Model Performance Report

Simon Green[1], Abdulrahman Altahhan[2]

School of Computing, University of Leeds, UK
[1] MSc, Artificial Intelligence
[2] Senior Teaching Fellow in Artificial Intelligence
{od21sg, a.altahhan}@leeds.ac.uk

Results are updated in realtime from the evaluations.

1 Results and Model Comparison

This report presents the performance evaluation of four reinforcement learning models: GenTRPO, TRPOER, and TRPOR. These models are implemented using **Stable Baselines3** and utilize **mini-batch gradient descent** approach. We benchmark against TRPO and PPO [3, 4]. The entropy calculations guiding the models are based on the entropy of each mini-batch.

1.1 EnTRPO (Trust Region Policy Optimization Method with Entropy Regularization)

The EnTRPO model extends TRPO by incorporating an entropy regularization term in the policy objective and adds a replay buffer with a specific 97.5% reward buffer reset logic [2]. While we do not present evaluations for this model, we include it in the discussion since it was used to inform some of the design decisions for other models evaluated. In the paper, the author does not discuss whether the replay buffer is prioritized or not, nor could we find any implementation or results for this model.

1.2 TRPOR (TRPO with Entropy Regularization)

This model extends TRPO by introducing entropy regularization only in the policy objective. The entropy coefficient hyperparameter guides the degree of regularization, ensuring a balance between exploration and exploitation. The entropy guiding this model is computed at the batch level, dynamically adjusting policy updates. The policy objective with entropy regularization is as follows:

$$J(\theta) = \mathrm{E}_{s,\, a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} \, A(s,a) \right] + \beta \; \mathrm{E}_{s \sim \rho^\pi} \left[H\left(\pi_\theta(\cdot \mid s)\right) \right] \tag{1}$$

where θ is the current policy parameter vector, θ_{old} is the previous policy parameter vector, $\pi_\theta(a \mid s)$ and $\pi_{\theta_{\text{old}}}(a \mid s)$ are the current and old policy probabilities, $A(s,a)$ is the advantage function, ρ^π is the state distribution under the current policy, β is the entropy regularization coefficient, and $H\left(\pi_\theta(\cdot \mid s)\right)$ is the entropy of the policy at state s.

1.3 TRPOER (TRPO with Entropy Regularized Experience Replay)

This model extends TRPO by incorporating entropy-based experience replay along with an additional policy entropy regularization term. In TRPOER, a prioritized experience replay buffer is employed where experiences are sampled according to the entropy of the current mini-batch, modulated by a hyperparameter coefficient. This bidirectional adaptive sampling mechanism adjusts both the number and the direction of sampled experiences to optimize learning. The adaptive sampling function is formulated as follows:

$$S(H,\kappa) = \begin{cases} S_{\max}, & \text{if } \sigma(H) \geq |\kappa|, \quad \kappa \geq 0 \\ S_{\min}, & \text{if } \sigma(H) < |\kappa|, \quad \kappa \geq 0 \\ S_{\max}, & \text{if } \sigma(H) \leq |\kappa|, \quad \kappa < 0 \\ S_{\min}, & \text{if } \sigma(H) > |\kappa|, \quad \kappa < 0 \end{cases} \tag{2}$$

where $S(H, \kappa)$ represents the number of replay samples drawn based on entropy H of the mini-batch, and κ is the sampling coefficient acting as an adaptive cutoff threshold. The function $\sigma(H)$ normalizes entropy into the range $[0, 1]$ using a sigmoid transformation:

$$\sigma(H) = \frac{1}{1 + e^{-H}} \tag{3}$$

ensuring bounded entropy-driven prioritization. For $\kappa > 0$, sampling is triggered when the normalized entropy $\sigma(H)$ *exceeds* the threshold $|\kappa|$, whereas for $\kappa < 0$, sampling is triggered when $\sigma(H)$ *falls below* the threshold. This bidirectional gating mechanism enables adaptive experience replay that dynamically adjusts to policy entropy variations during training.

The underlying hypothesis for TRPOER originates from previous experiments with modified EnTRPO variants (namely, EnTRPOLow and EnTRPOHigh). In those experiments, only the replay buffer was sampled, without directly incorporating an entropy term into the policy objective. Although these variants sometimes outperformed standard TRPO in certain environments, their performance was inconsistent. This observation motivated the development of an adaptive sampling strategy—controlled by the sampling coefficient κ—that dynamically adjusts the replay mechanism to achieve robust performance across diverse scenarios. We also observed that the EnTRPO model with Prioritized Experience Replay (PER) outperformed the EnTRPO without, hence motivating the use of prioritized experience replay in TRPOER.

1.4 GenTRPO (Generative Experience Replay Trust Region Policy Optimization with Entropy Regularization)

Quite a mouth full, we'll find a better name. The GenTRPO algorithm extends the Trust Region Policy Optimization with Entropy Regularization (TRPOER) framework [3,4] by incorporating a generative model to augment the experience replay buffer. The key idea is to leverage synthetic experiences generated by a forward dynamics model to complement real experiences, thereby improving exploration and sample efficiency [1].

In the GenTRPO framework, the experiences used for policy updates are sampled from a replay buffer. The sampling strategy ensures that half of the samples in each batch are real experiences collected from the environment, while the other half are generated by the forward dynamics model. This combination of real and synthetic data balances model fidelity with exploratory richness, enabling the policy to generalize effectively while maintaining stability during optimization.

The generative component of GenTRPO relies on a forward dynamics model inspired by the intrinsic curiosity module [1]. The forward dynamics model comprises an encoder and a dynamics predictor. The encoder maps raw states s into a compact latent space representation $h(s)$, capturing the essential features of the environment. The dynamics predictor then takes the latent state $h(s)$ and action a as input and predicts the next latent state $h(s')$, effectively modeling the transition function $P(s'|s, a)$. The error of this model, expressed as

$$\mathcal{F}(s, a, s', r) = \frac{1}{2} \left\| g(h(s), a) - h(s') \right\|^2 \tag{4}$$

where $g(h(s), a)$ is the predicted latent state, $h(s')$ is the true latent state, and $\| \cdot \|$ represents the Euclidean norm. This error quantifies how accurately the forward dynamics model predicts the latent state transitions. It is used to compute intrinsic motivation, encouraging the agent to explore transitions that are harder to predict, thereby fostering exploration [5].

2 Model Performance Table

The table below summarizes the models' performance in terms of mean and standard deviation of rewards, along with maximum and minimum rewards recorded during training. A higher mean reward suggests better overall performance, while lower standard deviation indicates increased stability.

	Ant-v5	Pendulum-v1	InvertedDouble Pendulum-v5	Humanoid-v5
PPO	$1173.61M$ $515.51\mu \pm 326.48\sigma$ $1895E, 10R$	$-0.02M$ $-207.56\mu \pm 118.01\sigma$ $2892E, 10R$	$9359.93M$ $1090.87\mu \pm 2910.66\sigma$ $6965E, 10R$	$1245.44M$ $464.88\mu \pm 298.07\sigma$ $15385E, 9R$
TRPO	$1960.61M$ $1328.68\mu \pm 284.88\sigma$ $1067E, 10R$	$-0.20M$ $-141.97\mu \pm 115.55\sigma$ $2600E, 10R$	$9359.78M$ $7516.76\mu \pm 3879.62\sigma$ $5767E, 10R$	$1247.77M$ $369.87\mu \pm 311.76\sigma$ $9569E, 10R$
TRPOER	$1349.85M$ $639.65\mu \pm 465.91\sigma$ $1882E, 8R$	$-0.07M$ $-173.49\mu \pm 156.48\sigma$ $2859E, 10R$	$9351.73M$ $2673.08\mu \pm 2620.77\sigma$ $5070E, 9R$	$897.72M$ $371.33\mu \pm 359.99\sigma$ $2799E, 4R$
TRPOR	$3590.62M$ $1662.87\mu \pm 1171.26\sigma$ $1655E, 7R$	$-0.22M$ $-197.32\mu \pm 215.34\sigma$ $4508E, 10R$	$9357.32M$ $8045.50\mu \pm 2918.29\sigma$ $9123E, 5R$	$1416.67M$ $620.72\mu \pm 381.70\sigma$ $9225E, 10R$

3 Performance Analysis Through Plots

The following plots visualize different aspects of model performance.

3.1 Resampled Rewards and Outlier Removal

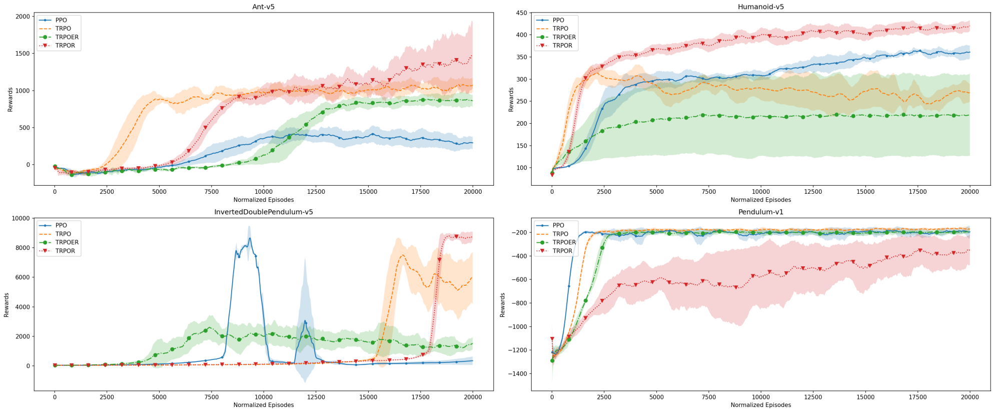

Fig. 1. Resampled Rewards with Outlier Removal

This plot presents reward distributions after applying smoothing and outlier removal techniques, filtering out misleading fluctuations.

3.2 Learning Stability

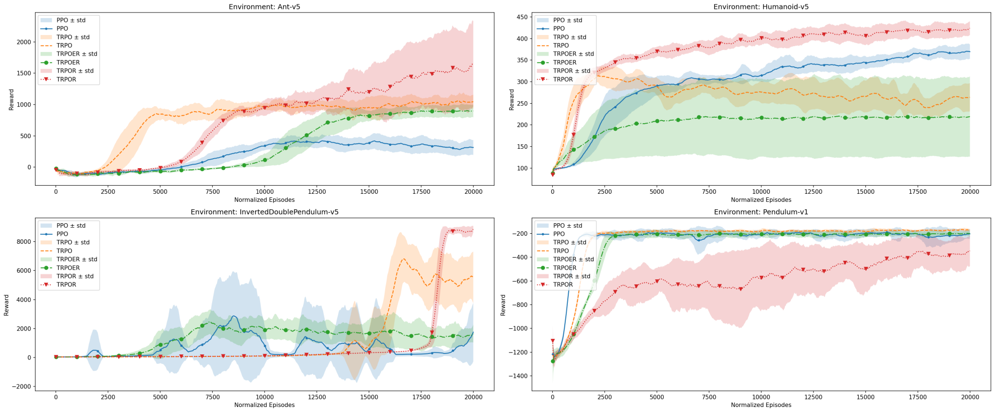

Fig. 2. Learning Stability for Different Models

Learning stability is evaluated based on the smoothness of the reward curve. A more stable learning process exhibits a steadily increasing reward trajectory, whereas high variance suggests instability due to sensitivity to hyperparameters.

3.3 Learning Stability (Coefficient of Variation)

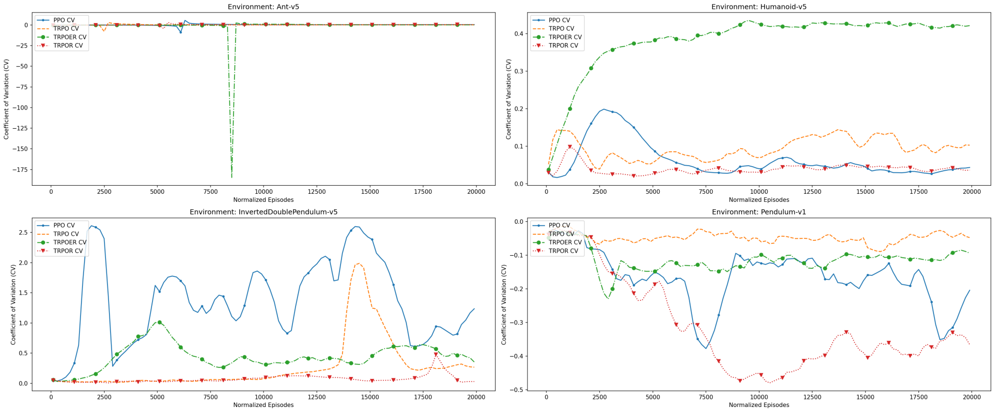

Fig. 3. Learning Stability (Coefficient of Variation)

The coefficient of variation (CV) provides a normalized measure of stability. A lower CV signifies less volatile performance, whereas a higher CV indicates inconsistency due to randomness in training.

3.4 Sample Efficiency

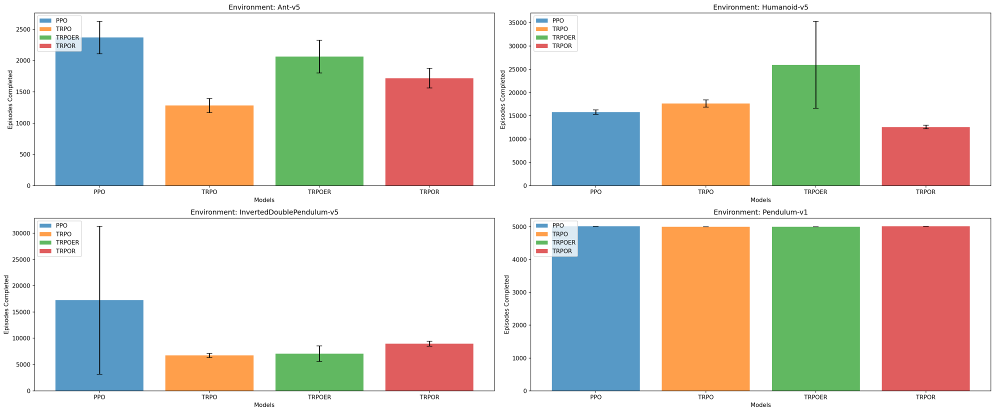

Fig. 4. Sample Efficiency Across Models

Sample efficiency measures how quickly a model improves with limited training episodes. Higher sample efficiency is desirable, especially in data-scarce scenarios.

3.5 Combined Sample Efficiency Results

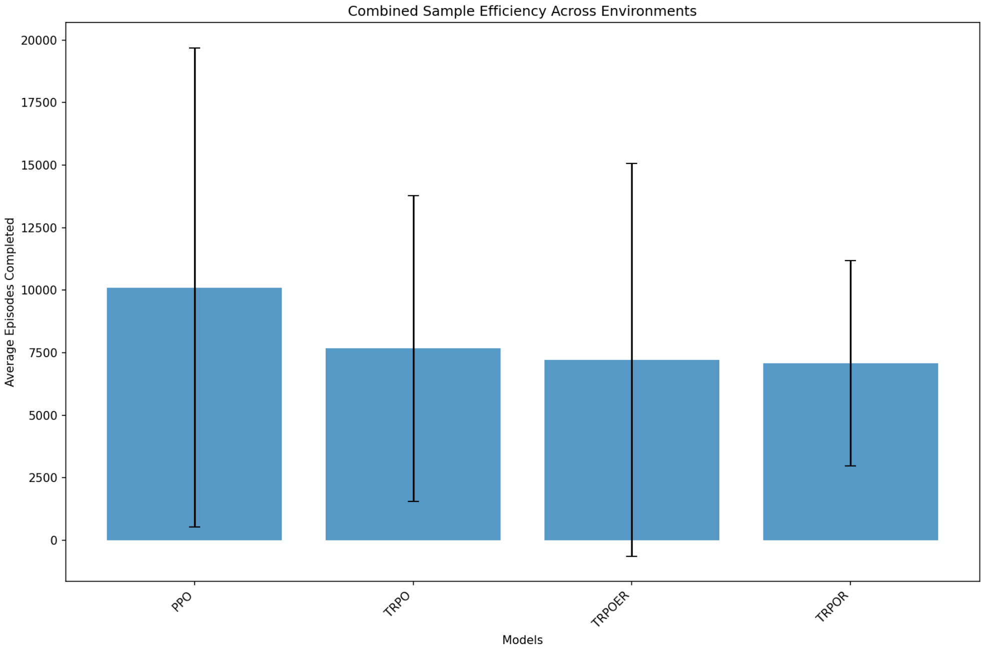

Fig. 5. Combined Sample Efficiency Results

The combined sample efficiency plot aggregates results across all environments, showing how different models perform in terms of data efficiency.

3.6 Raw Data

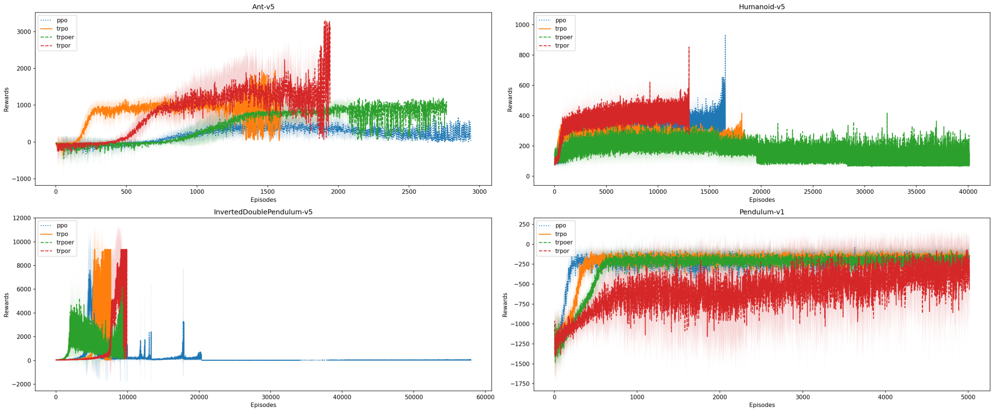

Fig. 6. Raw Reward Data for Different Models

The raw data plot displays the recorded reward values without any smoothing. It provides insights into the actual training process and variability in rewards.

References

1. Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017.
2. Sahar Roostaie and Mohammad Mehdi Ebadzadeh. Entrpo: Trust region policy optimization method with entropy regularization, 2021.
3. John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.
4. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
5. Renhao Wang, Kevin Frans, Pieter Abbeel, Sergey Levine, and Alexei A. Efros. Prioritized generative replay, 2024.