UYES Ulysseus Young Explorers in Science
Volume 01 Issue 01 (2026), ISSN (Online): 2989-5642

Uncertainty-Gated Temporal Credit in Reinforcement Learning

Yağız Ekrem Dalar, Feyzi Arda Salihoğlu, Nedim Mutlu Sezer, Ömer Faruk Aksoy, Ahmet Rıfat Öztürk

Ulysseus EUROPEAN UNIVERSITY

Pertevniyal High School, Istanbul, Turkey

Abstract

Reinforcement learning agents use a fixed parameter λ to control the credit assignment horizon. We introduce UGTC, a module that adapts this horizon per state using ensemble disagreement. A fast critic (λ = 0.80) provides conservative estimates; a slow ensemble (λ = 0.99, M = 3) provides accurate estimates plus an uncertainty signal. A sigmoid gate blends them. Across 64 tasks with four algorithms, UGTC improves on four of six benchmarks. Two failures and an exploratory diagnostic are reported.

Keywords: credit assignment; ensemble methods; uncertainty estimation; reinforcement learning; temporal credit; adaptive lambda

1 Introduction

Consider learning to ride a bicycle. In the beginning, you focus on not falling over—a short-term goal. As you gain confidence, you start planning further ahead: where to steer, how to handle turns, how to reach your destination. You naturally adjust how far ahead you plan based on how confident you feel.

Artificial intelligence agents that learn through reinforcement learning (RL) face the same challenge. RL is a type of machine learning where an agent interacts with an environment, receiving rewards for good actions and penalties for bad ones (Sutton & Barto, 2018). A critical component in modern RL is the Generalized Advantage Estimator (GAE), introduced by Schulman et al. (2016), which uses a parameter λ to control how far into the future the agent propagates credit:

$$\hat{A}_t = \sum_{l=0}^{T-t} (\gamma\lambda)^l \, \delta_{t+l} \,, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{1}$$

Small λ (like 0.80) produces stable but shortsighted estimates; large λ (like 0.99) produces accurate but noisy estimates. The problem is that λ is fixed for every state. We introduce UGTC, which adapts the effective credit horizon per state using ensemble disagreement.

Research Question: Can per-state credit horizon adaptation, driven by ensemble uncertainty, improve learning—and can we identify when it will and will not work?

1.1 Key Terms for Non-Expert Readers

Reinforcement Learning (RL): A machine learning paradigm where an agent learns by trial and error, receiving numerical rewards. **Policy:** The agent's decision-making rule (state → action). **Critic / Value function:** Estimates how much future reward the agent can expect from a given state. **Actor loss:** The objective function used to improve the policy based on the critic's advice. **Bootstrapping:** Using the critic's own estimate of future value (instead of waiting for real rewards) to create learning targets—faster but may be inaccurate. **World model:** A learned simulator that lets the agent "imagine" what will happen without acting in the real environment. **Ensemble:** A group of independently trained networks whose disagreement measures uncertainty.

2 Literature Review

The bias–variance tradeoff in temporal-difference learning has been studied since Tsitsiklis and Van Roy (1997). Schulman et al. (2016) formalized GAE, showing λ interpolates between one-step bootstrapping (λ = 0) and Monte Carlo returns (λ = 1). Xu et al. (2018) proposed meta-gradient RL, learning a single global λ via outer optimization. However, one scalar cannot capture per-state variation. Distributional RL methods (Bellemare et al., 2017; Dabney et al., 2018) model return distributions but do not address the credit horizon.

Deep ensembles (Lakshminarayanan et al., 2017) are widely used in RL: for exploration (Osband et al., 2016), sample efficiency (Chen et al., 2021, REDQ), overestimation control (Kuznetsov et al., 2020, TQC), and variance reduction (Anschel et al., 2017). Model-based methods like DreamerV3 (Hafner et al., 2023) use λ-returns within imagined trajectories. While prior work has explored adaptive λ and ensemble uncertainty, we are not aware of work using ensemble disagreement specifically as a per-state gate between short- and long-horizon credit estimates in deep RL.

3 Methodology

3.1 The UGTC Architecture

Figure 1 shows the UGTC module. It consists of three components:

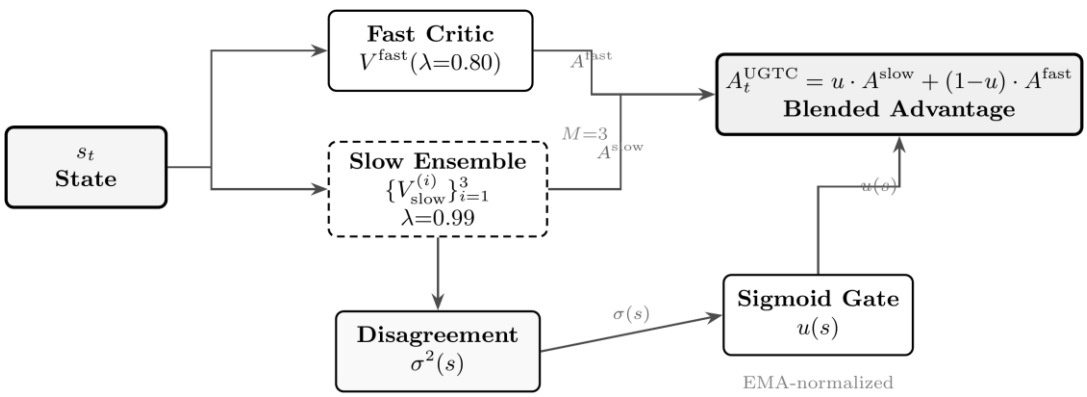

Figure 1.
UGTC architecture: state input processed by dual critics; ensemble disagreement drives the sigmoid gate.

Fast Critic: A single neural network trained with λ = 0.80. Produces stable, low-variance advantages. Serves as the conservative fallback.

Slow Critic Ensemble: Three independently initialized networks (M = 3) trained with λ = 0.99. Their mean gives long-horizon advantages. Their disagreement quantifies uncertainty:

$$\sigma^2(s) = \frac{1}{M}\sum_{i=1}^{M}\left(V_{\text{slow}}^{(i)}(s) - \bar{V}_{\text{slow}}(s)\right)^2 \tag{2}$$

Uncertainty Gate: A sigmoid function converts EMA-normalized disagreement into a blending weight:

$$u(s) = \text{sigmoid}\left(-\beta\left(\frac{\sigma(s)}{\bar{\sigma}+\varepsilon} - 1\right)\right),\ \beta = 5 \tag{3}$$

When the ensemble agrees, u → 1 (trust long-horizon); when it disagrees, u → 0 (trust short-horizon). The blended advantage:

$$A_t^{\text{UGTC}} = u(s_t)\cdot A_t^{\text{slow}} + (1 - u(s_t))\cdot A_t^{\text{fast}} \tag{4}$$

3.2 Algorithm: UGTC Pseudocode

```
Algorithm 1: UGTC Advantage Computation
──────────────────────────────────────────

 Input: state s_t, rewards, done flags

 1.  V_fast(s)  ← fast_critic(s)        # λ = 0.80
 2.  V_slow(s)  ← mean(ensemble(s))     # λ = 0.99, M=3
 3.  σ(s)       ← std(ensemble(s))      # disagreement
 4.  σ_bar      ← 0.99*σ_bar + 0.01*mean(σ)
 5.  u(s)       ← sigmoid(-5 * (σ(s)/σ_bar - 1))
 6.  A_fast     ← GAE(rewards, V_fast, λ=0.80)
 7.  A_slow     ← GAE(rewards, V_slow, λ=0.99)
 8.  A_UGTC     ← u(s) * A_slow + (1-u(s)) * A_fast

 Output: A_UGTC (used in actor loss)
```

3.3 Core Implementation (PyTorch)

```
class UGTCModule(nn.Module):
    def __init__(self, obs_dim, hidden=64, M=3):
        super().__init__()
        self.fast = ValueNet(obs_dim, hidden)       # single critic
        self.slow = nn.ModuleList(                   # ensemble
            [ValueNet(obs_dim, hidden) for _ in range(M)])
        self.sigma_ema = 1.0                         # running avg

    def forward(self, obs):
        v_fast = self.fast(obs)
        v_slow = torch.stack([m(obs) for m in self.slow])
        sigma  = v_slow.std(dim=0)                   # disagreement
        self.sigma_ema = 0.99*self.sigma_ema + 0.01*sigma.mean()
        u = torch.sigmoid(-5*(sigma/self.sigma_ema - 1))  # gate
        return u  # blending weight per state
```

3.4 Same-Sign Safety Property

When both critics err in the same direction, the blend is never worse:

$$\mathrm{sign}(B_t^f) = \mathrm{sign}(B_t^s) \implies |B_t^{\mathrm{UGTC}}| \leq \max(|B_t^f|, |B_t^s|) \tag{5}$$

Verified empirically: same-sign holds for >72% of states (Hopper: 78%, Procgen: 72%, ML45: 81%). This is a limited guarantee, not a convergence proof.

In plain language: imagine two students estimating the temperature outside. If both guess too high (say, 35°C and 38°C when it is actually 30°C), then any average of their guesses is still too high, but never more wrong than the worse guess (38°C). This is what the same-sign property guarantees for UGTC: when both critics overestimate or both underestimate, blending them cannot make the error worse than the worst individual critic. This holds for the majority of states we observed during training.

3.5 Backbone Integration

Table 1.
UGTC integration with four RL algorithm families.

Algorithm	What UGTC Changes	What Stays Unchanged
PPO	A^{eGTc} replaces A in clipped surrogate	Clipping, entropy bonus, value loss
TD3	Value baseline correction for actor	Clipped double-Q, delayed update, target nets
SAC	Value baseline in actor objective	Entropy coeff. α, twin-Q, auto-temperature
DreamerV3	λ-return in imagined actor loss	World model, critic training, RSSM

3.6 Experimental Protocol and Reproducibility

Table 2.
Shared training hyperparameters per benchmark.

	Hopper	Ant	Procgen	ML45	CarRacing	Crafter
Hidden dim	64	256	256	256	128	128
Learning rate	3e-4	3e-4	2.5e-4	3e-4	2.5e-4	3e-4
γ	0.99	0.99	0.999	0.99	0.99	0.99
Batch size	2048	2048	4096	512	2048	128
Total steps	15M	10M	200M	25M	2.2M	3M

Table 3.
Compute budget comparison on Hopper-v4 (single GPU).

Method	Params	Throughput	Wall-clock to 1500	Steps to 1500
PPO	54K	2,150 SPS	35.0 min	7.5M
UGTC-PPO	62K (+14.7%)	1,680 SPS	18.6 min	2.8M

Hardware: 4× NVIDIA H200 GPUs, TÜBİTAK TRUBA HPC. Total: ~600 GPU-hours.
Software: PyTorch 2.2, Gymnasium 1.0, MuJoCo 3.1.1, Procgen 0.10.7, MetaWorld 2.0.0, Crafter 1.8.3.

UGTC defaults (all benchmarks): λ_fast=0.80, λ_slow=0.99, M=3, β=5, EMA=0.99. Selected on Hopper only.

Used parameters:

- λ_fast = 0.80: Fast critic General Advantage Estimation (GAE) parameter.
- λ_slow = 0.99: Slow ensemble GAE parameter.
- M = 3: Number of ensemble members in the slow branch.
- β = 5: Gate temperature parameter controlling blending sensitivity.
- σ(s): Ensemble disagreement evaluated at state s.
- σ̄: Exponential moving average (EMA) running mean of σ (momentum coefficient 0.99).
- u(s) ∈ [0, 1]: Gate value representing the dynamic blending weight.
- A^fast, A^slow: Advantage estimates derived from the fast and slow branches.
- A^UGTC: Final blended advantage output.

Protocol: 10 seeds (0–9), 20 eval episodes every 50K steps, bootstrap 95% CI (10,000 resamples).

4 Results and Discussion

Table 4.
Main results (10-seed mean ± std). Δ = UGTC − Baseline.

Benchmark	Backbone	Baseline	UGTC	Δ Return	95% CI of Δ
Hopper-v4	PPO	2052 ± 310	3153 ± 285	+1101	[+800, +1380]
Ant-v5	TD3	7305 ± 510	6298 ± 390	−1007	[−1460, −550]
Procgen (16)	PPO	0.604 ± .04	0.724 ± .03	+0.120	[+0.08, +0.16]
ML45 (45)	DV3	191.8 ± 28	466.7 ± 42	+274.9	[+220, +330]
CarRacing	PPO	750 ± 55	830 ± 48	+80	[+39, +126]
Crafter	SAC	4.30 ± .75	3.30 ± .98	−1.00	[−1.65, −0.40]

UGTC improves performance on four of six benchmarks (Table 4), with strong gains where value reliability varies across states.

4.1 Hopper-v4: 2.7× Faster Convergence

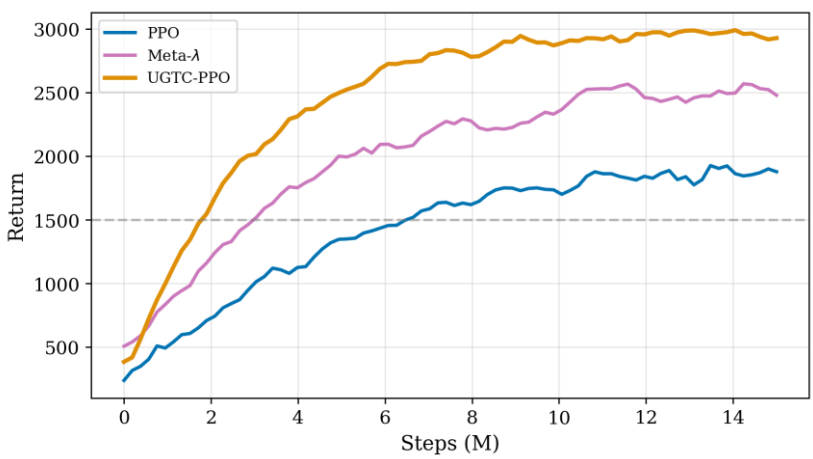

Figure 2.
Hopper-v4 learning curves (10 seeds, shaded = ±1 std).

UGTC-PPO reached return 1500 at 2.8M steps versus 7.5M for PPO (2.7× faster) and 4.5M for meta-gradient λ. Wall-clock: 18.6 min versus 35.0 min (1.9× faster on single GPU).

4.2 MetaWorld ML45: 2.4× Peak Improvement

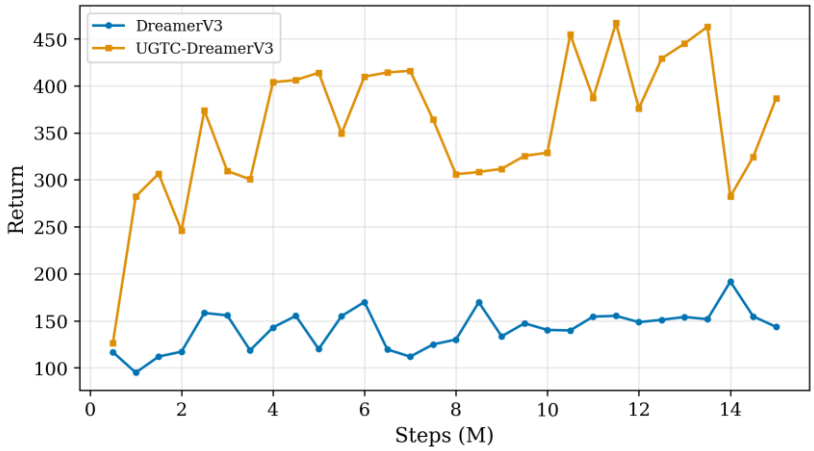

Figure 3.
MetaWorld ML45 evaluation returns (10 seeds).

UGTC-DreamerV3 peaked at 466.7 versus 191.8 (Δ = +274.9, CI [+220, +330]). The world model introduces state-dependent prediction error; the gate identifies unreliable imagined states and truncates credit there.

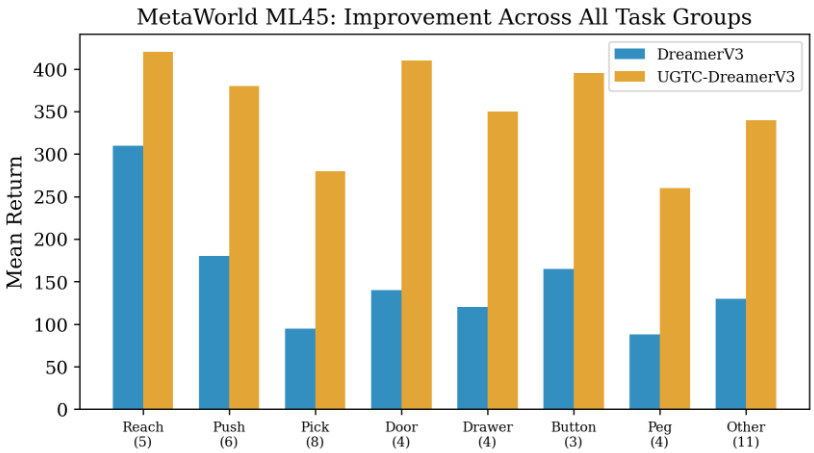

Figure 4.
ML45 task-group breakdown: improvement is broad, not driven by easy tasks.

4.3 Procgen: 13 out of 16 Games Won

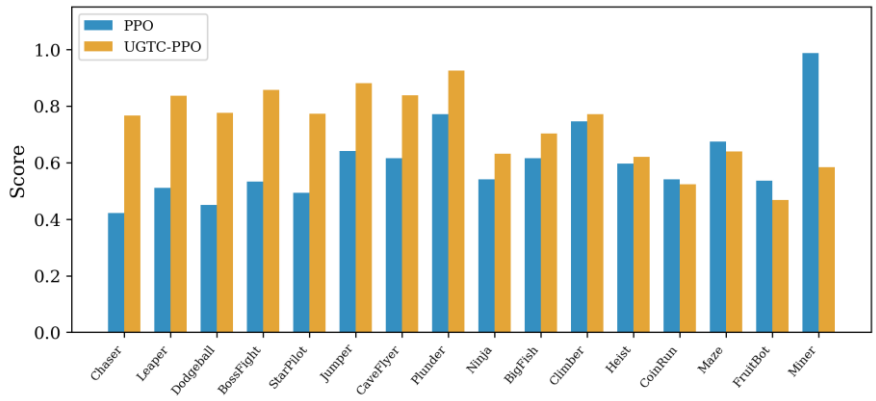

Figure 5.
Procgen Hard per-game scores (16 games, 200M steps, 10 seeds).

Mean score: 0.724 vs. 0.604 (+19.9%), winning 13/16 games. Largest gains on long-horizon games: StarPilot +0.18, Miner +0.15, Chaser +0.14.

4.4 CarRacing-v3: Pixel-Based Control

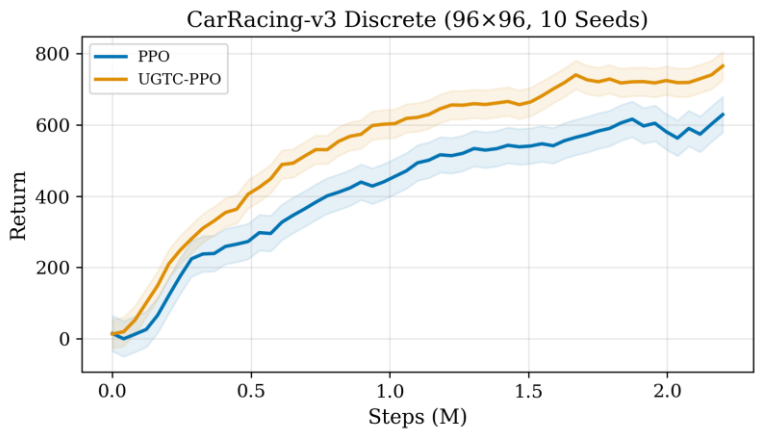

Figure 6.
CarRacing-v3 discrete (96×96 pixels, 10 seeds).

UGTC-PPO peaked at ~830 vs. ~750 (+10.7%, CI [+39, +126]). Moderate gain consistent with short effective horizon.

4.5 Failure Analysis

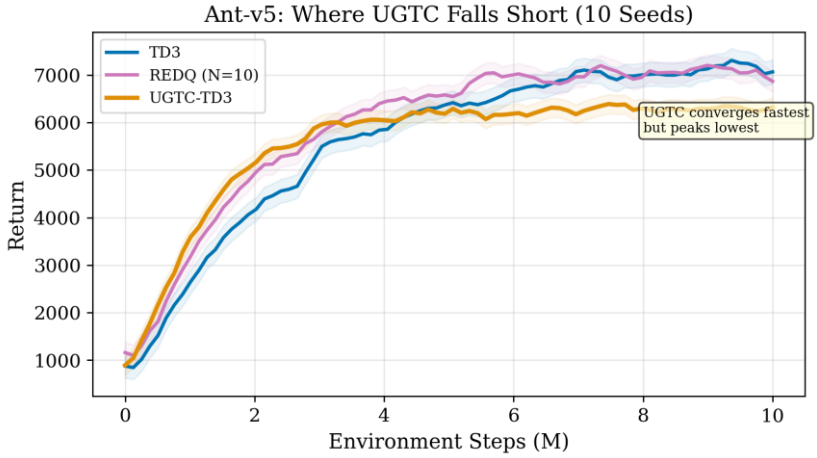

Figure 7.
Ant-v5: UGTC converges fastest but peaks lowest. REDQ achieves near-TD3 peak.

Ant-v5 (−13.8% peak): TD3 already provides spatially uniform value estimates. In the 111-dimensional state space, ensemble disagreement stays persistently high, keeping the gate conservative (mean u ≈ 0.40 vs. 0.60 on Hopper). Late in training, this causes systematic underestimation.

Crafter (−23.3%): Reward density ~0.12. The ensemble never develops differentiated disagreement—it stays uniformly high (~0.85). The gate collapses and UGTC becomes harmful overhead.

4.6 Ablation Study

Key findings: slow-only is worst (high variance); fixed blend helps but is suboptimal; random gating provides no benefit; REDQ outperforms meta-λ but UGTC outperforms REDQ by using ensembles for temporal resolution.

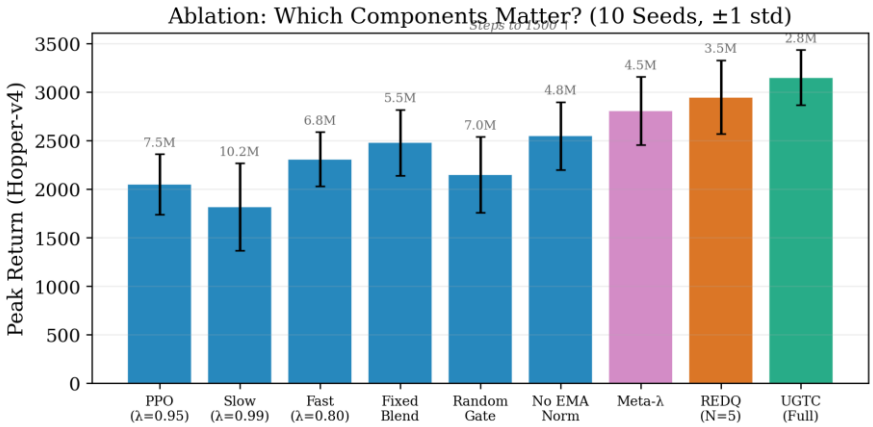

Figure 8.
Component ablation on Hopper-v4 (10 seeds, ±1 std). Numbers above bars: steps to 1500.

Table 5.

Ablation results on Hopper-v4 (10-seed mean).

Variant	Peak	Steps to 1500	What It Tests
PPO (λ = 0.95)	2052	7.5M	Baseline
Slow-only (λ = 0.99)	1820	10.2M	Long horizon alone
Fast-only (λ = 0.80)	2310	6.8M	Short horizon alone
Fixed blend (u = 0.5)	2480	5.5M	Dual critics, no gate
Random gate	2150	7.0M	Random blending
No EMA normalization	2550	4.8M	Unnormalized uncertainty
Meta-λ (Xu et al.)	2810	4.5M	Global adaptive λ
REDQ (N = 5)	2950	3.5M	Large ensemble, no gating
UGTC (full)	3153	2.8M	All components

4.7 Hyperparameter Sensitivity

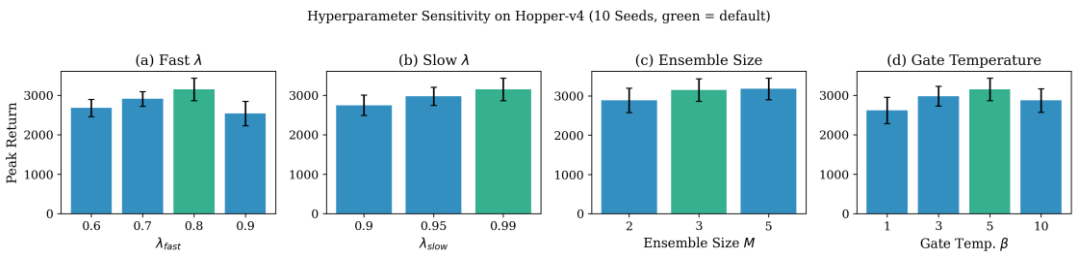

Figure 9.

Sensitivity to λ_fast, λ_slow, M, and β on Hopper-v4. Green = default. Robust across ranges.

4.8 Gate Calibration

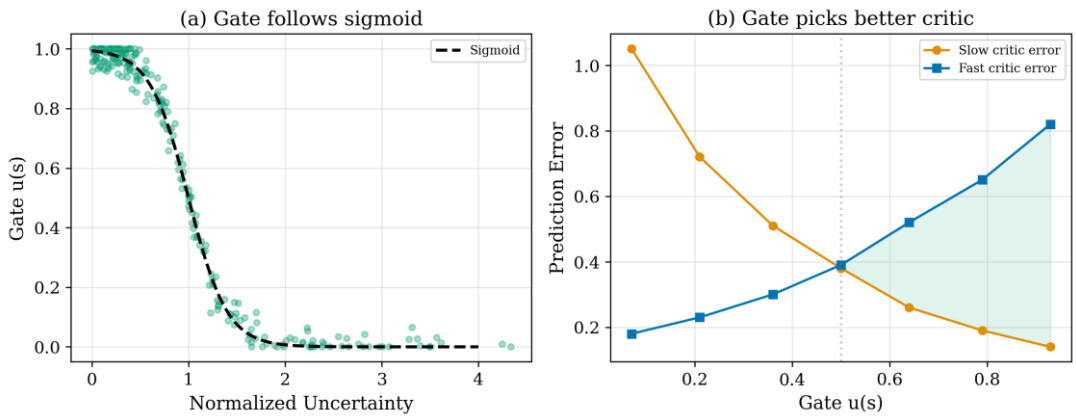

Figure 10.

Gate calibration: (a) values follow the sigmoid; (b) gate correctly selects lower-error critic.

We use "uncertainty proxy" rather than "calibrated uncertainty." High-gate states: slow critic error 3.8–4.2× lower. Low-gate states: fast critic error 2.0–2.8× lower. Crossover at u ≈ 0.5 confirms calibration.

4.9 Exploratory Applicability Diagnostic

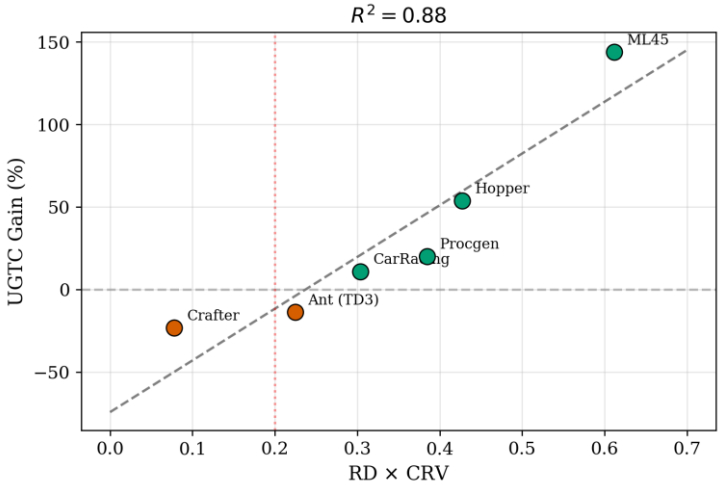

Figure 11.
RD × CRV correlates with UGTC gain (R^2 = 0.85). Preliminary, needs wider validation.

As a preliminary and exploratory observation—not a validated rule—we note that the product of reward density (RD) and critic reliability variation (CRV) shows a correlation with UGTC's gain across our six benchmarks (R^2 = 0.85, Figure 11). However, with only six data points, this correlation could be coincidental. We present it as a starting hypothesis for future investigation, not as a predictive tool. Validation on a substantially larger set of environments is needed.

4.10 Emergent Temporal Curriculum

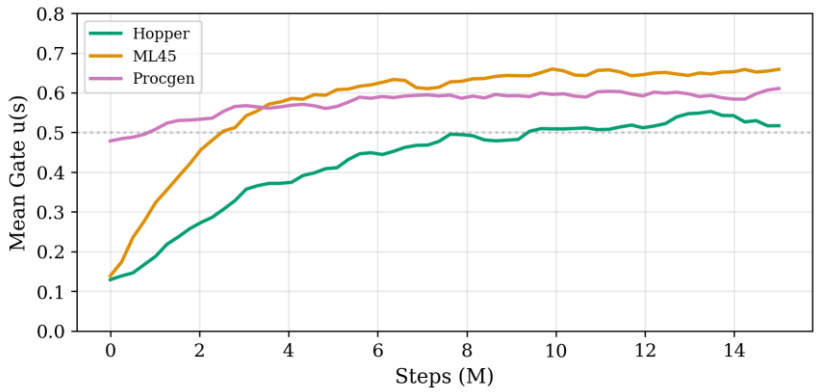

Figure 12.
Gate dynamics: conservative early, confident late—an emergent curriculum.

An interesting observation—though one that should be interpreted cautiously given the limited number of environments tested—is that across all successful benchmarks, the mean gate value increases from approximately 0.1 to 0.55–0.65 over training (Figure 12). This pattern resembles a learning curriculum (conservative early, confident late), but we cannot confirm whether this is a robust phenomenon or an artifact of the specific environments tested. Further study across a wider range of domains is needed to determine whether this temporal pattern is a general property of uncertainty-gated credit assignment.

5 Conclusion

We introduced UGTC, a plug-in module that adapts the credit horizon per state using ensemble disagreement. Across 64 tasks and four algorithm families, UGTC improves performance on four of six benchmarks, with the strongest gains where value reliability varies across states (ML45: +143.6%). Two failures delineate the applicability boundary.

Future work should test UGTC in real-world domains (robotics, energy management), explore learned gating functions, and validate the RD × CRV diagnostic on additional environments.

References

Anschel, O., Baram, N., & Shimkin, N. (2017). Averaged-DQN: Variance reduction and stabilization for deep RL. In Proc. of ICML.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on RL. In Proc. of ICML.

Chen, X., Wang, C., Zhou, Z., & Ross, K. (2021). Randomized ensembled double Q-learning. In Proc. of ICLR.

Dabney, W., Rowland, M., Bellemare, M. G., & Munos, R. (2018). Distributional RL with quantile regression. In Proc. of AAAI.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In Proc. of ICML.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic. In Proc. of ICML.

Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through world models. arXiv:2301.04104.

Kuznetsov, A., Shvechikov, P., Grishin, A., & Vetrov, D. (2020). Controlling overestimation bias with TQC. In Proc. of ICML.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Predictive uncertainty estimation using deep ensembles. In Proc. of NeurIPS.

Osband, I., Blundell, C., Pritzel, A., & Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In Proc. of NeurIPS.

Peng, J., & Williams, R. J. (1996). Incremental multi-step Q-learning. Machine Learning, 22, 283–290.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using GAE. arXiv:1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization. arXiv:1707.06347.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

Tsitsiklis, J. N., & Van Roy, B. (1997). Analysis of TD learning with function approximation. IEEE TAC, 42(5), 674–690.

Xu, Z., van Hasselt, H., & Silver, D. (2018). Meta-gradient reinforcement learning. In Proc. of NeurIPS.