Delightful Policy Gradient

Ian Osband

Keywords: policy gradient, reinforcement learning, gradient estimation, variance reduction, scaling

Summary

Standard policy gradients weight each sampled action by advantage alone, regardless of how likely that action was under the current policy. This creates two pathologies: within a single decision context (e.g. one image or prompt), a rare negative-advantage action can disproportionately distort the update direction; across many such contexts in a batch, the expected gradient over-allocates budget to contexts the policy already handles well. We introduce the *Delightful Policy Gradient* (DG), which gates each term with a sigmoid of *delight*, the product of advantage and action surprisal (negative log-probability). For K-armed bandits, DG provably improves directional accuracy in a single context and, across multiple contexts, shifts the expected gradient strictly closer to the supervised cross-entropy oracle. This second effect is not variance reduction: it persists even with infinite samples. Empirically, DG outperforms REINFORCE, PPO, and advantage-weighted baselines across MNIST, transformer sequence modeling, and continuous control, with larger gains on harder tasks.

Contribution(s)

1. We introduce the Delightful Policy Gradient (DG), which gates each sampled gradient term by a sigmoid of *delight*, the product of advantage and action surprisal (negative log probability). DG amplifies rare successes and suppresses rare failures, yielding a simple drop-in replacement for standard policy gradients that requires no importance ratios.
   Context: Prior variance-reduction methods (baselines, control variates (Kool et al., 2019)) reduce variance but leave the expected gradient direction unchanged. Trust-region methods (Schulman et al., 2015; 2017) clip importance ratios to limit policy change, while advantage-weighted methods (Peng et al., 2019; Abdolmaleki et al., 2024) reweight by advantage alone. Neither uses action surprisal to reshape the update.
2. In tabular K-armed bandits, we prove that DG improves policy-gradient updates through two distinct mechanisms. Within a single decision context, DG reduces directional variance by suppressing noise from low-probability negative-advantage actions (Proposition 1). Across multiple contexts, DG changes the bias of the expected gradient, shifting it strictly closer to the supervised cross-entropy oracle, even in the infinite-sample limit (Proposition 2).
   Context: Standard policy gradients allocate gradient budget in proportion to current success probability, creating a self-reinforcing dynamic in which easy contexts dominate while harder ones stall (Williams, 1992). DG instead redistributes budget toward harder contexts, identifying a more balanced allocation rule for long-run learning under bandit feedback.
3. Empirically, DG outperforms REINFORCE, PPO, and advantage-weighted baselines across MNIST, transformer sequence modeling, and continuous control. The gains grow with problem difficulty: on Token Reversal, DG exhibits a smaller empirical scaling exponent than all baselines.
   Context: Together, these experiments show that DG's mechanism is not a tabular artifact: MNIST diagnoses gradient geometry, Token Reversal tests scaling in sequential learning with Transformers, and the DeepMind Control Suite (Tassa et al., 2018) tests density-based surprisal in continuous action spaces.

Delightful Policy Gradient

Ian Osband[1]
iosband@google.com

[1]**Google DeepMind**

Abstract

Standard policy gradients weight each sampled action by advantage alone, regardless of how likely that action was under the current policy. This creates two pathologies: within a single decision context (e.g. one image or prompt), a rare negative-advantage action can disproportionately distort the update direction; across many such contexts in a batch, the expected gradient over-allocates budget to contexts the policy already handles well. We introduce the *Delightful Policy Gradient* (DG), which gates each term with a sigmoid of *delight*, the product of advantage and action surprisal (negative log-probability). For K-armed bandits, DG provably improves directional accuracy in a single context and, across multiple contexts, shifts the expected gradient strictly closer to the supervised cross-entropy oracle. This second effect is not variance reduction: it persists even with infinite samples. Empirically, DG outperforms REINFORCE, PPO, and advantage-weighted baselines across MNIST, transformer sequence modeling, and continuous control, with larger gains on harder tasks.

1 Introduction

Policy gradient methods underpin some of the most consequential systems in modern AI, from superhuman game play (Silver et al., 2017; Vinyals et al., 2019) to large language model alignment (Ouyang et al., 2022). In deep learning, optimizers often normalize gradients or constrain effective step size (Kingma & Ba, 2015; You et al., 2020), making update direction a primary determinant of learning progress. Most prior work therefore asks how to estimate the policy-gradient direction with lower variance or follow it more safely. We ask a different question: is the policy-gradient direction itself the right one to follow?

We show that, in many settings, it is not. Policy gradients weight each per-sample term by advantage, regardless of how probable the sampled action was under the current policy (Williams, 1992; Sutton et al., 1999). Within a single decision context, a rare negative-advantage action can disproportionately distort the update direction, even though the policy already avoids it. Across contexts, the problem is deeper: the expected policy-gradient direction systematically over-allocates gradient budget to contexts the policy already solves well. A classifier allocates more gradient budget to an image it already gets right 99% of the time than to one it gets right 50%; likewise, an easy prompt dominates a harder one simply because the model already succeeds on it. This bias is not a finite-sample artifact: it persists even with infinite data.

We propose a simple fix: gate each gradient term with a sigmoid of *delight*, the product of advantage and action surprisal, where surprisal is the negative log-probability of the sampled action under the current policy. The resulting estimator is the *Delightful Policy Gradient* (DG): one sigmoid, one multiply, and a temperature η that we fix to 1 throughout. Within a single context, DG suppresses perpendicular noise from unlikely negative-advantage actions (Prop. 1), improving directional accuracy; this variance effect vanishes as batch size grows. Across contexts, DG shifts the expected gradient strictly closer to the supervised cross-entropy oracle (Prop. 2); this directional effect persists even in the infinite-sample limit.

We build this case progressively across theory and experiments. In tabular K-armed bandits, we isolate the two mechanisms analytically (Section 4). On MNIST contextual bandits, we directly confirm both effects and show that DG closes roughly half the gap to supervised cross-entropy (Section 3). On token reversal with transformers, DG's advantage compounds with difficulty, yielding a smaller scaling exponent than all baselines (Section 5). On continuous control across 28 environments, DG matches or exceeds baselines without task-specific tuning (Section 6).

2 Delightful Policy Gradient

We consider the standard episodic reinforcement learning setting. At each timestep t, the agent observes a history $\mathcal{H}_t$, samples an action $A_t \sim \pi_\theta(\cdot \mid \mathcal{H}_t)$, and receives reward R_t. Standard policy gradients form per-sample updates $g_t = U_t \nabla_\theta \log \pi_\theta(A_t \mid \mathcal{H}_t)$, where U_t denotes an advantage estimate. Thus each score term is weighted by advantage alone, regardless of how likely the sampled action was under the current policy. DG modulates each term by a second quantity: *action surprisal*.

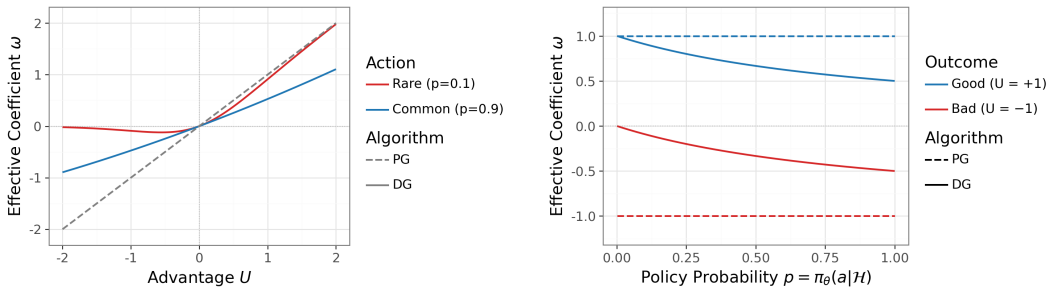
2.1 Definitions

The action surprisal is $\ell_t = -\log \pi_\theta(A_t \mid \mathcal{H}_t)$, which is large when the chosen action is unlikely under the current policy in that decision context. This surprisal is policy-relative: it measures how unlikely the action was under the policy, not how common that action is in the environment. It is also action-relative rather than outcome-relative: ℓ_t depends only on the probability the policy assigned to the sampled action, not on whether the resulting reward was good or bad.

We define *delight* as $\chi_t = U_t \ell_t$, the product of advantage and action surprisal, which is large when an unlikely action has high advantage.[1] The gate is $w_t = \sigma(\chi_t/\eta)$ for the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$, where $\eta > 0$ controls sharpness. DG therefore replaces the standard term g_t with the gated term $w_t\, g_t$. In other words, DG keeps the standard policy-gradient term but rescales it according to delight.

This gate weights score terms asymmetrically. When a rare action has positive advantage ($\chi_t \gg 0$), the gate opens and $w_t \approx 1$; we call such events *breakthroughs*. When a rare action has negative advantage ($\chi_t \ll 0$), the gate closes and $w_t \approx 0$; we call such events *blunders*. For actions the policy already favors, the surprisal is small, so the gate stays near $\frac{1}{2}$.

The intuition for this asymmetry is simple. A *blunder*—a low-probability action with negative advantage—is already being avoided; pushing it down further has limited value. A *breakthrough*—a low-probability action with positive advantage—is a discovery the policy should exploit, and increasing its probability also creates more opportunities to learn from it in the future. Standard policy gradients treat these two cases symmetrically; DG preserves breakthroughs while attenuating blunders. The sigmoid gate also arises from a local entropy-regularized objective over gate values (Appendix A). Figure 1 visualizes the resulting effective coefficient on the gradient $\nabla_\theta \log \pi$.



(a) Coefficient ω vs. advantage U.

(b) Coefficient ω vs. action probability p.

Figure 1: Effective coefficient $\omega = w \cdot U$ weighting $\nabla_\theta \log \pi$. DG amplifies breakthroughs (rare successes) and suppresses blunders (rare failures); PG (dashed) is probability-blind.

[1] We write χ for the Greek *chara* (delight).

2.2 Estimator and Implementation

Operationally, DG simply multiplies each standard policy-gradient term $g_t = U_t \nabla_\theta \log \pi_\theta(A_t \mid \mathcal{H}_t)$ by the gate $w_t = \sigma(\chi_t/\eta)$. This is a drop-in replacement for standard policy gradients:

$$\Delta\theta \;\propto\; \sum_{t\in\mathcal{B}} w_t\, g_t \;=\; \sum_{t\in\mathcal{B}} \sigma(\chi_t/\eta)\, U_t\, \nabla_\theta \log \pi_\theta(A_t \mid \mathcal{H}_t). \tag{1}$$

Because the gate reweights samples, DG changes the expected update direction, not merely its variance. At an optimal policy, on-policy advantages vanish, so delight vanishes and optimal policies are stationary points of DG for all $\eta > 0$ (Appendix A). We use $\eta = 1$ throughout and find it robust across experiments. Algorithm 1 gives pseudocode; relative to PG, DG adds one sigmoid and one multiply per sample.

Algorithm 1 Delightful Policy Gradient (discrete actions)

Require: Batch $\mathcal{B}$, policy π_θ, temperature η=1

$\Delta\theta \leftarrow 0$
for $t \in \mathcal{B}$ **do**
  $\ell_t \leftarrow -\log \pi_\theta(A_t \mid \mathcal{H}_t)$ ▷ Surprisal
  $\chi_t \leftarrow U_t \cdot \ell_t$ ▷ Delight
  $w_t \leftarrow \sigma(\chi_t/\eta)$ ▷ Gate
  $\Delta\theta \leftarrow \Delta\theta + w_t\, U_t\, \nabla_\theta \log \pi_\theta(A_t \mid \mathcal{H}_t)$
return $\Delta\theta$

For continuous actions, density-based surprisal makes $\chi = U\ell$ sensitive to action scaling. In our control experiments, clipping log-densities to $[-10, 10]$ was sufficient (Algorithm 2 in Appendix E.1); when action scales vary substantially across dimensions, one can also whiten $\tilde{\chi}_t = (\chi_t - \mu_\chi)/(\sigma_\chi + \epsilon)$ before gating.

3 MNIST Diagnostic

We cast MNIST classification as a one-step contextual bandit. Given an image, the learner predicts a digit and observes only whether that prediction was correct, not the label itself. This makes MNIST a clean test of whether policy-gradient updates recover the gradient geometry that standard supervised learning gets from full labels, without the additional complications of sequential control.

Formally, given an image X, the agent samples $A \in \{0, \ldots, 9\}$ from the policy and receives reward $R = \mathbb{I}\{A = Y\}$, where Y is the true label. The label itself is never revealed to the learner. We train a two-layer ReLU network with Adam on batches of $B = 100$ images.

To diagnose gradient quality, we compare each batch update against two oracle directions, both computed from the true labels and therefore unavailable to the learner:

$$g^*_{\mathrm{PG}} = \textstyle\sum_{x\in\mathcal{B}} p(x)\, \nabla_\theta \log \pi_\theta(Y \mid x) \qquad \text{(PG oracle)},$$
$$g^*_{\mathrm{CE}} = \textstyle\sum_{x\in\mathcal{B}} \nabla_\theta \log \pi_\theta(Y \mid x) \qquad \text{(cross-entropy oracle)},$$

where $p(x) := \pi_\theta(Y \mid x)$ is the probability assigned to the correct label. The PG oracle weights each image by its current success probability, whereas the cross-entropy oracle weights all images equally. Equivalently, these directions correspond to maximizing $\sum p(x)$ and $\sum \log p(x)$, respectively. All results average over 100 seeds; shaded regions show ± 1 standard error. Figure 2 shows that DG learns faster than PG, closing roughly half the gap to supervised cross-entropy (CE) on this configuration. CE requires labels; PG and DG see only rewards. The architecture and optimizer are identical across methods; the only difference is how each method weights its gradients.

To test whether this advantage is merely variance reduction, Figure 3 varies both the baseline and the number of independent action samples drawn per image, denoted S. The advantage takes the form

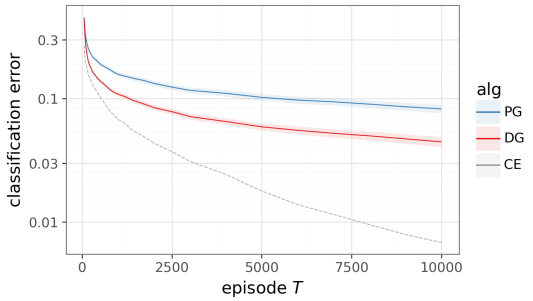


Figure 2: MNIST classification error. Supervised CE requires labels; PG and DG do not.

$U = R - b$, and we consider four baselines: $b = 0$ (zero), $b = 0.5$ (constant), $b = \hat{\mathbb{E}}[R \mid x]$ using the agent's own probability estimate (expected), and $b = \mathbb{E}[R \mid x]$ using the true label probability (oracle). As S increases, PG converges from above to the error floor set by the exact PG oracle g^*_{PG} (dashed line in Figure 3). DG surpasses this floor for every baseline: with the expected baseline, DG at $S = 1$ already matches the level that PG approaches only as $S \to \infty$. Because this gain persists at large S, it cannot be explained by variance reduction alone: DG changes the expected gradient direction itself.

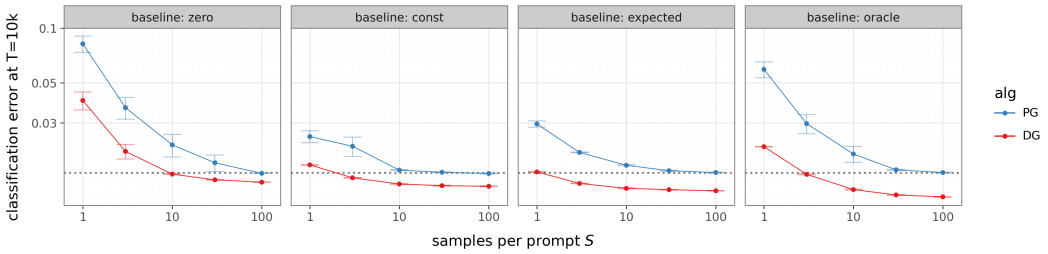


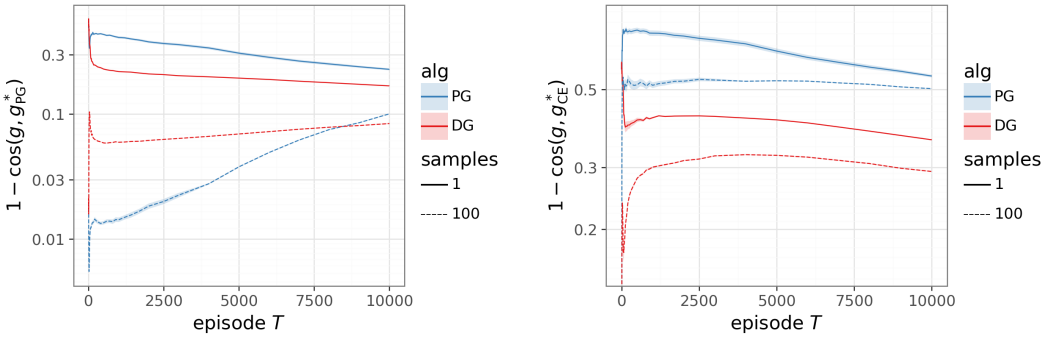
Figure 3: Classification error at $T = 10$k vs. samples per image S, faceted by baseline. Dashed line: error from the exact PG-oracle gradient g^*_{PG}, PG's best achievable direction.

Figure 4 separates the two mechanisms by measuring alignment to both oracles at $S = 1$ and $S = 100$. Against g^*_{PG} (a), DG's advantage shrinks at $S = 100$: this is the within-context variance effect disappearing as sampling noise is reduced. Against g^*_{CE} (b), DG's advantage persists at $S = 100$: this is the cross-context directional effect, which remains even when sampling noise is negligible. The only difference between the two oracles is how they weight images: the PG oracle allocates more budget to images the model already classifies correctly, whereas the cross-entropy oracle treats every image equally. DG compresses these weights toward equality, reallocating gradient budget from well-solved images to harder ones. Section 4 formalizes both mechanisms in settings where the true gradients are analytically tractable.

Appendix B provides extensive ablations showing that DG's gains persist across learning rates, batch sizes, network widths, and baselines. Validation error tracks training error closely, indicating no overfitting. The multiplicative form $\chi = U \cdot \ell$ also outperforms additive alternatives and entropy regularization. Taken together, these results suggest that DG addresses a core policy-gradient mismatch that appears even in canonical classification under bandit feedback.

4 Tabular Analysis

The MNIST diagnostics revealed two effects: DG denoises the gradient estimate (Figure 4a) and rotates its expected direction toward the cross-entropy oracle (Figure 4b). To isolate these mechanisms analytically, we move to the tabular setting, replacing the neural network with explicit tables over actions, and study K-armed bandits under simple assumptions: symmetry, orthogonality, and normalized steps.



(a) Misalignment to g^*_{PG}.

(b) Misalignment to g^*_{CE}.

Figure 4: Gradient misalignment over training at $S = 1$ (solid) and $S = 100$ (dashed). **(a)** DG's advantage diminishes with S (variance). **(b)** DG's advantage persists at $S = 100$ (directional).

The MNIST, transformer, and control experiments violate all of these assumptions; the tabular analysis isolates mechanism rather than modeling the full applications. We parameterize policies by logits z, so $\phi(a) = \nabla_z \log \pi(a) = e_a - \pi$.[2] To model direction-limited optimization, each step takes the form $z \leftarrow z + \alpha g/\|g\|$: every update has norm α, so better direction is the only way to learn faster. This is a useful idealization of large-scale training regimes in which adaptive optimization and gradient clipping can attenuate differences in raw gradient norm, making update direction increasingly important for progress (Appendix C.1).

4.1 Single context: variance reduction

We begin with a single context: a K-armed bandit with one correct action y^* among $K \geq 3$ choices, like classifying a single MNIST image but with the neural network replaced by an explicit probability table. The reward is $R = \mathbb{I}\{A = y^*\}$. In any single context of this form, the reward and cross-entropy oracles coincide. The score identity $\sum_a \pi(a)\,\phi(a) = 0$ implies $\mathbb{E}[g_{\mathrm{PG}}] = \pi(y^*)\,\phi(y^*) \propto g^*_{\mathrm{CE}}$. The only way to improve gradient quality is therefore to reduce perpendicular sampling noise. We write $\Pi_\perp$ for projection orthogonal to ∇J and $\mathrm{Var}_\perp(g) = \mathbb{E}\|\Pi_\perp(g)\|^2$ for the perpendicular variance.

Symmetry yields closed-form equalities. We specialize to a policy with $\pi(y^*) = 1 - \varepsilon$ and $\pi(a) = \varepsilon/(K-1)$ for each incorrect action, with baseline $b \in (0, 1)$. The DG gate then takes a common value w_+ on the correct action and w_- on all incorrect actions, with $w_- < \frac{1}{2} < w_+$.[3]

Proposition 1 (Variance reduction in symmetric bandits)**.** *In the bandit above, for any $\eta > 0$:*

(i) *DG preserves the expected gradient direction: $\mathbb{E}[g_{\mathrm{DG}}] = s \cdot g^*_{\mathrm{PG}}$, where $s = (1-b)\,w_+ + b\,w_- > 0$.*

(ii) *DG reduces perpendicular variance by exactly w_-^2: $\mathrm{Var}_\perp(g_{\mathrm{DG}}) = w_-^2 \cdot \mathrm{Var}_\perp(g_{\mathrm{PG}})$.*

(iii) *For the batch mean $\bar{g} = \frac{1}{B}\sum_{i=1}^B g_i$, in the regime where $\bar{g}$ concentrates around its mean,*

$$\frac{1 - \mathbb{E}[\cos(\bar{g}_{\mathrm{DG}},\, g^*_{\mathrm{PG}})]}{1 - \mathbb{E}[\cos(\bar{g}_{\mathrm{PG}},\, g^*_{\mathrm{PG}})]} \;\approx\; \frac{w_-^2}{s^2} \;<\; 1. \tag{2}$$

DG always reduces the alignment gap; both methods converge to cosine 1 as $B \to \infty$.

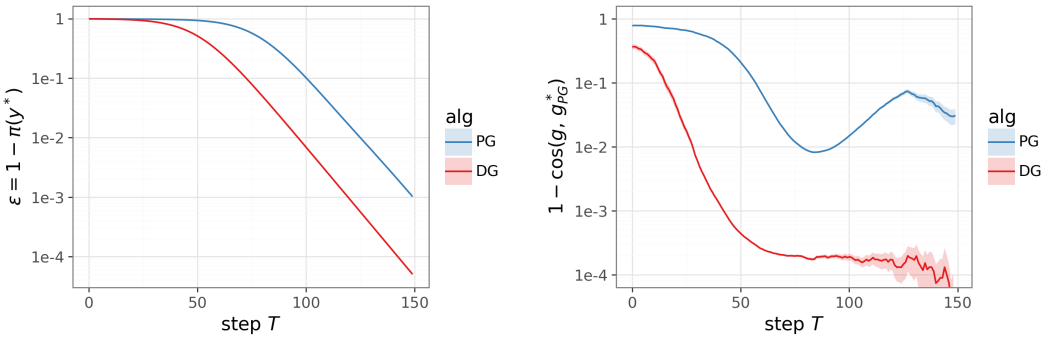
The ratio w_-^2/s^2 measures the fraction of PG's cosine gap that DG retains. Since $\sigma(-x) \leq e^{-x}$, the gate satisfies $w_- \leq \sqrt{\varepsilon/(K-1)}$ for $b = \frac{1}{2}$ and $\eta = 1$, giving $w_-^2/s^2 \leq 16\,\varepsilon/(K-1)$. Evaluating the exact ratio confirms that the reduction is substantial: for $K{=}100$ at $\varepsilon{=}0.5$, DG retains only 4% of PG's cosine gap; at $\varepsilon{=}0.1$, only 1%.

[2] Elsewhere ϕ denotes the score in general θ-space; under logits, it simplifies to $e_a - \pi$.

[3] Explicitly: $w_+ = \sigma\big((1-b)\log\frac{1}{1-\varepsilon}/\eta\big)$ and $w_- = \sigma\big(-b\log\frac{K-1}{\varepsilon}/\eta\big)$.

Beyond symmetry. Without uniform incorrect-action probabilities, each incorrect action receives a different gate, so $\mathbb{E}[g_{\mathrm{DG}}]$ is no longer exactly collinear with ∇J. However, tail suppression still holds: $w_-(a) \le \pi(a)^{b/\eta}$, so rare actions are damped more aggressively than under PG. As the policy concentrates ($\varepsilon \to 0$), the directional bias vanishes and the variance bound tightens (Appendix C.3).

We validate this picture with K=100, B=100, α=0.1, and η=1, averaged over 100 seeds. Despite identical step magnitudes, DG converges faster (Figure 5a) and maintains lower misalignment throughout training (Figure 5b). Late in training, incorrect actions become rare but each delivers a large perpendicular kick; PG's misalignment rebounds while DG's stays suppressed.



(a) Error $\varepsilon = 1 - \pi(y^*)$. (b) Misalignment $1 - \cos(g,\, g^*_{\mathrm{PG}})$.

Figure 5: Symmetric bandit, normalized steps (K=100, B=100, α=0.1).

4.2 Multiple contexts: directional improvement

For a single context, the reward and cross-entropy oracles coincide (Section 4.1), so DG can only reduce variance. In practice, however, a single gradient step must improve many contexts at once: each MNIST image is a separate K=10 classification problem, and the update must allocate learning across all of them. With N independent contexts at each step (a *contextual bandit*), the two oracles diverge. PG allocates gradient budget proportional to p_n, the probability of the correct action in context n; CE allocates equally, making direction itself a degree of freedom. MNIST confirmed this: DG surpasses PG's performance floor even at large S (Figure 3).

Formally, suppose the agent faces N independent contexts at each step, sums their gradients, and takes a single normalized step. Each context n contributes an orthogonal gradient component v_n; let $p_n := \pi_n(y_n)$ denote the probability of the correct action. With baseline $b = 0$, only correct actions contribute, and the three gradient directions differ only in how they weight contexts:[4]

$$\begin{aligned} g^*_{\mathrm{CE}} &= \textstyle\sum_n v_n && \text{(cross-entropy: equal weight)}, \\ g^*_{\mathrm{PG}} &= \textstyle\sum_n p_n\, v_n && \text{(PG: weight } \propto p_n\text{)}, \\ \mathbb{E}[g_{\mathrm{DG}}] &= \textstyle\sum_n p_n\, \sigma\big((-\log p_n)/\eta\big)\, v_n && \text{(DG: weights compressed by gate)}. \end{aligned} \tag{3}$$

PG weights each context by p_n, so well-solved contexts dominate the gradient. The gate $\sigma((-\log p_n)/\eta)$ is larger when p_n is small and smaller when p_n is large, partially cancelling the p_n weighting and redistributing budget toward hard contexts (Figure 4b).

These three directions are not arbitrary. Under normalized steps, $c_n \propto p_n$ is greedy-optimal for $\Delta(\sum p_n)$ and $c_n \propto 1$ for $\Delta(\sum \log p_n)$ (Lemma 1). PG's direction is myopically optimal but self-reinforcing: since $\nabla p_n = p_n v_n$, improving an easy context makes its gradient larger, attracting even more budget next step. DG moves weights toward $c_n \propto 1$, rebalancing budget to hard contexts where each step yields more long-run progress. As with the Kelly criterion (Kelly, 1956), greedy single-step optimality does not imply optimal long-run compounding.

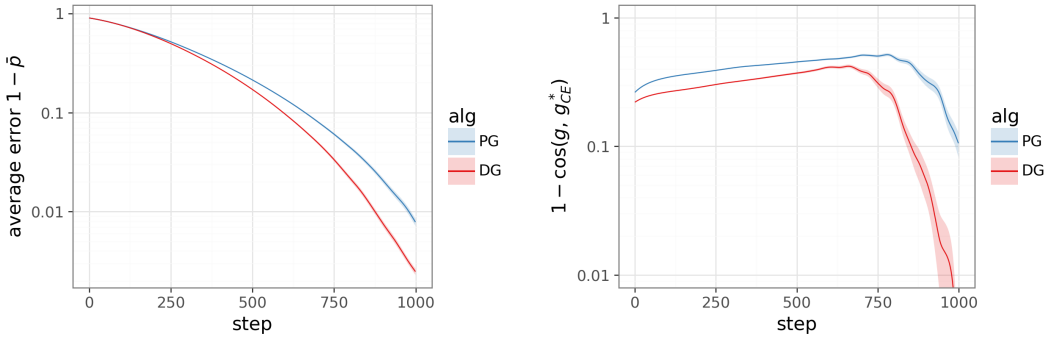
[4]With baseline $b > 0$, DG additionally suppresses incorrect-action variance (Prop. 1). Setting $b = 0$ isolates the cross-context reweighting for this subsection.

Lemma 1 (Greedy directions under normalized steps)**.** *Under a normalized step with $g = \sum_n c_n v_n$: the maximizer of $\Delta(\sum p_n)$ is $c_n \propto p_n$, and the maximizer of $\Delta(\sum \log p_n)$ is $c_n \propto 1$.*

Proposition 2 (Directional improvement toward cross-entropy)**.** *Let $N = 2$ with $p_1 \neq p_2$ and $\eta > 1/2$. Then $\cos\big(\mathbb{E}[g_{\mathrm{DG}}],\ g^*_{\mathrm{CE}}\big) > \cos\big(g^*_{\mathrm{PG}},\ g^*_{\mathrm{CE}}\big)$. DG's expected direction is strictly closer to the cross-entropy oracle than PG's, for any batch size, including the limit $B \to \infty$.*

Proof sketch. The cosine between $c_1 v_1 + c_2 v_2$ and $v_1 + v_2$ is maximized at ratio $r = c_1/c_2 = 1$ (Appendix C.5). The DG weight function $h(p) = p\,\sigma((-\log p)/\eta)$ is increasing for $\eta > 1/2$, and the sigmoid factor is decreasing in p, so DG compresses PG's ratio: $1 < r_{\mathrm{DG}} < r_{\mathrm{PG}}$, achieving higher cosine. Appendix C.6 extends to arbitrary N via a path argument; Figure 6 confirms the compression at N=100.

We validate with N=100 independent contexts, K=10 actions each, $\mathcal{N}(0,1)$ logit init, α=0.1, averaged over 100 seeds. At each step we compute *exact* population gradients (no sampling noise), so PG follows g^*_{PG} exactly. Despite this, DG converges faster (Figure 6a): its rebalancing toward hard contexts compounds over many steps. The cosine gap (Figure 6b) persists throughout training, confirming a directional effect rather than variance reduction.



(a) Average error $1 - \bar{p}$.

(b) Misalignment to g^*_{CE}.

Figure 6: N=100 independent contexts, K=10 actions, $\mathcal{N}(0,1)$ init, exact gradients. DG converges faster (a) by rebalancing gradient budget to hard contexts (b).

The two mechanisms map onto a bias–variance decomposition. For a single context, $g^*_{\mathrm{PG}} = g^*_{\mathrm{CE}}$, so DG's advantage is pure variance reduction and vanishes as $B \to \infty$ (Prop. 1). Across multiple contexts, Proposition 2 shows something stronger: DG improves the expected update itself, not just its noisy estimate. The induced bias is beneficial, rotating the gradient toward cross-entropy and rebalancing learning toward hard contexts. This means DG's advantage is not a finite-sample artifact: even with exact gradients, standard policy gradients can allocate update budget poorly, and DG corrects that defect.

5 Transformer Sequence Modeling

The MNIST and bandit experiments isolated DG's effect on update geometry in single-step problems. We now test the same mechanism in a sequential setting with memory, autoregressive generation, and temporal credit assignment.

We introduce the TOKEN REVERSAL task (Figure 7). An input sequence $x_{1:H}$ of length H is drawn uniformly from a vocabulary of size M, and a decoder-only Transformer autoregressively predicts the reverse sequence, $\hat{y}_t = x_{H-t+1}$. This is a controlled model of token-space reasoning: the learner must attend to the input, retain sequence structure, and generate a coherent output autoregressively. Because the output space has size M^H, fully correct sequences become exponentially rarer as horizon and vocabulary grow. We report *sequence error*, the fraction of output tokens predicted incorrectly; full details are in Appendix D.

We compare DG against REINFORCE, PPO (Schulman et al., 2017), and PMPO (Abdolmaleki et al., 2024), each tuned over its key hyperparameters (Appendix D.1). All methods share the same Transformer architecture, optimizer, and compute budget; results average over 30 seeds.

On a default configuration (H=10, M=2, K=1,000 episodes), Figure 8 shows that DG converges faster and reaches lower sequence error than all baselines. Multiplicative advantage $\times$ surprisal gating provides a qualitatively different signal from additive entropy bonuses or trust-region constraints. UCB-style mixtures $(1-\alpha)U + \alpha\ell$ improve over REINFORCE but do not match DG (Appendix D.4).

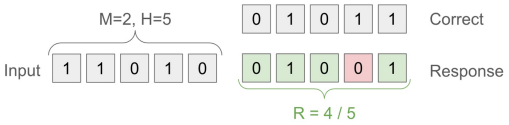


Figure 7: Token Reversal task.

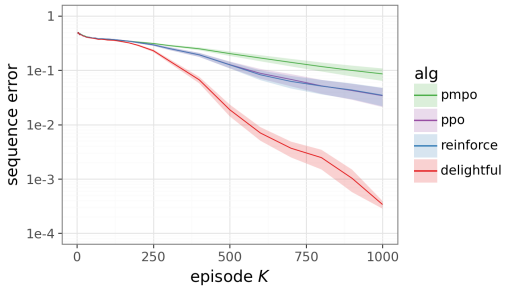
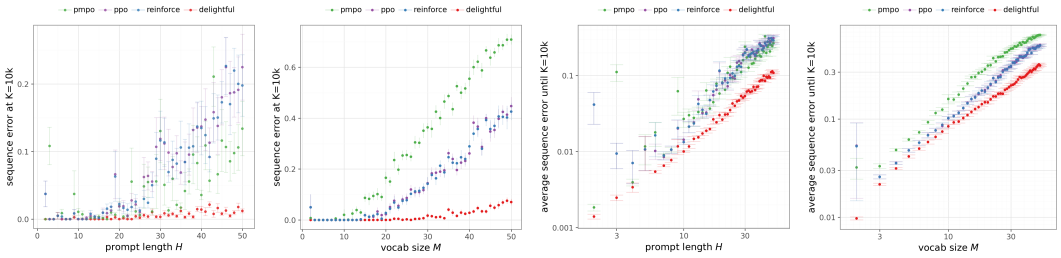


Figure 8: Sequence error on Token Reversal.

To test whether DG's advantage grows with difficulty, we increase H and M while holding the training budget fixed at K=10k episodes. Figure 9a–b shows final sequence error; Figure 9c–d shows cumulative error on log-log axes. A sharp contrast emerges: baseline performance deteriorates rapidly beyond a complexity threshold, whereas DG degrades much more gracefully. The log-log plots reveal approximate power-law scaling, with DG achieving a smaller exponent; its advantage compounds with difficulty.



(a) Final error vs. H (b) Final error vs. M (c) Avg error vs. H (d) Avg error vs. M

Figure 9: Scaling with task complexity. **(a,b)** Final sequence error after K=10k episodes; baselines degrade beyond a threshold while DG degrades more gracefully. **(c,d)** Log-log cumulative error reveals approximate power-law scaling; DG achieves a smaller exponent.

The scaling advantage reflects the mechanism identified in the bandit analysis: as M^H grows, reallocating gradient weight from high-surprisal failures to high-surprisal successes becomes increasingly valuable. Appendix D.3 tests eight task variants; DG leads in every configuration, with larger margins in harder settings. If these trends transfer to larger-scale sequence generation, where long horizons and large vocabularies make correct outputs increasingly rare, the advantage of rebalancing gradient weight could be substantial.

6 Continuous Control

Token Reversal showed that DG's advantage grows with task complexity in a discrete sequential setting. We now test whether the same mechanism transfers to continuous control, where the policy defines an action density rather than a discrete distribution.

We evaluate on the DeepMind Control Suite (Tassa et al., 2018): 28 environments, 10 million steps, and 3 seeds per environment. All methods share the same actor–critic architecture, critic algorithm (Retrace (Munos et al., 2016) with replay), and optimizer; only the policy update rule differs. DG is applied only to the actor, reweighting on-policy score terms without importance ratios. For continuous actions, delight uses clipped log-densities (Section 2.2).

DG matches or exceeds the best baseline on a majority of environments and avoids catastrophic failures: PPO collapses on `humanoid:run`, while REINFORCE fails on `hopper:hop`. On `humanoid:run`, DG discovers a successful gait while all baselines plateau (Figure 11). Across all 84 runs, DG is never the worst method (Figure 10) and achieves the lowest average regret throughout training (Figure 12). It also remains competitive with highly tuned MPO (Abdolmaleki et al., 2018) and SAC (Haarnoja et al., 2018) despite using no task-specific tuning (Appendix E.4). These results show that delight-based reweighting remains effective with continuous action densities.

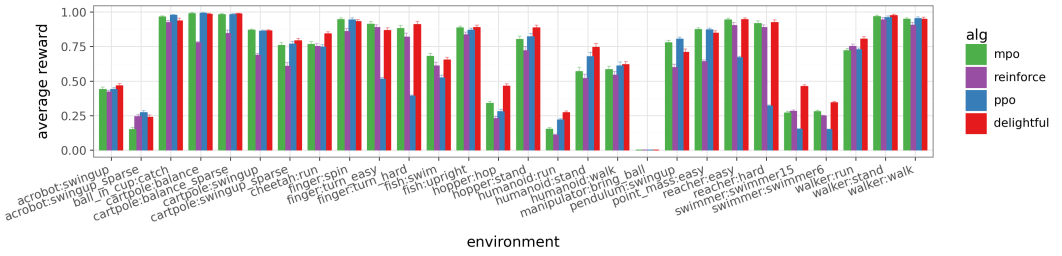


Figure 10: Average reward across 28 Control Suite environments. DG (red) is never the worst method.

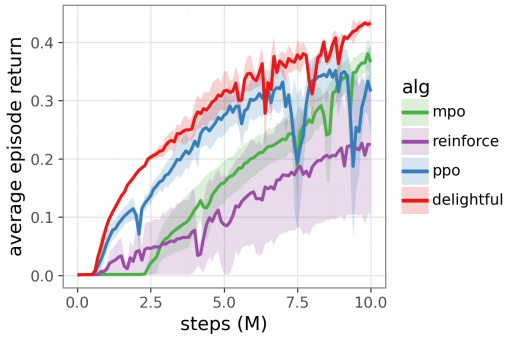


Figure 11: Learning curve on `humanoid:run`.

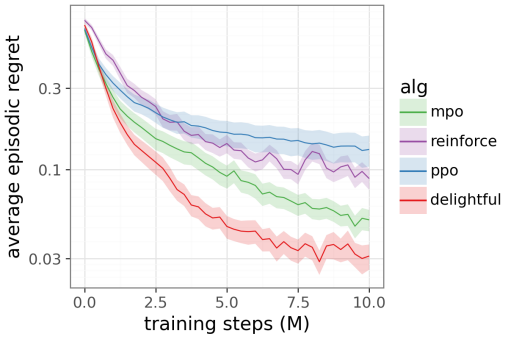


Figure 12: Aggregate regret across 28 tasks.

7 Related Work

Many policy-gradient methods can be viewed through the effective score weight ω_t multiplying $\nabla_\theta \log \pi_\theta$. This lens makes clear which methods distinguish rare from common actions and which do not (Figure 13). For standard PG, $\omega_t = U_t$ is linear in advantage and blind to action probability. For DG, $\omega_t = \sigma(U_t \ell_t / \eta)\, U_t$ depends on both advantage and surprisal, allowing rare successes and rare failures to be treated asymmetrically.

Trust-region methods. The natural policy gradient (Kakade, 2001) and its trust-region successors TRPO, PPO (Schulman et al., 2015; 2017), and GRPO (Shao et al., 2024) constrain or precondition the update, keeping ω_t in a narrow band around U_t. This limits large updates but also attenuates rare high-advantage events. DG instead modulates ω_t using only the current policy and requires no importance ratios.

Variance reduction. Leave-one-out baselines (Kool et al., 2019), generalized advantage estimation (Schulman et al., 2016), and other control-variate methods reduce gradient variance without changing the expected direction. DG's single-context mechanism (Prop. 1) achieves a related effect

through gating rather than control variates, but its cross-context rebalancing (Prop. 2) changes the expected direction itself—an effect that persists even with perfect variance reduction.

Advantage-weighted methods. AWR (Peng et al., 2019) and PMPO (Abdolmaleki et al., 2024) set $\omega_t = f(U_t)$ for some increasing function of advantage, but remain blind to surprisal. Gradient budget therefore continues to flow toward predictable actions until their advantage is driven toward zero. DG makes ω_t depend on surprisal directly, so rare and common actions are weighted differently even at the same advantage.

Entropy and intrinsic motivation. Entropy regularization (Haarnoja et al., 2018) and curiosity-based methods (Pathak et al., 2017) change the learning signal indirectly by modifying the reward or objective. DG instead leaves the reward unchanged and modifies the policy-gradient coefficient itself, reallocating gradient budget toward surprising task-relevant outcomes. Its distinct behavior relative to entropy bonuses is confirmed empirically in the MNIST experiments (Section 3).

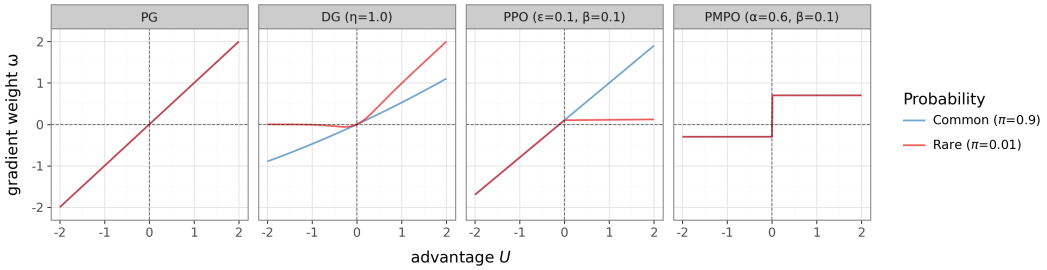


Figure 13: Score weight ω vs. advantage U for common (π=0.9) and rare (π=0.01) actions. DG treats rare successes and rare failures asymmetrically; PG and PMPO are blind to action probability, while PPO clips both tails through importance-ratio constraints.

8 Conclusion

This paper identifies a basic mismatch in standard policy-gradient learning. In the K-armed bandits we analyze, rare failures inject disproportionate noise within a context, and the expected gradient over-allocates budget to contexts the policy already handles well. MNIST, transformer, and control experiments suggest both effects persist beyond the tabular setting. DG corrects both effects by gating each score term with delight, the product of advantage and surprisal, amplifying rare successes and suppressing rare failures. The first effect, variance reduction, vanishes with infinite samples; the second, a beneficial bias toward the cross-entropy oracle, does not. We characterize both analytically and confirm them empirically across bandits, MNIST, transformer sequence modeling, and continuous control.

More broadly, these results suggest that delight is a useful primitive for decision learning under evaluative feedback. DG does more than reduce variance: it changes how finite update budget is allocated across samples and contexts. From this perspective, the method reveals that standard policy gradients use a suboptimal weighting rule for learning from evaluative feedback. Formal convergence guarantees remain open, as does the question of how far this mechanism transfers to sparse-reward settings, offline RL, and large-scale transformer training and RLHF.

Acknowledgements

We thank Ben Van Roy, Satinder Singh, Tor Lattimore, and Jincheng Mei for detailed reviews and feedback on earlier drafts. John Aslanides, Yotam Doron, Georg Ostrovski, Hubert Soyer, and Blanca Huergo contributed to the codebase and helped shape the experimental infrastructure. We are grateful to Raia Hadsell, Zoubin Ghahramani, Demis Hassabis, and Satinder Singh for fostering the research environment that made this work possible. Finally, we thank Dan Zigmond and Barry Kerzin for inspiration from Buddhist philosophy and feedback on the wider research into delightful learning.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Abbas Abdolmaleki, Bilal Piot, Bobak Shahriari, Jost Tobias Springenberg, Tim Hertweck, Rishabh Joshi, Junhyuk Oh, Michael Bloesch, Thomas Lampe, Nicolas Heess, et al. Preference optimization as probabilistic inference. *arXiv e-prints*, pp. arXiv–2410, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870, 2018.

Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems 14*, 2001.

John L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4): 917–926, 1956.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *Deep Reinforcement Learning Meets Structured Prediction, ICLR Workshop*, 2019.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pp. 1046–1054, 2016.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787, 2017.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proc. of ICML*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *Proc. of ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. of NIPS*, 1999.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

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

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

A The Delightful Gate: Derivation and Properties

We derive the sigmoid gate and establish basic properties.

Entropy-regularized gate selection. Treat each sampled score term g_t as a candidate update that is applied with weight $w \in [0,1]$. For fixed delight χ, we choose w to maximize a linear reward plus an entropy cost:

$$\max_{w\in[0,1]} \chi\, w + \eta\, H(w), \qquad H(w) = -w\log w - (1-w)\log(1-w). \tag{4}$$

Differentiating and setting to zero:

$$\frac{\partial}{\partial w}\big[\chi w + \eta H(w)\big] = \chi - \eta \log\left(\frac{w}{1-w}\right) = 0 \quad \Longrightarrow \quad w^* = \sigma\left(\frac{\chi}{\eta}\right). \tag{5}$$

Since H is strictly concave, this is the unique global maximizer.

Softplus potential. Substituting w^* into the objective yields the optimal value:

$$\Psi_\eta(\chi) = \max_{w\in[0,1]} \big\{\chi w + \eta H(w)\big\} = \eta \log\Big(1 + e^{\chi/\eta}\Big) = \eta\, \mathrm{softplus}(\chi/\eta). \tag{6}$$

This softplus potential provides a local variational interpretation of the DG gate. It saturates for $\chi \ll 0$ (suppressing updates from disfavored actions) and grows linearly for $\chi \gg 0$ (preserving updates from surprising successes).

Temperature interpolation. The temperature η controls gate sharpness and interpolates between two regimes. In the *hedonic* limit ($\eta \to \infty$), $w^* \to 1/2$ for bounded χ, recovering standard PG up to a constant: the gate ignores surprisal and responds only to advantage. In the *enlightened* limit ($\eta \to 0$), $w^* \to \mathbb{I}\{\chi > 0\}$, a hard gate that passes all positive-delight samples equally and fully suppresses negative ones. We use $\eta = 1$ throughout, which provides meaningful asymmetry without hard thresholding.

Stationarity at optimal policies. At an optimal policy π^*, on-policy advantages vanish on the support. Since the DG update is $w \cdot U \cdot \nabla_\theta \log \pi_\theta$ and $U = 0$ on supported actions, the update vanishes. Optimal policies are therefore stationary points of DG for all $\eta > 0$.

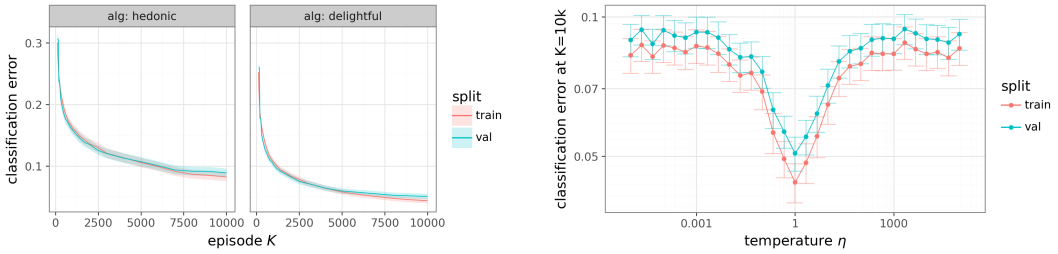
B MNIST Experimental Details

We provide supplementary details and robustness analyses for the MNIST contextual bandit experiments in Section 3.

Setup. We parameterize the policy π_θ as a two-layer MLP with ReLU activations and hidden width 100. Training proceeds in online batches: each step, the agent collects $B = 100$ trajectories, computes gradients, and updates parameters with Adam (learning rate 10^{-3}). We train for $K = 10{,}000$ episodes unless otherwise noted.

B.1 Generalization

A natural concern is that prioritizing high-surprisal events might cause overfitting. Figure 14 shows validation error tracks training error closely for both PG and DG, and the optimal temperature $\eta \approx 1$ is consistent across splits. DG's gains persist on held-out data.



(a) Learning curves (train vs. val).

(b) Temperature sweep (train vs. val).

Figure 14: Generalization on MNIST. **(a)** Validation error tracks training error; DG achieves lower error on both. **(b)** The optimal temperature $\eta \approx 1$ is consistent across train and validation.

B.2 Comparison with Entropy Regularization

A natural alternative is additive entropy regularization, augmenting the objective with $\alpha H(\pi_\theta)$. Figure 15 sweeps α for PG and η for DG. Entropy regularization is sensitive to its coefficient: small α provides negligible benefit, while large α collapses performance by forcing excessive stochasticity. DG achieves a lower error floor than any entropy-regularized baseline and maintains its advantage across a wide range of η.

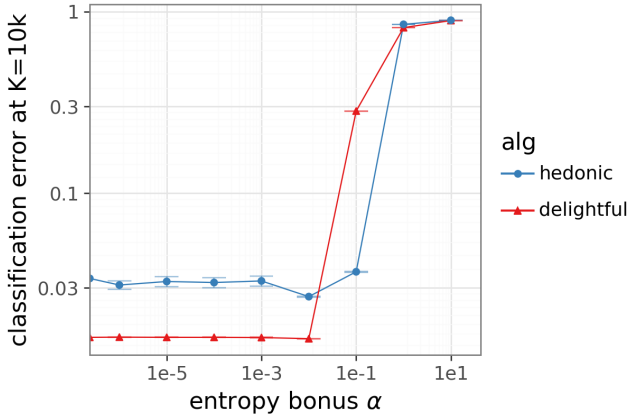


Figure 15: Entropy regularization vs. DG temperature. Large entropy coefficients degrade accuracy; DG is robust across η and consistently achieves lower error.

B.3 Robustness to Learning Rate

Figure 16 sweeps learning rate $\alpha \in [10^{-5}, 10^{-2}]$. DG consistently outperforms PG across the effective range, with a wider basin of low error around $\alpha \in [10^{-4}, 10^{-3}]$.

B.4 Robustness to Batch Size

Figure 17 sweeps batch size $B \in [1, 1000]$. Larger batches reduce gradient variance and improve both methods, but DG maintains a consistent advantage. This confirms that DG's benefit is not merely variance reduction—it provides a distinct signal that complements larger batches.

B.5 Robustness to Network Width

Figure 18 varies hidden width $W \in [1, 2048]$. Both methods exhibit a sweet spot in capacity, but DG achieves lower error at every width.

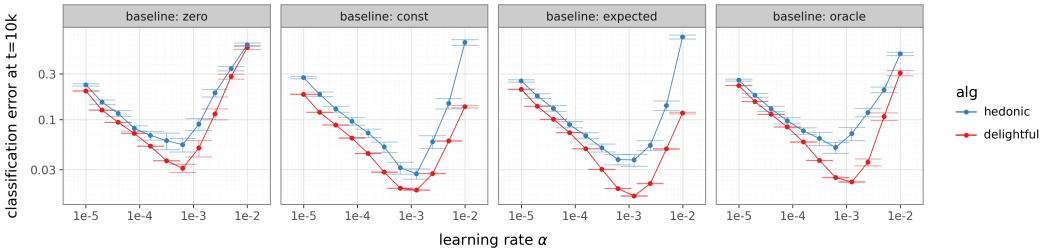


Figure 16: Learning rate sensitivity. DG (red) outperforms PG (blue) across all baselines and learning rates.

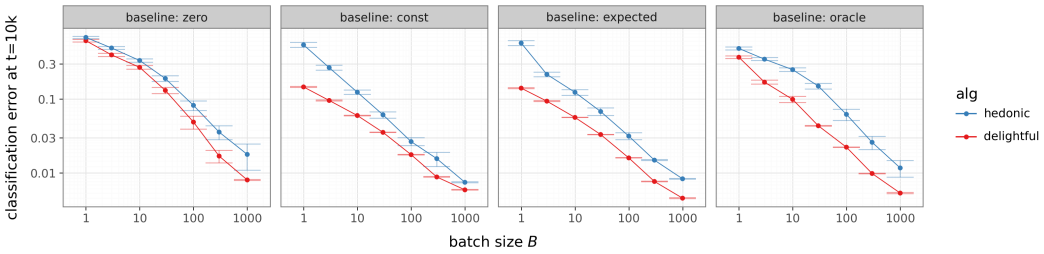


Figure 17: Batch size sensitivity. DG maintains its advantage across all batch sizes.

B.6 Alternative Definitions of Delight

We investigate whether alternative functional forms could improve on $\chi = U \cdot \ell$.

Additive mixtures. Inspired by UCB, we consider $\chi_\alpha^{\text{UCB}} = (1-\alpha)U + \alpha\ell$, interpolating between pure advantage ($\alpha = 0$) and pure surprisal ($\alpha = 1$). Figure 19(a) shows additive mixtures consistently underperform the multiplicative form.

Surprisal exponent. We generalize to $\chi_\beta = U \cdot \ell^\beta$. Figure 19(b) shows the optimum is exactly $\beta = 1$; deviations in either direction degrade performance. Within this family of alternatives, the simple product $\chi = U \cdot \ell$ performs best.

C Proofs for Tabular Analysis

This section collects the proofs for Section 4. We first justify the normalized-step model, then prove Propositions 1 and 2 and their extensions.

C.1 Cosine Controls Progress

The tabular analysis assumes normalized steps $z \leftarrow z + \alpha g/\|g\|$. The following proposition shows that under this update rule, expected improvement scales linearly with $\cos(g, \nabla J)$, so a higher-cosine gradient estimator translates directly into faster learning.

Proposition 3 (Cosine controls progress)**.** *Let J have L-Lipschitz gradient. For the normalized update $z^+ = z + \alpha g/\|g\|$,*

$$\mathbb{E}[J(z^+) - J(z) \mid z] \geq \alpha \|\nabla J(z)\| \cdot \mathbb{E}[\cos(g, \nabla J(z)) \mid z] - \tfrac{L}{2}\alpha^2.$$

Proof. By smoothness: $J(z^+) \geq J(z) + \langle \nabla J, \alpha g/\|g\| \rangle - \frac{L}{2}\alpha^2$. Taking expectations and using $\langle \nabla J, g/\|g\| \rangle = \|\nabla J\| \cos(g, \nabla J)$. □

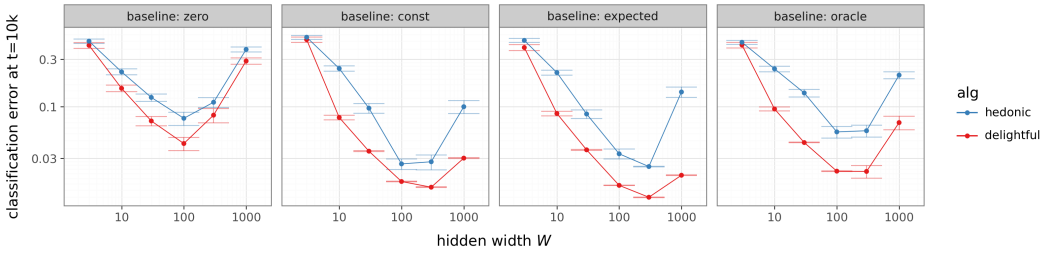
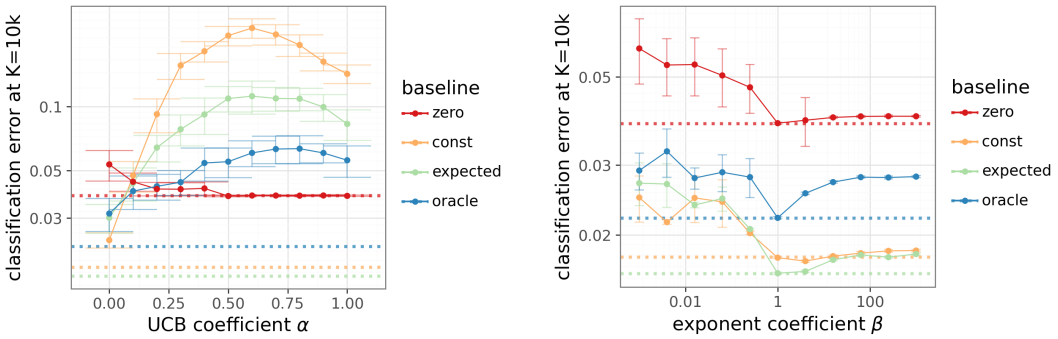


Figure 18: Network width sensitivity. DG outperforms PG across all capacity levels.



(a) Additive mixtures.

(b) Surprisal exponent β.

Figure 19: Alternative definitions of delight. **(a)** Additive mixtures underperform the multiplicative form (dotted). **(b)** The simple product ($\beta = 1$) is optimal.

C.2 Proof of Proposition 1 (Variance Reduction)

We use the symmetric bandit setup from Section 4.1: K actions, correct action y^*, $\pi(y^*) = 1 - \varepsilon$, $\pi(a) = q := \varepsilon/(K-1)$ for $a \neq y^*$, baseline b, and gate values w_+ (correct) and w_- (incorrect). The proof proceeds in three parts: (i) a symmetry lemma shows that all expected gradients are collinear, (ii) the perpendicular variance factors exactly, and (iii) a Taylor expansion converts the variance ratio into a cosine-gap ratio.

Part (i): Direction preservation. Let $u := \phi(y^*) = e_{y^*} - \pi$. We show $\sum_{a \neq y^*} \phi(a)$ is proportional to u, which forces the expected DG gradient to be collinear with $\nabla J = (1 - \varepsilon)\, u$.

Lemma 2 (Symmetry identity)**.** $\displaystyle\sum_{a \neq y^*} \phi(a) = -\frac{(K-1)(1-\varepsilon)}{\varepsilon}\, \phi(y^*).$

Proof. Since $\sum_{a=1}^{K} \phi(a) = \sum_a (e_a - \pi) = \mathbf{1} - K\pi$, we have $\sum_{a \neq y^*} \phi(a) = \mathbf{1} - K\pi - \phi(y^*)$. We verify this is proportional to $\phi(y^*)$ component by component. In component y^*: $1 - K(1-\varepsilon) - \varepsilon = -(K-1)(1-\varepsilon)$. In component $a \neq y^*$: $1 - Kq - (-q) = 1 - (K-1)q = 1 - \varepsilon$. Since $\phi(y^*)$ has component ε in position y^* and $-q = -\varepsilon/(K-1)$ in other positions, the ratio is $-(K-1)(1-\varepsilon)/\varepsilon$ in both components. □

Now compute the expected DG gradient:

$$\begin{aligned}\mathbb{E}[g_{\mathrm{DG}}] &= (1-\varepsilon)\, w_+\, (1-b)\, \phi(y^*) + \sum_{a \neq y^*} q\, w_-\, (-b)\, \phi(a) \\ &= (1-\varepsilon)\, w_+\, (1-b)\, \phi(y^*) + w_-\, (-b)\, q \cdot \Big(- \frac{(K-1)(1-\varepsilon)}{\varepsilon} \Big) \phi(y^*) \\ &= (1-\varepsilon) \Big[(1-b)\, w_+ + b\, w_- \Big]\, \phi(y^*) \\ &= s \cdot \nabla J,\end{aligned}$$

where $s = (1-b)\, w_+ + b\, w_-$ and we used $q(K-1) = \varepsilon$. Since $w_+, w_- > 0$ and $b \in (0,1)$, we have $s > 0$, confirming the direction is preserved.

Part (ii): Perpendicular variance is exactly $w_-^2 \cdot \mathrm{Var}_\perp(g_{\mathrm{PG}})$. Since $\nabla J \propto \phi(y^*)$, the projection $\Pi_\perp \phi(y^*) = 0$. Therefore the correct-action sample contributes zero perpendicular energy under both PG and DG. For incorrect actions, $g_{\mathrm{DG}}(a) = w_- \cdot g_{\mathrm{PG}}(a)$ pointwise, so $\Pi_\perp(g_{\mathrm{DG}}(a)) = w_- \cdot \Pi_\perp(g_{\mathrm{PG}}(a))$.

Note that $\mathbb{E}[\Pi_\perp(g)] = \Pi_\perp(\mathbb{E}[g]) = 0$ for both PG and DG (from Part (i), both means are parallel to ∇J), so $\mathrm{Var}_\perp$ equals the second moment. The perpendicular variance is:

$$\begin{aligned}\mathrm{Var}_\perp(g_{\mathrm{DG}}) &= \mathbb{E}\, \|\Pi_\perp(g_{\mathrm{DG}})\|^2 = \sum_{a \neq y^*} \pi(a)\, w_-^2\, b^2\, \|\Pi_\perp(\phi(a))\|^2 \\ &= w_-^2 \sum_{a \neq y^*} \pi(a)\, b^2\, \|\Pi_\perp(\phi(a))\|^2 = w_-^2 \cdot \mathrm{Var}_\perp(g_{\mathrm{PG}}).\end{aligned}$$

Part (iii): Alignment gap ratio. When $\bar{g}$ concentrates near $\mu := \mathbb{E}[\bar{g}]$, write $\bar{g} = \mu + \xi$ with ξ small. Since μ is parallel to ∇J (Part (i)), $\cos(\bar{g}, \nabla J) = \cos(\mu + \xi, \mu)$. Taylor-expanding to second order in ξ:

$$1 - \cos(\mu + \xi, \mu) \approx \frac{\|\xi_\perp\|^2}{2\|\mu\|^2},$$

where $\xi_\perp = \Pi_\perp(\xi)$ is the perpendicular component. Taking expectations:

$$1 - \mathbb{E}[\cos(\bar{g}, \nabla J)] \approx \frac{\mathrm{Var}_\perp(\bar{g})}{2\, \|\mathbb{E}[\bar{g}]\|^2} = \frac{\mathrm{Var}_\perp(g)}{2B\, \|\mathbb{E}[\bar{g}]\|^2}.$$

For PG: $\|\mathbb{E}[\bar{g}_{\mathrm{PG}}]\| = \|\nabla J\|$. For DG: $\|\mathbb{E}[\bar{g}_{\mathrm{DG}}]\| = s\, \|\nabla J\|$. Using Part (ii):

$$\frac{1 - \mathbb{E}[\cos(\bar{g}_{\mathrm{DG}}, \nabla J)]}{1 - \mathbb{E}[\cos(\bar{g}_{\mathrm{PG}}, \nabla J)]} \approx \frac{w_-^2 \cdot \mathrm{Var}_\perp(g_{\mathrm{PG}})/(s^2 \|\nabla J\|^2)}{\mathrm{Var}_\perp(g_{\mathrm{PG}})/\|\nabla J\|^2} = \frac{w_-^2}{s^2}.$$

C.3 Extension to Non-Symmetric Bandits

The symmetric assumption in Proposition 1 gives clean equalities, but the noise-suppression mechanism extends to general policies. The coincidence of oracles does not require symmetry: in any single-context bandit, the score identity $\sum_a \pi(a)\, \phi(a) = 0$ forces $\mathbb{E}[g_{\mathrm{PG}}] = \pi(y^*)\, \phi(y^*) \propto g^*_{\mathrm{CE}}$. The directional improvement of Proposition 2 is therefore a fundamentally multi-context phenomenon; in a single context, DG can only reduce variance.

Variance suppression. For any incorrect action a with negative advantage ($U < 0$) and surprisal $\ell(a) = -\log \pi(a)$, the DG gate satisfies

$$w(a) = \sigma(-b\, \ell(a)/\eta) \;\leq\; e^{-b\, \ell(a)/\eta} = \pi(a)^{b/\eta}.$$

For rare actions ($\pi(a) \ll 1$), this bound is small: the gate suppresses their contribution by at least $\pi(a)^{b/\eta}$. In a general (non-symmetric) bandit, each incorrect action a contributes $\pi(a)\, w(a)^2\, b^2\, \|\Pi_\perp(\phi(a))\|^2$ to $\mathrm{Var}_\perp(g_{\mathrm{DG}})$. Using the bound above:

$$\mathrm{Var}_\perp(g_{\mathrm{DG}}) \;\le\; \sum_{a \neq y^*} \pi(a)^{1+2b/\eta}\, b^2\, \|\Pi_\perp(\phi(a))\|^2.$$

When $b/\eta > 0$, the exponent $1 + 2b/\eta > 1$ ensures that rare actions are suppressed more aggressively than under PG (which has exponent 1). The exact equality of Proposition 1(ii) becomes an inequality, but the qualitative conclusion—DG suppresses perpendicular noise from rare incorrect actions—holds without symmetry.

Directional bias. Without symmetry, each incorrect action receives a different gate value, so $\mathbb{E}[g_{\mathrm{DG}}]$ is no longer exactly parallel to ∇J. This introduces a small perpendicular bias—a directional cost, since the two oracles already coincide in a single context. The bias vanishes as the policy concentrates on the correct action ($\varepsilon \to 0$): all incorrect-action contributions shrink, and the gate values converge.

C.4 Proof of Lemma 1 (Greedy Directions)

We derive the greedy-optimal direction for each objective by Cauchy–Schwarz. Let $a_n := \|v_n\|^2$ then for an update $z \leftarrow z + \alpha\, g/\|g\|$ with $g = \sum_n c_n v_n$ and $\|g\|^2 = \sum_n c_n^2 a_n$ (by orthogonality):

Arithmetic objective. $\Delta(\sum_n p_n) = \sum_n \langle \nabla_{z_n} p_n, \alpha c_n v_n/\|g\| \rangle$. Since $\nabla_{z_n} p_n = p_n v_n$, each term is $\alpha c_n p_n a_n/\|g\|$. So $\Delta(\sum p_n) = \alpha \sum_n c_n p_n a_n / \sqrt{\sum_m c_m^2 a_m}$.

To maximize over c, set $x_n = c_n\sqrt{a_n}$ and $y_n = p_n\sqrt{a_n}$. The ratio $\sum x_n y_n/\|x\|$ is maximized by Cauchy–Schwarz when $x \propto y$, i.e., $c_n \propto p_n$.

Log objective. $\Delta(\sum_n \log p_n) = \sum_n \langle \nabla_{z_n} \log p_n, \alpha c_n v_n/\|g\| \rangle$. Since $\nabla_{z_n} \log p_n = v_n$, each term is $\alpha c_n a_n/\|g\|$. Maximized when $c_n \propto a_n/a_n = 1$ (again by Cauchy–Schwarz with $y_n = \sqrt{a_n}$). □

C.5 Proof of Proposition 2 ($N = 2$)

The key tool is a monotonicity lemma: the cosine between a weighted sum and the equal-weight sum is uniquely maximized when the ratio of weights equals one.

Lemma 3 (Two-vector cosine monotonicity)**.** *For v_1, v_2 orthogonal with norms $a_1, a_2 > 0$, define $C(r) := \cos(r\, v_1 + v_2,\ v_1 + v_2)$ for $r > 0$. Then $C(r)$ is uniquely maximized at $r = 1$ and strictly decreasing as r moves away from* 1.

Proof. By orthogonality:

$$C(r) = \frac{r\, a_1 + a_2}{\sqrt{(r^2 a_1 + a_2)(a_1 + a_2)}}.$$

Since $C(r) > 0$ on $(0, \infty)$, C is maximized where $C(r)^2$ is. Define $Q(r) := C(r)^2 \cdot (a_1 + a_2) = (r\, a_1 + a_2)^2/(r^2 a_1 + a_2)$. Differentiating by the quotient rule and simplifying the numerator:

$$Q'(r) = \frac{2a_1 a_2 (1 - r)(r\, a_1 + a_2)}{(r^2 a_1 + a_2)^2}.$$

Since $a_1, a_2 > 0$ and $r\, a_1 + a_2 > 0$, we have $Q'(r) > 0$ for $r < 1$ and $Q'(r) < 0$ for $r > 1$. Thus Q (and hence C) is uniquely maximized at $r = 1$. □

Proof of Proposition 2. The PG population direction has ratio $r_{\mathrm{PG}} = p_1/p_2$. The DG population direction has ratio $r_{\mathrm{DG}} = h(p_1)/h(p_2)$, where $h(p) = p\,\sigma((-\log p)/\eta)$.

Assume WLOG $p_1 > p_2$, so $r_{\mathrm{PG}} > 1$. Since h is increasing, $r_{\mathrm{DG}} = h(p_1)/h(p_2) > 1$. Since the gate $\sigma((-\log p)/\eta)$ is decreasing, $r_{\mathrm{DG}} = (p_1/p_2)\cdot[\sigma((-\log p_1)/\eta)/\sigma((-\log p_2)/\eta)] < r_{\mathrm{PG}}$. Thus $1 < r_{\mathrm{DG}} < r_{\mathrm{PG}}$, and by Lemma 3, $C(r_{\mathrm{DG}}) > C(r_{\mathrm{PG}})$. □

C.6 Directional Improvement for General N

The $N = 2$ proof (Appendix C.5) uses direct ratio compression. For general N, we interpolate continuously between PG and DG weights and show the squared cosine strictly increases along the path.

Proposition 4 (Directional improvement, general N)**.** *Let $N \geq 2$ contexts with orthogonal score vectors v_n of norms $a_n > 0$. Let $\eta > 1/2$, and suppose the p_n are not all equal. Then*

$$\cos\big(\mathbb{E}[g_{\mathrm{DG}}],\ g_{\mathrm{CE}}\big)\ >\ \cos\big(\mathbb{E}[g_{\mathrm{PG}}],\ g_{\mathrm{CE}}\big).$$

Proof. Define the interpolated weights $c_n(t) = p_n\,[\sigma((-\log p_n)/\eta)]^t$ for $t \in [0,1]$. At $t = 0$, $c_n(0) = p_n$ (PG weights); at $t = 1$, $c_n(1) = p_n\,\sigma((-\log p_n)/\eta)$ (DG weights). Define the squared-cosine proxy

$$\Phi(t) := \frac{\big(\sum_n c_n(t)\,a_n\big)^2}{\sum_n c_n(t)^2\,a_n},$$

which is proportional to $\cos^2\big(\sum c_n(t)\,v_n,\ \sum v_n\big)$. We show $\Phi'(t) > 0$ for all $t \in [0,1)$.

Setting $\lambda_n := \log\sigma((-\log p_n)/\eta) < 0$ (since $\sigma < 1$ on $(0,1)$), we have $c_n'(t) = c_n(t)\,\lambda_n$. Differentiating $\Phi = A^2/B$ with $A = \sum c_n a_n$ and $B = \sum c_n^2 a_n$:

$$\Phi'(t) = \frac{2A}{B^2}\Big[A'\,B - A\,B'/2\Big]\ \propto\ \sum_{n,m} c_n\,c_m^2\,a_n\,a_m\,(\lambda_n - \lambda_m).$$

Antisymmetrizing:

$$\Phi'(t)\ \propto\ \sum_{n<m}(\lambda_n - \lambda_m)\,a_n\,a_m\,c_n\,c_m\,(c_m - c_n). \tag{7}$$

We check the sign of each term. Since $\eta > 1/2$, the function $p \mapsto p\,[\sigma((-\log p)/\eta)]^t$ is increasing for all $t \in [0,1]$.[5] Therefore $c_n(t)$ preserves the ordering of the p_n. Since λ_n is strictly decreasing in p_n (as $\sigma((-\log p)/\eta)$ is decreasing), for each pair $n < m$ with $p_n \neq p_m$:

- $p_n > p_m \implies c_n > c_m$ and $\lambda_n < \lambda_m$, so $(\lambda_n - \lambda_m)(c_m - c_n) > 0$.
- $p_n < p_m \implies c_n < c_m$ and $\lambda_n > \lambda_m$, so $(\lambda_n - \lambda_m)(c_m - c_n) > 0$.

Each factor $a_n\,a_m\,c_n\,c_m$ is positive, so every non-degenerate term in (7) is strictly positive. Since not all p_n are equal, $\Phi'(t) > 0$ for all $t \in [0,1)$, giving $\Phi(1) > \Phi(0)$. Both cosines are positive, so $\cos(\mathbb{E}[g_{\mathrm{DG}}], g_{\mathrm{CE}}) > \cos(\mathbb{E}[g_{\mathrm{PG}}], g_{\mathrm{CE}})$. □

On the $\eta > 1/2$ condition. The condition $\eta > 1/2$ ensures that the DG weight function $p \mapsto p\,\sigma((-\log p)/\eta)$ is monotonically increasing, which the path argument requires to preserve the ordering of weights. Since we use $\eta = 1$ throughout, this condition is always satisfied. For $\eta \leq 1/2$ the weight function can be non-monotone near $p = 1$, and Proposition 2 may fail; exploring this regime is an interesting direction for future work.

[5] Write $p\,\sigma((-\log p)/\eta)^t = p/(1 + p^{1/\eta})^t$. Its derivative has the sign of $1 + (1 - t/\eta)\,p^{1/\eta}$, which at worst ($t$=1, $p\to$1) equals $2 - 1/\eta > 0$.

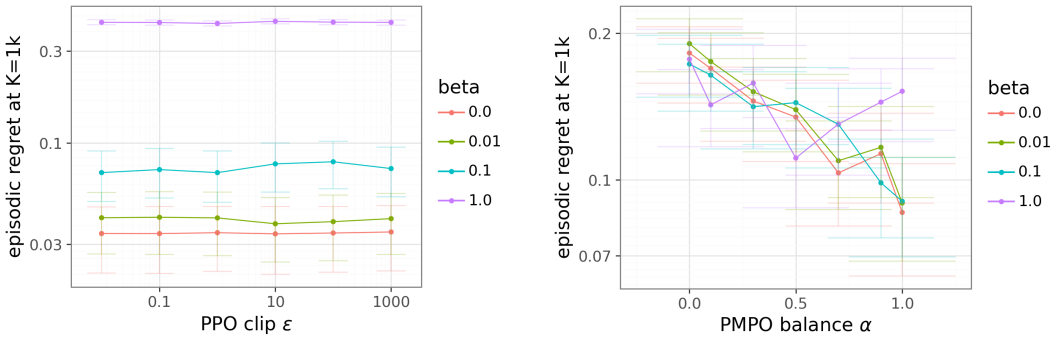
D Transformer Sequence Modeling

We provide experimental details and robustness checks for the Token Reversal experiments in Section 5.

Experimental Setup. The agent is a decoder-only Transformer with causal attention, model dimension $d_{\text{model}} = 64$, 2 layers, and 2 attention heads. We use a distributed actor-learner architecture with 10 parallel actors, each collecting batches of 10 trajectories, yielding 100 episodes per gradient step. All agents are trained with Adam using a standard empirical mean baseline for variance reduction. Unless otherwise noted, the training budget is $K = 1{,}000$ episodes.

D.1 Baseline Tuning

To ensure fair comparison, we tuned hyperparameters for PPO and PMPO. For PPO, we swept the clipping parameter ϵ and KL penalty β; Figure 20a shows the algorithm is insensitive to ϵ, and the KL penalty does not help on this deterministic task. For PMPO, we swept the weighting threshold α and KL penalty β; Figure 20b shows boundary values ($\alpha \approx 0$ or 1) outperform intermediate values. Even the best-tuned PPO (Regret ≈ 0.03) and PMPO (Regret ≈ 0.08) lag behind DG (Regret < 0.01).



(a) Tuning PPO (ϵ, β). (b) Tuning PMPO (α, β).

Figure 20: Hyperparameter tuning for baselines. Neither PPO nor PMPO matches DG despite extensive sweeps.

D.2 Robustness

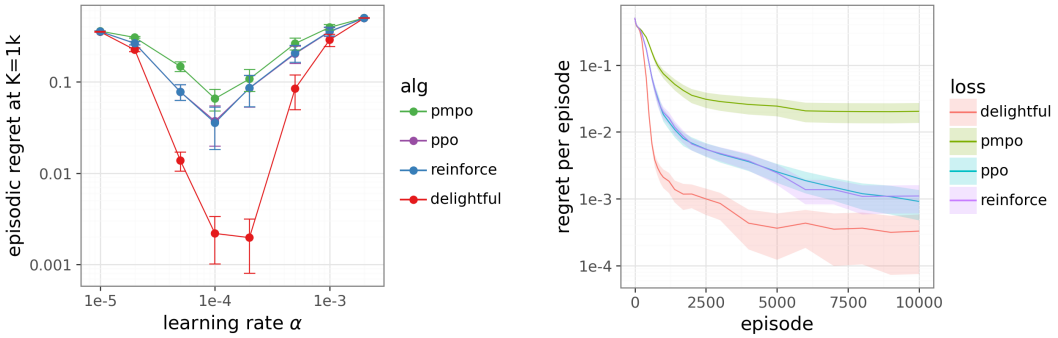
We validate robustness on the default task ($H = 10$, $M = 2$). Figure 21a shows regret at $K = 1{,}000$ across learning rates; DG consistently outperforms baselines over a wide effective range. Figure 21b extends training to $K = 10{,}000$ episodes; baselines do not catch up, confirming the advantage is not due to faster early learning alone.

D.3 Task Variations

To ensure findings generalize beyond reversal, we test four target logics:

- **Copy:** $y_i = x_i$
- **Flip:** $y_i = 1 - x_i$
- **Reverse Copy:** $y_i = x_{H-i+1}$ (the default)
- **Reverse Flip:** reverse and negate

We also vary reward structure. **Bag-of-Tokens** gives credit for each correct token regardless of position; **Sequential** gives credit only up to the first mistake, making credit assignment harder. Figure 22 illustrates the difference.

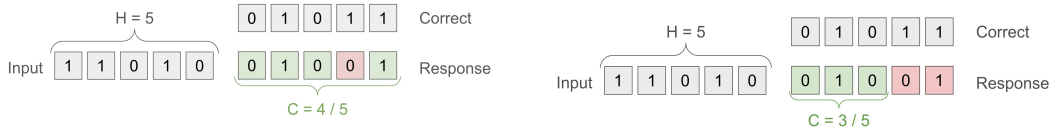


(a) Learning rate sensitivity.

(b) Extended training ($K = 10$k).

Figure 21: DG's advantage is robust to learning rate **(a)** and persists asymptotically **(b)**.

Figure 23 shows learning curves for all eight configurations. DG achieves the lowest regret in every setting. The gap is often larger in the harder Sequential settings, where baselines struggle with truncated reward streams.



(a) Bag-of-Tokens (dense).

(b) Sequential (strict).

Figure 22: Reward structures. Bag-of-Tokens credits all correct tokens; Sequential stops at the first error.

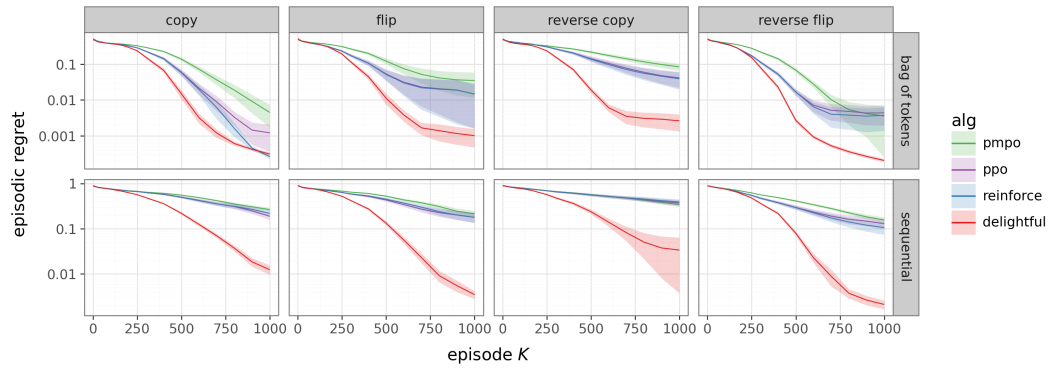


Figure 23: Learning curves across 8 task variants. Top: Bag-of-Tokens. Bottom: Sequential. DG (red) achieves the lowest error in all configurations.

D.4 Multiplicative vs. Additive Delight

We compare DG to UCB-style additive mixtures:

$$\chi_\alpha^{\mathrm{UCB}} = (1-\alpha)U + \alpha\ell. \tag{8}$$

Figure 24 sweeps $\alpha \in [0, 1.25]$ and $\eta \in \{0.2, 0.5, 1, 2, 5\}$. Additive mixtures can outperform REINFORCE (black dashed) but never approach DG (red dashed). The best additive achieves regret ≈ 0.1 versus DG's ≈ 0.04. Additive bonuses treat surprisal symmetrically, encouraging the policy to

chase high-surprisal actions even when advantage is negative. DG's multiplicative gate suppresses such blunders, filtering them from the update.

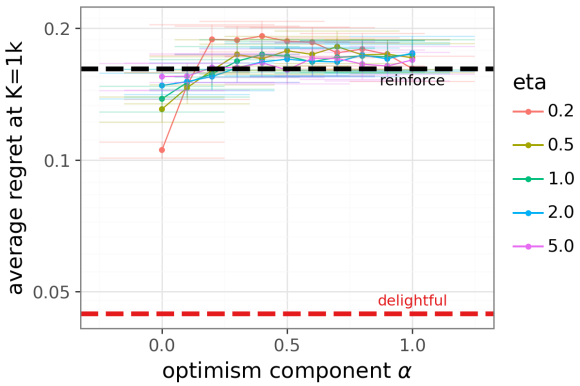


Figure 24: Additive mixtures (colored lines) vs. multiplicative DG (red dashed). No additive configuration matches DG.

E Detailed Control Suite Results

We provide experimental details and extended results for Section 6.

E.1 Experimental Setup

Architecture. All methods use the same actor-critic architecture: a 2-layer MLP with 256 hidden units for both actor and critic. The actor outputs mean and diagonal covariance of a Gaussian policy; the critic estimates state-action values. We use the Retrace algorithm (Munos et al., 2016) for off-policy correction.

Optimization. All methods use Adam with learning rate 3×10^{-4}. We train for 10 million environment steps with 4 parallel actors, a replay buffer of size 2×10^6, and batch size 256. Target networks are updated every 100 learner steps.

DG-specific details. For continuous actions, surprisal $\ell = -\log \pi(a \mid s)$ can take large values. We clip surprisal to $[-10, 10]$ before computing delight. We also normalize rewards using an exponential moving average (decay 0.999) for critic stability; this normalization is applied to all methods. The gate temperature is $\eta = 1$, consistent with all other experiments.

Algorithm 2 provides the continuous-action variant used in all control experiments.

Algorithm 2 Delightful Policy Gradient (continuous actions)

Require: Batch $\mathcal{B}$, Gaussian policy $\pi_\theta(\cdot \mid s)$, temperature η=1, clip bound C=10

$\Delta\theta \leftarrow 0$
for $t \in \mathcal{B}$ **do**
  $\ell_t \leftarrow \text{clip}(-\log \pi_\theta(A_t \mid \mathcal{H}_t),\ -C,\ C)$
  $\chi_t \leftarrow U_t \cdot \ell_t$ ▷ Delight
  $w_t \leftarrow \sigma(\chi_t/\eta)$ ▷ Gate
  $\Delta\theta \leftarrow \Delta\theta + w_t\, U_t\, \nabla_\theta \log \pi_\theta(A_t \mid \mathcal{H}_t)$
return $\Delta\theta$

Baselines. We compare against three baselines within our codebase:

- **PG:** Standard policy gradient with Retrace critic and no gating.
- **PPO:** Clipped surrogate objective with $\epsilon = 0.2$.
- **MPO:** Softmax-weighted updates with temperature $\eta = 1.0$.

All baselines use identical architecture, optimizer, and replay settings.

E.2 Per-Environment Learning Curves

Figure 25 displays individual learning curves for all 28 Control Suite environments. DG (red) consistently matches or exceeds baseline performance, with notable improvements on exploration-heavy tasks such as `acrobot:swingup`, `finger:turn_hard`, and `humanoid:run`.

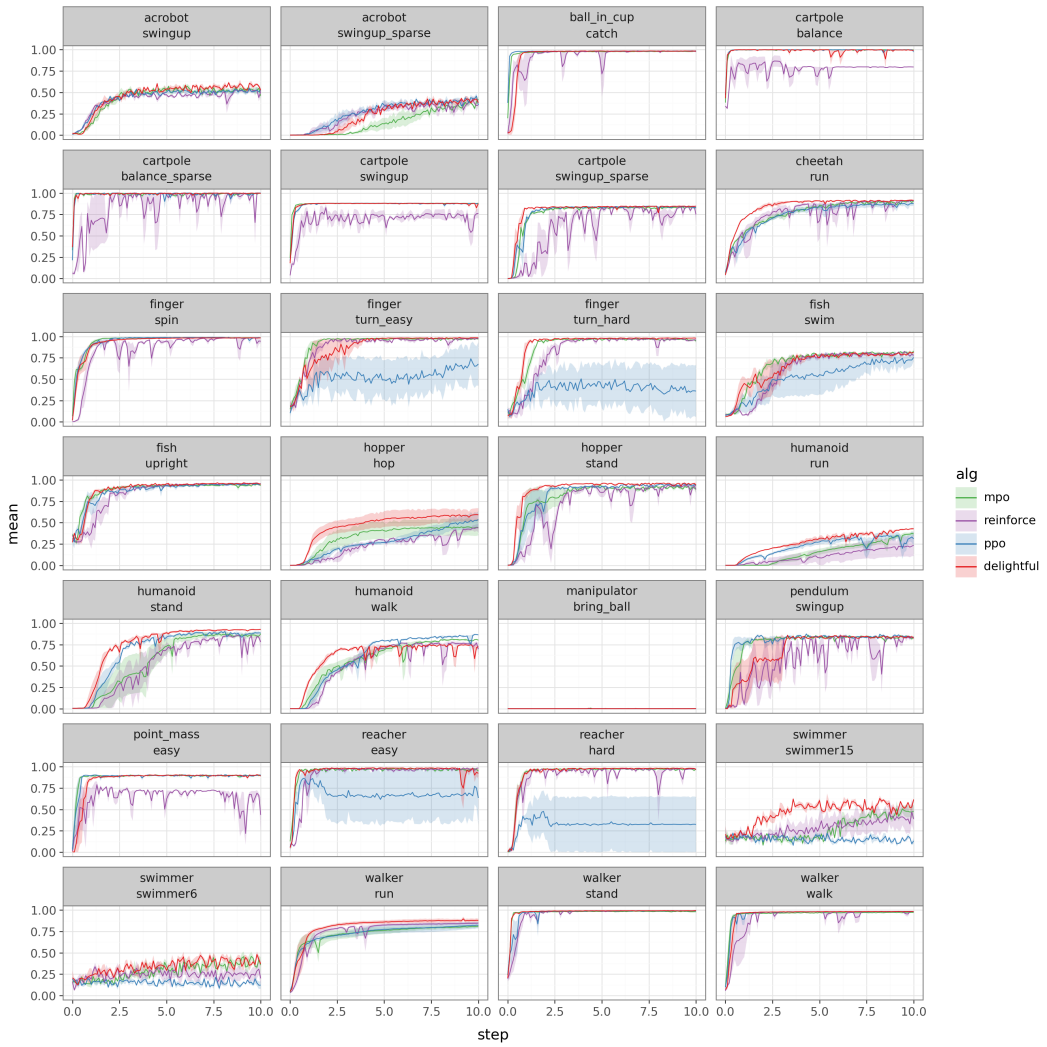
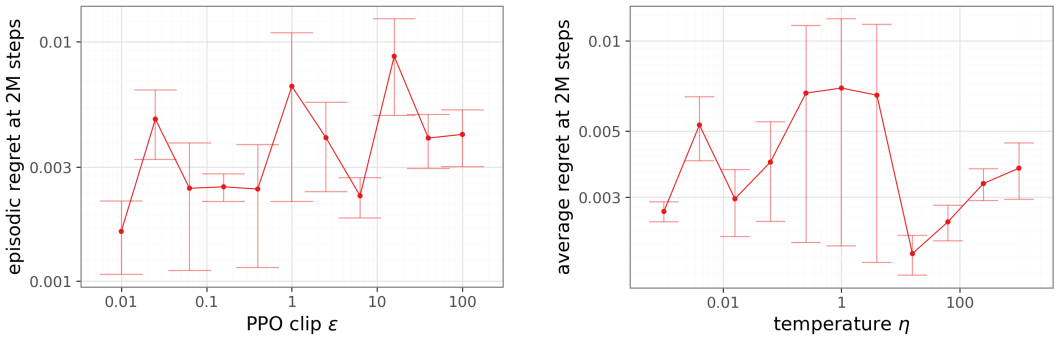


Figure 25: Learning curves for 28 Control Suite environments. DG (red) consistently matches or exceeds the hedonic baseline (purple), PPO (blue), and MPO (green).

E.3 Baseline Hyperparameter Sensitivity

To ensure fair comparison, we verified that default hyperparameters are reasonable for PPO and MPO. Figure 26 shows sensitivity sweeps on `cartpole:swingup`. For PPO, we sweep clip parameter $\epsilon \in [0.01, 100]$; for MPO, we sweep temperature $\eta \in [0.001, 1000]$. Neither sweep reveals a clear

optimum that substantially outperforms the defaults ($\epsilon = 0.2$, $\eta = 1.0$). We therefore use these standard values throughout.



(a) PPO clip ϵ.

(b) MPO temperature η.

Figure 26: Hyperparameter sensitivity on `cartpole:swingup`. Default values (dashed) perform comparably to alternatives.

E.4 Comparison to Tuned External Implementations

We also benchmark DG against highly-optimized external implementations: **MPO (Tuned)** with adaptive temperature and **SAC (Tuned)** with automatic entropy adjustment. These use separate codebases with extensive per-task tuning. This comparison is especially stringent because regret is defined relative to the best performance achieved by *any* method: $\text{Regret}_k = R_{\text{best}} - R_k$.

Figure 27 shows aggregate regret over training. Even against these optimized baselines, DG achieves the lowest average regret. Figure 28 breaks down final performance by environment, confirming that DG's advantage is broad-based rather than driven by outliers. DG achieves low regret across domains ranging from `cartpole` to `humanoid` and `dog`.

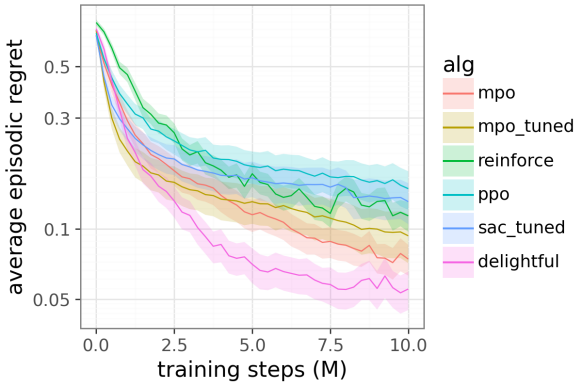


Figure 27: Aggregate regret against tuned SOTA baselines. DG (pink) outperforms tuned MPO (gold) and SAC (blue).

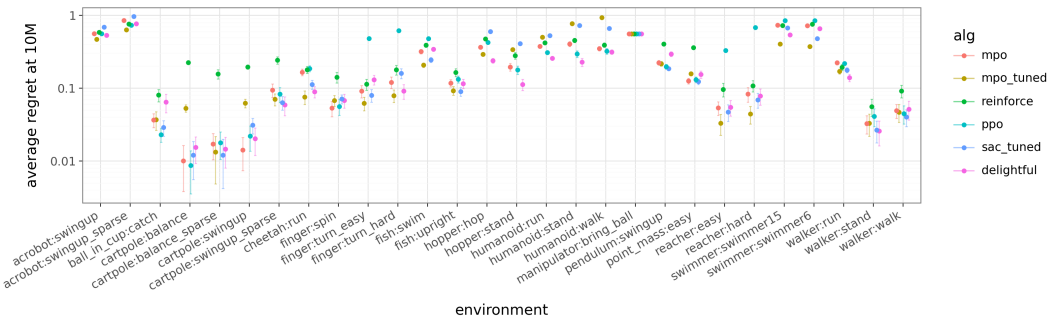


Figure 28: Per-environment regret at 10M steps. DG (pink) consistently achieves low regret across the suite.