Policy Gradient and Behaviour Cloning on CartPole-v1

1 Setup

CartPole-v1: pole on a cart, two actions (push left/right), +1 reward per surviving timestep, max 500 steps. State is four-dimensional (cart position, velocity, pole angle, angular velocity). Five PG variants are trained, then the best policy serves as expert for behaviour cloning.

Architecture. Policy: MLP $[4 \to 64 \to 64 \to 2]$ with ReLU, outputting two logits fed into `Categorical(logits=...)`. Two logits rather than a single sigmoid for numerical stability and because it generalizes to > 2 actions. Value network (for methods that need it): same shape but $[4 \to 64 \to 64 \to 1]$, predicting $V(s)$.

Hyperparameters. 2000 episodes, batch size 10 (so 200 gradient updates), Adam with lr $= 10^{-3}$, $\gamma = 0.99$, seed 42. Policy and value nets use separate optimizers — shared-trunk setups caused gradient interference in early experiments.

Limitation. All results are from a single seed. Some of the observed differences between methods may not survive multi-seed testing.

2 Policy Gradient Results

Method	Final eval (mean ± std)	Converges to 450+?
Vanilla PG	303.0 ± 60.3	yes, but unstable
PG + Avg Baseline	478.7 ± 54.5	yes
PG + Value Baseline	486.8 ± 53.9	yes
PG + RLOO (K=10)	140.9 ± 26.6	no
PG + Value + Entropy	488.1 ± 48.5	yes

Table 1: Final evaluation over 100 episodes (stochastic policy).

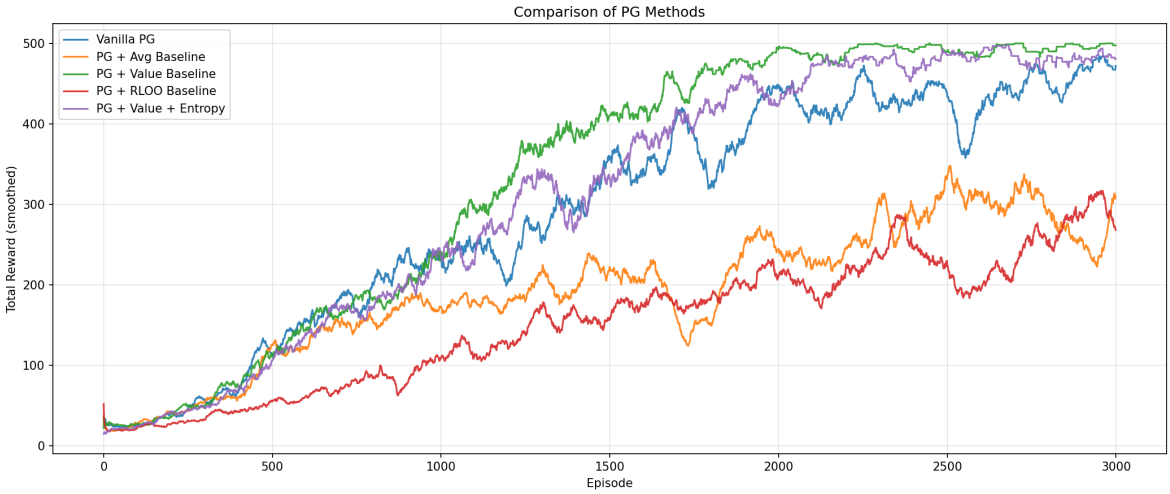

Figure 1: Per-episode reward (smoothed), all methods. RLOO plateaus and then degrades; the other four reach 400+ by episode 1500.

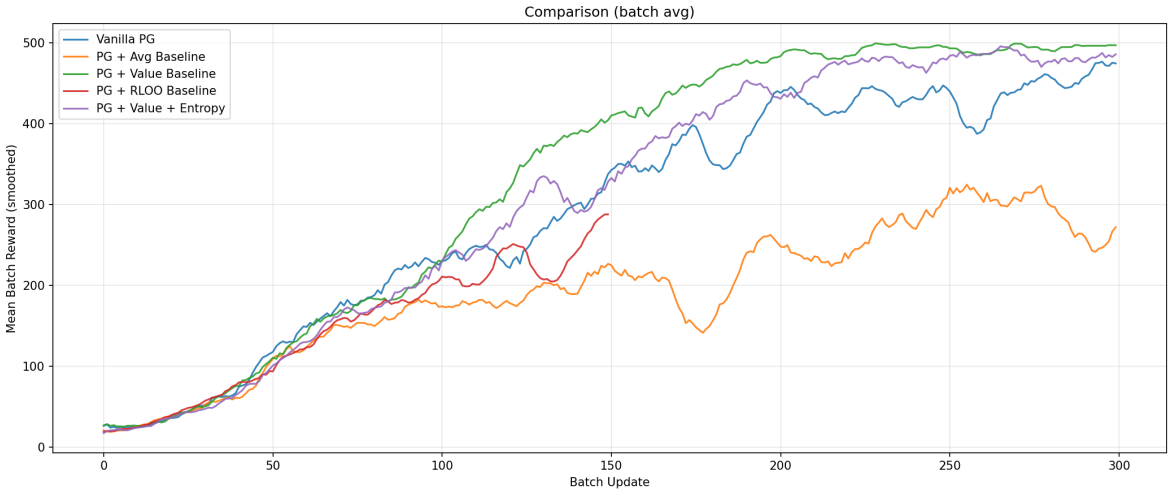

Figure 2: Mean batch reward per update. Same data, coarser view.

2.1 Vanilla PG

REINFORCE with full return G_t as weight:

$$L = -\frac{1}{B}\sum_{i=1}^{B}\sum_{t=0}^{T}\log \pi_\theta(a_t|s_t)\, G_t$$

Converges but slowly — reward hits 349 at episode 1000, peaks near 497 at 1800. Final eval (303) is well below the training peak, which points to high gradient variance causing the policy to "forget" good behavior. This is the expected failure mode of unbaselined REINFORCE.

2.2 Average-Reward Baseline

Subtracts a running mean of episode returns:

$$L = -\frac{1}{B} \sum \log \pi(a_t|s_t) \, (G_t - \bar{G})$$

where $\bar{G}$ is updated incrementally. Final eval 478.7, much better than vanilla. Training is still bumpy though (491.9 at episode 1200, drops to 163.9 at 1400, partially recovers). The baseline doesn't depend on state, so the advantage estimate is noisy for states whose true value is far from the global mean.

2.3 Value Baseline

State-dependent baseline $V(s)$ trained alongside the policy:

$$L_\pi = -\sum \log \pi(a_t|s_t) \, (G_t - V(s_t)) \tag{1}$$

$$L_V = \text{MSE}(V(s_t), \, G_t) \tag{2}$$

Most stable method: monotonic rise to 491.4, final eval 486.8. Per-state advantage estimates have lower variance than a global baseline, which is the textbook argument for actor-critic.

2.4 RLOO

Leave-one-out baseline: for trajectory i, baseline is the mean return of the other $K-1$ trajectories in the batch:

$$b_i = \frac{1}{K-1} \sum_{j \neq i} G_j$$

Unbiased, no extra network. But with K=10 and 2000 episodes, we only get 200 gradient updates. The method starts learning (271.0 at episode 1200) then degrades to 115.3. CartPole reward is essentially bimodal (fail $\approx$ 20 or succeed $\approx$ 500), so even a crude one-trajectory baseline is informative — update frequency matters more than baseline quality here. With K=2 (1000 updates) this would likely work much better.

2.5 Entropy Regularization

Adds an entropy bonus to the value-baseline loss:

$$L = L_\pi - \beta \, H(\pi)$$

with β linearly decayed from 0.1 to 0.001. High initial entropy encourages exploration; decay lets the policy commit once it has found good behavior. Final eval 488.1 — best overall, though the margin over plain value baseline (486.8) is small. The main visible benefit is smoother training without the sharp dips seen in other methods.

2.6 Discussion

On CartPole the gap between baseline methods (avg, value, entropy) is small. The task is simple enough that vanilla PG can solve it; baselines mainly improve stability, not asymptotic performance. RLOO's failure is about update count, not baseline quality — a useful reminder that batch size trades off against optimization steps under a fixed episode budget. For harder environments the differences would be larger.

3 Behaviour Cloning

3.1 Expert and Basic BC

The expert is the best RL policy (PG + Value + Entropy), evaluated stochastically at 485.0 ± 59.9. From 200 expert trajectories (≈100k state-action pairs, filtered for reward ≥ 450), a fresh MLP is trained with cross-entropy:

$$L_{\text{BC}} = -\sum \log \pi_\theta(a_{\text{expert}}|s)$$

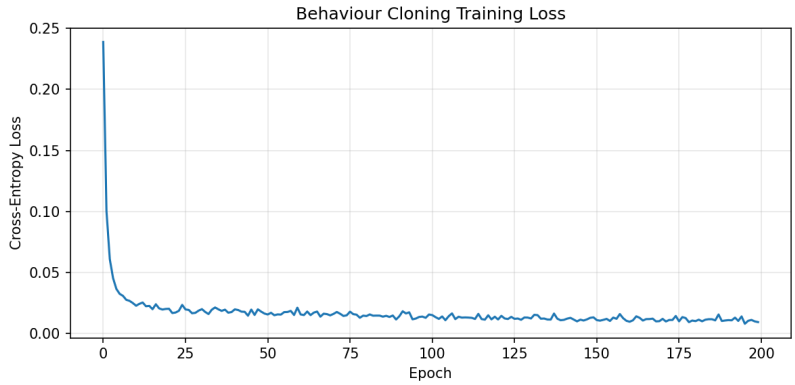

Figure 3: BC training loss. Converges quickly; the loss floor (≈0.516) reflects irreducible uncertainty near the decision boundary.

BC achieves 497.6 ± 19.0 — actually better than the expert. This makes sense: the training data is filtered (only good trajectories), and the deterministic MLP averages out the expert's stochastic mistakes near the boundary.

3.2 Experiment 1: Dataset Size

Trajectories	Pairs (s, a)	Reward
10	5,000	483.3 ± 58.9
50	25,000	497.8 ± 15.5
100	49,915	480.1 ± 78.7
200	99,912	485.9 ± 48.4

Table 2: BC works well even at 10 trajectories. CartPole's decision boundary in $\mathbb{R}^4$ is simple enough that ∼5000 points already cover it.

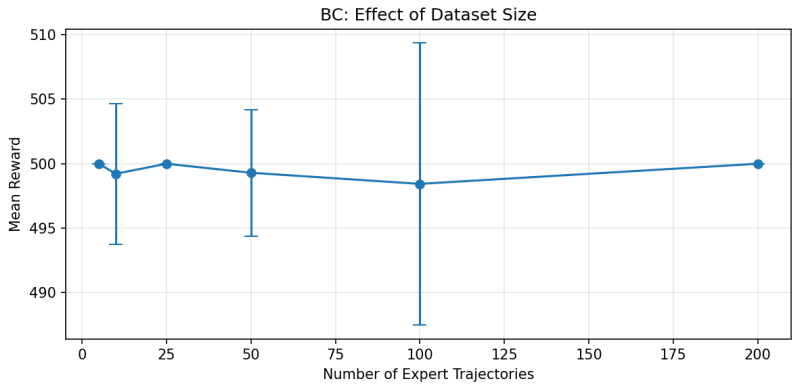

Figure 4: BC reward vs. number of expert trajectories. No clear phase transition at these sizes — the smallest dataset already has 5000 transitions.

The lack of a visible phase transition is itself informative: for a 4D binary classification problem, 5000 labeled points is already plenty. A denser sweep at lower counts (e.g. 1–3 trajectories, or measuring in raw transitions from 100 to 5000) would likely reveal the transition point.

3.3 Experiment 2: Noisy Expert

Noise prob	Reward
0%	445.6 ± 104.9
5%	451.1 ± 97.8
10%	473.6 ± 69.6
20%	322.0 ± 147.0
30%	193.7 ± 138.4

Table 3: Noise here means replacing the expert action with a random action with the given probability.

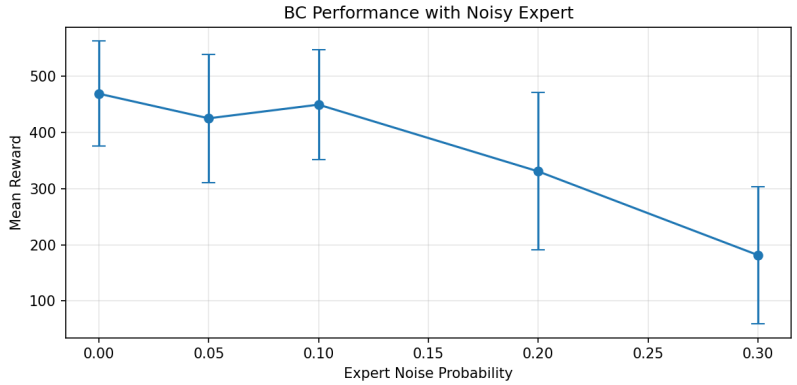

Figure 5: BC reward vs. expert noise. Robust up to ∼10%, sharp degradation between 10% and 20%.

Up to 10% noise, BC is robust: with binary actions, the majority-vote signal overwhelms the noise. Between 10–20% there's a sharp transition. At 30% the clone is barely better than random.

One odd detail: 0% noise (445.6) underperforms 10% (473.6). This is a single seed so it may be noise, but the direction is consistent with a label-smoothing effect — small noise prevents overfitting to the expert's systematic preferences in ambiguous states near the decision boundary.

Note that noise also affects trajectory length: a noisy expert falls sooner, so the dataset contains fewer late-episode states. This compounds the label corruption — worse labels *and* worse state coverage.

3.4 Experiment 3: Distribution Shift

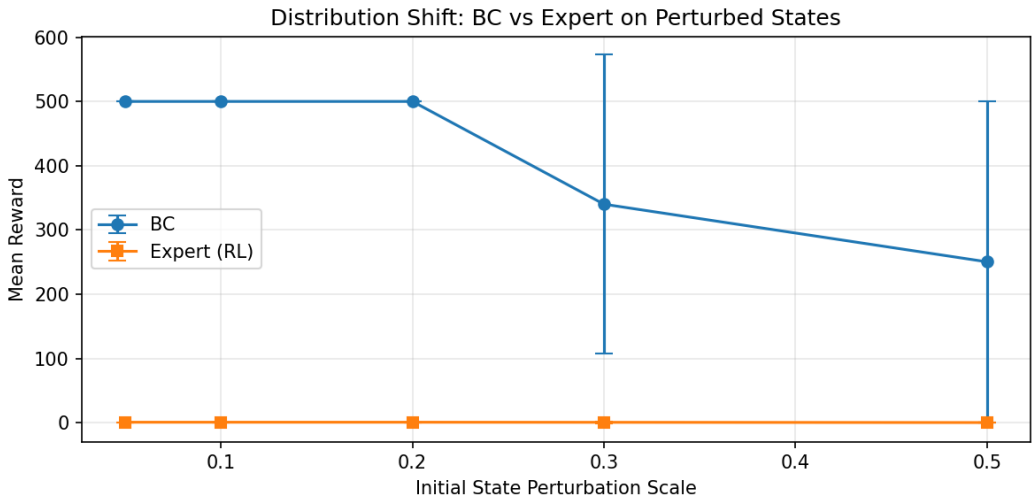

Figure 6: State visitation distributions: expert vs. BC policy (200 trajectories). On this dataset size the distributions are nearly identical — CartPole is too simple for distribution shift to manifest.

With enough data, BC and the expert visit the same states. Distribution shift — the core failure mode of BC — would show up more clearly with less data (the clone enters states not seen during training, makes bad predictions, drifts further) or with longer horizons. The theoretical compounding bound is $O(\varepsilon T^2)$ where ε is per-step error and T is horizon. On-policy RL avoids this by construction: the policy always trains on states it actually visits.

4 Takeaways

1. Baselines improve training stability on CartPole but don't change the asymptotic result much. Value baseline is the most stable; average baseline is a cheap alternative.
2. RLOO with K=10 fails here because 200 gradient updates isn't enough. The fix is smaller K, not a better baseline.
3. Entropy regularization with a linear schedule gives the best result, though the margin over value baseline alone is slim on this task.
4. BC works very well on CartPole given $\geq$ 5000 transitions. The task is a simple binary classification in $\mathbb{R}^4$.

5. Noise degrades BC nonlinearly: robust below $\sim$10%, sharp collapse between 10–20%.

6. All of this is single-seed. Several of these conclusions might not hold up under multi-seed testing — particularly the small differences between baseline methods.