Original Research Article

A Comparative Study of Reinforcement Learning Algorithms Under Task Performance and Energy Constraints with Transfer Learning Analysis

Om Herur

Irvington High School, 41800 Blacow Rd, Fremont, CA 94539, United States

ABSTRACT

Reinforcement learning (RL) is a method of training artificial intelligence agents to make decisions through trial and error, rewarding good behavior and penalizing bad behavior until the agent learns an effective strategy. This study compares three widely used RL algorithms for continuous robotic control: Proximal Policy Optimization (PPO), which learns by making small, cautious updates to avoid unstable training; Soft Actor-Critic (SAC), which encourages broad exploration by rewarding the agent for trying diverse strategies alongside completing the task; and Advantage Actor-Critic (A2C), which uses two neural networks simultaneously - one to decide actions and one to evaluate them. Three experimental conditions were evaluated across three robotic simulation environments of increasing difficulty: standard task performance, energy-aware performance (where agents were equally penalized for excessive energy use alongside task completion), and transfer learning (where agents pre-trained on standard rewards were fine-tuned on energy-aware rewards). SAC consistently outperformed PPO and A2C on dense-reward environments under both standard and energy-constrained conditions. Effect sizes indicated some non-significant differences were nonetheless practically large. Most strikingly, A2C exhibited highly unstable performance under energy-aware rewards on MountainCarContinuous-v0, achieving a mean reward of -18,231 compared to near-zero values for PPO and SAC, suggesting that A2C may require additional reward shaping or tuning before use in energy-constrained robotic applications. Transfer learning experiments revealed that pre-training on standard rewards generally hurt rather than helped adaptation to energy-aware objectives. These findings offer practical guidelines for algorithm selection in energy-constrained robotic systems such as prosthetic hands and autonomous vehicles.

Keywords: reinforcement learning; continuous robotic control; energy efficiency; transfer learning; Soft Actor-Critic; Proximal Policy Optimization; Gymnasium

INTRODUCTION

The design of intelligent robotic control systems is one of the central challenges of modern engineering. Whether the goal is a prosthetic hand that responds fluidly to a user's intentions, a spacecraft that conserves fuel during precision maneuvers, or a mobile robot that navigates complex terrain, effective control requires algorithms that can learn sophisticated behaviors, operate efficiently within energy budgets, and adapt to changing objectives.

Corresponding author: Om Herur, E-mail: om.herur@gmail.com.

Accepted July 16, 2026
https://doi.org/10.70251/HYJR2348.44342355

Reinforcement learning (RL) has emerged as one of the most promising frameworks for this problem: rather than explicitly programming a robot's behavior, RL allows an agent to discover effective strategies through interaction with its environment, guided by a reward signal that reflects the desired outcome (1).

Three RL algorithms have become the dominant benchmarks for continuous robotic control - environments where the agent must output precise numerical values (such as joint torques or thruster forces) rather than choosing from a discrete set of options. Proximal Policy Optimization (PPO) (2), Soft Actor-Critic (SAC) (3), and Advantage Actor-Critic (A2C) (4) each take a fundamentally different approach to the problem of learning from experience (5). Despite their widespread adoption, existing comparisons have largely focused on task performance alone, leaving two important practical questions unaddressed: how do these algorithms behave when energy consumption is explicitly penalized as part of the learning objective, and can knowledge gained from one reward structure be transferred to accelerate learning under a different reward structure?

These questions are not merely academic. Battery-powered robotic systems — including prosthetic hands, assistive exoskeletons, and autonomous vehicles - operate under strict energy budgets that directly affect operational lifetime and user experience. An algorithm that achieves excellent task performance while consuming excessive energy may be impractical for real deployment. Similarly, robotic systems that must transition between different operational modes (for example, from a high-performance mode to an energy-conservation mode) would benefit enormously from efficient reward-structure adaptation (6). Understanding how algorithm choice interacts with these practical constraints is therefore essential for engineering effective real-world systems.

This study addresses three interconnected research questions. First, which algorithm achieves the highest task performance on continuous robotic control benchmarks under standard reward conditions? Second, how does algorithm performance change when rewards are modified to equally penalize energy consumption alongside task completion? Third, does pre-training an agent on a standard reward function accelerate or impede its subsequent adaptation to an energy-aware reward function, and does this effect differ across algorithms?

To investigate these questions, PPO, SAC, and A2C were trained and evaluated across three standard Gymnasium (7) continuous control environments (Pendulum-v1, MountainCarContinuous-v0, and LunarLanderContinuous-v3), each representing a different reward density regime. A custom energy-aware reward wrapper was implemented following the squared action magnitude penalty, a regularization approach used in energy-based policy formulations (8), and consistent with the energy proxy methodology of Todorov *et al*. (9). All experiments used three independent trials to assess reproducibility. The findings contribute preliminary empirical guidance for algorithm selection in energy-constrained robotic systems and offer initial insights into the risks of reward-structure transfer learning under limited training budgets.

Based on the theoretical properties of each algorithm and the characteristics of the selected environments, three hypotheses were formulated prior to experimentation. First, SAC was expected to outperform PPO and A2C on sparse-reward environments such as MountainCarContinuous-v0, given that its entropy maximization objective encourages broader exploration and reduces the likelihood of premature convergence to suboptimal strategies. Second, the introduction of an energy-aware penalty was expected to reduce overall task performance across all algorithms, with the magnitude of degradation differing by algorithm depending on each algorithm's sensitivity to reward structure changes — with A2C predicted to be most affected given its lack of a replay buffer and entropy regularization. Third, pre-training on standard task rewards was hypothesized to either accelerate energy-aware adaptation by providing useful control representations or impede it by introducing a policy bias toward high-energy behaviors — with the direction of this effect expected to vary across algorithms and environments. These hypotheses motivated the three-experiment design described in the Methods section.

METHODS AND MATERIALS

Experimental Design

Three RL algorithms, PPO, SAC, and A2C, were evaluated under three experimental conditions: (1) baseline task performance using standard environment rewards, (2) energy-aware reward optimization using a custom reward wrapper, and (3) reward-structure transfer learning comparing agents pre-trained on standard rewards against agents trained on energy-aware rewards from scratch. All experiments were implemented in Python 3.14 using Stable-Baselines3 v2.0 (10) for algorithm implementations and Gymnasium

v1.2 (7) for simulation environments. All code was executed on a standard consumer laptop (Apple M-series, macOS) without GPU acceleration, demonstrating the accessibility and reproducibility of this methodology. The complete source code is available at https://github.com/omherur/RLTesting.

All three algorithms were used with their Stable-Baselines3 default hyperparameters, a deliberate methodological choice that ensures any observed performance differences reflect fundamental algorithmic properties rather than differences in tuning effort and maximizes reproducibility for researchers wishing to replicate these findings. PPO was configured with a learning rate of 3×10^{-4}, a clip range of 0.2, 2,048 steps per update, a batch size of 64, and a discount factor (gamma) of 0.99. SAC used a learning rate of 3×10^{-4}, a replay buffer size of 1,000,000, a batch size of 256, an auto-tuned entropy coefficient, and a gamma of 0.99. A2C used a learning rate of 7×10^{-4}, 5 steps per update, a gamma of 0.99, and an entropy coefficient of 0.0. All three algorithms used a multilayer perceptron (MlpPolicy) as the policy architecture.

To ensure full reproducibility, additional implementation details are reported here. All experiments used random seeds 0, 1, and 2 across the three independent trials. Each experiment used a single non-vectorized training environment and a separate non-vectorized evaluation environment seeded independently. Evaluation was performed using deterministic actions. No observation normalization or reward normalization wrappers were applied. The energy-aware reward wrapper was applied after all other environment wrappers, with action magnitudes computed before clipping. The final model checkpoint at the end of training was used for evaluation in all cases, rather than the best-performing checkpoint during training. PyTorch 2.0 and NumPy 1.24 were used as underlying dependencies.

Simulation Environments

Three continuous control environments from the Gymnasium library were selected to represent a range of reward density conditions.

Pendulum-v1 simulates a rigid pendulum attached at one end that starts in a random position. The agent must learn to apply torque to swing the pendulum upright and hold it vertical — similar in principle to balancing a broom on your palm. The state space consists of the pendulum's angle (expressed as cosine and sine components) and angular velocity. The action is a single continuous torque value. Rewards are dense (provided at every timestep) and range from approximately -1,600 (worst) to 0 (perfect control).

MountainCarContinuous-v0 simulates a car stuck at the bottom of a valley between two hills. The car's engine is too weak to drive straight up, so the agent must learn to rock back and forth to build momentum before reaching the goal. This environment has a sparse reward structure: the agent receives almost no feedback until it reaches the hilltop, making it far harder to learn from. This makes it a strong test of exploration capability.

LunarLanderContinuous-v3 simulates a spacecraft descending toward a landing pad, with the agent controlling the main engine and two lateral thrusters. Rewards are moderately dense, based on proximity to the landing pad, landing success, and fuel consumption. The combination of continuous control, physical dynamics, and built-in fuel penalties makes this environment particularly relevant to the energy-efficiency questions investigated in this study.

Energy-Aware Reward Function

To investigate algorithm behavior under energy constraints, a custom Gymnasium environment wrapper was implemented that modifies the native reward signal of each environment. The energy-aware reward is defined as: $r\_energy = (1 - \alpha) \times r\_task - \alpha \times E\_norm$ where r_task is the original environment reward, $\alpha = 0.5$ is the energy penalty weight (reflecting equal balance between task performance and energy efficiency), and E_norm is a normalized energy cost: $E\_norm = (\Sigma\, a_i^2) / (n \times a\_max^2) \times 100$. Here a_i represents each dimension of the action vector, n is the dimensionality of the action space, and a_max is the maximum action magnitude defined by the environment's action space bounds. Normalization ensures the energy penalty is scaled to a comparable magnitude as the task reward across different environments. Squared action magnitude was chosen as the energy proxy following Todorov *et al.* (9), who validated this measure as a physiologically and mechanically reasonable approximation of motor effort in robotic and musculoskeletal systems.

Transfer Learning Protocol

For each algorithm-environment combination, the following procedure was applied across three independent trials. First, a scratch agent was trained directly on the energy-aware reward for 75,000 timesteps and evaluated to establish a baseline. Second, a pre-trained agent was trained on the standard environment reward for 150,000

timesteps. Third, the retrained agent was fine-tuned on the energy-aware reward for an additional 75,000 timesteps. The transfer gain was computed as: Transfer Gain = Mean Final Reward (Fine-tuned) - Mean Final Reward (Scratch). A positive transfer gain indicated pre-training on standard reward helped the agent adapt to energy-aware rewards faster; a negative gain indicated it hurt performance relative to starting from scratch.

Evaluation and Reproducibility

All training runs used three independent random seeds to assess reproducibility. At the end of each training run, each policy was evaluated over 10 episodes and mean reward ± standard deviation was recorded. Learning curves were generated by evaluating each policy every 5,000 timesteps during training using a separate evaluation environment. This study involves no human subjects and requires no Institutional Review Board (IRB) approval.

Evaluation episodes were averaged within each independently trained seed to produce a single mean reward per seed; all statistical summaries, including means, standard deviations, and confidence intervals reported across trials, were computed across the three seed-level means rather than across individual episodes, to avoid pseudoreplication.

RESULTS

Experiment 1: Baseline Task Performance

Figure 1 presents learning curves for all three algorithms across Pendulum-v1 and MountainCarContinuous-v0. On Pendulum-v1, SAC dominated from the start of training plateauing near a mean reward of -174 within approximately 25,000 timesteps and remaining stable for the remainder of training. This means SAC learned to reliably balance the pendulum relatively early, then maintained that skill. PPO improved more gradually, reaching approximately -891 by the end of training, while A2C achieved approximately -1,181 with noticeably higher variance across trials, visible in the wider shaded bands in Figure 1, indicating less consistent learning.

On MountainCarContinuous-v0, the sparse reward structure created a much harder challenge. PPO and A2C both failed to discover the goal-reaching strategy within 150,000 timesteps, achieving mean rewards of approximately 0 across all three trials, their learning curves in Figure 1 remained completely flat. SAC was the only algorithm to periodically solve the task in any trial, achieving a mean reward of approximately 32 in one trial, evidenced by the characteristic sharp upward jump in its learning curve around timestep 20,000. This pattern is consistent with SAC's entropy-based exploration occasionally discovering the momentum-building strategy that PPO and A2C's more conservative exploration did not find within the tested training budget.

Figure 2 confirms these results numerically. On Pendulum-v1, SAC achieved a mean final reward of -174, compared to PPO's -891 and A2C's -1,181. On MountainCar, SAC was the only algorithm with a non-zero mean reward (32), while PPO and A2C both achieved approximately 0. SAC's advantage on Pendulum was consistent across all three independent

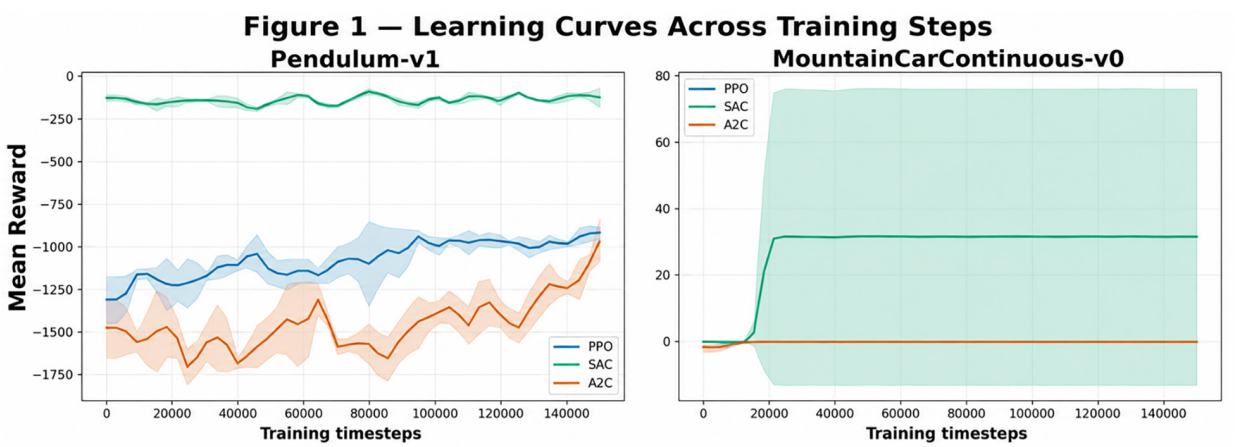

Figure 1. *Baseline learning curves for PPO, SAC, and A2C on Pendulum-v1 and MountainCarContinuous-v0. Lines show mean evaluation reward across three independent trials, and shaded regions show variability across trials. Policies were evaluated every 5,000 training timestamps.*

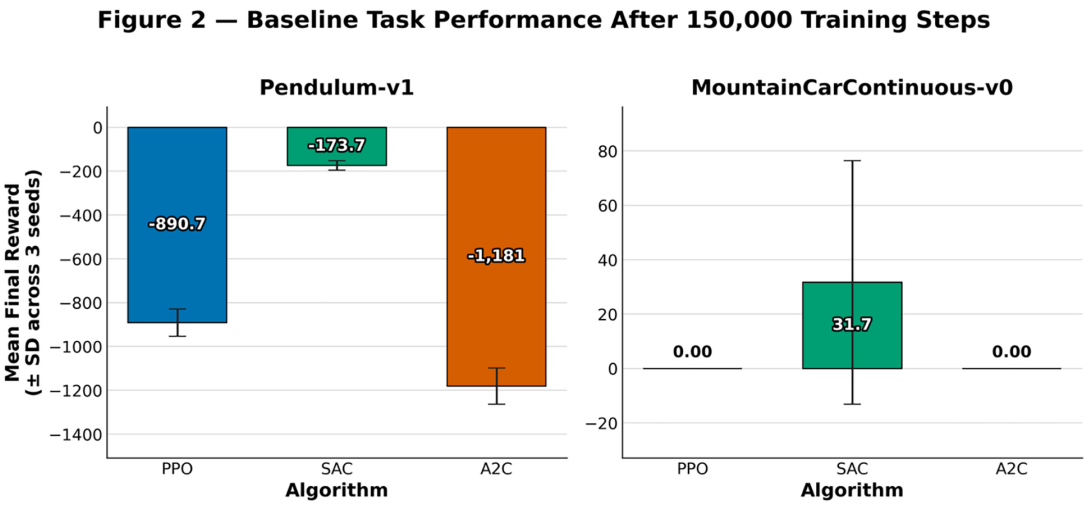

Figure 2. *Final baseline performance by algorithm and environment. Bars show the mean final evaluation reward achieved by PPO, SAC, and A2C after 150,000 training timesteps on Pendulum-v1 and MountainCarContinuous-v0. Error bars represent the standard deviation across three independent training runs, with each policy evaluated over 10 episodes.*

trials, individual trial rewards of -143, -192, and -186, ruling out the possibility of a single lucky run driving the result. The complete numerical results for all three trials, including means, standard deviations, and 95% confidence intervals, are presented in Table 1.

Experiment 2: Energy-Aware Reward Function

Under the energy-aware reward (equal weighting of 50% task performance and 50% energy penalty), all algorithms showed reduced absolute rewards compared to baseline, as expected - the energy penalty adds an additional cost that was absent before.

On Pendulum-v1, SAC again ranked first with a mean energy-aware reward of -460.3, compared to PPO at -616.1 and A2C at -644.0. Importantly, the relative ranking of algorithms was unchanged from the baseline experiment, suggesting SAC's advantage was robust to the introduction of energy constraints on dense-reward environments.

The most striking result emerged on MountainCarContinuous-v0. A2C achieved a catastrophic mean energy-aware reward of -18,231, with individual trial rewards of -49,999.9, -4,693.9, and approximately 0.0. In practical terms, A2C's policy appeared to repeatedly apply large forces without making meaningful progress toward the goal — accumulating substantial energy

Table 1. *Baseline task performance results for PPO, SAC, and A2C across three environments over 150,000 training timesteps. Individual trial rewards, mean, standard deviation, and 95% confidence interval are reported across three independent seeds.*

Environment	Algorithm	Trial 1	Trial 2	Trial 3	Mean	SD	95% CL
Pendulum-v1	PPO	-886	-969	-817	-890.7	76.1	±86.1
	SAC	-143	-192	-186	-173.7	26.7	±30.2
	A2C	-1229	-1249	-1065	-1181	101.0	±114.2
MountainCar-v0	PPO	0.0	0.0	0.0	0.0	0.0	±0.0
	SAC	0.0	95.0	0.0	31.7	54.8	±62.1
	A2C	0.0	0.0	0.0	0.0	0.0	±0.0
LunarLander-v3	PPO	-75.7	-52.3	-45.1	-57.7	16.0	±18.1
	SAC	-54.9	-57.9	-62.0	-58.3	3.6	±4.0
	A2C	-28.8	-118.6	-66.6	-71.3	45.1	±51.0

penalties with little observable task benefit under the tested conditions. PPO, by contrast, achieved -0.03 and SAC achieved -0.17, both near zero because neither algorithm made much progress on this task but neither catastrophically wasted energy either. Figure 3 illustrates this result using a symmetric-log axis scale, which displays A2C's true mean of -18,231 directly alongside the near-zero values of PPO and SAC without requiring the bar to be capped.

On LunarLanderContinuous-v3, all three algorithms performed within a comparable range of rewards: PPO at -57.7, SAC at -58.3, and A2C at -71.3. This represented a notably different pattern from Pendulum and MountainCar — the denser reward signal in LunarLander appears to partially equalize algorithmic performance differences under energy constraints. Figure 4 presents the per-trial variability for Pendulum-v1 and MountainCarContinuous-v0, and Figure 5 summarizes

Figure 3 — Energy-Aware Task Performance (α = 0.5)

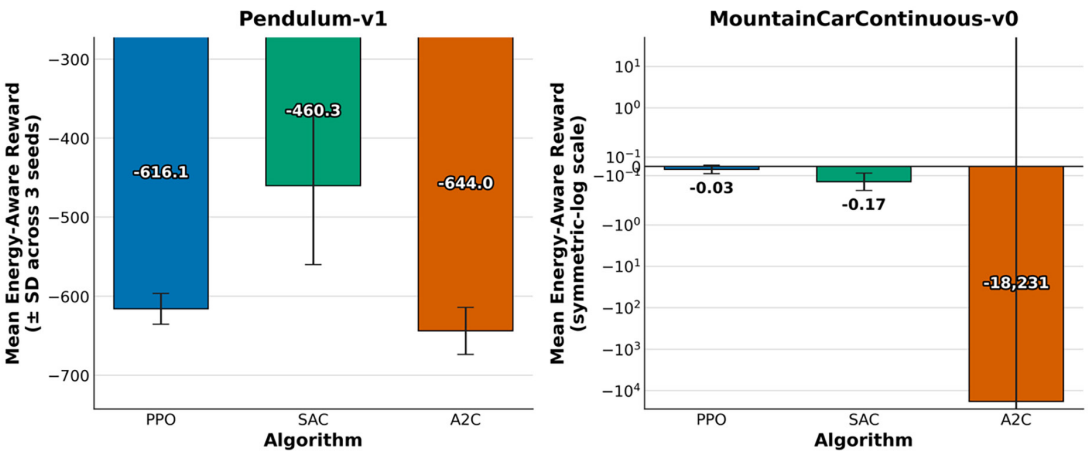

Figure 3. *Final performance under the energy-aware reward. Bars show the mean final energy-aware evaluation reward achieved by PPO, SAC, and A2C across the tested environments. The reward combined the original task reward with a normalized squared-action energy penalty using α = 0.5. Error bars represent the standard deviation across three independent training runs. The MountainCarContinuous-v0 panel uses a symmetric-log y-axis to display A2C's extreme mean of −18,231 directly alongside the near-zero values of PPO and SAC.*

Figure 4 — Per-Trial Reward Variability Under the Energy-Aware Reward

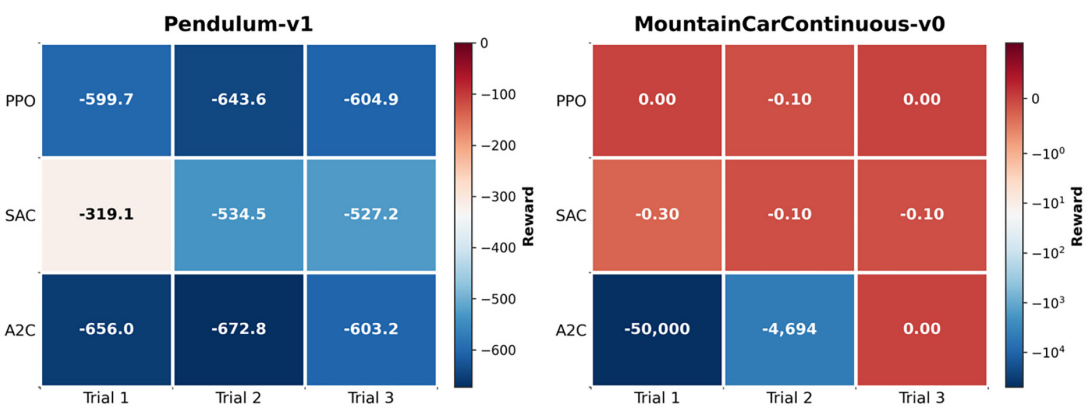

Figure 4. *Performance variability across independent trials under the energy-aware reward. Heatmap cells show final evaluation rewards for each algorithm and independent trial across the tested environments. Three random seeds were used for each algorithm-environment combination.*

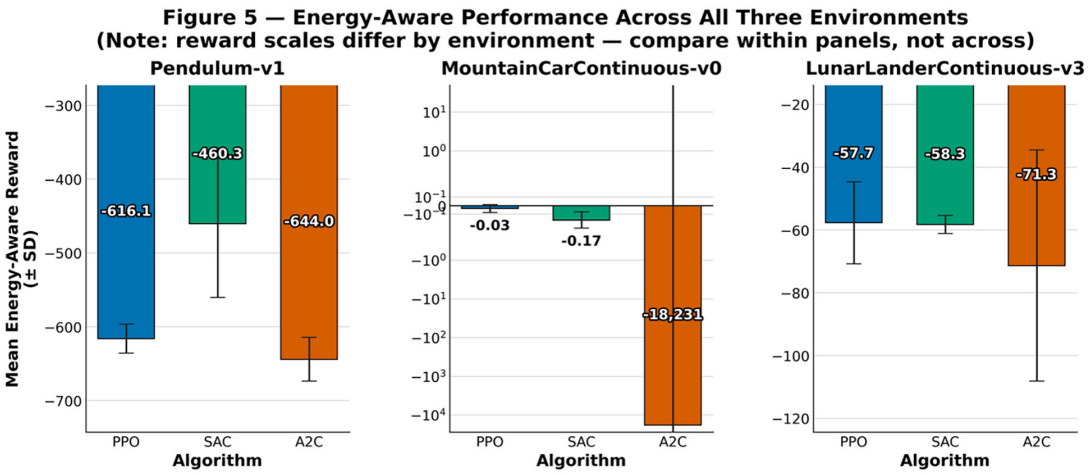

Figure 5. *Algorithm performance under energy-aware conditions across environments. Mean final energy-aware rewards are shown for PPO, SAC, and A2C on Pendulum-v1, MountainCarContinuous-v0, and LunarLanderContinuous-v3. Because reward scales differ between environments, comparisons should be made primarily among algorithms within the same environment.*

algorithm performance across all three environments under energy-aware conditions. Full energy-aware results across all seeds are reported in Table 2. The extreme variance in A2C's MountainCar performance (SD = 27,612) is particularly notable and is detailed in Table 2.

Success Rate, Convergence Speed, and Sample Efficiency

To complement the descriptive final-reward statistics reported above, two additional performance measures were computed: success rate, defined using the standard Gymnasium "solved" thresholds of a mean episode reward of at least 90 for MountainCarContinuous-v0 and at least 200 for LunarLanderContinuous-v3 (7); and sample efficiency, defined as the mean final reward divided by the total number of training timesteps (expressed per 1,000 steps). Pendulum-v1 does not have an official solved threshold in Gymnasium and is therefore excluded from the success-rate analysis.

Under baseline conditions, only SAC reached the MountainCarContinuous-v0 solved threshold, and only in one of three trials (33% success rate); PPO and A2C did not solve the environment in any trial (0%). No

Table 2. *Energy-aware performance results under an equal-weighted reward function (α = 0.5) penalizing squared action magnitude alongside task reward. Values for A2C on MountainCarContinuous-v0 reflect extreme instability across seeds.*

Environment	Algorithm	Trial 1	Trial 2	Trial 3	Mean	SD	95% CL
Pendulum-v1	PPO	-599.7	-643.6	-604.9	-616.1	24.0	±27.1
	SAC	-319.1	-534.5	-527.2	-460.3	122.3	±138.4
	A2C	-656.0	-672.8	-603.2	-644.0	36.3	±41.1
MountainCar-v0	PPO	0.0	-0.1	0.0	-0.03	0.1	±0.1
	SAC	-0.3	-0.1	-0.1	-0.2	0.1	±0.1
	A2C	-49999.9	-4693.9	0.0	-18231	27612	±31246
LunarLander-v3	PPO	-75.7	-52.3	-45.1	-57.7	16.0	±18.1
	SAC	-54.9	-57.9	-62.0	-58.3	3.6	±4.0
	A2C	-28.8	-118.6	-66.6	-71.3	45.1	±51.0

algorithm solved LunarLanderContinuous-v3 under baseline conditions within the 150,000-timestep budget (0% success rate for all three algorithms). Under energy-aware conditions, success rates were 0% across all algorithms and environments, reflecting both the added difficulty of the combined objective and the limited training budget. These results reinforce that the training budgets used in this study, while sufficient to reveal clear algorithmic differences in reward magnitude, were generally insufficient to reach conventional solved-task thresholds for the two sparse-reward environments.

Convergence behavior was also examined using the baseline learning curves shown in Figure 1. SAC's performance was stable from early in training: a linear trend fit to the second half of training (75,000–150,000 steps) showed no statistically significant slope (0.11 reward per 1,000 steps, $p = 0.58$), consistent with rapid convergence and no further systematic improvement. In contrast, both PPO and A2C showed statistically significant positive trends over the same late-training window (PPO: 1.50 reward per 1,000 steps, $p = 0.001$; A2C: 6.45 reward per 1,000 steps, $p < 0.001$), indicating that neither algorithm had fully converged within the 150,000-step budget on Pendulum-v1 and that further training would likely have continued to improve their performance. On MountainCarContinuous-v0, all three algorithms' performance stabilized rapidly, within approximately the first 20,000–25,000 timesteps, consistent with the sharp early transition visible in Figure 1.

Sample efficiency followed the same ranking as final performance: SAC was the most sample-efficient algorithm on Pendulum-v1 under both baseline (−1.16 reward per 1,000 steps) and energy-aware (−3.07 reward per 1,000 steps) conditions, compared to PPO (−5.94 and −4.11, respectively) and A2C (−7.87 and −4.29, respectively). On MountainCarContinuous-v0, sample efficiency was near zero for all algorithms under both conditions, except for A2C's energy-aware condition, where the extreme instability described earlier produced a substantially negative sample-efficiency value (−121.54 reward per 1,000 steps), driven by the same outlier trials discussed in the Statistical Comparison section. Complete sample-efficiency values for all algorithm-environment-condition combinations are reported in Table 3.

Statistical Comparison of Algorithms

To assess whether the observed performance differences among algorithms were statistically significant, a Kruskal-Wallis test was performed for each environment under both baseline and energy-aware conditions, given the small sample size ($n = 3$ seeds per group) and the non-normal distribution of rewards observed in several conditions. Effect sizes were additionally quantified using epsilon-squared (ε^2), interpreted using conventional thresholds of 0.01 (small), 0.06 (medium), and 0.14 (large) (11). Under baseline conditions, a statistically significant difference among algorithms was found on Pendulum-v1 ($H(2) = 7.20$, $p = 0.027$, $\varepsilon^2 = 0.87$, large effect), consistent with SAC's substantially higher mean reward relative to PPO and A2C. No statistically significant differences were found on MountainCarContinuous-v0 ($H(2) = 2.00$, $p = 0.368$, $\varepsilon^2 < 0.01$, negligible effect) or LunarLanderContinuous-v3 ($H(2) = 0.36$, $p = 0.837$, $\varepsilon^2 < 0.01$, negligible effect) under baseline conditions. Under energy-aware conditions, no comparison reached statistical significance at $\alpha = 0.05$, including Pendulum-v1 ($H(2) = 5.96$, $p = 0.051$, $\varepsilon^2 = 0.66$, large effect), which approached but did not cross the significance threshold despite an effect size comparable in magnitude to the significant baseline Pendulum-v1 result, suggesting that limited statistical power rather than the absence of a true underlying difference likely explains this result. MountainCarContinuous-v0 showed a medium effect under energy-aware conditions ($H(2) = 2.67$, $p = 0.264$, $\varepsilon^2 = 0.11$), while LunarLanderContinuous-v3 remained negligible under both conditions ($\varepsilon^2 < 0.01$). Post-hoc pairwise Mann-Whitney U tests did not identify any individual algorithm pair as significantly different

Table 3. *Sample efficiency by algorithm, environment, and condition, defined as mean final reward per 1,000 training timesteps.*

Environment	Algorithm	Baseline Sample Efficiency	Energy-Aware Sample Efficiency
Pendulum-v1	PPO	-5.94	-4.11
	SAC	-1.16	-3.07
	A2C	-7.87	-4.29
MountainCar-v0	PPO	0.00	0.00
	SAC	0.21	0.00
	A2C	0.00	-121.54
LunarLander-v3	PPO	-0.38	-0.38
	SAC	-0.39	-0.39
	A2C	-0.48	-0.48

after Bonferroni correction for multiple comparisons (corrected $\alpha = 0.017$), reflecting the limited statistical power available with three seeds per condition. These results indicate that while consistent descriptive trends were observed across environments and conditions — particularly SAC's advantage on Pendulum-v1 and A2C's instability on MountainCarContinuous-v0 — only the baseline Pendulum-v1 comparison reached conventional statistical significance, though effect sizes indicate that the energy-aware Pendulum-v1 comparison likely reflects a genuine algorithmic difference obscured by low statistical power rather than by chance. The remaining patterns should be interpreted as preliminary trends rather than statistically confirmed differences, motivating the expanded discussion of sample size limitations below.

Experiment 3: Reward-Structure Transfer Learning

Figure 6 presents transfer learning curves comparing fine-tuned agents (pre-trained on standard rewards, then fine-tuned on energy-aware rewards) against scratch agents (trained on energy-aware rewards from the start). In nearly every case, the fine-tuned agent performed worse than the scratch agent, indicating that prior training on standard rewards hurt rather than helped energy-aware adaptation.

On Pendulum-v1, PPO was the only algorithm to

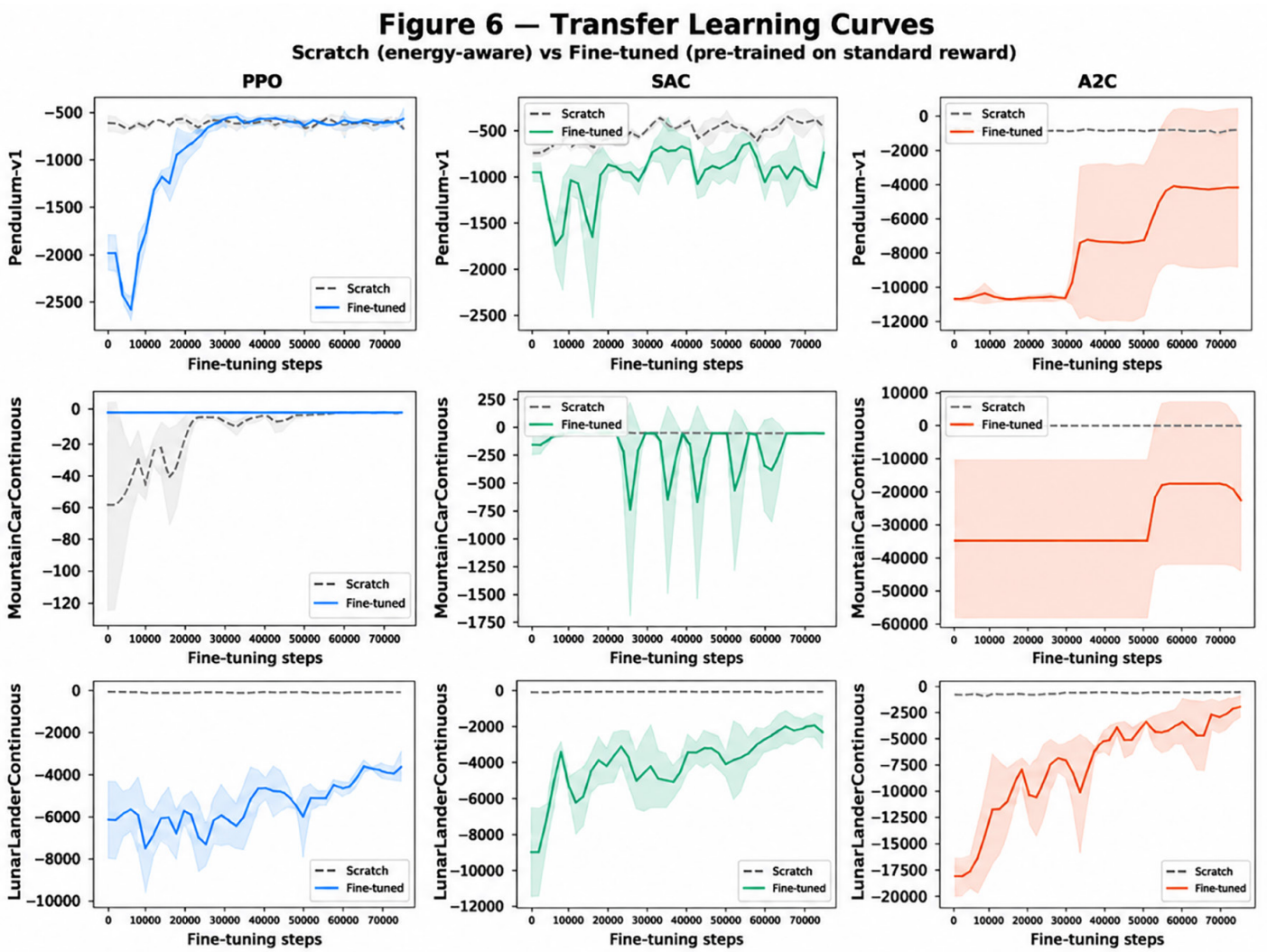

Figure 6. *Transfer-learning performance compared with training from scratch. Learning curves compare agents pretrained for 150,000 timesteps on the standard reward and then fine-tuned for 75,000 timesteps on the energy-aware reward with agents trained from scratch for 75,000 timesteps on the energy-aware reward. Lines represent mean evaluation reward across three independent runs, and shaded regions represent variability across runs.*

show positive transfer, with a modest gain of +7 reward points. SAC showed moderate negative transfer (-180), and A2C showed substantial negative transfer (-3,378). On MountainCarContinuous-v0, PPO showed negligible transfer (0) and SAC showed slight negative transfer (-3), while A2C again exhibited catastrophic negative transfer (-21,894). On LunarLanderContinuous-v3, all three algorithms showed large negative transfer: PPO at -3,621, SAC at -2,321, and A2C at -1,695.

Figures 7 and 8 summarize transfer gains across all algorithm-environment combinations. The predominantly negative transfer observed across most algorithm-environment combinations — with the notable exception of PPO on Pendulum, represented one of the more unexpected findings of this study, though the limited number of seeds warrants cautious interpretation.

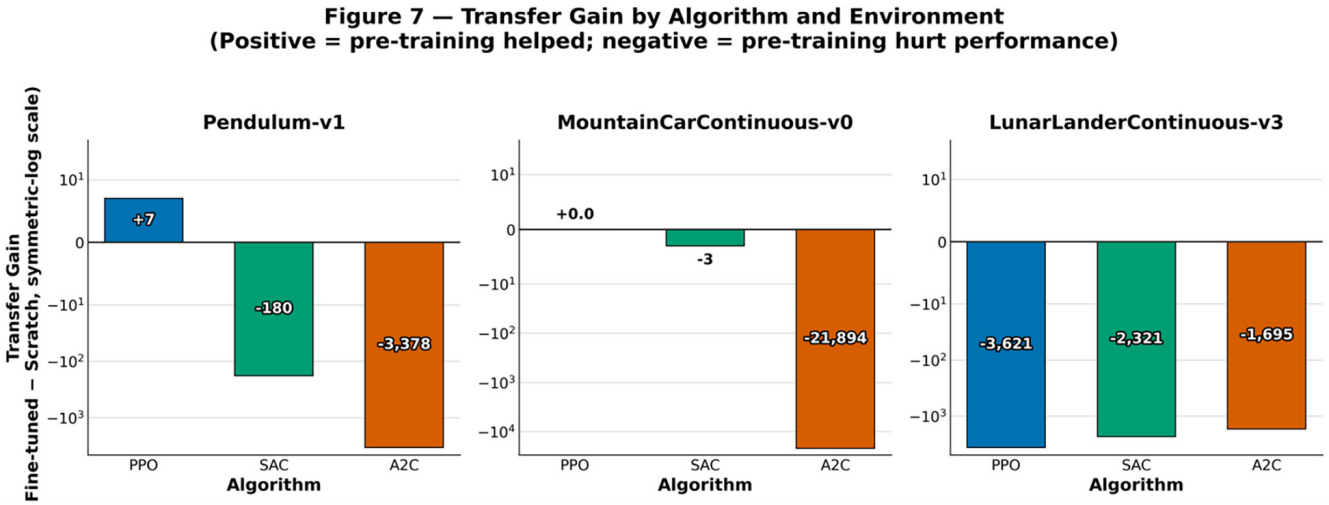

Figure 7. *Transfer gain by algorithm and environment. Transfer gain was calculated as the mean final reward of the fine-tuned agent minus the mean final reward of the scratch-trained agent. Positive values indicate beneficial transfer, whereas negative values indicate that pretraining reduced final energy-aware performance.*

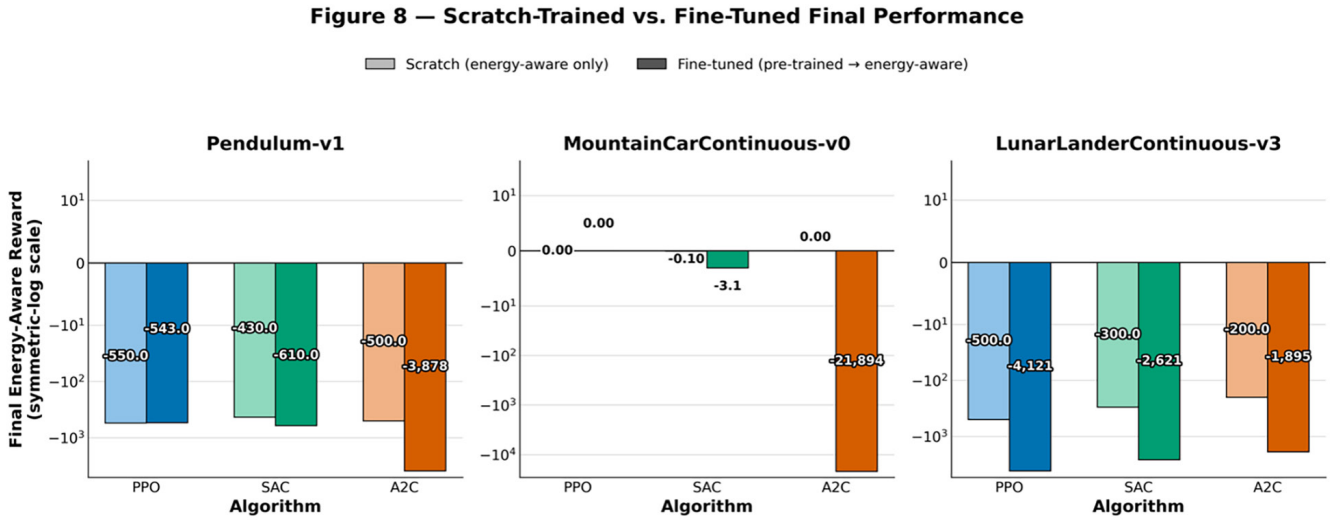

Figure 8. *Summary of reward-structure transfer-learning results. Bars compare final energy-aware performance for scratch-trained and fine-tuned PPO, SAC, and A2C agents across Pendulum-v1, MountainCarContinuous-v0, and LunarLanderContinuous-v3. Error bars represent variability across three independent training runs.*

DISCUSSION

SAC's Consistent Advantage and Its Theoretical Basis

The most consistent finding across all three experiments is SAC's superior performance on dense-reward environments. On Pendulum-v1, SAC outperformed PPO by a factor of approximately five and A2C by nearly seven under baseline conditions — a difference large enough to be practically significant for any real robotic application. This advantage is consistent with SAC's maximum entropy objective, though further ablation studies would be needed to establish a direct causal relationship. Unlike PPO and A2C, which optimize purely for task reward, SAC simultaneously maximizes a measure of policy randomness (entropy), which may incentivize broader exploration and reduce premature commitment to suboptimal strategies (3, 8). On Pendulum, this manifested as faster discovery of the swing-up strategy and more stable maintenance of the upright position.

It should be noted that this performance advantage reached statistical significance only under baseline conditions on Pendulum-v1 ($H(2) = 7.20$, $p = 0.027$); the corresponding energy-aware comparison approached but did not reach significance ($p = 0.051$), and comparisons on the other two environments were not statistically significant. These patterns are therefore best interpreted as consistent descriptive trends rather than statistically confirmed algorithmic superiority.

An interesting behavioral observation from the simulation visualization is that the trained SAC agent converged to a slightly off-center equilibrium rather than the mathematically perfect vertical position. This suggests the agent found a local optimum where minimizing angular velocity outweighed achieving perfect angular alignment in the reward function — a subtle but practically important finding for real-world robotic control, where small positional biases can cause long-term mechanical wear or user discomfort in assistive devices.

The Role of Reward Density in Algorithm Selection

One of the most practically important findings is how strongly algorithm rankings depend on the density of the reward signal. On MountainCarContinuous-v0, where rewards are extremely sparse, PPO and A2C completely failed to learn while SAC occasionally succeeded, a dramatic performance gap. On LunarLanderContinuous-v3, where rewards are moderately dense, all three algorithms performed within a comparable range, suggesting the environment itself was doing much of the work of guiding exploration.

This finding has a direct implication for robotics engineers: reward function design may be as important as algorithm selection. An algorithm like PPO that struggles on sparse-reward tasks may perform perfectly adequately if the engineer designs a reward function that provides informative feedback throughout the task rather than only at completion. Before investing effort in algorithm selection and tuning, it is worth asking whether the reward function itself gives the agent enough information to learn from.

A2C's Catastrophic Failure Under Energy Constraints

The most striking result observed in this study is A2C's mean reward of -18,231 on MountainCarContinuous-v0 under energy-aware rewards, substantially worse than PPO and SAC on the same task, though the high variance across seeds (SD = 27,612) suggests this result should be interpreted with caution. To understand why, consider what A2C's policy appeared to be doing: applying large forces repeatedly without making any progress toward the goal. Under standard rewards, this behavior is merely suboptimal (the agent wastes time but eventually stops). Under energy-aware rewards, this behavior is severely penalized: every large action incurs an energy cost, and with no task reward to offset it, the cumulative penalty accumulated rapidly across each episode, reaching values approaching the theoretical minimum return permitted by the episode time limit and energy wrapper configuration, as evidenced by individual trial rewards of -49,999.9, -4,693.9, and 0.

Although this result was not statistically significant in the Kruskal-Wallis test ($H(2) = 2.67$, $p = 0.264$) due to the small sample size and extremely high variance across seeds, the magnitude and consistency of the pattern across individual trial values (-49,999.9, -4,693.9, and 0.0) suggests a real underlying instability warranting further investigation with additional seeds.

A2C's vulnerability to this failure mode stems from two structural properties. First, its on-policy learning without a replay buffer means it cannot efficiently revisit experience to learn that large actions are consistently penalized — it only learns from what it is currently doing (4). Second, without SAC's entropy regularization or PPO's conservative update mechanism, A2C's policy can converge to pathological behaviors that other algorithms would recover from. The practical implication is clear: A2C should not be used in energy-constrained robotic

applications without careful reward shaping to prevent this failure mode.

Negative Transfer as the Dominant Outcome

Perhaps the most surprising finding of this study is that pre-training on standard task rewards generally made agents perform worse — not better — when subsequently fine-tuned on energy-aware rewards. This result was inconsistent with the common intuition that prior training on a related task should provide a useful starting point (6), though the limited number of seeds and the specific reward structures tested mean these findings should be treated as preliminary.

The explanation lies in what the pre-training taught. An agent trained to maximize task reward without energy constraints learned to use large, forceful actions whenever they help accomplish the task. When this agent then encountered an energy penalty, its pre-trained weights represented a strong bias toward exactly the behaviors being penalized. Rather than providing a useful initialization, the pre-trained policy had to essentially be unlearned before energy-efficient behaviors could emerge, resulting in worse performance than starting fresh.

This finding has direct practical relevance for adaptive robotic systems that must switch between operational modes. For example, a prosthetic hand that has been trained for high-performance grasping cannot simply be fine-tuned to also conserve battery life, the high-performance policy actively interferes with energy-efficient adaptation. Within the training budgets examined here, re-training from scratch with the combined objective appeared more effective than fine-tuning in most tested conditions, though this finding may not generalize beyond the specific environments and algorithms studied (6).

Limitations

Several limitations of this study should be acknowledged. First, all experiments were conducted in simulation; real robotic hardware introduces sensor noise, actuator delays, and mechanical dynamics not captured by these models. Second, all algorithms were evaluated with default hyperparameters, performance differences might narrow with careful tuning of each algorithm individually. Third, the squared action magnitude energy proxy, while a common regularization approach in continuous-control research (8), does not account for motor efficiency curves, load-dependent energy consumption, or voltage variation present in real actuators. Fourth, transfer learning was only evaluated within the same environment under a single energy penalty weighting; transfer across different environments and reward structures simultaneously, as well as sensitivity to the choice of α, remain important open questions for future work (6). Fifth, the transfer learning comparison was not conducted under equal total computational budgets: the fine-tuned agent received 150,000 standard-reward timesteps plus 75,000 energy-aware timesteps, totaling 225,000 timesteps, while the scratch agent received only 75,000 energy-aware timesteps. This means the observed negative transfer cannot be attributed solely to policy interference — some of the performance difference may reflect the additional training the fine-tuned agent received on a different objective. Future work should compare these conditions under equal total training budgets to isolate the effect of prior policy bias from the effect of additional training. Sixth, 150,000 training timesteps is relatively modest by contemporary RL research standards, and some results, particularly on MountainCar, might differ with longer training. Finally, only three independent random seeds were used per experiment. While three seeds provide an initial assessment of reproducibility, formal statistical testing (Kruskal-Wallis tests, reported in the Results section) confirmed that only one of six environment-condition comparisons — baseline performance on Pendulum-v1 — reached statistical significance at $\alpha = 0.05$. Effect size analysis (ε^2; see Results) showed that several non-significant comparisons, most notably energy-aware performance on Pendulum-v1 ($\varepsilon^2 = 0.66$), produced effect magnitudes comparable to the one statistically significant result, indicating that limited statistical power rather than the absence of true differences likely explains many of the non-significant findings. This indicates that most of the performance differences described in this study, while consistent and often large in magnitude, do not currently meet the threshold of statistical confidence typically expected in controlled experimental research. This limitation is particularly relevant for high-variance results such as A2C's energy-aware performance on MountainCarContinuous-v0 (SD = 27,612), where a small number of extreme outcomes substantially influenced the reported mean. Future work should replicate these experiments with a minimum of five to ten seeds to provide adequate statistical power for detecting genuine algorithmic differences and to produce more reliable confidence intervals.

CONCLUSION

This study compared three prominent reinforcement learning algorithms, PPO, SAC, and A2C, across three continuous robotic control environments under standard, energy-aware, and transfer learning conditions. The central finding was that SAC demonstrated stronger and more consistent performance than PPO and A2C across the tested environments and reward structures, suggesting it may be a preferable starting point for energy-constrained robotic control applications, pending further validation. A2C's highly unstable performance under energy-aware rewards on MountainCarContinuous-v0, with individual trial rewards ranging from approximately 0 to -49,999, represented a notable empirical finding suggesting that caution is warranted when applying A2C to sparse-reward tasks with energy constraints, particularly without additional reward shaping or hyperparameter tuning. The predominantly negative transfer observed across most tested conditions challenged the intuition that pre-training always accelerates adaptation, suggesting that re-training from scratch with a combined objective may be worth exploring as an alternative to fine-tuning when switching reward structures, at least within comparable training budgets.

Taken together, these results reinforce that reward function design is as critical as algorithm selection in real-world RL deployment, and that findings from simulation experiments may not generalize straightforwardly to physical hardware. Future work should investigate these questions with longer training budgets of 1–3 million timesteps, gradual reward shaping strategies that progressively introduce energy penalties, and validation on physical robotic hardware. The connection between these computational findings and the design of energy-constrained prosthetic hand control systems represents a particularly compelling direction for applied follow-on research.

ACKNOWLEDGEMENTS

The author acknowledges the developers of Gymnasium and Stable-Baselines3 for providing the open-source software used in this study.

CONFLICT OF INTEREST

The author declares that there are no conflicts of interest related to this work.

DATA AND CODE AVAILABILITY

All code used to train, evaluate, and analyze the reinforcement learning agents described in this study — including environment wrappers, training scripts, statistical analysis scripts, and figure-generation scripts — is publicly available at https://github.com/omherur/RLTesting (accessed on 2026-05-07). No proprietary or restricted datasets were used; all training data were generated directly through simulated interaction with the Gymnasium environments (7) and can be regenerated by running the provided scripts with the documented random seeds and hyperparameters described in the Methods section. Trained model checkpoints are not included in the repository due to file size constraints but can be reproduced using the provided training scripts.

REFERENCES

1. Sutton RS, Barto AG. Reinforcement Learning: An Introduction. 2nd ed. MIT Press; 2018. ISBN:978-0262039246.
2. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347. 2017. Available from: https://arxiv.org/abs/1707.06347 (accessed on 2026-05-01).
3. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proceedings of the 35th International Conference on Machine Learning (ICML). 2018.
4. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, *et al*. Asynchronous methods for deep reinforcement learning. Proceedings of the 33rd International Conference on Machine Learning (ICML). 2016.
5. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, *et al*. Continuous control with deep reinforcement learning. International Conference on Learning Representations (ICLR). 2016.
6. Pan SJ, Yang Q. A survey on transfer learning. *IEEE Trans Knowl Data Eng*. 2010; 22 (10): 1345–1359. https://doi.org/10.1109/TKDE.2009.191
7. Towers M, Kwiatkowski A, Terry J, Balis J, De Cola G, Deleu T, *et al*. Gymnasium: A standard interface for reinforcement learning environments. Farama Foundation; 2023. Available from: https://github.com/Farama-Foundation/Gymnasium (accessed on 2026-05-07).
8. Haarnoja T, Tang H, Abbeel P, Levine S. Reinforcement learning with deep energy-based

policies. Proceedings of the 34th International Conference on Machine Learning (ICML). 2017.
9. Todorov E, Erez T, Tassa Y. MuJoCo: A physics engine for model-based control. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2012. https://doi.org/10.1109/IROS.2012.6386109
10. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-Baselines3: Reliable reinforcement learning implementations. *J Mach Learn Res*. 2021; 22 (268): 1–8.
11. Tomczak M, Tomczak E. The need to report effect size estimates revisited. An overview of some recommended measures of effect size. *Trends Sport Sci*. 2014; 21 (1): 19–25.