

PID-Gated Continual Prompt Adaptation Under Noisy Feedback

Zhentao Xu

Independent Researcher
Sunnyvale, CA, USA
frankxu@umich.edu

Abstract

Deploying LLM agents in non-stationary environments requires continually revising their instructions as the task drifts, yet rewriting after every reported mistake can turn noisy feedback into persistent errors. This raises an often-overlooked scheduling question in prompt adaptation: *when* should feedback trigger a lasting instruction update, and at what editor-call cost? Taking inspiration from proportional-integral-derivative (PID) control, we introduce a family of lightweight gates that react to the current error, accumulate evidence over time, and roll back deterioration after an edit. We benchmark these gates against both simple baselines and advanced prompt optimizers (TextGrad and GEPA), over a diverse suite of drifting tasks, LLM parameter sizes, feedback-noise levels, and random seeds. PID-gated continual prompt adaptation yields a more favorable Pareto trade-off between task performance (high accuracy, low error) and editor-call budget than the baselines considered, pointing to control-theoretic scheduling as a simple, general tool for robust prompt adaptation under unreliable feedback. We open source our code at <https://github.com/zhentao-xu/pid-gated-agent-learning>.

1 Introduction

Consider an agent guessing a hidden integer under noisy high/low feedback: it holds a belief about the plausible range $[L, U]$ that must tighten as evidence accumulates, yet a single mislabeled “higher” reply can widen the wrong bound and lock in errors for many rounds. This is the general dilemma of adapting instructions in a changing environment: revising after every reported mistake can turn a useful rule into a persistent source of errors, while waiting leaves the agent following an outdated rule. The challenge is deciding when a mistake warrants a lasting change.

Language models offer several ways to learn from feedback: revising an answer [11], retaining verbal experience across attempts [18], or optimizing prompts through textual feedback [26] and candidate search [1, 24]. In each, however, the scheduling decision is fixed: revisions fire after every attempt or after every observed error subject to a call ceiling, offering no principled way to withhold an edit when the feedback itself might be wrong. We instead ask a complementary scheduling question: *when should feedback trigger a persistent instruction update?* This connects prompt adaptation to concept drift [7] and continual learning [23].

Proportional-integral-derivative (PID) control [3] suggests three complementary responses: reacting to current error, accumulating evidence over time, and monitoring deterioration after an edit [4]. We translate this intuition into discrete editing gates and compare them against online TextGrad and GEPA using one LLM-based editor and structured predictors. Across our empirical study (Section 5, Table 1, Figure 2), the PID gates collectively yield a more favorable Pareto trade-off between task performance and editor-call budget than the baselines.

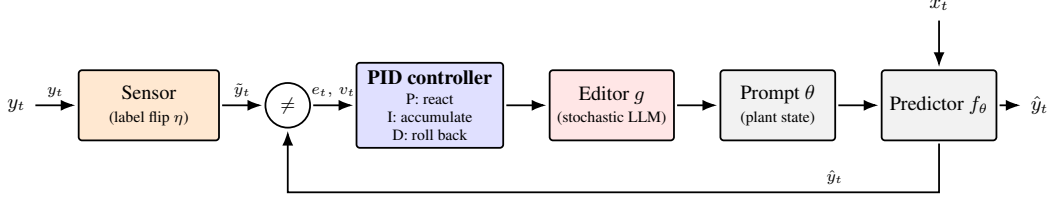


Figure 1: The regulated textual feedback loop. Signals enter from outside: target y_t from the left, input x_t from above. The Sensor flips the label with probability η (labels are either categorical, as in Parity, or bucketed categorical, as in Guess number / moving target). The comparator produces the observed error e_t (binary for Parity, bucketed distance for numeric tasks); when the controller fires, editor g rewrites the prompt θ .

2 Related Work

Self-evolving LLM agents and noisy evaluators. Methods that improve LLM behavior from natural-language feedback split by *what* they update. In the textual loop, Self-Refine [11], Reflexion [18], TextGrad [26], OPRO [24], and GEPA [1] revise prompts, chains-of-thought, or candidate instructions in response to observed errors, and skill-library agents such as Voyager [20] and Skill-Weaver [28] accumulate reusable textual skills. In the parameter loop, STaR [27], Self-Rewarding Language Models [25], RLHF [13], and Constitutional AI [5] convert feedback into weight updates. Our PID gates target the textual loop, where every edit is a stochastic LLM rewrite of a persistent artifact, so the scheduling question of *when* to invoke the editor is especially acute. The feedback signal itself is unreliable in both regimes: LLM-as-judge suffers position, verbosity, and self-preference biases [14, 22, 29], sycophancy [17], and models cannot reliably self-correct [8]. This motivates evidence-gated editing rather than reacting to every reported error.

Control theory for learning and continual adaptation. Feedback control has long informed machine learning: SGD-momentum as PI control [2], KL-weight scheduling in VAEs as PID regulation [16], constrained RL as a PID-Lagrangian [19], and deep-learning optimizers as continuous-time PID [6]. Our work instantiates PID intuition *outside* the parameter space of a differentiable model, on natural-language artifacts driven by a stochastic LLM editor. On the continual-learning side, catastrophic forgetting [12] and the stability-plasticity dilemma have driven decades of research [7, 15, 21, 23]. Classical parameter-space defenses (elastic weight consolidation [9], replay memory [10]) have natural prompt-space analogues in the buffered evidence and rollback rules used here.

3 Method: PID-Gated Adaptation

3.1 System overview

Figure 1 lays out the five-component closed loop. At each step t , a drifting environment presents input x_t with hidden target y_t ; the predictor emits $\hat{y}_t = f(\theta_t, x_t)$ under the current prompt θ_t ; a noisy sensor returns an observed label $\tilde{y}_t$ that disagrees with y_t at rate η ; the comparator then produces an observed error e_t (binary $\mathbf{1}[\tilde{y}_t \neq y_t]$ for categorical tasks, or a bucketed distance for numeric ones) and, on mismatch, a natural-language verbalization v_t . A PID controller reads the histories $\{e_\tau, v_\tau\}_{\tau \leq t}$ and decides whether to invoke the LLM editor g , which returns $\theta_{t+1} = g(\theta_t, \{v_\tau\})$. Because g is a stochastic actuator with no gradient and θ is a persistent state whose bad edits compound, the controller’s role is scheduling (*when* and on what evidence to call g) rather than choosing the prompt itself; the goal is to track y_t at low editing cost while remaining robust to corruption in $\tilde{y}_t$. In the guess-number game, for instance, θ_t encodes the agent’s belief about the plausible value range as integer bounds $[L, U]$; on each turn (x_t), the LLM predictor f emits a numeric guess ($\hat{y}_t$) conditioned on this belief; the environment scores it against a hidden target (y_t) and returns high/low/correct feedback with a distance bucket ($\tilde{y}_t$) that the sensor may flip; the comparator packages this into (e_t, v_t) , e.g., “guess was too low, distance 10–19”; when the controller fires, the LLM editor g rewrites $[L, U]$, tightening the belief for θ_{t+1} .

3.2 PID-inspired Gating

We map each classical PID component onto a discrete editing gate. All three gates share the same eligibility check (unused call budget, error evidence since the last reset, no mandatory pause) and invoke the same LLM editor g , which receives the current prompt and up to K recent observed examples (including correct ones; $K = 8$ in our experiments); invalid proposals retain the prompt and every model call counts against the budget. The gate controls only *when* g is called, never the content of θ_{t+1} .

Proportional Gate (P). The proportional term reacts to the current error: the *Proportional Gate* fires whenever the observer detects an error, i.e., $e_t = 1$ in the binary case or $e_t \geq \tau_P$ when the error is a non-negative bucketed distance, and eligibility holds. It is the most responsive schedule while the most vulnerable to isolated corruption in $\tilde{y}_t$.

$$a_t = \mathbf{1}[e_t \geq \tau_P] \mathbf{1}[\text{eligible at } t]. \quad (1)$$

Integral Gate (I). The integral term accumulates evidence over time. The *Integral Gate* maintains a decayed error sum and resets A_t to zero after any attempted edit.

$$A_t = \alpha A_{t-1} + e_t, \quad a_t = \mathbf{1}[A_t \geq \tau_I] \mathbf{1}[\text{eligible at } t]. \quad (2)$$

Derivative Gate (D). The derivative term watches for post-edit deterioration. On firing, the *Derivative Gate* snapshots the current prompt, holds the replacement fixed for W observations, and restores the snapshot if the post-edit error mean rises by at least τ_D over the pre-edit mean. Full update order is in Appendix A.

$$r_t = \mathbf{1}[\bar{e}_{\text{post}} - \bar{e}_{\text{pre}} \geq \tau_D]. \quad (3)$$

4 Experiments

4.1 Tasks

We design three simple but representative tasks to test the self-evolving agent’s capability across different output types (categorical and numeric) and drift patterns (sudden reversal, abrupt state shift, and smooth continuous drift). Together they probe how quickly the agent detects change, how it accumulates evidence under noisy feedback, and how gracefully it tracks a target that never stops moving. Hidden targets, boundaries, and initial rules are seed-randomized and never shown to the editor, so the study isolates LLM-mediated policy editing from end-to-end agent reasoning.

- **Parity** (categorical, sudden rule flips). Inputs are integers 1–10; the hidden rule assigns Red/Blue by parity, and the mapping reverses near steps 30, 60, and 90. Feedback is a categorical color, corrupted independently with probability η . Stresses categorical rule detection under discrete reversals.
- **Guess number** (numeric, abrupt state shifts). The agent guesses a hidden integer in $[1, 100]$ that shifts abruptly and periodically; feedback is high/low/correct paired with a distance bucket. Stresses numeric adaptation when the target state jumps.
- **Track moving target** (numeric, smooth continuous drift). The hidden integer follows a smooth continuous trajectory (e.g., sinusoidal, rectangular, or exponential); our implementation uses a sinusoid completing ~ 1.5 cycles over 120 steps, with no discrete change points. Stresses continuous tracking without visible boundaries a controller could anchor to.

4.2 Experimental Setup

We evaluate seven policies (No Edit, Fixed Cadence, Proportional (P), Integral (I), Derivative (D), TextGrad, GEPA) on the three tasks under three feedback-corruption levels $\eta \in \{0, 0.1, 0.2\}$ with $T = 120$ steps per trajectory, using the same LLM as both predictor and editor; results are mean $\pm$ std over 5 seeds per cell. No Edit keeps the initial prompt unchanged for the full trajectory (zero edits, zero editor calls) and serves as a no-adaptation reference; Fixed Cadence fires an edit every 15 steps regardless of the error signal and serves as a data-independent control. Seeded inputs, hidden targets, and controller settings are paired across corruption levels for matched comparison. Evaluation

metrics are task-specific: classification accuracy for **Parity** (higher is better), mean absolute error (MAE) for **Guess number** (lower is better), and root-mean-squared error (RMSE) for **Track moving target** (lower is better). Model calls are counted at every generation stage; tokens and deterministic candidate evaluations are reported separately. TextGrad and GEPA use the default parameters from their published libraries; interface and accounting details are given in Appendix B. We report primary results on Qwen3-14B (Table 1) and a scaling-up ablation on Qwen3.8-27B (Table 2).

5 Results and Analysis

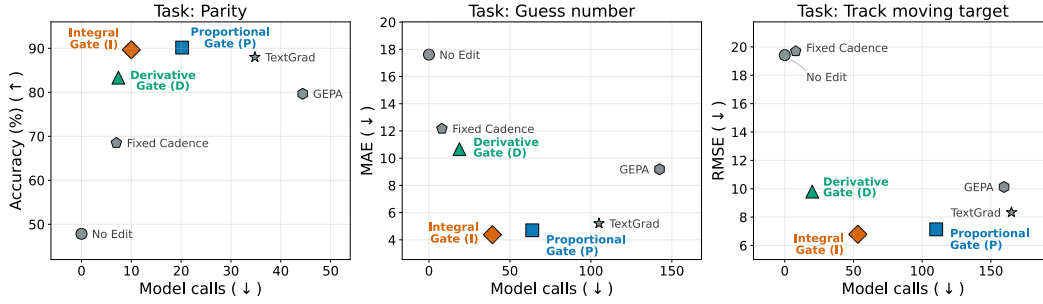


Figure 2: Cost-quality snapshot at $\eta = 0.1$ on Qwen3-14B (one marker per method, mean over 5 seeds). Higher accuracy and lower MAE/RMSE are better; fewer calls are cheaper. Points near the top-left (or bottom-left, for MAE/RMSE) indicate a better cost-quality trade-off.

Table 1: Primary results on Qwen3-14B at $\eta = 0.1$ (10% feedback corruption, 5 random seeds). Bold marks the best mean per column, excluding No Edit and Fixed Cadence.

Policy	Parity		Guess number		Track moving target	
	Acc.(%) ↑	Calls ↓	MAE ↓	Calls ↓	RMSE ↓	Calls ↓
No Edit	47.8 ± 5.8	0.0 ± 0.0	17.61 ± 3.05	0.0 ± 0.0	19.42 ± 2.05	0.0 ± 0.0
Fixed Cadence	68.5 ± 10.1	7.0 ± 0.6	12.17 ± 3.54	8.0 ± 0.0	19.70 ± 4.10	8.0 ± 0.0
TextGrad	88.0 ± 4.4	34.8 ± 13.1	5.20 ± 1.47	105.0 ± 21.1	8.33 ± 1.57	165.0 ± 3.3
GEPA	79.7 ± 6.4	44.4 ± 7.6	9.18 ± 1.55	142.4 ± 18.3	10.12 ± 0.91	159.6 ± 9.2
Proportional Gate (P)	90.2 ± 3.6	20.2 ± 5.1	4.70 ± 2.76	63.8 ± 14.3	7.14 ± 1.42	110.2 ± 5.5
Integral Gate (I)	89.7 ± 2.5	10.0 ± 3.0	4.38 ± 1.15	39.2 ± 8.0	6.79 ± 2.38	53.2 ± 1.2
Derivative Gate (D)	83.3 ± 4.9	7.4 ± 2.1	10.66 ± 2.19	18.8 ± 1.2	9.80 ± 0.84	20.0 ± 0.0

Table 2: Scaling-up ablation on Qwen3.8-27B at $\eta = 0.1$ (10% feedback corruption, 5 random seeds). Bold marks the best mean per column, excluding No Edit and Fixed Cadence.

Policy	Parity		Guess number		Track moving target	
	Acc.(%) ↑	Calls ↓	MAE ↓	Calls ↓	RMSE ↓	Calls ↓
No Edit	47.8 ± 5.8	0.0 ± 0.0	17.61 ± 3.05	0.0 ± 0.0	19.42 ± 2.05	0.0 ± 0.0
Fixed Cadence	61.0 ± 15.7	7.2 ± 0.7	14.59 ± 4.48	7.8 ± 0.4	20.03 ± 5.50	8.0 ± 0.0
TextGrad	87.3 ± 5.9	34.8 ± 13.9	3.92 ± 1.39	99.6 ± 20.2	7.03 ± 1.22	160.8 ± 8.8
GEPA	79.7 ± 6.4	44.4 ± 7.6	8.27 ± 1.75	107.0 ± 10.6	10.31 ± 0.98	161.6 ± 3.5
Proportional Gate (P)	89.2 ± 3.7	22.6 ± 6.4	3.93 ± 1.34	67.6 ± 19.5	6.45 ± 1.20	108.0 ± 3.9
Integral Gate (I)	89.7 ± 3.7	10.0 ± 3.7	4.19 ± 1.61	39.8 ± 10.1	6.34 ± 1.49	52.4 ± 2.0
Derivative Gate (D)	82.3 ± 4.3	8.0 ± 1.9	8.85 ± 1.27	19.2 ± 1.0	10.25 ± 2.14	19.8 ± 0.4

Table 1 summarizes the primary comparison on Qwen3-14B at $\eta = 0.1$: on every task, a PID gate attains the top quality metric among learned schedules (90.2% accuracy on Parity for Proportional; 4.38 MAE on Guess number and 6.79 RMSE on Track for Integral), while all PID gates issue fewer LLM calls than TextGrad and GEPA. The Integral Gate occupies a particularly efficient operating point: it matches or beats TextGrad on every quality metric while using roughly 2.7–3.5× fewer

editor calls, and the Derivative Gate uses the fewest calls on each task at a modest quality trade-off. The same cost-quality ordering replicates on a larger LLM (Table 2): on Qwen3.8-27B, the Integral Gate takes the top Parity accuracy (89.7%) and the top Track RMSE (6.34) at roughly $3\times$ fewer calls than TextGrad, with Guess-number MAE within 0.3 of the best. Absolute quality tracks editor capacity, but the Pareto ordering between PID gates and the optimizer baselines is preserved. Figure 2 visualizes the 14B snapshot: PID points sit up and to the left of the TextGrad/GEPA cluster, i.e., comparable or higher quality at substantially lower cost. This ordering is consistent across all three noise levels $\eta \in \{0, 0.1, 0.2\}$ (Appendix C, Figure 3): the PID gates occupy a better cost–quality frontier than TextGrad and GEPA under both clean and corrupted feedback.

6 Conclusion and Future Work

We introduced PID-gated scheduling (Proportional, Integral, Derivative) that decides *when* to invoke LLM prompt editors for continual prompt adaptation under drifting tasks and noisy feedback. Across three drifting tasks, the PID gates collectively yield a more favorable Pareto trade-off between task performance and editor-call budget than the baselines considered, typically matching the strongest optimizer at a fraction of its call cost (Section 5, Tables 1–2, Figure 2). Future work includes the *how-much* question (edit magnitude), for which a natural extension is an “edit gain” that limits the magnitude of the change from θ_t to θ_{t+1} so one controller regulates both frequency and magnitude; and online auto-tuning of the gate thresholds, borrowing established methods from classical control.

References

- [1] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziem, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. URL <https://arxiv.org/abs/2507.19457>.
- [2] Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang. A PID controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [3] Karl Johan Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, 2nd edition, 1995.
- [4] Karl Johan Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.
- [5] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [6] Song Chen, Jiaxu Liu, Pengkai Wang, Chao Xu, Shengze Cai, and Jian Chu. Accelerated optimization in deep learning with a proportional-integral-derivative controller. *Nature Communications*, 15(1):10263, 2024.
- [7] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, 2014.
- [8] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024.
- [9] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.
- [10] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [11] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [12] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, 1989.
- [13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

- [14] Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- [15] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.
- [16] Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek Abdelzaher. ControlVAE: Controllable variational autoencoder. In *International Conference on Machine Learning (ICML)*, 2020.
- [17] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. Towards understanding sycophancy in language models. In *International Conference on Learning Representations (ICLR)*, 2024.
- [18] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [19] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID lagrangian methods. In *International Conference on Machine Learning (ICML)*, 2020.
- [20] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research (TMLR)*, 2024.
- [21] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.
- [22] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
- [23] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.
- [24] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.
- [25] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *International Conference on Machine Learning (ICML)*, 2024.
- [26] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4.
- [27] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [28] Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.
- [29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

A Update Order and Gate Characterization

Algorithm 1 PID-inspired gate editing (Proportional, Integral, Derivative). One pass over T steps; the gate G selects the trigger condition and whether the post-edit regression check is armed. The Derivative gate reuses the Integral trigger ($A \geq \tau_I$) and adds a pre/post error-mean regression check that can roll back the edit.

Require: prompt θ ; editor g ; gate $G \in \{P, I, D\}$; decay α ; thresholds τ_P, τ_I, τ_D (τ_I is shared by I and D); guard window W ; history size K ; call cap B ; horizon T

- 1: $A \leftarrow 0$; $\mathcal{H} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$ (recent-error queue, length W); pending $\leftarrow \emptyset$ (holds $(\theta_{\text{snap}}, \bar{e}_{\text{pre}}, \text{post-window})$ for D); cooldown $\leftarrow 0$; calls $\leftarrow 0$; evidence $\leftarrow \text{FALSE}$ $\triangleright$ True once any error has been seen since the last reset
- 2: **for** $t = 0, \dots, T - 1$ **do**
- 3: Predict $\hat{y}_t = f(\theta, x_t)$; observe $\tilde{y}_t$; set $e_t = 1[\hat{y}_t \neq \tilde{y}_t]$ (categorical) or a bucketed distance (numeric)
- 4: Append $(x_t, \hat{y}_t, \tilde{y}_t, e_t)$ to $\mathcal{H}$; retain the latest K entries
- 5: restored $\leftarrow \text{FALSE}$ $\triangleright$ set if a D-gate rollback just happened; skips accumulator update
- 6: **if** $G = D$ **and** pending $\neq \emptyset$ **then** $\triangleright$ regression check
- 7: Append e_t to pending's post-edit window
- 8: **if** W post-edit observations collected **then**
- 9: **if** $\bar{e}_{\text{post}} \geq \bar{e}_{\text{pre}} + \tau_D$ **then**
- 10: Restore $\theta \leftarrow \theta_{\text{snap}}$; $A \leftarrow 0$; evidence $\leftarrow \text{FALSE}$; restored $\leftarrow \text{TRUE}$
- 11: **end if**
- 12: pending $\leftarrow \emptyset$
- 13: **end if**
- 14: **end if**
- 15: Append e_t to $\mathcal{R}$ (drop oldest if $|\mathcal{R}| > W$)
- 16: **if not** restored **then**
- 17: $A \leftarrow \alpha A + e_t$; **if** $e_t \geq 1$ **then** evidence $\leftarrow \text{TRUE}$
- 18: **end if**
- 19: **if** cooldown > 0 **then**
- 20: cooldown $\leftarrow \text{cooldown} - 1$; **continue**
- 21: **end if**
- 22: fire $\leftarrow (G=P \wedge e_t \geq \tau_P) \vee (G \in \{I, D\} \wedge A \geq \tau_I)$
- 23: **if** fire **and** evidence **and** calls $< B$ **then**
- 24: **if** $G = D$ **and** $|\mathcal{R}| = W$ **then**
- 25: pending $\leftarrow (\theta, \bar{e}_{\text{pre}} = \frac{1}{W} \sum \mathcal{R}, \text{empty post-window})$
- 26: **end if**
- 27: $\theta \leftarrow g(\theta, \mathcal{H})$; calls $\leftarrow \text{calls} + 1$
- 28: $A \leftarrow 0$; evidence $\leftarrow \text{FALSE}$; **if** $G = D$ **then** cooldown $\leftarrow W$
- 29: **end if**
- 30: **end for**

Default settings (shared across gates): $\alpha = 0.9$, $W = 5$, $K = 8$, $B = 180$. Gate-specific: $\tau_I = 1.5$ (used by I and D), $\tau_D = 0.15$ (D only), $\tau_P = 1$ (P only, binary-error case).

B Published Optimizers as Online Baselines

TextGrad [26] and GEPA [1] use their pinned official packages, with inference routed exclusively through the common LLM-based editor. Both trigger after two observed errors since the previous invocation and require at least three remaining calls. Each restarts from the currently deployed prompt and receives at most eight past/current observations. Neither accesses clean evaluation metrics or future observations. These adaptations compare online use of the algorithms with measured resource costs, rather than reproducing their original task suites or claiming that all implementations are equivalent.

TextGrad. Version 0.1.8 uses an explicit engine for each of three stages: `TextLoss` critiques the current structured prompt against the task specification and observed feedback; `backward` produces textual gradient feedback; `TextualGradientDescent` proposes an updated variable. Gradient memory is zero. Loss and backward stages request concise critiques of at most 100 words; the update retains the package’s required variable tags. Deterministic prediction is outside the autograd graph: the optimized variable is the structured policy prompt itself. Missing tags or an invalid structured proposal retain the deployed prompt and consume all three calls.

GEPA. Version 0.1.4 starts a fresh search population from the deployed prompt, uses instance-level Pareto parent selection, and permits at most three model reflections per invocation. Accepted proposals must strictly improve on a common training minibatch before full validation. Alternating history entries form disjoint training and validation subsets when at least four observations are available, with the newest in training; smaller histories reuse the same observations in both. Minibatches contain at most three training examples. Optional merging is disabled, and perfect-score minibatches may skip reflection. Validation is past noisy evidence, not an independent estimate of future task performance.

The common client counts all reflection calls. Candidate-example evaluations include rejected and malformed candidates and are logged separately. A nominal metric-evaluation limit provides an additional termination condition for skipped reflections; the official search checks it between iterations, so realized evaluations can exceed that nominal limit. Every proposal, selected parent, fitness evaluation, validation score, and final selection is retained. An invocation containing an invalid proposal may still deploy a different valid candidate; invalid-proposal count is therefore distinct from final prompt changes. Unexpected model or library failures halt the run rather than being relabeled successful no-ops.

C Robustness Across Feedback-Noise Levels

Figure 3 extends the primary comparison (Figure 2) by plotting all three noise levels $\eta \in \{0, 0.1, 0.2\}$ together, with marker face shade encoding noise (hollow $\rightarrow$ half-filled $\rightarrow$ solid). Across every corruption level and on every task, the PID gates sit on a better cost–quality frontier than TextGrad and GEPA: they reach comparable or higher quality at a substantially lower editor-call cost. The Integral Gate, in particular, holds its operating point across the whole noise range, matching or beating the strongest baseline at a fraction of its call count. This replicates the Pareto conclusion of Section 5 and shows that the ordering is not an artifact of the single noise level highlighted in the main text.

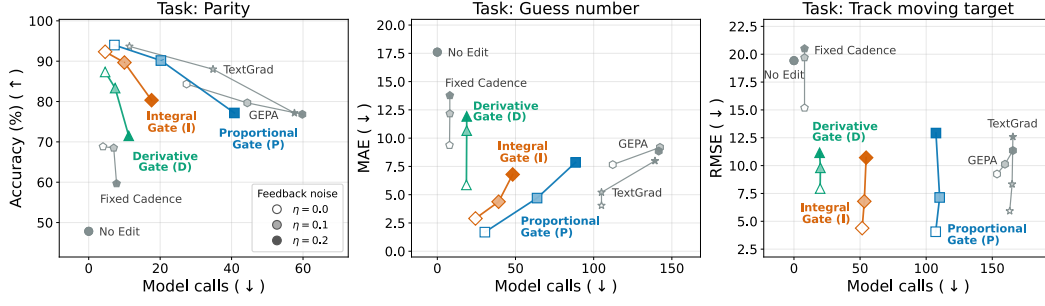


Figure 3: Cost–quality tradeoff across all three feedback-noise levels on Qwen3-14B (mean over 5 seeds). Marker face shade encodes noise: hollow $\eta = 0$, half-filled $\eta = 0.1$, solid $\eta = 0.2$. Lines connect the three noise points for each method. Points near the top-left (bottom-left for MAE/RMSE) indicate a better cost–quality trade-off. PID gates sit on a better frontier than TextGrad and GEPA at every noise level.

D Notation

Only recurring symbols are listed. One-shot symbols keep their meaning inline. The letter g denotes the LLM editor throughout.

Table 3: Recurring symbols.

Symbol	Meaning
t, T	Step index and horizon; $T = 120$ steps per trajectory.
x_t, y_t	Input and hidden ground-truth target; y_t never reaches the editor.
$\hat{y}_t, \tilde{y}_t$	Predictor output $\hat{y}_t = f(\theta_t, x_t)$ and observed (possibly corrupted) feedback.
e_t	Observed error: $\mathbf{1}[\hat{y}_t \neq \tilde{y}_t]$ for categorical tasks, or a bucketed distance for numeric ones.
θ_t	Current prompt (plant state); the editor’s target of change.
f, g	LLM predictor $\hat{y}_t = f(\theta_t, x_t)$ and LLM editor $\theta_{t+1} = g(\theta_t, \mathcal{H})$ (invoked when the gate fires).
$\mathcal{H}, K$	Rolling editor history: up to $K = 8$ most recent observed examples.
A_t, α, τ_I	Decayed error accumulator $A_t = \alpha A_{t-1} + e_t$; decay $\alpha = 0.9$; threshold $\tau_I = 1.5$.
τ_P, τ_D	Proportional-gate threshold ($\tau_P = 1$ in the binary case); Derivative-gate restoration threshold $\tau_D = 0.15$.
W	Guard window: number of observations used for the pre- and post-edit error means ($W = 5$).
B	Nonbinding call cap per trajectory ($B = 180$).
η	Feedback corruption probability (label-flip rate), $\eta \in \{0, 0.1, 0.2\}$.