha	success_rate	mean_return	num_episodes
alpha	1.000000	0.344144	0.910095	0.476731
success_rate	0.344144	1.000000	0.368832	0.304927
mean_return	0.910095	0.368832	1.000000	0.475783
num_episodes	0.476731	0.304927	0.475783	1.000000

Figure 3: Correlation table

5 Conclusion

5.1 Discussion

ur experiments demonstrate that potential-based reward shaping improves PPO's performance on Reacher-v5 compared to the baseline without shaping. Among the methods tested, the decay300K variant trained for 1M steps achieved the highest success rate (26%), outperforming baseline (11%) by more than double. Both L2 and L2-squared shaping variants also consistently improved performance over the baseline, although their sensitivity to training duration differed: L2 benefited from extended training, while L2-squared's performance plateaued or declined after longer training.

During training, shaping terms are most useful during the early stages, where the environment reward alone may provide sparse or weak signals for exploration. Decaying the shaping term over time allows the agent to benefit from strong initial guidance, but eventually transition to optimizing the true environment reward without additional bias. In contrast, maintaining a constant strong shaping term, especially with aggressive penalties like L2-squared shaping, can interfere with fine-tuning late in training, leading to instability or reduced final performance.

Correlation analysis also revealed that mean episode return does not strongly predict success rate outcomes. Instead, smoother and more stable learning curves, rather than higher peak returns, were more consistently associated with higher success rates. These findings suggest that shaping strategies that stabilize policy behavior can be more important for task success than simply maximizing training rewards.

5.2 Limitations and future work

This study is limited to a single environment, Reacher-v5, which has relatively short 50-step episodes. Results may not fully generalize to longer-horizon tasks or environments with more complex reward structures. In particular, I observed that when evaluating agents under longer episode lengths, their behavior deteriorates substantially. The policies appear to overfit to the 50-step training horizon, optimizing only for short-term reward reduction without maintaining stable goal-reaching behaviors over extended durations. Furthermore, I maintained fixed PPO hyperparameters across all experiments without conducting tuning sweeps, which may have constrained peak performance. The chosen hyperparameters were based on the performance of PPO baseline at the beginning of the project.

Future work could extend this analysis to a broader set of continuous control tasks, including FetchReach, HalfCheetah, or Ant environments (all from MuJoCo). Exploring meta-learning approaches to adapt the shaping function $\Phi(s)$ during training, as well as investigating nonlinear or reward-adaptive decay schedules, could further enhance the effectiveness of reward shaping strategies in deep reinforcement learning.

References

[1] Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., & Nowé, A. (2015, July). Reinforcement Learning from Demonstration through Shaping. In *IJCAI* (pp. 3352-3358).

[2] Devlin, S. M., & Kudenko, D. (2012, June). Dynamic potential-based reward shaping. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*(pp. 433-440). IFAAMAS.

[3] Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016, June). Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning* (pp. 1329-1338). PMLR

[4] Raffin, Antonin, et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations." Journal of Machine Learning Research (2021).

[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[6] Wan, Y., Korenkevych, D., & Zhu, Z. (2025). An Empirical Study of Deep Reinforcement Learning in Continuing Tasks. *arXiv preprint arXiv:2501.06937*.

[7] Wiewiora, E. (2003). Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19, 205-208.

Appendix A

The complete data is uploaded to github and plots are stored in the jupyter notebook `plot_results.ipynb` in this repository:

`https://github.com/hungnguyendinh1999/RL-Reacher`

Running `eval_runs.py` will read from `data/*/ppo_model.pt` to automatically run evaluations of 1000 episodes on each model and produce the `results.csv` file for analysis.

For DEMO script, please read the README file for full instructions on running the `demo_run_rollout.py` script.

Below are 2 graphs that were excluded because these are extension to the experiment parameters.

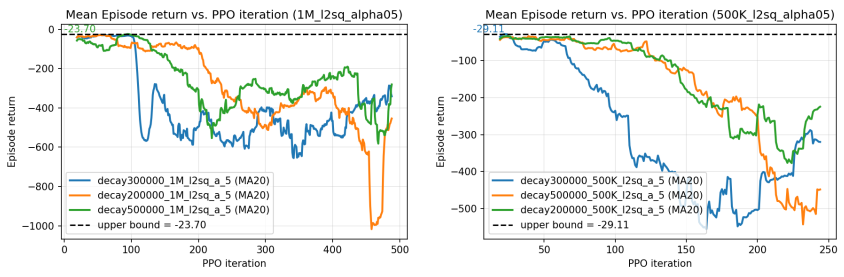


Figure 4: Experiment-only: Decay variants with L2-squared as shaping function

These are certainly very interesting, they quickly plunge to much worse mean return, which aligns with our understanding that L2-squared is sensitive to difference in errors, even with the help of timed-decay shaping.

Appendix B

run	variant_base	num_episodes	alpha	success_rate	mean_return
decay300000_1M	decay300K	1M	1	0.258	-67.56
l2_1M	l2	1M	1	0.253	-81.81
l2sq_seed0	l2sq	500K	1	0.225	-56.34
l2_seed0	l2	500K	1	0.223	-121.18
l2sq_500K_alpha_5	l2sq	500K	0.5	0.208	-809.02
l2sq_1M	l2sq	1M	1	0.188	-67.92
decay500000_1M	decay500K	1M	1	0.181	-57.44
decay300000_500K_alpha_5	decay300K	500K	0.5	0.17	-907.34
decay300000_seed	decay300K	500K	1	0.159	-244.34
decay200000_seed	decay200K	500K	1	0.155	-61.68
decay200000_1M	decay200K	1M	1	0.135	-94.79
decay500000_seed	decay500K	500K	1	0.128	-97.37
decay200000_500K_alpha_5	decay200K	500K	0.5	0.115	-775.14
none_seed0	none	500K	1	0.112	-158.01
decay500000_500K_alpha_5	decay500K	500K	0.5	0.111	-378.21
none_1M	none	1M	1	0.107	-128.58
l2_500K_alpha_5	l2	500K	0.5	0.083	-1177.37

Table 2: Full table of results of experiments

run	variant_base	num_episodes	success_rate	mean_return
decay300000_1M_l2sq_a_5	decay300K	1M	0.207	-276.29
decay200000_500K_l2sq_a_5	decay200K	500K	0.187	-232.65
decay300000_500K_l2sq_a_5	decay300K	500K	0.158	-347.05
decay500000_1M_l2sq_a_5	decay500K	1M	0.132	-424.73
decay500000_500K_l2sq_a_5	decay500K	500K	0.123	-1111.2
decay200000_1M_l2sq_a_5	decay200K	1M	0.113	-662.76

Table 3: Experiment-only: Decay variants with L2-squared as shaping function