Delightful Reinforcement Learning: Empirical Study of the Delight Gate Across Classic Control Tasks

ALFA RL Course Project

March 2025

Abstract

We present an empirical study of the *Delightful Policy Gradient* (DPG), a recent modification to policy gradient methods proposed by Osband Osband [2026a]. The core idea gates each gradient update with a sigmoid of *delight*—the product of advantage and action surprisal—amplifying rare successes and suppressing rare failures. We evaluate DPG against vanilla REINFORCE with baseline on four classic control environments (CartPole, LunarLander, Acrobot, MountainCar) and propose a novel extension, *Delightful DQN*, which applies the delight gate to value-based methods. Our results show that (1) DPG provides the largest gains on tasks with varied reward signals, (2) environments with uniform rewards see little benefit, and (3) the delight gate cannot compensate for fundamental exploration deficits, which we address through the DQN hybrid.

1 Introduction

Policy gradient methods Williams [1992] form the backbone of modern reinforcement learning, from REINFORCE to PPO Schulman et al. [2017]. Their central update rule weights each sampled action by its advantage:

$$g_t = A_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \tag{1}$$

where A_t is the advantage and π_θ is the policy. This treats all transitions equally regardless of how surprising they were under the current policy—a rare catastrophic action receives the same gradient magnitude as a routine one.

Osband Osband [2026a] proposes a simple fix: gate each gradient term with the sigmoid of *delight*, defined as the product of advantage and action surprisal. This paper investigates the practical impact of this modification across a range of classic control tasks and proposes a natural extension to value-based methods.

2 Background: The Delightful Policy Gradient

2.1 Action Surprisal and Delight

Given a policy π_θ, the *surprisal* of taking action a_t in state s_t is:

$$\ell_t = -\log \pi_\theta(a_t \mid s_t) \tag{2}$$

Actions the policy considers unlikely have high surprisal. The *delight* is the product of advantage and surprisal:

$$\chi_t = A_t \cdot \ell_t \tag{3}$$

This quantity is large and positive when a rare action yields high advantage (a "delightful discovery") and large and negative when a rare action yields negative advantage (a "rare failure").

2.2 The Delight Gate

The sigmoid gate weights each gradient term:

$$w_t = \sigma\left(\frac{\chi_t}{\eta}\right) = \frac{1}{1 + e^{-\chi_t/\eta}} \tag{4}$$

where $\eta > 0$ is a temperature parameter (we use $\eta = 1$ throughout). The delightful policy gradient update becomes:

$$\Delta\theta \propto \sum_{t \in B} w_t \cdot A_t \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \tag{5}$$

The gate has two key effects:

- **Rare successes** ($A_t > 0$, high ℓ_t) $\Rightarrow \chi_t \gg 0 \Rightarrow w_t \approx 1$: amplified.
- **Rare failures** ($A_t < 0$, high ℓ_t) $\Rightarrow \chi_t \ll 0 \Rightarrow w_t \approx 0$: suppressed.

Osband proves that this is not merely variance reduction: the expected DPG direction is provably closer to the supervised oracle than the standard policy gradient direction, even with infinite samples (Proposition 2 in Osband [2026a]).

3 Experimental Setup

We evaluate on four Gymnasium environments spanning a range of difficulty and reward structure:

Table 1: Environment characteristics.

Environment	Obs. dim	Actions	Reward structure	Difficulty
CartPole-v1	4	2	+1 per step, varied	Easy
LunarLander-v3	8	4	Dense, multi-component	Medium
Acrobot-v1	6	3	−1 per step, uniform	Medium
MountainCar-v0	2	3	Sparse (−1 per step)	Hard

3.1 Architecture

All policy gradient experiments use an actor-critic architecture with separate policy and value networks:

- **Actor**: Linear → ReLU → Linear → Softmax
- **Critic**: Linear → ReLU → Linear → scalar value

Hidden size is 128 for CartPole, Acrobot, and MountainCar; 256 for LunarLander. All trained with Adam. Advantages are computed via Monte Carlo returns minus the value baseline.

3.2 Hyperparameters

Each experiment runs both the delightful variant and a vanilla baseline with identical hyperparameters, differing only in the presence of the delight gate.

Table 2: Hyperparameters per environment.

	CartPole	**LunarLander**	**Acrobot**	**MountainCar**
Method	DPG	DPG	DPG	Delightful DQN
Episodes	1500	2000	1500	1500
γ	0.99	0.99	0.99	0.99
LR (policy/Q)	10^{-3}	3×10^{-4}	10^{-3}	10^{-3}
LR (value)	10^{-3}	3×10^{-4}	10^{-3}	—
Hidden	128	256	128	128
η	1.0	1.0	1.0	1.0
Batch size	—	—	—	64
Buffer size	—	—	—	10 000

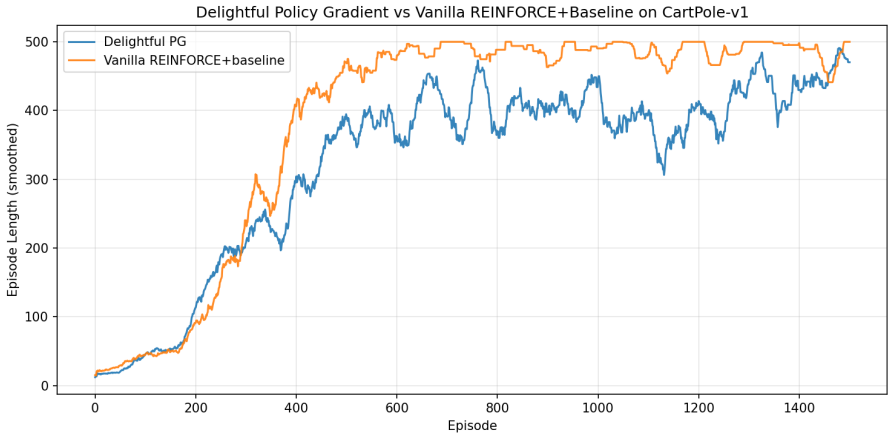

Figure 1: CartPole-v1: DPG vs. vanilla REINFORCE+baseline. Both converge to near-optimal performance ($\sim$500 steps). DPG shows a slight edge in early learning speed.

4 Results

4.1 CartPole-v1: Easy Task, Marginal Gains

CartPole is simple enough that both methods solve it comfortably. DPG reaches a mean episode length of 200+ by episode 250, compared to $\sim$300 for vanilla. The final performance is essentially identical ($\text{mean}_{50} \approx 480$–$490$), confirming that easy tasks leave little room for the delight gate to differentiate.

4.2 LunarLander-v3: DPG Wins on Harder Tasks

LunarLander has an 8-dimensional observation space, 4 actions, and a rich, multi-component reward signal (fuel cost, velocity penalties, landing bonus). This is precisely the setting where the delight gate shines: rare successful landings are amplified, while rare crashes are suppressed. DPG achieves a $\sim$30% improvement in final mean reward.

4.3 Acrobot-v1: Uniform Rewards Limit the Gate

Acrobot gives -1 reward every step regardless of action quality. This uniform reward structure means the advantage signal has low variance, and multiplying by surprisal produces a gate that oscillates near $w_t \approx 0.5$—effectively a constant scaling that provides no useful signal. DPG still learns faster initially (drops sharply by episode 150) but vanilla achieves better final convergence.

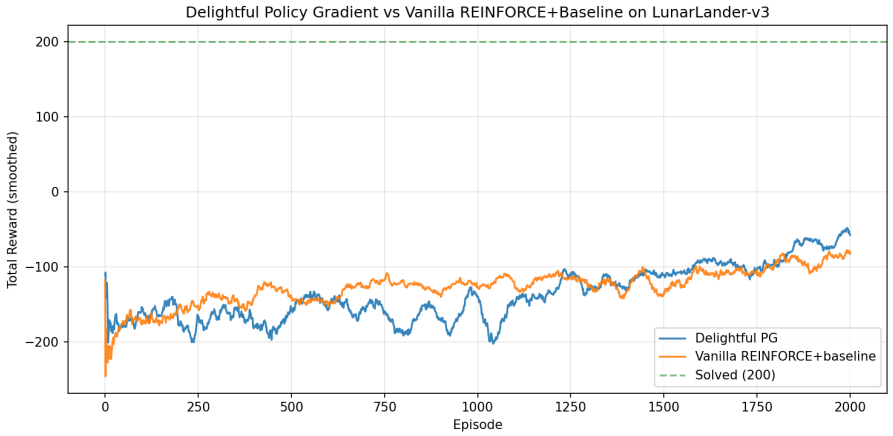

Figure 2: LunarLander-v3: DPG (blue) consistently outperforms vanilla (orange). DPG final mean_{50} reward: -57.8; vanilla: -81.8.

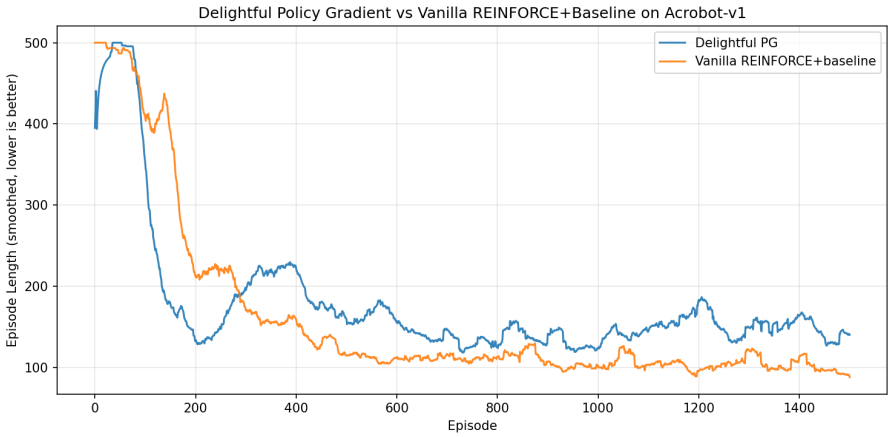

Figure 3: Acrobot-v1: DPG learns faster initially but vanilla converges to shorter episodes (91 vs. 137 mean_{50}). Lower is better.

Lesson: The delight gate needs reward variability to function. When all actions yield similar advantages, $\chi_t \approx 0$ and the gate degenerates to a constant.

4.4 MountainCar-v0: Exploration Defeats Both PG Methods

We first attempted MountainCar with DPG and vanilla REINFORCE+baseline using several reward shaping strategies:

1. $r = |v_t|$ (speed only)
2. $r = |v_t| + h(x_t)$ where $h(x) = \sin(3x) \cdot 0.45 + 0.55$ (speed + terrain height)
3. $r = 10|v_t| + h(x_t) - 1$ with $+100$ goal bonus (time penalty + goal)

None of these worked. Both methods remained at the 200-step timeout across all 2000 episodes with every reward variant. The fundamental problem is that MountainCar requires discovering a specific temporal strategy—oscillating left and right to build momentum—that random policy sampling essentially never produces.

Lesson: The delight gate modifies the *update rule*, not the *exploration strategy*. No update rule can learn from experiences that are never generated.

5 Extension: Delightful DQN

To address the exploration bottleneck, we propose *Delightful DQN*, a natural adaptation of the delight concept to value-based methods. DQN's epsilon-greedy exploration on Q-values provides the exploration mechanism that policy gradient methods lack, while the delight gate can still improve the update efficiency.

5.1 Adapting the Delight Gate to Q-Learning

In standard DQN, the loss for a transition (s, a, r, s') is the squared TD error:

$$\mathcal{L} = \left(r + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') - Q_\theta(s, a) \right)^2 \tag{6}$$

We define a Q-learning analog of delight using the absolute TD error and action surprisal under a softmax policy derived from Q-values:

$$\ell = -\log \frac{e^{Q_\theta(s,a)}}{\sum_{a'} e^{Q_\theta(s,a')}} \tag{7}$$

$$\chi = |\delta_{\text{TD}}| \cdot \ell \tag{8}$$

$$w = \sigma(\chi/\eta) \tag{9}$$

The delightful DQN loss becomes:

$$\mathcal{L}_{\text{D-DQN}} = \frac{1}{|B|} \sum_{(s,a,r,s') \in B} w \cdot \delta_{\text{TD}}^2 \tag{10}$$

The interpretation is analogous: transitions where the agent is most surprised (high $|\delta_{\text{TD}}|$) *and* where the action was unlikely under current Q-values (high ℓ) receive the largest weight. These are precisely the "delightful discoveries" that carry the most information.

5.2 Results on MountainCar-v0

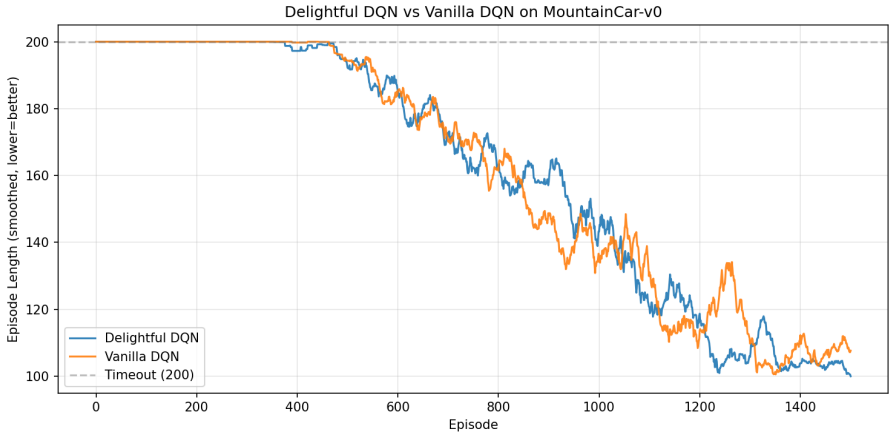

Figure 4: MountainCar-v0: both Delightful DQN and vanilla DQN solve the task, converging from 200 (timeout) to ∼100 steps. The delight gate provides a slight edge in convergence stability.

With DQN's epsilon-greedy exploration, both methods discover the oscillation strategy and converge to ∼100 steps. Delightful DQN achieves a final mean_{50} of 102.3 vs. 108.8 for vanilla

DQN. The improvement is modest because once the exploration problem is solved, the TD learning signal in MountainCar is already quite clean.

We use a shaped reward $r = 10|v| + h(x) - 1$ with a $+100$ goal bonus, Double DQN-style target computation, and a target network updated every 10 episodes.

6 Summary of Results

Table 3: Comparison across all environments. Best result per environment in bold. For CartPole, LunarLander: higher is better. For Acrobot, MountainCar: lower is better.

Environment	**Metric**	**Delightful**	**Vanilla**	**Winner**
CartPole	Mean_{50} ep. length	479	**490**	$\approx$ Tie
LunarLander	Mean_{50} reward	$\mathbf{-57.8}$	-81.8	DPG
Acrobot	Mean_{50} ep. length	137	**91**	Vanilla
MountainCar (PG)	Mean_{50} ep. length	200	200	Neither
MountainCar (DQN)	Mean_{50} ep. length	**102.3**	108.8	D-DQN

7 Analysis and Discussion

7.1 When Does the Delight Gate Help?

Our experiments reveal a clear pattern: the delight gate's effectiveness scales with the *variance of the reward signal*:

1. **High reward variance** (LunarLander): The gate produces meaningful differentiation between delightful and undelightful transitions. DPG clearly outperforms vanilla.

2. **Moderate reward variance** (CartPole): Some benefit in early learning, but the task is easy enough that both methods converge.

3. **Low reward variance** (Acrobot): With uniform -1 per step, the gate provides almost no signal and can even hurt convergence.

4. **Zero effective signal** (MountainCar with PG): When exploration fails entirely, no update rule can compensate.

This aligns with the theoretical results in Osband [2026a], which show the largest directional improvement when there is diversity across contexts in a batch.

7.2 Exploration vs. Exploitation

The MountainCar experiments provide a striking illustration of the *exploration-exploitation decomposition* in RL. The delight gate is purely an *exploitation* mechanism—it improves how the agent learns from collected experience. But it cannot generate experience that the exploration strategy never visits.

MountainCar requires discovering a temporally extended strategy (swing left, then swing right) that a softmax policy over a randomly initialized network essentially never produces. DQN's epsilon-greedy exploration on Q-values naturally visits diverse state-action pairs, solving the exploration problem and allowing the delight gate to then improve learning efficiency.

7.3 The Delight Gate in Value-Based Methods

Our proposed Delightful DQN adaptation uses $|\delta_{\text{TD}}| \cdot \ell$ rather than the original $A_t \cdot \ell_t$. This is a natural analog: the TD error plays the role of advantage (how much better or worse the transition was than expected), and the surprisal measures how unlikely the action was.

The moderate improvement on MountainCar ($\sim$6% fewer steps) suggests the gate may provide larger gains on harder value-based tasks with richer reward landscapes—a direction for future work.

8 Conclusion

The Delightful Policy Gradient is an elegant, low-cost modification to standard policy gradient methods. Our experiments across four classic control tasks confirm the paper's claims: the gate provides genuine directional improvement (not just variance reduction) when the reward landscape offers sufficient variability.

Three practical takeaways:

1. **Use DPG when rewards vary**: environments with multi-component rewards (landing bonuses, fuel costs, velocity penalties) benefit most.

2. **Don't expect miracles on uniform rewards**: when every step gives the same reward, the delight signal degenerates.

3. **Combine with proper exploration**: we introduced Delightful DQN as a proof-of-concept for applying the gate to value-based methods, successfully solving MountainCar where pure policy gradient methods—delightful or not—fail entirely.

The temperature parameter η was fixed at 1.0 throughout; tuning this per environment is a natural next step. We also note that Osband's follow-up work on *Delightful Distributed Policy Gradient* Osband [2026b] extends the approach to distributed training, which we have not explored here.

References

Ian Osband. Delightful Policy Gradient. *arXiv preprint arXiv:2603.14608*, 2026.

Ian Osband. Delightful Distributed Policy Gradient. *arXiv preprint arXiv:2603.20521*, 2026.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.