DIAL: Direction-Informed Adaptive Learning for LLM Agents

Ziming Li[1], Jiatan Huang[1], Xiaoguang Guo[1], Guiling Wang[2], Chuxu Zhang[1]*
{ziming.li,jiatan.huang,xiaoguang.guo,chuxu.zhang}@uconn.edu
gwang@njit.edu
[1]University of Connecticut, [2]New Jersey Institute of Technology

Abstract

Adaptive test-time compute for LLM agents aims to invoke extra computation only when it improves performance. Existing methods typically use confidence-, uncertainty-, or difficulty-based gates, assuming a fixed direction from the gating signal through compute need to the value of computation. This makes gating a utility-calibration problem: gating signals should align with whether extra computation improves the final outcome over the base policy. We show that this alignment is unstable: the same signal predicts rollout benefit in one setting and rollout harm in another, with reversals across environments and backbones even when the task is fixed. Wrong-direction gates can therefore worsen performance by precisely selecting harmful states. This reversal reflects a deeper distinction between compute need and compute suitability: a high uncertainty signal may indicate decision-difficult states where rollouts help compare alternatives, or intervention-unsuitable states where the current context does not support useful rollout-based improvement. Under this two-source model, fixed-direction gates are unreliable across heterogeneous settings. To address this, we propose DIAL (**D**irection-**I**nformed **A**daptive **L**earning), a sparse gate trained from signal-agnostic counterfactual exploration to learn the utility direction of state features per (environment, backbone). Across six environments and three backbones, DIAL yields a stronger overall success-cost trade-off than fixed-direction baselines. Our code is open-sourced at https://github.com/zmli6/DIAL.

1 Introduction

Large language models (LLMs) have rapidly advanced across a wide range of tasks, from question answering and code generation to embodied decision making [1–6]. However, a single forward pass is often insufficient for complex tasks that require evaluating alternatives, verifying intermediate results, or searching over candidate actions [7–9]. This has motivated test-time compute, where additional computation at inference time improves decision quality [10, 11].

However, test-time compute is expensive: each decision point may require multiple rollouts or candidate evaluations [3, 7]. Applying it uniformly across diverse agent settings is wasteful, since only a fraction of steps benefit. This has motivated *adaptive gating*: using uncertainty signals to decide *when* extra compute is worthwhile. Prior work explores various mechanisms, including entropy thresholding [12–14], calibrated confidence [15, 16], vote disagreement [17], and learned difficulty estimators based on hidden states or reinforcement learning [18–21]. These methods differ in mechanism but share a common assumption: larger uncertainty, disagreement, or difficulty signals indicate greater value of additional computation. We call this the *fixed-direction* assumption. In interactive agents, this makes gating a utility-calibration problem: the signal must predict the value of computation, namely whether invoking the optimizer improves the final task outcome. This differs from standard confidence calibration, which aligns confidence with action correctness.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We test this assumption empirically using token entropy, the most common signal in prior work [12–14]. The fixed-direction assumption predicts $\rho(\text{entropy}, U) > 0$: higher entropy should mark states where rollout has positive value. Figure 1 finds this violated on half of six agent environments: high entropy can predict that extra compute hurts rather than helps. Table 1 further shows that the same task can flip sign across model backbones, so a signal's utility semantics are determined by the (environment, backbone) pair, not by the environment or backbone alone. This reversal is consequential: when the assumed direction is wrong, sharper gates worsen performance by more precisely selecting harmful states. The most predictive signal also varies across environments (Table 6), so adaptive compute cannot rely on a fixed signal with a fixed direction. This raises a central question: *if signal-utility direction is not fixed, can adaptive compute still be principled?*

Answering this requires understanding why direction reverses in the first place. Hidden in the fixed-direction assumption is a deeper conflation: uncertainty or difficulty measures whether a state may benefit from more compute, but not whether that compute will actually help. We trace this reversal to two coexisting state types with opposite signal-utility relationships. *Intervention-unsuitable* states (Type I) arise when the current context does not support useful rollout-based improvement (e.g., missing evidence, misleading partial observations, or weak grounding); high uncertainty there predicts negative value of computation, since additional rollouts amplify unresolved ambiguity rather than resolve it. *Decision-difficult* states (Type D, e.g., coding tasks with multiple viable implementations) arise when multiple options exist; rollouts can sample different ones and find the best, so high uncertainty predicts positive value. The mixture depends on what the agent already encodes, so it is determined by the (environment, backbone) pair; the same uncertainty signal therefore carries opposite meanings depending on which type dominates, an instance of Simpson's paradox [22, 23]: the marginal signal-utility correlation reflects the mixture of two subgroups whose internal correlations point in opposite directions.

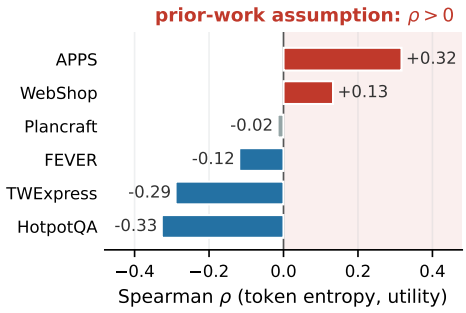


Figure 1: **The fixed-direction assumption fails on half of the environments.** Spearman correlation $\rho(\text{entropy}, U)$ across six environments on Qwen3-4B.

This is why uncertainty alone cannot reliably gate: two states at the same uncertainty level can have opposite utility, so no function of uncertainty alone can distinguish them. A reliable gate must therefore use additional features that correlate with state type, such as step count as a proxy for information accumulation. This also clarifies why replacing thresholds with learned difficulty estimators or confidence calibration does not automatically resolve the problem. Calibration aligns confidence with action correctness, whereas adaptive gating requires alignment with the counterfactual utility of a particular optimizer. Difficulty estimators may identify that a state is hard, but hardness can mean either that rollouts can productively compare alternatives or that the current context is unsuitable for improvement. Moreover, a learned gate trained only on uncertainty-triggered states can inherit the same fixed-direction assumption through selection bias, baking the prior into the training distribution rather than testing it. The decomposition further yields testable predictions about how $\rho(\sigma, U)$ changes with the Type I/Type D mixture, which we verify on held-out data in §5.4.

This motivates data-driven direction learning: utility depends on interaction outcomes and cannot generally be inferred from model-internal properties alone. Building on this analysis, we propose DIAL (**D**irection-**I**nformed **A**daptive **L**earning), which learns the signal-utility direction for each (environment, backbone) setting from data. DIAL collects signal-agnostic paired counterfactual outcomes by running the optimizer and the base policy from the same state, then fits a sparse linear gate over a small set of state features augmented by LLM-proposed task-specific signals. Across six environments and three LLM backbones, DIAL consistently improves the success-cost trade-off over fixed-direction baselines, triggering frequently when rollouts help and rarely when they do not.

2 Related Work

Prior work on adaptive test-time compute uses diverse mechanisms, but shares a common assumption: gating signals have a fixed, setting-independent relationship with compute value. We organize the literature by how each line embeds this assumption.

Gating on uncertainty signals. Signal-based methods allocate extra computation when token confidence, entropy, or disagreement exceeds a threshold, encoding the fixed-direction assumption directly in their trigger rule. In reasoning settings, this takes the form of extended chain-of-thought generation [12–14, 24, 15, 16]. In agent settings, vote-based methods [17] invoke an arbiter when candidate actions disagree, and verbalized uncertainty methods [25] trigger reflection when self-reported confidence is low. All of these methods hardcode the direction from uncertainty to the value of computation; we show this direction is itself a function of (environment, backbone) and cannot be hardcoded.

Gating on learned difficulty. Hidden-state probes [18, 19] estimate task difficulty from internal representations, and RL-based methods [20, 21, 26, 27] learn think/no-think policies end-to-end. Although these methods commit to neither a specific signal nor an explicit threshold, their evaluations primarily focus on homogeneous reasoning benchmarks such as GSM8K [28] and GPQA [29], where higher difficulty is typically aligned with greater benefit from additional reasoning. The difficulty-compute relationship is thus never tested for reversal. We test it explicitly in heterogeneous agent environments, where it fails.

Confidence calibration vs. utility calibration. Classical confidence calibration aligns a model's stated confidence with its action correctness; recent work shows that calibration quality is domain-dependent [30–33]. Our problem is distinct: *utility calibration* aligns gating signals with the value of additional computation, not action correctness. Rational metareasoning [34, 35] formalizes this VOC target but classically assumes VOC ≥ 0; we show this fails when the same model both proposes and evaluates actions (§3.3). Beyond domain-dependent quality, the *sign* of the signal-to-utility correlation reverses across (environment, backbone) pairs, so even correctness-calibrated confidence is insufficient for gating. Full discussion in Appendix A.

3 Why Fixed-Direction Gating Fails

In this section, we first formalize the adaptive gating problem (§3.1), then measure signal-utility correlations across diverse environments to test the fixed-direction assumption (§3.2), and finally model why the assumption fails (§3.3).

3.1 Problem Setup

We study LLM-based agents that solve tasks through multi-step interaction with an environment. At each step t, the agent observes state s_t, selects action a_t from its base policy π, and transitions to s_{t+1}. An environment-specific *test time optimizer* T (e.g., rollout evaluation, K-variant sampling) can be invoked at any step to potentially improve action quality. Let $\sigma(s)$ denote a scalar gating signal at state s (e.g., token entropy, confidence, disagreement, or a learned difficulty score). We define the *optimizer utility* as:

$$U(T, s) = \mathbb{E}[R(\tau) \mid a{=}T(s)] - \mathbb{E}[R(\tau) \mid a{=}\pi(s)], \tag{1}$$

where $R(\tau)$ is the episode return. $U > 0$ indicates the optimizer improves the outcome; $U \leq 0$ indicates it is wasteful or harmful. The *adaptive gating problem* is to learn a gate $g : \mathcal{S} \to \{0, 1\}$ that triggers T only when $U > 0$.

3.2 Signal-Utility Direction Reverses Across Environments and Backbones

The signal-utility direction is not fixed. Table 1 reports Spearman correlations between token entropy and optimizer utility across 6 environments and 3 model backbones [36–38]. The correlation reverses along two axes. First, *the same signal can reverse across backbones with the task held fixed*: among environments with at least two significant nonzero correlations, sign disagreement across backbones is common. The most extreme case is FEVER, which flips from $\rho{=}-0.156$ on Phi-3.5 to $\rho{=}+0.428$ on Llama-3.1, a sign change driven entirely by the choice of LLM. The signal's semantics are

Table 1: Signal-utility correlations across 6 environments and 3 backbones. ρ: Spearman of token entropy with utility U; direction varies across both. Cons.: sign agreement across backbones (✓same, ✗flip, —: insufficient). *: $p{<}0.05$.

Environment	Qwen3	Phi-3.5	Llama-3.1	Cons.
HotpotQA	-0.327^*	$+0.184^*$	-0.346^*	✗
WebShop	$+0.133^*$	$+0.335^*$	$+0.287^*$	✓
FEVER	-0.119^*	-0.156^*	$+0.428^*$	✗
APPS	$+0.317^*$	-0.024	-0.249^*	✗
TWExpress	-0.290^*	$+0.000$	$+0.000$	—
Plancraft	-0.016	$+0.000$	-0.176^*	—

therefore not a property of the environment alone, but of how the agent's prior knowledge interacts with it. Second, *within a single backbone, the direction varies across environments*: on Qwen3, entropy is negatively correlated with utility in information-seeking environments (FEVER, HotpotQA, TWExpress) but positively in code generation (APPS) and web navigation (WebShop), with near-zero correlation in weak-signal environments (Plancraft).

Signal identity also varies across environments. Beyond direction, the most informative signal itself differs across environments: `step_count` dominates in most settings, but WebShop relies on `num_available_actions` instead (Appendix F.1), and no single signal is reliably informative across all environments (Appendix F.4).

Wrong direction is harmful, not just uninformative. This direction mismatch is not benign: a gate that assumes the wrong direction systematically falls *below* the no-trigger baseline. On strong-signal environments, reversing the gate's direction collapses success rate by 23–37 points (Table 3), making it strictly worse than doing nothing in mismatched (environment, backbone) pairs.

The reversal is not an artifact of entropy scales. A natural concern is that cross-backbone reversal could reflect incomparable raw entropy scales across tokenizers. We rule this out: per-(environment, backbone) quantile normalization preserves both Spearman rank ordering (trivially) and Pearson sign on the strong-correlation cells (Appendix G.1). The reversal is also robust to monotone reward transformations $U \to \alpha U$ ($\alpha > 0$) and to entropy transformations $\sigma \to \sigma/T$, $\sigma \to \sigma^{\alpha}$, and $\sigma \to \log \sigma$, ruling out reward parameterization and entropy-pipeline calibration as alternative explanations (Appendix G.2). Together, these findings suggest that uncertainty reflects latent state structure that varies across (environment, backbone) pairs, which we formalize next.

3.3 Why Does Direction Reverse? A Two-Source Model

Model. We model direction reversal as arising from two coexisting regimes within each environment, not from two disjoint classes of tasks. The following linear form is a regime-level abstraction: it specifies the sign of the signal-utility relationship within each regime, not a mechanistic claim that all states within a regime share a single underlying cause. Operationally, Type I denotes regimes with a negative within-regime signal-utility slope, regardless of whether the cause is missing evidence, weak grounding, invalid intermediate states, or rollout-induced compounding errors. At each decision step, a state can be of either type:

- **Type I (intervention-unsuitable):** Utility is *negatively* related to entropy: $U_I(s) \sim -\alpha H(s) + \varepsilon_I$, with $\alpha > 0$ and zero-mean noise ε_I. These are states where the current context does not provide a reliable basis for the rollout optimizer to improve the action; missing evidence in evidence-seeking tasks is one common cause, alongside misleading partial observations, weak grounding, or compounding rollout errors.
- **Type D (decision-difficult):** Utility is *positively* related to entropy: $U_D(s) \sim +\beta H(s) + \varepsilon_D$, with $\beta > 0$ and zero-mean noise ε_D. These are states where uncertainty reflects multiple viable alternatives, so rollouts can productively sample, compare, or verify candidate actions.

Here α and β are the within-type slopes: they capture how strongly entropy maps to utility within each state type. A real environment $\mathcal{E}$ contains a mixture of both types; let $p_I(\mathcal{E})$ denote the fraction of Type I states. The marginal entropy-utility correlation across all states is a weighted blend of the two within-type slopes:

$$\rho(\mathcal{E}) \;\approx\; \beta - (\alpha + \beta)\, p_I(\mathcal{E}). \tag{2}$$

At the two extremes, $\rho = +\beta$ when the environment is pure Type D ($p_I = 0$) and $\rho = -\alpha$ when it is pure Type I ($p_I = 1$). In between, ρ crosses zero at $p_I^* = \beta/(\alpha + \beta)$: environments above this threshold show negative ρ (Type I dominated), those below show positive ρ (Type D dominated), and those near it show negligible signal as the two effects cancel. This expression captures the sign-level dependence under a simplified linear model and is not intended as an estimator of Spearman ρ.

Implication: σ is not sufficient for optimizer utility. Two states with identical σ but opposite latent types have opposite-sign utility. An uncertainty-only gate $g(\sigma(s))$ that lacks features distinguishing latent state types therefore cannot guarantee non-negative value across heterogeneous settings (Proposition 1, proof in Appendix B.1). Recovering sufficiency requires augmenting σ with features that distinguish the two state types, directly motivating the multi-feature design of §4.

Backbone-level reversal is a prediction, not an anomaly. The fraction $p_I(\mathcal{E})$ defined above depends on more than just the environment. A state is intervention-unsuitable only relative to what

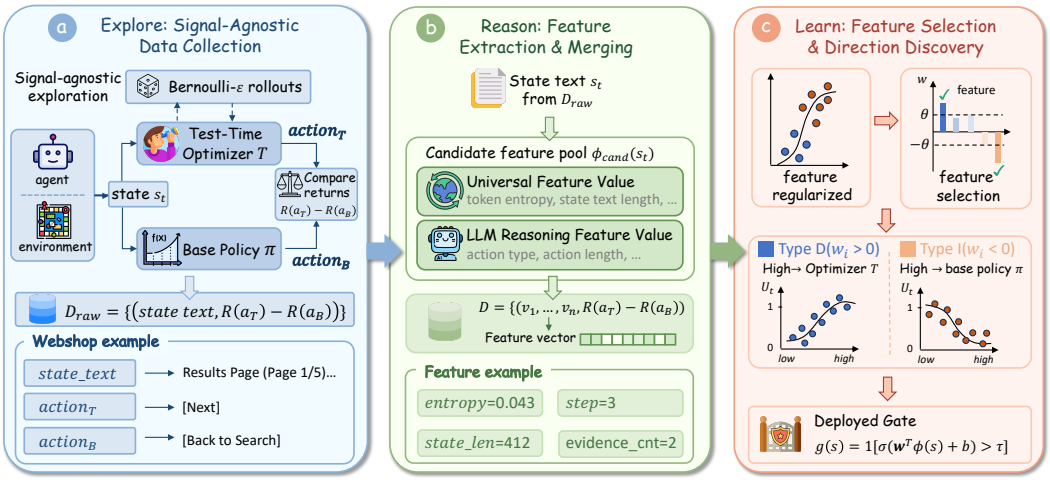


Figure 2: **The DIAL pipeline. (a) Explore**: paired counterfactual rollouts via Bernoulli-ε yield raw data $\mathcal{D}_{\text{raw}}=\{(\text{state text}, R(a_T)-R(a_B))\}$. **(b) Reason**: from each state text, a candidate feature pool $\phi_{\text{cand}}(s)$ combines universal features with LLM-proposed task-specific features, yielding $\mathcal{D}=\{(\phi(s), U)\}$. **(c) Learn**: fit an ℓ_1-regularized logistic gate $g(s)$; signed weights jointly select features and recover per-environment direction (Type D: $w_i>0$; Type I: $w_i<0$).

the agent already encodes: the same observation can be Type I for a model that lacks the relevant knowledge or grounding and Type D for one that has it. The effective mixture is therefore indexed by the (environment, backbone) pair, not the environment alone. This explains the backbone-level sign flips of Table 1, most extreme on FEVER between Phi-3.5 and Llama-3.1: backbone changes shift p_I, and when they cross p_I^*, ρ flips (Appendix G.6).

Theoretical grounding. This direction reversal is a special case of Simpson's paradox [22, 39]: aggregating subpopulations with opposing within-group trends reverses the aggregate correlation. Our decomposition adapts the epistemic/aleatoric distinction [40] to adaptive compute: Type I states often exhibit epistemic-like uncertainty (e.g., information deficit or weak grounding), while Type D states exhibit aleatoric-like diversity (multiple viable paths). Rather than a post-hoc explanation, this decomposition makes testable predictions: in §5.4 we derive three (temporal dynamics, cross-environment consistency, and signal identity alignment) and verify each on held-out data.

4 Direction-Informed Adaptive Learning

DIAL reframes gating from difficulty estimation to intervention-utility estimation: it trains on counterfactual optimizer utility rather than difficulty or correctness proxies, and uses signal-agnostic exploration so that the learned gate can discover when each feature indicates positive or negative rollout value instead of inheriting a fixed direction from the data collection rule. The analysis in §3 implies three constraints for any cross-setting adaptive gate. It must (i) *discover* the signal-utility direction per (environment, backbone) pair rather than hardcoding it; (ii) draw from a *multi-signal feature pool* that approximates the latent state type $\tau(s)$, since no single signal is reliable across settings; and (iii) collect training data *without conditioning on* the very signal whose direction is unknown, otherwise the data is biased toward the prior assumption. DIAL satisfies these three constraints in three stages (Fig. 2): randomized exploration (§4.1), feature construction (§4.2), and sparse gate fitting (§4.3).

4.1 Explore: Signal-Agnostic Data Collection

The exploration stage collects the (s, U) pairs used to fit the gate, under one central constraint: the collection policy must remain independent of any signal whose semantics we are trying to learn. A gate trained on uncertainty-triggered states would systematically under-sample the states that determine ρ's sign, baking the prior assumption into the training data instead of testing it.

DIAL therefore uses the simplest policy consistent with this constraint. At each step t in episode i, a uniform random coin flip with probability $\varepsilon_{\text{explore}}$ determines whether to trigger the optimizer T

($z_t^{(i)}$=1) or use the base action ($z_t^{(i)}$=0). For each step with $z_t^{(i)}$=1, we estimate utility by paired counterfactual rollouts from the same state snapshot: we fork the environment, execute both the optimizer's chosen action and the base action via truncated rollouts of horizon H, and record which yields a higher return. We set $U_t^{(i)} = 1$ if the optimizer's action wins, and 0 otherwise; we binarize the continuous utility of Eq. (1) for logistic-gate fitting, which preserves the sign that drives gating decisions. The Bernoulli trigger eliminates selection bias, and the paired design eliminates baseline bias; the remaining approximation is the truncation horizon H. Aggregating across N_{explore} episodes yields the dataset:

$$\mathcal{D} = \left\{ \left(\phi(s_t^{(i)}),\ U_t^{(i)} \right) : z_t^{(i)} = 1 \right\}_{t,i}, \tag{3}$$

where $\phi(s)$ is the feature vector. Per-environment estimation details are in Appendix D.1.

4.2 Reason: Feature Construction

The next stage builds the candidate feature pool $\phi_{\text{cand}}(s)$. This pool must be rich enough to capture each environment's informative signals, but not so environment-specific that deployment requires manual engineering per task.

DIAL's candidate pool combines two layers. The first is a small set of universal features defined identically across environments: `step_count`, `token_entropy`, `evidence_count`, `num_available_actions`, and `is_finish` (definitions in Table 9). All universal features are computed from the state text via either direct parsing (e.g., `evidence_count`) or the agent's existing LLM forward pass (e.g., `token_entropy`, `is_finish`); no additional data source is required. Signals an environment does not naturally expose (e.g., `evidence_count` outside QA-style tasks) default to zero, requiring no per-environment engineering. The second is an LLM-generated layer: an LLM receives a summary of $\mathcal{D}$ and proposes task-specific feature hypotheses $\phi_{\text{LLM}}(s)$, forming $\phi_{\text{cand}}(s) = \phi_{\text{univ}}(s) \cup \phi_{\text{LLM}}(s)$. The ℓ_1 penalty in the next stage automatically discards uninformative proposals, so the LLM layer can propose freely without degrading gate quality. We show in §5.3 that the universal pool alone retains most of DIAL's gains, and the LLM layer adds further gains in environments where universal features are inadequate. We provide the prompt template and per-environment ablation in Appendix C.3 and Appendix F.2.

4.3 Learn: Sparse Linear Gate

Given $\mathcal{D}$ and ϕ_{cand}, DIAL fits an ℓ_1-regularized logistic regression with the binary cross-entropy loss, where the regularization strength λ controls sparsity over ϕ_{cand} so that uninformative features drop out. Let $(\mathbf{w}^*, b^*)$ denote the fitted weights and bias. The deployed gate is

$$g(s) = \mathbf{1}\left[\text{sigmoid}(\mathbf{w}^{*\top}\phi(s) + b^*) > \tau\right], \tag{4}$$

with threshold τ set by cross-validation on $\mathcal{D}$. At deployment, the gate requires only a feature evaluation and a single sigmoid: no extra LLM calls, no K-way voting, no calibration query. This zero per-step LLM overhead after the one-time exploration phase contrasts with prior gating methods that incur per-step costs (e.g., CATTS triggers K=5 forward passes per step, AUQ issues a confidence query every step).

Signed weights as a diagnostic. The ℓ_1 penalty simultaneously performs feature selection and direction discovery. The sign of each fitted weight provides an interpretable per-environment estimate of how the feature relates to optimizer utility: $w_i > 0$ indicates that feature i behaves as a decision-difficulty proxy in this setting, where larger values increase expected rollout value; $w_i < 0$ indicates an intervention-unsuitability proxy, where larger values mark states in which rollouts are less likely to help; and $w_i \approx 0$ indicates that the feature is uninformative. The complete pipeline, including online adaptation for drifting environments and detailed cost accounting, is in Appendix C.

5 Experiments

5.1 Experimental Setup

We evaluate DIAL along three axes: does environment-aware gating improve the success-cost frontier over fixed-direction baselines (§5.2); which components drive the gains (§5.3); and do the Two-Source Model's predictions hold on held-out data (§5.4).

Table 2: Full results across 6 environments and 3 backbones. SR (%) and Cost (×base, total deployment tokens normalized by base_only). Best and second-best mean SR per environment. †: methods requiring calibration data. Cost includes all LLM calls: base proposer, gate overhead (CATTS K=5 voting, AUQ confidence query), and rollout calls.

Method	HotpotQA		WebShop		FEVER		TWExpress		Plancraft		APPS	
	SR↑	Cost↓	SR↑	Cost↓	SR↑	Cost↓	SR↑	Cost↓	SR↑	Cost↓	SR↑	Cost↓
Qwen3-4B-Instruct												
CaTS†	93.2 ± 1.4	10.54	30.5 ± 2.2	3.45	48.2 ± 3.6	25.48	96.7 ± 0.8	3.37	20.3 ± 2.8	4.14	66.2 ± 0.0	2.61
SEAG†	67.5 ± 4.8	9.34	28.0 ± 4.9	2.84	47.3 ± 3.2	17.19	97.3 ± 0.3	2.60	22.8 ± 0.8	3.52	66.0 ± 0.0	2.75
CoRefine†	68.2 ± 4.9	9.28	27.5 ± 3.8	2.78	47.8 ± 2.0	17.19	97.5 ± 0.5	2.57	20.8 ± 3.0	3.44	67.5 ± 0.0	2.70
CATTS	68.3 ± 4.6	10.53	16.0 ± 0.5	5.55	32.2 ± 0.3	16.46	97.5 ± 0.5	2.83	23.0 ± 2.0	6.53	60.8 ± 0.0	6.00
AUQ	95.0 ± 0.5	10.29	35.7 ± 8.3	5.83	38.7 ± 4.0	18.04	95.5 ± 0.0	1.95	22.2 ± 1.6	6.64	64.7 ± 1.0	3.28
s1_budget	94.0 ± 0.5	8.27	17.8 ± 2.6	2.91	44.2 ± 2.0	19.21	95.0 ± 0.5	2.02	16.3 ± 2.4	3.02	69.0 ± 1.3	2.65
DIAL	95.2 ± 0.3	8.02	43.8 ± 4.9	2.50	49.8 ± 2.5	16.51	99.0 ± 0.3	1.81	23.3 ± 1.5	3.63	73.0 ± 1.0	2.61
Phi-3.5-mini-Instruct												
CaTS†	68.3 ± 2.8	5.86	41.7 ± 3.8	2.52	19.8 ± 2.0	14.51	68.2 ± 4.2	1.00	13.5 ± 0.5	1.44	30.5 ± 1.8	3.55
SEAG†	88.3 ± 3.1	7.82	36.5 ± 2.8	2.43	13.5 ± 3.0	5.61	92.5 ± 1.5	2.58	14.7 ± 0.8	2.29	27.8 ± 0.8	2.59
CoRefine†	87.7 ± 2.8	7.83	35.8 ± 3.3	2.44	13.7 ± 3.3	5.73	92.7 ± 1.5	2.56	15.2 ± 1.3	2.24	28.5 ± 0.3	2.60
CATTS	42.0 ± 1.5	7.98	28.7 ± 1.9	4.86	12.7 ± 1.3	7.65	86.7 ± 2.3	4.00	13.5 ± 0.0	5.96	27.7 ± 0.0	6.10
AUQ	31.2 ± 1.4	5.47	37.0 ± 2.0	3.11	8.5 ± 2.3	3.18	92.5 ± 1.5	1.82	12.7 ± 0.3	2.24	27.2 ± 0.3	2.03
s1_budget	91.2 ± 2.9	5.26	43.0 ± 4.5	3.89	20.0 ± 5.4	8.45	93.7 ± 0.8	2.05	13.7 ± 0.8	1.73	34.5 ± 0.3	2.93
DIAL	92.3 ± 2.8	5.38	57.3 ± 1.3	2.56	20.5 ± 3.0	7.44	96.7 ± 0.8	1.83	16.8 ± 0.8	1.33	36.8 ± 2.0	2.87
Llama-3.1-8B-Instruct												
CaTS†	94.2 ± 1.3	7.21	36.0 ± 6.3	2.51	56.3 ± 2.3	14.82	35.8 ± 5.4	3.00	22.8 ± 0.3	3.42	55.2 ± 1.3	4.81
SEAG†	94.0 ± 2.0	9.82	27.5 ± 3.9	1.86	53.5 ± 2.9	12.57	76.5 ± 0.9	3.85	21.8 ± 3.5	2.70	57.2 ± 1.6	4.41
CoRefine†	94.2 ± 1.5	9.42	27.5 ± 3.8	1.82	53.5 ± 3.3	12.64	76.0 ± 0.0	3.56	21.5 ± 2.2	2.69	57.2 ± 1.3	4.37
CATTS	94.0 ± 2.0	11.11	11.3 ± 3.8	4.74	14.2 ± 2.8	17.00	49.3 ± 0.0	4.11	27.0 ± 2.9	7.31	52.3 ± 1.0	6.70
AUQ	95.0 ± 1.5	10.54	37.8 ± 4.3	4.54	52.8 ± 2.3	25.08	63.2 ± 0.9	3.58	27.0 ± 1.5	8.21	46.0 ± 0.5	6.11
s1_budget	94.5 ± 0.5	8.08	11.2 ± 0.5	1.87	13.8 ± 1.0	8.56	76.1 ± 3.0	3.05	26.3 ± 0.8	2.53	52.5 ± 0.8	2.33
DIAL	95.5 ± 1.7	7.07	41.7 ± 4.5	1.68	54.7 ± 3.2	18.69	94.8 ± 5.1	2.55	27.0 ± 3.4	2.49	59.7 ± 1.8	4.29

Environments. We conduct evaluations across six environments: HotpotQA [41], APPS [42], WebShop [43], FEVER [44], TWExpress [45], and Plancraft [46]. These cover the full range of the Two-Source Model, from intervention-unsuitable-leaning (FEVER) through mixed (WebShop) to decision-difficulty-leaning (APPS) settings, including extreme rollout properties (TWExpress: rollout-safe; Plancraft: rollout-harmful). Full environment specifications are in Appendix D.1.

Baselines. All methods share the same agent and optimizer T; only the gating decision differs. Six fixed-direction or fixed-budget baselines instantiate a predetermined compute-allocation rule (high uncertainty, low confidence, disagreement, or a fixed token budget triggering additional computation): CaTS [13], SEAG [12], CoRefine [14], CATTS [17], AUQ [25], and s1_budget [15]. Full specifications are in Appendix D.2. We additionally include two bounds for reference (Appendix D.4): *base_only*, which never triggers the optimizer, and *always_trigger*.

Metrics. We use SR (success rate, ↑) and Cost (×base, ↓), total deployment tokens normalized by base_only per episode and including all LLM calls (base proposer, gate overhead, rollout); calibration and exploration overhead are excluded for all methods. A method Pareto-dominates another if SR $\geq$ and Cost $\leq$ with at least one strict inequality. Token cost constants are in Appendix D.3.

5.2 Main Results

DIAL improves the success-cost frontier over fixed-direction baselines. Table 2 reports SR and Cost (×base) across all 6 environments and 3 backbones; the Qwen3-4B Pareto frontier is visualized in Appendix G.6. DIAL achieves the highest SR in 16 of 18 (environment, backbone) cells. Three insights follow. First, *wrong direction is catastrophic, not merely costly*. On Qwen3-4B FEVER, an intervention-unsuitable-leaning environment, CATTS pays 16.46× base for K=5 voting yet reaches only 32.2% SR, below the 37.0% no-trigger baseline: fixed-direction gating in the wrong direction does not merely waste compute, it actively hurts. Second, *learning the right direction yields a better SR-cost trade-off, not just a retuned threshold*. On Qwen3-4B HotpotQA, DIAL trades 1.8 SR points for 1.3× less cost than always_trigger (95.2 vs. 97.0% at 8.02× vs. 10.63×); on WebShop, DIAL matches always_trigger at less than half the cost. The gap is too large to close by retuning thresholds without changing direction. Third, *the win generalizes across backbones*. The same DIAL pipeline takes 6/6 cells on both Qwen3-4B and Phi-3.5-mini and 4/6 on Llama-3.1-8B, even though entropy–utility correlations flip sign across model families (§3.2). Fixed-direction baselines

instead break per-backbone: AUQ on HotpotQA falls from 95.0% (Qwen3-4B) to 31.2% (Phi-3.5). The experiment in Appendix G.7 using Llama 4 Maverick shows the same pattern. On both WebShop and APPS, DIAL attains the highest SR among adaptive methods at the lowest cost.

5.3 Ablation Studies

The main results show that DIAL works; we now ask *why*. We ablate direction along two axes: its sign (direction reversal) and the gate's capacity to exploit it (gate complexity). We then characterize how DIAL's trigger rate adapts across rollout-headroom regimes and explain why fewer rollouts can improve accuracy. The regularizer ablation appears in Appendix F.3.

Direction is a first-order determinant of gate quality. To isolate the effect of direction, we reverse the sign of all learned weights in DIAL's fitted gate while keeping everything else unchanged, constructing an adversarially wrong-direction gate that preserves ranking structure but inverts trigger semantics. Table 3 shows that on strong-signal environments with $|\rho^*| > 0.4$, reversal collapses SR by 23–37%; on weak-signal Plancraft ($|\rho^*|$=0.016) the effect is negligible. The damage scales with signal strength: wrong direction is not a calibration error but a structural failure. This instantiates the wrong-direction harm result of Appendix B.2: a sharper gate fires precisely on the states where computation has negative value.

Table 3: Effect of reversing DIAL's learned direction. ΔSR: reversed − original. $|\rho^*|$ is the absolute Spearman correlation between the dominant signal and optimizer utility in each environment. Degradation scales with $|\rho^*|$.

Environment	DIAL SR	Reversed SR	Δ SR	$\lvert\rho^*\rvert$
FEVER	49.8%	13.0%	−36.8	0.619
HotpotQA	95.2%	58.2%	−37.0	0.494
WebShop	43.8%	20.6%	−23.2	0.444
APPS	73.0%	70.5%	−2.5	0.339
TWExpress	99.0%	96.2%	−2.8	0.290
Plancraft	23.3%	20.5%	−2.8	0.016

Gate capacity does not substitute for direction. A natural objection is that DIAL's gains come from gate simplicity rather than learned direction. With correct direction, a logistic gate, a 2-layer MLP, and a hidden-state probe (2560-d) all reach ≈95% SR on HotpotQA: capacity adds <1% (Figure 3). With wrong direction, the ordering inverts: the MLP drops to 45.3%, *below* the wrong-direction logistic at 62.0%. Direction consistently outweighs capacity, which is why DIAL's minimal linear gate suffices.

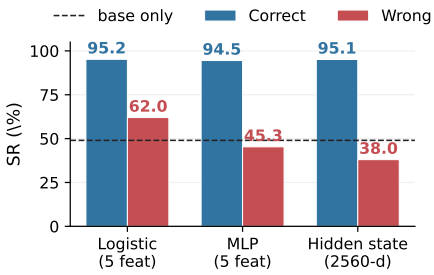


Figure 3: Gate complexity ablation on HotpotQA (Qwen3-4B). With correct direction all gates reach ≈95%; with wrong direction.

The universal feature pool carries most of the gain. Table 4 removes the LLM-proposed features and refits the gate on the five universal features alone. SR changes by less than two points in five of six environments. FEVER is the exception with a 9.1-point drop: there the proposer finds signals such as `has_claim` that the universal pool lacks. The LLM layer is therefore an extensibility mechanism for environments whose structure the universal pool does not capture, not the source of DIAL's gains.

Table 4: LLM feature ablation on Qwen3-4B. SR (%) with and without LLM-proposed features.

Environment	w/ LLM	w/o LLM	Δ
HotpotQA	95.2	93.8	+1.4
WebShop	43.8	42.7	+1.1
FEVER	49.8	40.7	+9.1
TWExpress	99.0	97.2	+1.8
Plancraft	23.3	23.4	−0.1
APPS	73.0	73.0	+0.0

Trigger rate adapts to rollout headroom. In rollout-harmful environments, baselines that fire aggressively pay a price beyond efficiency: on Plancraft (Δ=−7.0), s1_budget keeps triggering and falls below *base_only* (16.3% vs. 29.8%, see Appendix D.4); more compute makes performance *worse* than no gate at all. DIAL avoids this trap without ever observing the headroom: its trigger rate adapts from 73% on rollout-safe TWExpress, where almost any trigger helps, down to <20% on Plancraft, with a within-episode decay from 49% at step 0 to under 20% in late steps as the gate learns that further rollouts only add harm. This adaptive magnitude emerges from the per-environment signed weights alone: direction discovery already encodes whether the environment rewards or penalizes additional computation. The full per-step analysis is in Appendix G.8.

Why fewer rollouts can improve SR. The optimizer is not a monotone improvement operator. In intervention-unsuitable states, rollout evaluation starts from the same unreliable context as the base

policy and can amplify misleading evidence, invalid intermediate states, or compounding trajectory errors. Avoiding such negative-utility interventions therefore improves accuracy, not merely efficiency. DIAL's gain in low-headroom regimes comes from learning when *not* to invoke the optimizer, while still triggering in decision-difficult states where rollouts expose useful alternatives.

5.4 Two-Source Model Verification

The empirical landscape (§3.2) shows direction varies but does not pin down the Two-Source Model (§3.3) as the cause. We test the model with three complementary experiments that each rule out a different class of alternative explanation: within-episode temporal dynamics (P1, observational), within-task causal manipulation of the information mixture (P2, interventional), and cross-environment alignment of the dominant signal with the dominant state type (P3, population-level).

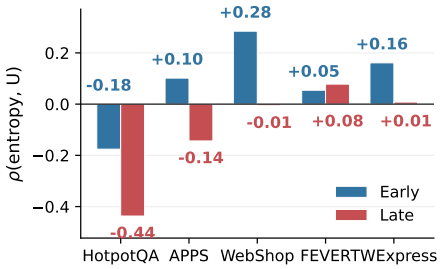


Figure 4: P1: ρ(entropy, U) shifts from early (blue, step $\leq$ median) to late (red); bars show Spearman ρ. Plancraft omitted ($|\rho|$=0.016).

P1: Temporal dynamics. The Two-Source Model predicts that ρ should shift downward after the early phase of the episode: as the agent gathers information, many Type D states are resolved early, so later-step uncertainty increasingly reflects residual Type I states. Figure 4 supports this pattern in environments with strong temporal signal. In HotpotQA, ρ shifts from -0.176 (early) to -0.437 (late). In WebShop, the strong positive early-phase signal (ρ=$+0.285$) vanishes in the late phase. In APPS, ρ shifts from non-significant positive ($+0.102$) to weak negative (-0.144). Full results are in Appendix G.3.

P2: Controlled information manipulation. P1 verifies a within-episode dynamic; we now intervene on p_I directly for *causal* evidence. We construct two interventions on HotpotQA: **InfoPoor** returns only the first sentence of each retrieved paragraph (increasing the Type I mixture), and **InfoRich** prepends gold-standard evidence to the prompt (reducing one major source of intervention-unsuitability, namely information poverty, and shifting states toward Type D). Appendix G.4 details the construction. Table 5 reports the result: restricting evidence makes `step_count` dominant (ρ=-0.608) while entropy is weak; injecting gold evidence makes entropy dominant (ρ=$+0.311$) while `step_count` weakens. As the only test in this section that intervenes on the latent mixture rather than observing it, P2 confirms that signal identity is a function of information structure.

Table 5: Controlled information manipulation on HotpotQA (Qwen3-4B): availability shifts which signal dominates, providing causal evidence for the Two-Source Model.

Condition	Base SR	ρ(ent.)	ρ(step)	DIAL SR
Original	49.0%	-0.041	-0.494	95.2%
InfoPoor	44.0%	$+0.119$	$\mathbf{-0.608}$	94.0%
InfoRich	82.0%	$\mathbf{+0.311}$	-0.147	88.3%

P3: Signal identity alignment. P2 manipulated information within one task; P3 asks whether the predicted alignment also holds *across* tasks. The Two-Source Model further predicts that the strongest signal in each environment should reflect its dominant state type: information-sufficiency proxies in Type I, decision-complexity proxies in Type D. Table 6 confirms this: `step_count` dominates in Type I environments, while `num_available_actions` and other choice-related features dominate in WebShop. The LASSO coefficient matrix and its interpretation as Two-Source proxies are in Appendix F.1.

Table 6: Strongest signal per environment aligns with the dominant regime: information-sufficiency proxies in Type I, decision-complexity in Type D.

Env	**Regime**	**Strongest Signal**	$\lvert\rho\rvert$	**Interpretation**
HotpotQA	Type I	step_count	0.494	Info accumulation
FEVER	Type I	step_count	0.619	Info accumulation
TWExpress	Type I	step_count	0.477	Exploration depth
WebShop	Mixed	num_avail_actions	0.444	Decision-space size
APPS	Type D	step_count	0.339	Debugging depth
Plancraft	Weak	has_output	0.162	Rollout harmful

The three tests are observational, interventional, and population-level respectively, and rule out distinct confounds. Cross-setting ρ consistency at fixed backbone is reported in Appendix G.5, and multi-backbone verification in Appendix G.6 provides additional robustness checks.

6 Limitations

FEVER is DIAL's weakest setting: useful rollouts concentrate in the earliest steps, which random exploration may miss, and on one backbone a fixed-direction baseline matches or slightly exceeds DIAL on the SR-cost frontier. Curriculum-based exploration is a natural extension. DIAL incurs a one-time exploration cost per (environment, backbone), amortized over subsequent inference; in exchange, it removes the need for manual direction tuning. DIAL also learns one gate per (environment, backbone) pair and does not transfer it to new settings. This is unavoidable under our own finding: a signal that predicts rollout benefit in one setting can predict rollout harm in another, so a gate fitted in one setting cannot be reused elsewhere without recalibration. DIAL therefore requires knowing which setting the agent is in and being able to run a calibration phase there. Both conditions can fail in practice: a coding agent may move from locating a bug to reading test output within a single trajectory, with no clear point at which to switch gates. A natural next step is to pool exploration data across settings and learn a single utility predictor conditioned on the state. We exclude end-to-end RL gating controllers, whose reward-cost objective and online training can confound direction-attribution.

7 Conclusion

Adaptive gating for LLM agents is not simply difficulty estimation. It requires learning when the available test-time optimizer can improve the final outcome. We show that the relationship between common gating signals and rollout utility can reverse across environments and model backbones. Gates that assume the wrong direction can worsen performance by selecting states where additional computation is harmful. Our two-source model accounts for these reversals by distinguishing compute need from compute suitability. Decision-difficult states can benefit from comparing alternative actions, while intervention-unsuitable states lack the conditions for reliable rollout-based improvement. Based on this distinction, DIAL uses signal-agnostic counterfactual exploration and a sparse feature-based gate to learn utility direction for each (environment, backbone) setting. Across six environments and three backbones, DIAL demonstrates that learning utility direction enables more effective compute allocation, improving the overall success-cost trade-off over fixed-direction baselines. These findings highlight that effective compute allocation depends not only on uncertainty, but also on whether additional computation can turn that uncertainty into better decisions.

Acknowledgments and Disclosures

This work was partially supported by the NSF under grants IIS-2528540, IIS-2334193, IIS-2340346 and CNS-2426514. This work also used computational resources provided through NSF ACCESS grant CIS260048 and CIS260811. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. The authors declare no competing interests.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[2] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

[3] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*, 2024.

[4] Jiazheng Li, Lu Yu, Qing Cui, Zhiqiang Zhang, Jun Zhou, Yanfang Ye, and Chuxu Zhang. Mass: Mathematical data selection via skill graphs for pretraining large language models. In *ICML*, 2025.

[5] Ziming Li, Xiaoming Wu, Zehong Wang, Jiazheng Li, Yijun Tian, Jinhe Bi, Yunpu Ma, Yanfang Ye, and Chuxu Zhang. Graph is a substrate across data modalities. *CoRR*, 2026.

[6] Qiwei Ma, Chunping Qiu, Xinjun Cheng, Xiaoyu Zhang, Puhong Duan, Ke Yang, Xudong Kang, and Shutao Li. Multimodal large language models for remote sensing image understanding: Domain-specific or general-purpose? *CoRR*, 2026.

[7] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

[8] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *EMNLP*, 2023.

[9] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, 2024.

[10] Zehong Wang, Fang Wu, Hongru Wang, Xiangru Tang, Bolian Li, Zhenfei Yin, Yijun Ma, Yiyang Li, Weixiang Sun, Xiusi Chen, and Yanfang Ye. Why reasoning fails to plan: A planning-centric analysis of long-horizon decision making in LLM agents. *CoRR*, 2026.

[11] King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for LLM agents. *CoRR*, 2025.

[12] Sungjae Lee, Hyejin Park, Jaechang Kim, and Jungseul Ok. Semantic exploration with adaptive gating for efficient problem solving with language models. In *ACL*, 2025.

[13] Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Cats: Calibrated test-time scaling for efficient llm reasoning. In *ICLR*, 2026.

[14] Chen Jin, Ryutaro Tanno, Tom Diethe, and Philip Teare. Corefine: Confidence-guided self-refinement for adaptive test-time compute. *CoRR*, 2026.

[15] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *EMNLP*, 2025.

[16] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *ACL*, 2025.

[17] Nicholas Lee, Lutfi Eren Erdogan, Chris Joseph John, Surya Krishnapillai, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Agentic test-time scaling for webagents. *CoRR*, 2026.

[18] Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. Diffadapt: Difficulty-adaptive reasoning for token-efficient LLM inference. *CoRR*, 2025.

[19] Yubo Zhu, Dongrui Liu, Zecheng Lin, Wei Tong, Sheng Zhong, and Jing Shao. The LLM already knows: Estimating llm-perceived question difficulty via hidden representations. In *EMNLP*, 2025.

[20] Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. In *EMNLP*, 2025.

[21] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM learns when to think. *CoRR*, 2025.

[22] Edward H Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society*, 1951.

[23] Jingwei Wang, Jianshan He, Weidi Xu, Ruopeng Li, and Wei Chu. Learning to discover various simpson's paradoxes. In *KDD*, 2023.

[24] Aman Sharma and Paras Chopra. Think just enough: Sequence-level entropy as a confidence signal for LLM reasoning. *CoRR*, 2025.

[25] Jiaxin Zhang, Prafulla Kumar Choubey, Kung-Hsiang Huang, Caiming Xiong, and Chien-Sheng Wu. Agentic uncertainty quantification. *CoRR*, 2026.

[26] Pranjal Aggarwal and Sean Welleck. L1: controlling how long A reasoning model thinks with reinforcement learning. *CoRR*, 2025.

[27] Davide Paglieri, Bartlomiej Cupial, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for LLM agents. *CoRR*, 2025.

[28] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, 2021.

[29] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, 2023.

[30] Linwei Tao, Yi-Fan Yeh, Minjing Dong, Tao Huang, Philip Torr, and Chang Xu. Revisiting uncertainty estimation and calibration of large language models. *CoRR*, 2025.

[31] Juyeon Heo, Miao Xiong, Christina Heinze-Deml, and Jaya Narain. Do llms estimate uncertainty well in instruction-following? In *ICLR*, 2025.

[32] Yasin Abbasi-Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your LLM: iterative prompting for estimating epistemic uncertainty. In *NeurIPS*, 2024.

[33] Jiatan Huang, Mingchen Li, Ziming Li, Sunjae Kwon, Hong Yu, and Chuxu Zhang. Counterfactual graph for multi-agent LLM calibration. *CoRR*, 2026.

[34] Stuart J. Russell and Eric Wefald. *Do the right thing - studies in limited rationality*. MIT Press, 1991.

[35] Nicolò De Sabbata, Theodore R. Sumers, and Thomas L. Griffiths. Rational metareasoning for large language models. *CoRR*, 2024.

[36] Qwen Team. Qwen3 technical report. *CoRR*, 2025.

[37] Llama Team. The llama 3 herd of models. *CoRR*, 2024.

[38] Microsoft. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, 2024.

[39] Judea Pearl. Comment: Understanding simpson's paradox. In *Probabilistic and Causal Inference: The Works of Judea Pearl*. 2022.

[40] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 2009.

[41] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

[42] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Andy Zou, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *NeurIPS*, 2021.

[43] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

[44] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *NAACL*, 2018.

[45] Peter Jansen and Marc-Alexandre Côté. Textworldexpress: Simulating text games at one million steps per second. In *EACL*, 2023.

[46] Gautier Dagan, Frank Keller, and Alex Lascarides. Plancraft: an evaluation dataset for planning with LLM agents. *CoRR*, 2024.

[47] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *CoRR*, 2025.

[48] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, 2025.

[49] Alex Graves. Adaptive computation time for recurrent neural networks. *CoRR*, 2016.

[50] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *CoRR*, 2017.

[51] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *ACL*, 2024.

[52] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[53] David Raposo, Samuel Ritter, Blake A. Richards, Timothy P. Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *CoRR*, 2024.

[54] Xuan Shen, Hangyu Zheng, Yifan Gong, Zhenglun Kong, Changdi Yang, Zheng Zhan, Yushu Wu, Xue Lin, Yanzhi Wang, Pu Zhao, and Wei Niu. Sparse learning for state space models on mobile. In *ICLR*, 2025.

[55] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *CoRR*, 2022.

[56] Eric Horvitz. Reasoning about beliefs and actions under computational resource constraints. In Laveen N. Kanal, Tod S. Levitt, and John F. Lemmer, editors, *UAI*, 1987.

[57] Falk Lieder and Thomas L Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 2020.

[58] Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *ICML*, 2024.

[59] Kou Misaki, Yuichi Inoue, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling LLM inference-time compute with adaptive branching tree search. *CoRR*, 2025.

[60] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

[61] Bradley C. A. Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, 2024.

[62] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

[63] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*, 2024.

[64] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *CoRR*, 2024.

[65] Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: advanced reasoning and learning for autonomous AI agents. *CoRR*, 2024.

[66] Zehong Wang, Yijun Ma, Connor R. Schmidt, Tianyi Ma, Weixiang Sun, Ziming Li, Xiaoguang Guo, Chuxu Zhang, Matthew Webber, and Yanfang Ye. Mdforge: Agentic molecular dynamics pipeline design under sparse simulator feedback. *CoRR*, 2026.

[67] Jiatan Huang, Zheyuan Zhang, Kaiwen Shi, Yanfang Ye, and Chuxu Zhang. Evolverouter: Co-evolving routing and prompt for multi-agent question answering. *CoRR*, 2026.

A Extended Related Work

A.1 Adaptive Compute: Reasoning vs. Agent Settings

The adaptive test-time compute literature spans two distinct settings. In the *reasoning* setting, methods optimize single-turn chain-of-thought generation on homogeneous benchmarks (MATH, GSM8K, GPQA). Signal-based approaches use token-level confidence or entropy to decide when to extend thinking [12–14, 24, 15, 16]. RL-based methods learn think/no-think policies through reinforcement learning [20, 21, 26]. Hidden-state probing methods train lightweight classifiers on internal representations to estimate task difficulty [18, 19], sometimes assuming a universal U-shaped entropy pattern that our cross-(environment, backbone) evidence disproves. Because reasoning benchmarks share similar task semantics, signal direction happens to be consistent across tasks, and the fixed-direction assumption is never tested.

In the *agent* setting, methods operate in multi-step interactive environments with external optimizers. Vote-based methods [17, 14] sample multiple candidate actions and trigger evaluation when disagreement is high. Verbalized-uncertainty methods [25] use self-reported confidence for agent reflection but assume fixed direction. Episode-level approaches [27] learn when to invoke a planner, and agentic RL methods [47] use entropy-based adaptive rollout at tool-call steps. All agent-setting methods inherit the fixed-direction assumption from the reasoning literature without verifying it. Our work is the first to systematically study signal semantics across heterogeneous agent environments *and backbones*, where we find that the direction reverses along both axes.

A.2 Relation to RLVR and General-Model Training

Reinforcement learning with verifiable rewards, or RLVR, trains the base policy to decide how much reasoning to spend on each problem [48, 26, 20, 21]. DIAL addresses a different decision. It keeps the base policy and the optimizer frozen and learns a separate gate that predicts whether invoking the optimizer at the current state will improve the final outcome. The two operate at different levels of the system. Consider an agent debugging a repository. An RLVR-trained policy may reason longer about which file to inspect before acting. DIAL decides whether the system should launch an external procedure at this step, such as a search, a verification pass, or a set of rollouts. One method shapes the computation inside the policy; the other allocates computation outside it. The two are therefore complementary.

Our findings also bear on how RLVR-style methods are supervised. Several RL-based methods learn to skip or shorten reasoning on problems the model already solves and to think longer on harder ones [20, 21]. This again assumes that difficulty implies compute need. Section 3 shows that the assumption fails in heterogeneous agent settings. The supervision target should instead be the counterfactual utility of additional computation: whether extra compute changes the outcome relative to acting immediately. DIAL's exploration phase produces exactly this label at the state level. Pooling these labels across environments and backbones would allow training a single utility predictor conditioned on the state, or a routing head inside a general model, in place of one gate per setting. This would remove the per-setting calibration requirement discussed in Section 6. We leave this integration to future work.

A.3 Adaptive Computation in Neural Networks

The idea of adapting computation to input difficulty predates LLMs. Adaptive Computation Time [49] allows RNNs to learn how many computational steps to take per input. Early exit methods [50, 51] attach classifiers at intermediate transformer layers and exit early for easy inputs. Mixture-of-Experts architectures [52, 53] activate only a subset of parameters per token, adapting capacity rather than depth. Sparse learning for efficient state space models [54] similarly reduces per-forward-pass cost. These methods adapt computation within a single forward pass. Our work operates at a different granularity: we adapt whether to invoke an external optimizer at each decision step, and the key challenge is not input difficulty but the *direction* of the signal-utility relationship.

A.4 Confidence Calibration vs. Utility Calibration

Classical confidence calibration. Kadavath et al. [55] show that larger models are well-calibrated on multiple-choice tasks and can estimate P(True) for their own outputs. Tao et al. [30] and Heo et al. [31] find that calibration quality varies substantially across task types, providing converging evidence that uncertainty semantics are domain-dependent. Abbasi-Yadkori et al. [32] propose decoupling epistemic and aleatoric uncertainty in LLM outputs, a distinction conceptually related to our Type I/Type D decomposition.

From confidence-to-correctness to confidence-to-utility. These works ask whether confidence is calibrated to the *correctness* of an action. Adaptive gating, however, requires a different alignment: whether a confidence-derived signal predicts the *value of additional computation*. We call this latter target *utility calibration*. The distinction matters because even a perfectly correctness-calibrated confidence may be insufficient for gating: a gate triggers only when extra computation improves the outcome, which depends on the environment and the optimizer, not just on whether the current action is right.

Our contribution. We extend these findings in two directions. First, we make the target distinction explicit: gating is a utility-calibration problem, not a correctness-calibration one. Second, we show that this strengthens the domain-dependence picture empirically: not only does calibration *quality* vary across tasks, but the *sign* of the signal-to-utility relationship reverses across (environment, backbone) pairs, so a gate using correctness-calibrated confidence under the wrong utility direction will systematically select harmful states.

A.5 Metareasoning and Value of Computation

Rational metareasoning [34, 56] formalizes the value of computation (VOC) as a guide for allocating cognitive resources. The classical framework assumes VOC ≥ 0 because an agent can always ignore computation results. Lieder and Griffiths [57] generalize this to resource-rational analysis, modeling cognition as optimal use of limited computational resources. Sabbata et al. [35] apply metareasoning to LLMs, analyzing when to extend chain-of-thought reasoning. Our work shows that the classical VOC ≥ 0 assumption is violated in the agent setting; we formalize this in Appendix B.3.

A.6 Simpson's Paradox in Machine Learning

Simpson's paradox [22, 39] occurs when a trend present in subgroups reverses upon aggregation. In machine learning, this phenomenon has been documented in fairness [23], causal inference, and treatment effect estimation. Our two-source model identifies a new instance: within Type I states, entropy negatively predicts utility; within Type D states, entropy positively predicts utility; but the marginal correlation depends on the mixture proportion p_I, which is itself an (environment, backbone)-indexed quantity, since a state's status as Type I or Type D depends on what the agent already encodes, so the same environment can sit on opposite sides of the reversal threshold p_I^* under different LLMs. This makes direction reversal a structural consequence of aggregating heterogeneous uncertainty sources, and predicts both the across-environment and across-backbone reversals reported in Table 1.

A.7 Search-Based Optimizers

A growing body of work targets *what* the test-time compute procedure does, rather than *when* it should be invoked. Tree-search and lookahead methods organize multi-step exploration through value propagation and AlphaZero-style rollouts [7, 3, 8, 10, 58, 59]. Repeated independent sampling with majority aggregation exploits smooth coverage gains as the sample budget grows [60, 61], while verifiers trained over intermediate reasoning steps rerank candidates or steer search [62, 63]. At a higher level, compute-allocation analyses and reasoning models study how a fixed budget should be distributed across these mechanisms or absorbed into long latent traces [9, 48]. Closer to our setting, agent-specific work adapts these mechanisms to interactive, multi-step environments by applying tree search to web tasks [64], coupling MCTS with preference learning over agent trajectories [65, 66], and characterizing scaling behavior for agent rollouts [11, 67]. All of this work specifies the rollout procedure that runs at a triggered step. Our contribution is orthogonal: a per-step gate decides

whether such a procedure should run, and the gate composes with any of the above as a drop-in test-time optimizer.

B Theoretical Material

This appendix collects the theoretical material referenced in Section 3: the formal statement and proof of the necessity-of-direction-discovery result (§B.1), the wrong-direction damage quantification (§B.2), and the VOC negativity analysis (§B.3).

B.1 Insufficiency of Uncertainty Signals: Statement and Proof

We formalize the central theoretical claim of the paper: any adaptive-compute policy that depends on an uncertainty signal σ alone cannot deliver non-negative value of computation across heterogeneous (environment, backbone) settings.

Proposition 1 (Insufficiency of Uncertainty Signals)**.** *Let $\sigma : \mathcal{S} \to \mathbb{R}$ be any uncertainty signal (e.g., token entropy, calibrated confidence, vote disagreement, verbalized uncertainty). Let $\mathrm{VOC}(s) = \mathbb{E}[R(\tau) \mid a{=}T(s)] - \mathbb{E}[R(\tau) \mid a{=}\pi(s)]$ denote the value of computation at state s. Then σ is* not a sufficient statistic *for* VOC *in the following sense: there exist (environment, backbone) settings $\mathcal{E}_1, \mathcal{E}_2$ and states $s_1 \in \mathcal{E}_1$, $s_2 \in \mathcal{E}_2$ with $\sigma(s_1) = \sigma(s_2)$ such that $\mathrm{VOC}(s_1) > 0$ and $\mathrm{VOC}(s_2) < 0$. Consequently, no policy of the form $\pi_\sigma(s) = g(\sigma(s))$ for any measurable $g : \mathbb{R} \to \{0, 1\}$ can simultaneously satisfy $\mathrm{SR}(\pi_\sigma, \mathcal{E}_1) \geq \mathrm{SR}(\mathrm{base}, \mathcal{E}_1)$ and $\mathrm{SR}(\pi_\sigma, \mathcal{E}_2) \geq \mathrm{SR}(\mathrm{base}, \mathcal{E}_2)$.*

The proposition strengthens the earlier "no fixed-direction gate works" formulation in two ways. First, it rules out *any* measurable $g(\sigma)$, not just monotone or threshold-based policies, so non-monotone or U-shaped calibrators are also covered. Second, it identifies the underlying obstruction as an information-theoretic property of σ itself, one that survives any post-processing of the signal.

Proof. By the two-source decomposition of §3.3, $\mathrm{VOC}(s)$ depends on a latent type $\tau(s) \in \{I, D\}$ that is not measurable from $\sigma(s)$ alone:

$$\mathrm{VOC}(s) \;=\; \begin{cases} -\alpha\,\sigma(s) + \varepsilon_I & \text{if } \tau(s) = I, \\ +\beta\,\sigma(s) + \varepsilon_D & \text{if } \tau(s) = D, \end{cases}$$

with $\alpha, \beta > 0$. Choose any signal value σ_0 in the support of both Type I and Type D states within their respective settings. Then there exist $s_1 \in \mathcal{E}_1$ with $\tau(s_1) = D$ and $\sigma(s_1) = \sigma_0$, giving $\mathbb{E}[\mathrm{VOC}(s_1) \mid \sigma_0] = \beta\sigma_0 > 0$ for sufficiently large σ_0, and $s_2 \in \mathcal{E}_2$ with $\tau(s_2) = I$ and $\sigma(s_2) = \sigma_0$, giving $\mathbb{E}[\mathrm{VOC}(s_2) \mid \sigma_0] = -\alpha\sigma_0 < 0$. Hence the conditional VOC is not a function of σ alone, establishing insufficiency.

For the policy claim, fix any $g : \mathbb{R} \to \{0, 1\}$. The expected one-step reward gain over base from playing π_σ is $\mathbb{E}_{s\sim\mathcal{E}}[g(\sigma(s)) \cdot \mathrm{VOC}(s)]$. Decomposing by latent type and writing $p_I(\mathcal{E})$ for the Type I fraction,

$$\begin{aligned} \mathbb{E}_{\mathcal{E}}[g(\sigma) \cdot \mathrm{VOC}] = &-\alpha\, p_I(\mathcal{E})\, \mathbb{E}_I[g(\sigma)\sigma] \\ &+ \beta\,(1 - p_I(\mathcal{E}))\, \mathbb{E}_D[g(\sigma)\sigma] + \mathrm{const}. \end{aligned}$$

Whenever g is non-trivially correlated with σ (which any informative gate must be), the two terms have opposite signs as p_I varies. In particular, for any fixed g there exist $\mathcal{E}_1, \mathcal{E}_2$ with $p_I(\mathcal{E}_1)$ and $p_I(\mathcal{E}_2)$ on opposite sides of the reversal threshold $p_I^* = \beta/(\alpha + \beta)$ of §3.3 such that the expected gain has opposite signs. The setting where it is negative violates the SR constraint relative to base, completing the argument. □

Corollary (necessity of direction discovery). The original "no fixed-direction gate works" result is the restriction of Proposition 1 to threshold policies $g(\sigma) = \mathbf{1}[d \cdot \sigma > \theta]$, $d \in \{+1, -1\}$. In that case both cases of the proof above collapse to: g's sign of correlation with σ is fixed at d, but VOC's sign of correlation with σ flips with p_I, so on the wrong side of p_I^* the gate triggers exactly when $\mathrm{VOC} < 0$ and incurs systematic harm.

What recovers sufficiency. The proposition pinpoints what an adaptive-compute method must condition on: any feature $\phi(s)$ that is informative about the latent type $\tau(s)$ recovers a sufficient

statistic for VOC, since $\mathrm{VOC}(s) = f(\sigma(s), \tau(s))$ is by construction a function of (σ, ϕ). DIAL's multi-feature gate (§4) is the minimal procedure that operationalizes this recovery: it augments σ with auxiliary universal features that empirically correlate with τ (e.g., `step_count` as a proxy for information accumulation, `num_available_actions` as a proxy for decision optionality) and lets the data identify which of them carry information about τ in each (environment, backbone) setting.

B.2 Wrong-Direction Damage: The Calibration-Worsens Mechanism

Table 3 is the empirical statement of the headline claim: a gate with reversed weights is, by construction, a more precisely calibrated *harm-targeting* mechanism, and the resulting SR collapse scales with signal strength $|\rho^*|$. The decomposition predicts the relationship quantitatively. By the wrong-direction lemma in App B.1,

$$\mathrm{SR}(g_{-d^*}, \mathcal{E}) - \mathrm{SR}(\mathrm{base}, \mathcal{E}) \;\approx\; -c \cdot |\rho^*(\mathcal{E})|,$$

where c is a constant that depends on the optimizer's harm magnitude. Empirically this scaling is clean: HotpotQA ($|\rho^*|$=0.494) loses 37.0%, FEVER ($|\rho^*|$=0.619) loses 36.8%, WebShop ($|\rho^*|$=0.444) loses 23.2%, while Plancraft ($|\rho^*|$=0.016) loses only 2.8%; the slope of damage in $|\rho^*|$ is the slope of ranking sharpness, since sharper-ranking gates concentrate their firing on higher-$|\sigma|$ states which, when the direction is wrong, are exactly the harmful ones.

Two environments show only minimal damage under reversal, both consistent with the same prediction (small $|\rho^*|$ or rollout-safe optimizer):

TWExpress is rollout-safe (Δ=+31.8). The optimizer rarely harms performance regardless of when it is triggered, so both directions achieve near-perfect SR (99.0%).

APPS shows a small negative ΔSR (-2.5%). This is consistent with its unstable signal direction across backbones (Table 1): Qwen3 shows positive ρ, Llama-3.1 shows negative ρ, and Phi-3.5 is non-significant. When the signal is this unstable, direction reversal has minimal impact and small fluctuations can favor either direction.

B.3 VOC Non-Negativity Scope

In the classical VOC framework [34], $\mathrm{VOC}(T, s) = \mathbb{E}[\max(V(a_T), V(a_{\mathrm{base}}))] - V(a_{\mathrm{base}}) \geq 0$ because the agent can *disregard* the computation result if unfavorable. Our setting violates this: when the agent uses an evaluator/optimizer T, the evaluator's assessment *is* the agent's decision. The agent cannot separately evaluate the evaluator's output because the evaluator *is* the evaluation mechanism.

Under evaluator-executor identity, VOC can be negative: $\mathrm{VOC}(T, s) = \mathbb{E}[V(a_T(s))] - V(a_{\mathrm{base}}(s))$. If T systematically selects worse actions in Type I states, $\mathrm{VOC} < 0$ for those states. This explains why wrong-direction gating is not merely wasteful ($\mathrm{VOC} = 0$) but actively harmful ($\mathrm{VOC} < 0$), and why *sharper-ranking* gates fire more precisely on these $\mathrm{VOC} < 0$ states, so ranking improvements amplify rather than mitigate the harm (Table 3 and App B.2). The classical VOC framework's $\mathrm{VOC} \geq 0$ guarantee fails for adaptive compute in agent settings precisely because the agent cannot "disregard" the optimizer's output, and any policy that depends only on σ (which cannot identify Type I states) will trigger T on a non-trivial fraction of $\mathrm{VOC} < 0$ states.

C DIAL Implementation Details

This appendix expands on DIAL's implementation: hyperparameters and training details (§C.1), universal feature definitions (§C.2), the optional LLM feature layer (§C.3), online adaptation for drifting environments (§C.4), and the LLM prompt template (§C.5).

C.1 Hyperparameters and Training Details

Phase 1 runs for $N_{\text{explore}} = 50$ episodes with $\varepsilon_{\text{explore}} = 0.5$. Before fitting in Phase 3, all features in ϕ_{cand} are standardized to zero mean and unit variance on $\mathcal{D}$, ensuring that the interpretable

Algorithm 1 DIAL: Direction-Informed Adaptive Learning

Require: Environment $\mathcal{E}$, agent policy π, optimizer T, universal feature set ϕ_{univ}, LLM $\mathcal{M}$, exploration budget N_{explore}, exploration rate $\varepsilon_{\text{explore}}$, gate threshold τ
Ensure: Deployed gate $g : \mathcal{S} \to \{0, 1\}$

// Phase 1: Explore

$\mathcal{D} \leftarrow \emptyset$
for episode $= 1, \ldots, N_{\text{explore}}$ **do**
  Run π in $\mathcal{E}$; at each step t, invoke T with probability $\varepsilon_{\text{explore}}$ independent of state
  Record (s_t, U_t) pairs into $\mathcal{D}$
end for

// Phase 2: Reason

$(p_{\mathcal{E}}, \mathcal{H}) \leftarrow \mathcal{M}(\text{summary}(\mathcal{D}))$ ▷ env profile + feature hypotheses
$\phi_{\text{cand}} \leftarrow \phi_{\text{univ}} \cup \text{extract}(\mathcal{H})$
Recompute $\phi_{\text{cand}}(s)$ for all $s \in \mathcal{D}$

// Phase 3: Learn

Standardize ϕ_{cand} on $\mathcal{D}$
Select C via 5-fold CV on held-out log-loss
Fit ℓ_1-regularized logistic regression:

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w}, b} \sum_{(\phi, U) \in \mathcal{D}} \ell\big(\sigma(\mathbf{w}^\top \phi + b),\, U\big) + \tfrac{1}{C}\|\mathbf{w}\|_1 \tag{5}$$

$\phi \leftarrow$ features with non-zero w_i^*
$g(s) \leftarrow \mathbf{1}[\sigma(\mathbf{w}^{*\top} \phi(s) + b^*) > \tau]$
return g

weight signs discussed in Section 4.3 are not confounded by scale differences across heterogeneous features. We induce sparsity using the ℓ_1-regularized logistic regression of Eq. (5), with ℓ the binary cross-entropy loss and regularization strength C selected by 5-fold cross-validation over $C \in \{0.01, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0\}$, optimizing held-out log-loss. Features with zero coefficient after fitting are dropped from the deployed gate. The decision threshold τ in Eq. (4) is selected during calibration on $\mathcal{D}$ and stored with the fitted gate; a fixed τ=0.5 is used only in controlled ablations that hold the threshold constant to isolate the component under study (e.g., Appendix F.3).

Operating threshold. Because τ is selected per setting, the operating points differ substantially across settings. Table 7 reports the median, minimum, and maximum selected τ across seeds for all 18 configurations. Values range from 0.050 to 0.950 with an overall median of 0.344, so no single fixed threshold suits all settings.

Table 7: Selected operating threshold τ per (environment, backbone), reported as median [min, max] across seeds.

Env	Qwen3-4B	Phi-3.5-mini	Llama-3.1-8B
HotpotQA	0.105 [0.050, 0.197]	0.399 [0.270, 0.509]	0.307 [0.215, 0.307]
WebShop	0.160 [0.105, 0.215]	0.270 [0.197, 0.344]	0.491 [0.197, 0.509]
FEVER	0.270 [0.142, 0.399]	0.491 [0.436, 0.509]	0.234 [0.197, 0.252]
TWExpress	0.050 [0.050, 0.068]	0.344 [0.087, 0.362]	0.454 [0.307, 0.491]
Plancraft	0.289 [0.215, 0.381]	0.730 [0.730, 0.950]	0.950 [0.454, 0.950]
APPS	0.381 [0.362, 0.546]	0.491 [0.399, 0.564]	0.362 [0.307, 0.436]

To quantify how the threshold affects the trade-off between success and compute, we replace the selected τ with fixed values $\tau \in \{0.25, 0.50, 0.75\}$ on Qwen3-4B HotpotQA and APPS and report paired differences relative to the selected threshold in matched evaluation runs. Table 8 shows the result. A threshold of 0.25 raises the trigger rate by 10 to 17 points without a reliable SR gain. A threshold of 0.75 lowers the trigger rate by 23 to 30 points but reduces SR significantly in both environments. The effect of 0.50 depends on the setting. The threshold therefore traces a frontier between success and compute rather than having a universally optimal value, which is why DIAL selects it per setting.

Table 8: Fixed threshold minus selected threshold on Qwen3-4B, paired differences over matched runs. ΔSR in percentage points with 95% CI; ΔTrigger in percentage points of trigger rate.

Env	Fixed τ	ΔSR	95% CI	ΔTrigger
HotpotQA	0.25	$+0.67$	$[-1.92, +3.25]$	$+9.68$
	0.50	-0.83	$[-4.63, +2.96]$	-8.90
	0.75	-3.33	$[-5.23, -1.44]$	-22.89
APPS	0.25	-0.33	$[-2.92, +2.25]$	$+17.23$
	0.50	-1.83	$[-2.55, -1.12]$	-15.04
	0.75	-3.50	$[-5.98, -1.02]$	-29.52

C.2 Universal Feature Definitions

Table 9 defines the universal features available in every environment.

Table 9: Universal feature definitions. All features are computed identically across environments; signals an environment does not naturally expose default to zero.

Feature	Definition
`step_count`	Current step index t in the episode
`token_entropy`	Entropy of the next-token distribution
`evidence_count`	Count of evidence/information items collected (QA/search envs)
`num_available_actions`	Count of currently-available actions (navigation envs)
`is_finish`	Whether the proposed action is the episode-terminating "finish"

Additional feature categories used only in the regularizer ablation. The expanded candidate pool of Appendix F.3 additionally includes three *derived* features (`step_ratio` = step_count/max_steps, `entropy_sq` = token_entropy^2, and `step_x_entropy` = step_count $\times$ token_entropy), 20 *hidden-state PCA* components extracted from the model's internal representations, and three *auto-extracted* features parsed from the state text (`state_length`, `num_numbers`, and `has_error`). The main DIAL configuration uses only the universal features above plus the LLM-generated layer (Appendix C.3).

C.3 LLM Feature Layer

For environments where universal features may be inadequate, DIAL's optional LLM feature layer automates the proposal of task-specific signals. The LLM receives a structured summary of the exploration dataset $\mathcal{D}$, including: (1) aggregate statistics (total episodes, steps, optimizer trigger rate, overall positive-utility fraction), (2) per-step breakdowns of trigger rate and utility by step index, and (3) representative examples of states where the optimizer was triggered, paired with their utility labels (positive vs. negative).

We use Qwen3-4B-Instruct-2507 as the proposer in all experiments. It is called once per gate during calibration; across the 18 settings, these calls consumed 29,404 tokens and 187.6 seconds in total. The resulting extractor is validated and frozen before deployment, so no proposer call occurs at inference time.

Leakage control. The proposer sees only calibration examples. Each example contains a truncated state, the proposed action, the step index, and the numeric counterfactual utility, together with aggregate statistics such as the positive-utility rate and mean utility. The proposer uses these examples only to propose candidate feature functions. It is not given held-out instances or their outcomes, and the generated extractor is frozen before held-out evaluation. It therefore cannot exploit held-out solution patterns during feature construction or evaluation.

Given this summary, the LLM is prompted to analyze patterns that distinguish positive-utility from negative-utility states and to produce a Python function that extracts exactly 5 task-specific features from the raw state text. Each feature must return a float value using only the Python standard library. The full prompt template is shown in Figure 5 (§C.5).

The LLM-proposed features $\phi_{\text{LLM}}(s)$ are combined with the universal features to form the candidate pool $\phi_{\text{cand}}(s) = \phi_{\text{univ}}(s) \cup \phi_{\text{LLM}}(s)$. The ℓ_1-regularized gate (§4.3) then selects or discards each feature based on its predictive value for rollout utility. This two-stage design (LLM proposes, LASSO disposes) ensures that uninformative LLM features are automatically filtered out. As shown in Table 15, environments with richer task-specific structure (WebShop, TWExpress) retain more LLM features, while Plancraft retains none.

C.4 Online Adaptation

For drifting environments, DIAL can optionally refit during deployment using ε-greedy exploration ($\varepsilon : 0.1 \to 0$) and periodic retraining. During deployment, $\varepsilon_{\text{deploy}}$ decays from 0.1 to 0 over the first 100 episodes, and the gate is refit every $k = 30$ episodes using the accumulated ε-greedy samples. Refitting reuses the same solver configuration as the initial training, so no additional hyperparameter tuning is required at deployment time.

This mode is *not free*: ε-greedy overrides ignore the current gate, so in Type-I-dominant environments a fraction of overridden steps will trigger harmful rollouts. For stationary environments, which are the setting of our main experiments, we recommend skipping online adaptation and relying on the offline-fitted gate.

C.5 LLM Reasoning Prompts

Figure 5 shows the full LLM prompt template. We use WebShop as the example, where LLM features contribute most meaningfully.

LLM Feature Layer Prompt Template

```
You are an AI agent that has been exploring an interactive environment.
During exploration, you sometimes performed ``rollouts'' (additional
computation) to improve your decisions.  Sometimes the rollout was useful
(positive utility), sometimes it was not.
Your task:  Analyze the patterns in your experience and write a Python
function that extracts features from the state text that could predict whether
a rollout would be useful.
```

[Exploration statistics and representative examples provided]

```
Requirements:
- Return a dict with string keys and float values.
- Use only Python standard library.
- Extract exactly 5 features based on the patterns you observe.
- Focus on features that DIFFER between useful and not-useful rollout cases.
```

Figure 5: Prompt template used in the LLM feature layer (Phase 2) to propose task-specific features from exploration data.

D Experimental Setup

D.1 Environment Specifications

We evaluate on 6 environments spanning diverse agent tasks. Table 10 summarizes their base performance, optimizer headroom, and optimizer type.

HotpotQA [41]. A multi-hop question answering task where the agent must retrieve and reason over multiple Wikipedia paragraphs to answer complex questions. The agent iteratively issues search queries and reads returned passages before producing a final answer.

WebShop [43]. A web navigation task where the agent shops for a product matching a natural language instruction. The agent navigates a simulated e-commerce website by searching, clicking product links, and selecting attributes (size, color) before purchasing.

FEVER [44]. A fact verification task where the agent must classify a claim as *Supported*, *Refuted*, or *Not Enough Info* based on evidence retrieved from Wikipedia. The agent searches for relevant passages and reasons over retrieved evidence to produce a verdict.

APPS [42]. A code generation task where the agent solves competitive programming problems. Given a problem description with input/output examples, the agent generates a Python solution and can iteratively debug based on test case feedback.

TWExpress [45]. A text-based game where the agent navigates a simulated environment to complete household tasks (e.g., finding and heating objects). The agent issues natural language commands (go, open, take, use) and observes textual descriptions of the resulting state.

Plancraft [46]. A crafting planning task where the agent must produce a target item by executing a sequence of crafting recipes in a Minecraft-inspired environment. The agent selects recipes and manages inventory to reach the goal item.

These environments cover the full range of the Two-Source Model: FEVER and HotpotQA are dominated by Type I states (information poverty), WebShop and APPS are dominated by Type D states (decision difficulty), TWExpress is rollout-safe ($\Delta{=}+31.8$), and Plancraft is rollout-harmful ($\Delta{=}-7.0$).

All three optimizer types rank candidates by environment-provided reward; they differ along two axes: what is varied (single action / full program / full action sequence) and how the rollout is aggregated (mean return of truncated rollouts / test-case pass / full-episode reward).

- **Per-action evaluation** (HotpotQA, FEVER, TWExpress, Plancraft). At the gated step, take the top-$K{=}5$ candidate actions proposed by the base policy. For each candidate a, fork the environment snapshot, execute a, and continue for $H{-}1{=}2$ further steps with the same LLM proposer ($\tau{=}0.7$), giving a truncated rollout of length $H{=}3$. Repeat $N{=}5$ independent rollouts per candidate and set $V(a) = \frac{1}{N}\sum_{n=1}^{N} R_n(a)$, where $R_n(a)$ is the env-provided score for the n-th truncated trajectory (task-specific: e.g., F1 for QA, episode return for games). Execute $a^* = \arg\max_a V(a)$.
- **K-variant sampling** (APPS). Sample $K{=}3$ complete code solutions and execute the first one that passes the provided test cases (objective check via the interpreter).
- **LLM-Propose-K** (WebShop). The LLM proposes $K{=}3$ full action sequences; each is simulated end-to-end in the environment, and the trajectory with the highest observed episodic reward is executed.

Utility estimation. In all environments, the utility label U_t used for gate training (§4.1) is computed by paired counterfactual evaluation from the same state snapshot. For per-action evaluation environments, the base policy's proposed action a_B is included in the top-K candidate set and evaluated with the same $N{\times}H$ rollout protocol as all other candidates; utility is $U_t = \mathbf{1}[V(a^*) > V(a_B)]$, where $V(a) = \frac{1}{N}\sum_{n=1}^{N} R_n(a)$. For K-variant sampling (APPS), the base solution is evaluated on the same test suite as the K variants, and $U_t = \mathbf{1}[\max_k \text{pass_rate}_k > \text{pass_rate}_{\text{base}}]$. Because all candidates are evaluated from identical initial conditions, this is a counterfactual estimate rather than an on-policy observation. The truncation horizon H introduces approximation bias relative to full-episode returns; we find results stable for $H \in [3, 10]$.

In all three cases, the optimizer itself has access to the env's scoring function; our method is about learning *when* this optimizer is worth invoking, not how it scores internally.

Table 10: Environment setup. Base/Always: SR without/with optimizer. Δ: rollout headroom.

Environment	Base SR	Always SR	Δ	Optimizer T
HotpotQA	49.0%	97.0%	+48.0	Per-action eval
WebShop	7.2%	43.0%	+35.8	LLM-Propose-K
FEVER	37.0%	99.8%	+62.8	Per-action eval
TWExpress	67.5%	99.3%	+31.8	Per-action eval
Plancraft	29.8%	22.8%	-7.0	Per-action eval
APPS	60.5%	79.5%	+19.0	K-variant sampling

D.2 Detailed Baseline Comparison

Table 11 provides a detailed comparison of all baselines evaluated in this paper. All six methods share a common assumption: a fixed mapping from signal value to compute need.

Table 11: Detailed comparison of baselines. All methods assume a fixed signal-utility direction. †Requires a separate labeled dataset for calibration.

Method	Signal	Direction	Granularity	Mechanism
CATTS [17]	Vote entropy + margin	High disagreement → trigger	Per-step	K=5 forward passes, arbiter on disagreement
SEAG† [12]	Mean token confidence	Low conf. → search	Per-problem	Platt scaling, search depth allocation
CaTS† [13]	Calibrated confidence	Low conf. → generate	Per-attempt	Self-calibration, Best-of-N early stopping
CoRefine† [14]	Conv1D confidence	Low conf. → refine	Per-problem	Halt / re-examine / redirect controller
s1_budget [15]	Token budget	Fixed budget → stop	Per-problem	Budget-constrained generation
AUQ [25]	Verbalized uncertainty	High uncert. → reflect	Per-step	Uncertainty-aware memory + reflection
DIAL	**Multi (auto)**	**Learned**	**Per-step**	Explore + reason + direction learning

D.3 Computational Cost Breakdown

Table 12 breaks down the per-method cost structure. Cost is reported as ×base: total *deployment* tokens normalized by base_only tokens per episode. Calibration or exploration overhead is excluded for all methods, ensuring a fair comparison of runtime efficiency. This metric captures hidden per-step overhead that raw rollout counts miss: CATTS incurs K=5 forward passes every step regardless of the gating decision, and AUQ issues a confidence query every step. Consistent with the cost accounting of §4, DIAL incurs a one-time exploration cost of N_{explore}=50 episodes per (environment, backbone) deployment and sub-second offline gate fitting; at deployment the gate adds zero per-step inference cost beyond the feature evaluation and a single sigmoid: no extra LLM call, no K-way voting, and no calibration query.

Table 12: Computational cost breakdown per method. Gate OH = additional LLM calls per step beyond the base proposer. Cost (×base) = total *deployment* tokens normalized by base_only tokens per episode, averaged across 6 environments. Calibration/exploration overhead is excluded for all methods. †Requires separate calibration data.

Method	**Gate OH/step**	**OH source**	**Avg ×b**
base_only	0	—	1.00
always_trigger	0	—	6.00
CATTS	$+K$=5 calls	Voting (fwd pass)	7.98
SEAG†	0	Reads logprob	6.37
CoRefine†	0	Reads entropy	6.33
CaTS†	0	Reads confidence	8.26
AUQ	+1 query	Confidence question	7.67
s1_budget	0	—	6.35
DIAL	**0**	**Learned gate**	**5.85**

D.4 DIAL vs. Bounds: Full Numbers

Table 13 reports DIAL's SR and Cost against the two bounds (*base_only*, *always_trigger*) across all 6 environments on Qwen3-4B. In environments with positive rollout headroom (HotpotQA, WebShop, TWExpress, APPS), DIAL achieves SR close to always_trigger at substantially lower cost. In the rollout-harmful environment Plancraft ($\Delta{=}-7.0$), DIAL learns to suppress triggering, losing only 6.5% relative to base_only while always_trigger loses 7.0%. FEVER remains the weakest case: DIAL improves over base_only (49.8% vs. 37.0%) but falls far short of always_trigger (59.8%), a limitation analyzed in §6.

Table 13: DIAL vs. bounds on Qwen3-4B. Δ: rollout headroom (always$-$base).

	base_only		**always_trigger**		**DIAL**	
Env	SR↑	Cost↓	SR↑	Cost↓	SR↑	Cost↓
HotpotQA	49.0	1.00	97.0	10.63	95.2	8.02
WebShop	7.2	1.00	43.0	5.56	43.8	2.50
FEVER	37.0	1.00	59.8	17.87	49.8	16.51
TWExpress	67.5	1.00	99.3	2.15	99.0	1.81
Plancraft	29.8	1.00	22.8	6.13	23.3	3.63
APPS	60.5	1.00	79.5	3.65	73.0	2.61

D.5 Compute Resources

All experiments are run on NVIDIA A100 GPUs with 40 GB of memory. Each backbone (Qwen3-4B, Llama-3.1-8B-Instruct, Phi-3.5-mini-instruct) fits on a single A100 in BF16, and we serve the model with vLLM for batched inference. The full set of reported experiments covers 6 environments $\times$ 3 backbones $\times$ 3 seeds, plus the DIAL exploration phase ($N_{\text{explore}}{=}50$ episodes per (environment, backbone) deployment) and ablations. We estimate the total compute used to produce the results in this paper at approximately 1,900 A100-GPU-hours, of which roughly 55% corresponds to evaluation rollouts (DIAL, the two bounds, and six fixed-direction baselines across all (environment, backbone, seed) combinations), 2% to DIAL exploration, and the remainder to ablations (LLM-feature variants, gate-capacity, wrong-direction reversal, regularizer, controlled-reversal interventions, and supplementary multi-backbone runs). The full research project, including preliminary experiments and configurations that did not make it into the paper, used roughly 2,600 A100-GPU-hours in total.

D.6 Asset Licenses

Table 14 lists the public assets used in this paper, together with their licenses. We use each asset within the terms of its license; all datasets and model weights are obtained from their official releases.

Table 14: Datasets and pre-trained models used in this paper.

Asset	**Type**	**License**
HotpotQA [41]	Dataset	CC BY-SA 4.0
APPS [42]	Dataset	MIT
WebShop [43]	Dataset+Env	MIT (Princeton NLP)
FEVER [44]	Dataset	CC BY-SA 3.0
TextWorld Express [45]	Environment	Apache 2.0
Plancraft [46]	Environment	MIT
Qwen3-4B [36]	Model weights	Apache 2.0
Llama-3.1-8B-Instruct [37]	Model weights	Llama 3.1 Community License
Phi-3.5-mini-instruct [38]	Model weights	MIT

E Full Results Analysis

Section 5.2 (Table 2) reports SR and Cost ($\times$base) for all methods across 6 environments and 3 backbones. Cost includes all deployment LLM calls (base proposer, gate overhead, rollouts) nor-

malized by base_only cost. Bounds comparison is in Table 13 (§D.4). This appendix elaborates per-backbone failure modes and cost anomalies that the aggregated table does not surface.

Qwen3-4B. DIAL achieves the highest SR across all six environments. The ×base metric reveals hidden costs of fixed-direction baselines: CATTS on APPS costs 6.00×base (due to K=5 per-step voting) despite barely triggering rollouts, while DIAL costs only 2.61× at 12.2% higher SR. FEVER is the cost outlier: DIAL still leads in SR (49.8%) but at 16.51×base, the highest cost among all environments. This failure mode is analyzed in §6.

Phi-3.5-mini. Fixed-direction baselines degrade sharply on this backbone. On FEVER, all six baselines score at or below 20%, while DIAL reaches 20.5%. On WebShop, DIAL (57.3%) exceeds the best baseline (s1_budget, 43.0%) by 14.3% at lower cost (2.56× vs. 3.89×).

Llama-3.1-8B. Direction reversal interacts with model capacity. On TWExpress, CaTS collapses to near-base performance (35.8% vs. base 36.5%) because it rarely triggers, while DIAL achieves 94.8%. On FEVER, DIAL (54.7%) is competitive with CaTS (56.3%), unlike Qwen3 where FEVER is the weakest environment. On APPS, AUQ drops to 46.0% (below the no-trigger baseline), demonstrating that wrong-direction gating actively harms performance on this backbone.

F Signal and Feature Analyses

This appendix collects empirical analyses of signal identity and signal sufficiency referenced in Section 3.2. §F.1 reports which features the ℓ_1 gate selects per environment and how their signed weights align with the Two-Source Model. §F.2 ablates the LLM feature layer. §F.3 stress-tests sparsity by ablating the regularizer under an expanded candidate pool. §F.4 reports the AUC hierarchy (single-signal < multi-signal < hidden-state probe) as the empirical signature of σ-insufficiency.

F.1 Per-Environment Feature Selection

The LASSO coefficient matrix identifies which auxiliary features carry information about the latent state type τ in each environment. These are exactly the components of $\phi(s)$ that lift σ toward sufficiency for VOC (App B.1). `step_count` is informative in 4 of 5 environments with significant signal: it proxies *information accumulation*, distinguishing late-step Type I states (where information has accumulated to a known shortage) from early-step Type D states (where multiple options are still viable). WebShop is the exception, relying on `num_available_actions` and task-specific features (`price_mentioned`, `action_is_click`); these proxy *decision optionality* directly, as expected for a choice-dominated (Type D) environment. In every case, the features LASSO retains are interpretable as proxies for the two-source decomposition rather than ad-hoc engineering choices.

F.2 LLM-Generated Features

LLM feature ablation. Table 4 in §5.3 compares DIAL with and without LLM-generated features. FEVER is the only environment where LLM features contribute meaningfully (+9.1%), where task-specific signals (`has_claim`, `text_length`) extend the universal feature pool. In the remaining environments, removing LLM features changes SR by <2%; in Plancraft and APPS the contribution is effectively zero. The LLM feature layer is therefore an automation convenience rather than a core driver of DIAL's gains.

Selected LLM features per environment. Table 15 reports the features generated by the LLM reasoning step (Phase 2) for each environment. The LLM receives exploration statistics and representative positive/negative examples, then produces a Python function extracting 5 task-specific features (the full prompt template is in §C.5). LASSO subsequently selects or discards each feature based on its predictive value for rollout utility.

Environments with rich task-specific structure benefit most from LLM features: WebShop's choice-dominated interface is well captured by `price_mentioned` and `action_is_click`, while TWExpress's exploration mechanics are reflected in `closed_ratio`

Table 15: LLM-generated features per environment. Each environment receives 5 LLM-proposed features; LASSO selects those with non-zero coefficients. Environments with richer task-specific structure (WebShop, TWExpress) retain more LLM features.

Environment	Sel./Gen.	LASSO-Selected LLM Features
WebShop	4/5	`price_mentioned`, `action_is_click`, `step_early`, `instruction_keyword_count`
TWExpress	4/5	`text_length`, `closed_ratio`, `action_look_around`, `already_open`
HotpotQA	3/5	`has_question`, `entity_count`, `fact_keyword_count`
FEVER	2/5	`text_length_normalized`, `has_claim`
APPS	1/5	`action_type`
Plancraft	0/5	(none selected)

and `action_look_around`. In contrast, Plancraft's rollout-harmful nature means no LLM feature improves prediction, and LASSO correctly filters all five proposals. This pattern aligns with the LLM ablation results (Table 4): the environment where LLM features contribute most to SR (FEVER, $+9.1\%$) is precisely where the LLM discovers unique signals (`has_claim`) that the universal feature pool lacks.

Figure 6 shows the complete LLM-generated feature extraction function for WebShop (seed 42), the environment with the most LLM features retained.

LLM Output for WebShop

```
def extract_features(state_text, step_count, action_text):
    import re
    features = {}
    # Instruction keyword density
    keywords = ["gluten free", "protein",
        "price lower than", "size", "color"]
    matches = sum(1 for kw in keywords
        if kw in state_text.lower())
    features['instruction_keyword_count'] = (
        float(matches) / len(keywords))
    # Early-step indicator
    features['step_early'] = (
        1.0 if step_count <= 2 else 0.0)
    # Click vs. search action
    features['action_is_click'] = (
        1.0 if action_text.startswith("click[")
        else 0.0)
    # Price constraint presence
    price_matches = len(re.findall(
        r"price lower than|price:|\$",
        state_text, re.IGNORECASE))
    features['price_mentioned'] = (
        float(price_matches) / 5.0)
    # Filter complexity
    filter_kws = ["size","color","flavor",
        "price","rating"]
    features['filter_count'] = sum(
        1 for kw in filter_kws
        if kw in state_text.lower()) / 5.0
    return features
```

Figure 6: Complete LLM-generated feature extraction function for WebShop (seed 42), the environment with the most LLM features retained by LASSO.

F.3 Regularizer Ablation

The signed-weight diagnostic of §4.3 and DIAL's robustness to noisy LLM-proposed features (§4.2) both rely on the *exact* sparsity induced by the ℓ_1 penalty in Eq. (5). To verify that ℓ_1 specifically, rather than any shrinkage or selection mechanism, is responsible for these properties, we ablate the regularizer while holding all other pipeline components fixed: same exploration dataset $\mathcal{D}$, same candidate feature pool ϕ_{cand}, same 5-fold CV protocol for hyperparameters, same threshold $\tau{=}0.5$ (fixed across variants for this ablation to isolate the regularizer's effect; main DIAL uses τ selected by cross-validation per Eq. (4)). We compare five variants:

- ℓ_1**-logistic** (DIAL default): joint sparsity and shrinkage.
- ℓ_2**-logistic** (Ridge): shrinkage only; all features retained with small non-zero weights.
- **No regularization**: plain logistic fit on ϕ_{cand}.
- **Elastic Net** (α=0.5, mixing ℓ_1 and ℓ_2).
- **MI top-k + unregularized**: hard feature selection by mutual information (top-3), then unregularized logistic on the retained features. This variant isolates *selection* from *shrinkage*.

Table 16 reports SR and the mean number of non-zero fitted weights across three environments spanning signal strength: HotpotQA ($|\rho^*|$=0.494, strong), WebShop ($|\rho^*|$=0.444, moderate), and Plancraft ($|\rho^*|$=0.016, weak).

Table 16: Regularizer ablation on three environments ordered by signal strength. SR (%); "#nz" is the mean number of non-zero fitted weights across three seeds. For this ablation, ϕ_{cand} is expanded to a stress-test pool of 31 features (5 universal + 3 derived + 20 hidden-state PCA + 3 auto-extracted) to evaluate each regularizer's behavior under a high-dimensional candidate set; adding the LLM feature layer brings the pool to ≈36. ℓ_1 is the only variant that achieves competitive SR *and* exact sparsity required by the signed-weight diagnostic.

	HotpotQA $\lvert\rho^*\rvert$=0.494		WebShop $\lvert\rho^*\rvert$=0.444		Plancraft $\lvert\rho^*\rvert$=0.016	
Regularizer	SR	#nz	SR	#nz	SR	#nz
ℓ_1 (DIAL)	94.8 ± 0.2	6.7	42.7 ± 6.7	6.3	27.2 ± 1.6	5.0
ℓ_2	94.3 ± 1.0	30	41.3 ± 5.8	30	26.3 ± 1.0	30
No regularization	94.0 ± 1.9	30	36.2 ± 7.3	30	27.8 ± 0.9	30
Elastic Net	94.3 ± 1.6	8.7	44.0 ± 1.8	6.0	27.2 ± 0.8	10
MI top-3 + unreg.	95.0 ± 1.1	3.0	38.5 ± 8.2	3.0	26.3 ± 1.2	3.0

Note. SR is reported as mean±std across three seeds (Elastic Net on WebShop is reported for a single seed and is excluded from std-based comparisons). "#nz" is the mean number of features with non-zero coefficient; for ℓ_2, no regularization, and MI top-k this number is fixed by construction. The L1 row uses the same fitting pipeline as the four other rows (the modified `principled_scg.py` with the LASSO step swapped per row), yielding internally consistent SR across the table; the slight numerical divergence from the corresponding cells in the main results (Table 2) on Plancraft reflects an implementation difference there (`scg_finetune_lr`, default sklearn logistic) rather than a methodological change. Across ℓ_1 runs, sparsity adapts to signal strength: HotpotQA and WebShop retain 6–7 features each, whereas Plancraft collapses to ≈5, absorbing the remainder into the bias and yielding a near-constant trigger probability consistent with that environment's low rollout headroom (Figure 10).

The five rows isolate three hypotheses underlying DIAL's regularizer choice.

ℓ_1 vs. ℓ_2: sparsity is required for diagnosis, not just for predictive performance. Both penalties shrink noisy coefficients toward zero, and on all three environments their gating SRs are within a single percentage point: 94.8 vs. 94.3 on HotpotQA, 42.7 vs. 41.3 on WebShop, 27.2 vs. 26.3 on Plancraft. The critical distinction is interpretability rather than performance: under ℓ_2, every one of the ≈30 candidate features retains a small non-zero weight, collapsing the clean {informative, uninformative} partition of §4.3 into a continuous ranking. The signed-weight summary no longer reads off which signals the environment treats as informative; it merely ranks them. Reports of "non-zero weights per environment" in the main text (e.g., the direction analyses of Table 3) are only interpretable under the exact sparsity that ℓ_1 provides.

ℓ_1 vs. unregularized logistic: robustness under weak signals. The unregularized fit is the most fragile variant in the table. On Plancraft ($|\rho^*|$≈0.02), no regularization actually overfits the training split into a higher mean SR than ℓ_1 (27.8 vs. 27.2) but with no diagnostic interpretation (#nz = 30 across all seeds). The damaging case is WebShop ($|\rho^*|$≈0.44, moderate signal): no regularization drops to 36.2 ± 7.3, fully 6.5 points below ℓ_1 at 42.7 ± 6.7, with the wider standard deviation reflecting seed-dependent overfitting onto whichever PCA components happen to pseudo-correlate

with utility on the training fold. ℓ_1's shrinkage absorbs this noise into the bias term and yields a gate that triggers more conservatively (lower `rollouts/ep` at the same SR; cf. Appendix G.8).

ℓ_1 vs. MI top-k + unregularized: joint selection+shrinkage vs. selection alone. Hard feature selection by mutual information separates the two roles ℓ_1 plays. MI top-3 recovers most of ℓ_1's SR on strong- and weak-signal environments (95.0 vs. 94.8 on HotpotQA, 26.3 vs. 27.2 on Plancraft), confirming that on those endpoints sparsity matters more than which exact features are kept. On WebShop the gap opens to 4 points (38.5 vs. 42.7, std $= 8.2$, the largest in the table), reflecting seed-dependent overfitting that ℓ_1's joint shrinkage prevents. MI top-k also requires tuning k per environment rather than CVing a single C, reintroducing the per-environment engineering cost DIAL aims to eliminate.

ℓ_1 vs. Elastic Net. Elastic Net (α=0.5) is statistically interchangeable with ℓ_1 on all three environments (within 1 pt SR; #nz higher by 1–5). We default to ℓ_1 as the strictly simpler choice: one hyperparameter instead of two, exact zeros instead of small-but-nonzero weights, and direct compatibility with the signed-weight diagnostic.

Summary. ℓ_1 is the only variant in this table that simultaneously delivers competitive SR, the exact sparsity required by the signed-weight diagnostic of §4.3, and a single CV-tunable hyperparameter; we use it as the default.

F.4 AUC Hierarchy: Empirical Signature of σ-Insufficiency

Table 17 is the empirical instantiation of the sufficient-statistic claim of §3.3: it measures, on held-out data, how much of VOC each gate family can recover. Single entropy is near chance (AUC≈0.50) in every environment, the operational signature of σ's insufficiency for VOC. Multi-signal logistic gates that augment σ with universal auxiliary features reach 0.74–0.92, recovering most of the available signal; hidden-state probes (which have access to the model's full internal representation, an upper bound on what any feature map $\phi(s)$ can extract) reach 0.79–0.99. The gap between the single-entropy row and the multi-signal row is therefore a quantitative measure of how much auxiliary conditioning is required to lift σ toward sufficiency for VOC; the residual gap between multi-signal and hidden-state probe is the sufficiency that universal features fail to capture (small in most environments). The bottleneck is signal *combination*, not gate *capacity*, and the multi-signal LR row is the empirical case for DIAL's design.

Table 17: AUC by gate family. Single entropy is near chance; multi-signal gates close most of the gap to hidden-state probes. FEVER and TWExpress are omitted because their short episodes yield insufficient held-out data for reliable AUC estimation.

Gate family	HotpotQA	APPS	WebShop	Plancraft
Single entropy	0.50	0.56	0.50	0.50
Best single signal	0.78	0.78	0.90	0.74
Multi-signal LR	0.85	0.76	0.92	0.74
Hidden-state probe	0.87	0.79	0.99	0.95

G Robustness and Additional Experiments

This appendix collects Two-Source Model verification data, robustness checks for direction reversal, and additional experimental analyses. §G.1 and §G.2 rule out scale and reward-distribution artifacts as causes of the observed sign reversal. §G.3, §G.5, and §G.6 verify the Two-Source Model's predictions across time, environment composition, and model backbone. §G.8 analyzes DIAL's deployment trigger behavior across environments. §G.7 evaluates DIAL on a newer backbone.

G.1 Robustness of Reversal to Entropy Quantile-Normalization

A natural concern with Table 1 is that cross-backbone reversal could reflect incomparable raw entropy scales across tokenizers rather than a structural sign change. We rule this out by replacing each

Table 18: Robustness of cross-backbone reversal to entropy quantile-normalization. We report Spearman $\rho(\sigma, U)$ (invariant under within-cell monotone transforms, so shared across raw/S1/S2/S3) and Pearson ρ under raw and three quantile schemes: **S1** ranks within each (env, backbone), **S2** within each backbone's pool, **S3** within each environment's pool. Listed are the **11 strong-signal cells** (raw Pearson $|\rho| \geq 0.13$) covering every reversal claim. **Robust** = sign stable vs. raw under all three schemes (CIs on the same side). ‡ Phi-3.5 APPS robust under S1, S2 but inconclusive under S3. Excluded (discussed in prose): 4 weak-signal cells ($|\rho| < 0.13$, raw CIs straddle zero) and 3 cells with insufficient logged data. *: $p < 0.05$, 1000-resample bootstrap.

			Spearman	**Pearson $\rho(\sigma, U)$**				
Backbone	**Env**	**N**	raw=S1=S2=S3	raw	S1	S2	S3	**Conclusion**
Llama-3.1	FEVER	840	$+0.428^*$	$+0.481$	$+0.418$	$+0.454$	$+0.432$	✓ Robust
Phi-3.5	WebShop	751	$+0.335^*$	$+0.384$	$+0.374$	$+0.382$	$+0.394$	✓ Robust
Llama-3.1	HotpotQA	244	-0.346^*	-0.344	-0.337	-0.341	-0.320	✓ Robust
Qwen3-4B	TWExpress	798	-0.290^*	-0.319	-0.290	-0.329	-0.324	✓ Robust
Llama-3.1	WebShop	948	$+0.287^*$	$+0.298$	$+0.287$	$+0.303$	$+0.321$	✓ Robust
Llama-3.1	APPS	475	-0.242^*	-0.217	-0.271	-0.233	-0.287	✓ Robust
Qwen3-4B	APPS	439	$+0.317^*$	$+0.203$	$+0.303$	$+0.265$	$+0.285$	✓ Robust
Phi-3.5	APPS	400	-0.129^*	-0.197	-0.104	-0.193	-0.054	✓ Robust
Qwen3-4B	FEVER	282	-0.119^*	-0.186	-0.123	-0.237	-0.192	✓ Robust
Phi-3.5	FEVER	824	-0.156^*	-0.158	-0.161	-0.160	-0.157	✓ Robust
Qwen3-4B	HotpotQA	1208	-0.327^*	-0.138	-0.330	-0.264	-0.323	✓ Robust

$\sigma(s)$ with its empirical quantile rank $Q(s) \in [0, 1]$ under three pooling schemes: **S1** ranks within each (environment, backbone) cell, **S2** ranks within each backbone's pool of states (across environments), and **S3** ranks within each environment's pool (across backbones). For each scheme we recompute Spearman $\rho(Q, U)$ and Pearson $\rho(Q, U)$ per cell with 1000-resample bootstrap confidence intervals.

Spearman: invariant by construction. Per-cell Spearman ρ is mathematically rank-invariant under any within-cell monotone transform of σ, so all three schemes give identical Spearman ρ to the raw column within each cell. We report this transparently: it confirms that rank structure is preserved but is not, in itself, a non-trivial test.

Pearson: scale-sensitive, the substantive test. Pearson ρ varies under non-linear normalization. We partition the 18 (env, backbone) cells by raw Pearson $|\rho|$ at the 0.13 threshold and report the 11 strong-signal cells ($|\rho| \geq 0.13$) in Table 18. **All 11 strong-signal cells are robust**: Pearson sign is stable under all three normalization schemes on 10 cells and 2/3 schemes on 1 cell (Phi-3.5 APPS, where S3 yields -0.054 with CI straddling zero). Critically, every cell that the paper's reversal claim invokes, including the cross-backbone anchors on HotpotQA, APPS, and FEVER, is in this strong block. The remaining 7 cells fall into two categories that we exclude from the table for brevity. **Weak-signal cells** (4 of 7: Qwen3 WebShop, Phi-3.5 HotpotQA, and Plancraft on {Qwen3, Llama-3.1}) all have raw $|\rho| < 0.13$ and CIs already straddling zero; under each of S1/S2/S3 they remain inconclusive, but since the paper does not claim reversal on these cells, this is expected sample-level noise rather than a refutation. **Insufficient-data cells** (3 of 7: TWExpress on {Phi-3.5, Llama-3.1} and Plancraft on Phi-3.5) have logged σ with ≤ 2 unique values or no positive U labels, making correlation analysis statistically ill-defined. The pattern in the strong block, where stability tracks signal strength and no strong-signal cell flips under any scheme, is what one expects if the reversal is structural rather than a tokenizer-scale artifact.

The cross-backbone reversal pattern on HotpotQA, APPS, and FEVER therefore survives all three normalization schemes, supporting the claim of §3.2 that the reversal is structural rather than a tokenizer-scale artifact.

G.2 Ruling Out Alternative Explanations: Reward and Entropy Calibration

We further verify that the reversal is not explained by reward parameterization or entropy estimator calibration.

Table 19: Robustness of $\rho(\sigma, U)$ to monotone transforms. Spearman ρ is rank-invariant under monotone transforms of either variable, so the $\sigma^{0.5}$, σ^2, and $\log\sigma$ columns must match raw up to numerical noise (a sanity check on the entropy pipeline); under $U \to \alpha U$ with $\alpha < 0$, sign flips as expected. Pearson ρ varies under non-linear σ transforms; the only Pearson sign flip is Phi-3.5/APPS under $\log\sigma$ ($+0.027$), where Spearman stays stable at -0.129. Excluded: three cells with $|\sigma_{\text{unique}}| \leq 2$ or no positive U labels (TWExpress on {Phi-3.5, Llama-3.1}, Plancraft on Phi-3.5), where correlation is ill-defined.

		Spearman ρ					**Pearson ρ**		
Backbone	**Env**	raw	$\alpha U, \alpha=-1$	$\sigma^{0.5}$	σ^2	$\log\sigma$	$\sigma^{0.5}$	σ^2	$\log\sigma$
Qwen3-4B	HotpotQA	-0.327	$+0.327$	-0.327	-0.327	-0.327	-0.201	-0.085	-0.317
Qwen3-4B	APPS	$+0.317$	-0.317	$+0.317$	$+0.317$	$+0.317$	$+0.273$	$+0.085$	$+0.305$
Qwen3-4B	WebShop	$+0.133$	-0.133	$+0.133$	$+0.133$	$+0.133$	$+0.127$	$+0.008$	$+0.238$
Qwen3-4B	FEVER	-0.119	$+0.119$	-0.119	-0.119	-0.119	-0.228	-0.140	-0.240
Qwen3-4B	TWExpress	-0.290	$+0.290$	-0.290	-0.290	-0.290	-0.305	-0.281	-0.264
Qwen3-4B	Plancraft	-0.016	$+0.016$	-0.016	-0.016	-0.016	-0.019	-0.031	-0.010
Phi-3.5	HotpotQA	$+0.184$	-0.184	$+0.184$	$+0.184$	$+0.184$	$+0.170$	$+0.042$	$+0.107$
Phi-3.5	APPS	-0.129	$+0.129$	-0.129	-0.129	-0.129	-0.161	-0.196	$+0.027$
Phi-3.5	WebShop	$+0.335$	-0.335	$+0.335$	$+0.335$	$+0.335$	$+0.399$	$+0.324$	$+0.296$
Phi-3.5	FEVER	-0.156	$+0.156$	-0.156	-0.156	-0.156	-0.153	-0.156	-0.049
Llama-3.1	HotpotQA	-0.346	$+0.346$	-0.346	-0.346	-0.346	-0.332	-0.335	-0.288
Llama-3.1	APPS	-0.242	$+0.242$	-0.242	-0.242	-0.242	-0.266	-0.150	-0.326
Llama-3.1	WebShop	$+0.287$	-0.287	$+0.287$	$+0.287$	$+0.287$	$+0.310$	$+0.258$	$+0.310$
Llama-3.1	FEVER	$+0.428$	-0.428	$+0.428$	$+0.428$	$+0.428$	$+0.431$	$+0.528$	$+0.354$
Llama-3.1	Plancraft	-0.041	$+0.041$	-0.041	-0.041	-0.041	-0.014	-0.035	-0.003

Reward transformations. Spearman $\rho(\sigma, U)$ is rank-invariant under positive monotone transforms of U, so $U \to \alpha U$ for $\alpha > 0$ exactly preserves ρ across all 15 analyzable cells; for $\alpha < 0$ the sign flips as expected (rank order reversed). Table 19 shows the $\alpha = -1$ column for completeness. Reward parameterization is therefore not a confound. Additive shifts $U \to U + c$ also preserve the correlation, but trivially flip the binary classifier label $\text{sign}(U)$ on states with $U = 0$; this is a degenerate property of the U distribution rather than a sensitivity, and we do not include label-flip data in the robustness verdict.

Entropy transformations. For $\sigma \to \sigma/T$ with $T \in \{0.5, 1, 2, 5\}$, both Spearman and Pearson ρ are exactly preserved (positive linear scaling). Under non-linear transforms $\sigma \to \sigma^\alpha$ with $\alpha \in \{0.5, 1, 2\}$ and $\sigma \to \log(\sigma + \epsilon)$, Spearman ρ remains invariant by rank-preservation on every cell; Pearson varies (as expected for non-linear transforms) but only one cell, Phi-3.5 / APPS under $\log\sigma$, exhibits a Pearson sign flip (Table 19). Spearman on that cell is stable at -0.129, so the structural reversal claim is unaffected. We report this single anomaly transparently rather than averaging it away.

Trajectory length. Trajectory length varies across environments and correlates with both entropy and utility. To rule out length as a confounder, we stratify by step count and recompute $\rho(\text{entropy}, U)$ within each stratum. Direction reversal persists within trajectory-length strata where utility variance is non-zero. In HotpotQA, ρ remains negative across all three strata (-0.18/-0.46/-0.42). The reversal is not an artifact of trajectory length.

Across the four alternative explanations considered (rollout noise, reward bias, search depth, calibration error), only rollout noise and search depth would require new rollouts to test directly. Reward bias and calibration error are post-hoc verifiable on the logged data and are ruled out above. We leave the rollout-noise and search-depth ablations as future work; the two-source mixture interpretation of §3.3 is consistent with all observational data and survives all post-hoc robustness checks within reach.

G.3 Temporal Dynamics (P1)

Table 20 reports the full early/late ρ data referenced in Figure 4. The Two-Source Model predicts that $\rho(\text{entropy}, U)$ should shift downward from early to later steps within each episode, as late-step uncertainty increasingly reflects the residual Type I component after information-gathering attempts.

Table 20: Temporal shift in ρ(entropy, U). Four of five testable environments show ρ shifting more negative from early to late steps, consistent with P1; FEVER is the exception. Plancraft omitted ($|\rho^*|$=0.016, too weak for meaningful comparison).

Env	Phase	ρ	95% CI	n	p	Shift
HotpotQA	early	-0.176	$[-0.260, -0.091]$	509	<0.001	$\Delta=-0.261$
	late	-0.437	$[-0.500, -0.370]$	699	<0.001	
APPS	early	$+0.102$	$[-0.013, +0.209]$	290	0.084	$\Delta=-0.246$
	late	-0.144	$[-0.281, +0.001]$	199	0.043	
WebShop	early	$+0.285$	$[+0.222, +0.346]$	600	<0.001	$\Delta=-0.291$
	late	-0.006	n.s.	473	0.895	
FEVER	early	$+0.054$	n.s.	200	0.446	$\Delta=+0.024$
	late	$+0.078$	n.s.	82	0.486	
TWExpress	early	$+0.161$	—	422	<0.001	$\Delta=-0.153$
	late	$+0.008$	n.s.	376	0.877	

The prediction holds across 4 of 5 testable environments: HotpotQA shows the strongest shift ($\Delta=-0.261$), with late-step entropy reflecting deeper information poverty after failed evidence retrieval. WebShop's strong positive early-phase signal ($\rho=+0.285$) vanishes in the late phase ($\rho=-0.006$, n.s.), consistent with the choice-dominated structure being resolved early. In APPS, ρ reverses from marginally positive ($+0.102$) to negative (-0.144), as late-step decision complexity gives way to residual information poverty.

FEVER is the exception: both phases show weak, non-significant positive ρ, consistent with FEVER's short episodes (median 8 steps) concentrating most rollout value at step 0–1, leaving insufficient late-step variation for a meaningful shift.

Why a static utility can have a time-varying correlation. A natural objection is that $U(s)$ is a fixed function of the state, so its correlation with $\sigma(s)$ should not change over the episode. The resolution is that the correlation at phase t is evaluated under the time-conditioned state distribution,

$$\rho_t = \mathrm{corr}_{s\sim p(s|t)}\big(\sigma(s), U(s)\big). \tag{6}$$

Even when σ and U are fixed functions of the state, ρ_t changes because trajectory survival and the composition of states in $p(s \mid t)$ change over the episode. Under the Two-Source Model, later phases contain a larger fraction of residual Type I states because many Type D states are resolved earlier, which shifts the marginal correlation toward the Type I relationship.

Three-phase estimate. To test whether the shift is a single early-to-later transition or a continuing decline, we split Qwen3-4B HotpotQA episodes into three phases and estimate ρ in each with an episode-cluster bootstrap. Table 21 reports the result. Relative to the early phase, the middle and late estimates shift by -0.284 and -0.242, while the middle and late estimates are close to each other. The evidence therefore supports one downward shift after the early phase rather than a monotone decline at every step, which is the form of P1 stated in §5.4.

Table 21: Three-phase Spearman ρ(entropy, U) on Qwen3-4B HotpotQA with 95% episode-cluster bootstrap intervals.

Phase	Traj.	Decisions	ρ	95% CI
Early, steps 0–2	200	509	-0.176	$[-0.265, -0.082]$
Middle, steps 3–5	117	317	-0.459	$[-0.594, -0.304]$
Late, steps 6–9	96	382	-0.418	$[-0.556, -0.260]$

G.4 Construction of the HotpotQA Interventions (P2)

The two interventions in §5.4 are operational manipulations of evidence availability. They perturb the state mixture relevant to the Two-Source Model; they do not assign individual states to Type I or Type D.

InfoPoor. After each `search[entity]` action, only the first sentence of the retrieved paragraph is returned to the agent.

InfoRich. All paragraphs named in the annotated gold supporting facts are inserted in full into the initial observation.

As an example, one evaluated question asks what the battle in which Giuseppe Arimondi died secured for Ethiopia. Both the bridge to the Battle of Adwa and the answer, sovereignty, occur after the first sentence of their supporting paragraphs. InfoPoor removes both decisive facts, whereas InfoRich provides both complete paragraphs from the start.

Across the 600 evaluated instances, InfoPoor makes only 57.5% of annotated supporting facts potentially accessible and leaves at least one fact inaccessible in 70.8% of instances. Table 22 confirms the evidence gap by deterministic replay of the evaluated decision points. The corresponding changes in base SR, DIAL SR, and signal correlations are reported in Table 5; the entropy correlation rises from -0.041 with 95% CI $[-0.125, 0.036]$ in the original setting to $+0.311$ with 95% CI $[0.186, 0.416]$ under InfoRich. InfoRich also raises base SR from 49.0% to 82.0%, confirming that it changes the base policy's operating regime and the remaining utility of additional computation.

Table 22: Gold supporting-fact coverage at evaluated decision points under the two HotpotQA interventions, measured by deterministic replay. Coverage before final: fraction of points at which all gold facts were visible before the final answer.

Condition	Points	Mean recall	Full coverage	Missing $\geq$1 fact	Coverage before final
InfoPoor	317	8.0%	2.2%	97.8%	3.5%
InfoRich	221	100.0%	100.0%	0.0%	100.0%

G.5 Cross-setting Consistency at Fixed Backbone

Beyond the within-episode (P1) and within-task interventional (P2) tests in the main text, the Two-Source Model implies that settings with similar p_I should exhibit similar ρ. With the backbone held fixed at Qwen3 (Table 1), this consistency holds: FEVER and HotpotQA (both search-based QA) share negative ρ, and WebShop and APPS (both involving choice among alternatives) share positive ρ. As predicted by the (environment, backbone)-indexed mixture of §3.3, however, task-structure similarity alone does not pin down ρ when the backbone changes: the same environment can sit on opposite sides of p_I^* under different LLMs (Appendix G.6). We therefore treat this cross-setting consistency as a complementary observational check rather than a standalone falsifiable prediction; the causal evidence in P2 is the more decisive test of the mixture-driven account.

G.6 Multi-Backbone Verification

We validate our core findings on three backbones from three vendors: Qwen3-4B-Instruct (Alibaba, 4B), Phi-3.5-mini-instruct (Microsoft, 3.8B), and Llama-3.1-8B-Instruct (Meta, 8B) across all 6 environments. Table 2 (§5.2) reports full numerical results, and Figure 7 below visualizes the Qwen3-4B Pareto frontier.

This appendix supports the (environment, backbone)-indexed formulation of §3.3: the same task changes p_I (and, when it crosses p_I^*, the sign of ρ) when the backbone changes, so backbone-level reversal is a prediction of the two-source model rather than a confound to be controlled for.

Figure 8 visualizes this direction variation, with black-outlined cells marking sign flips. The downstream consequence appears in Figure 9: on Phi-3.5 FEVER, most fixed-direction baselines score at or below base_only (7.2% base $\rightarrow$ 8.5–23% across baselines), while DIAL adapts to the different sign and reaches 20.5%. The fixed-direction assumption is brittle when the model changes; DIAL's per-(environment, backbone) direction discovery handles this transparently.

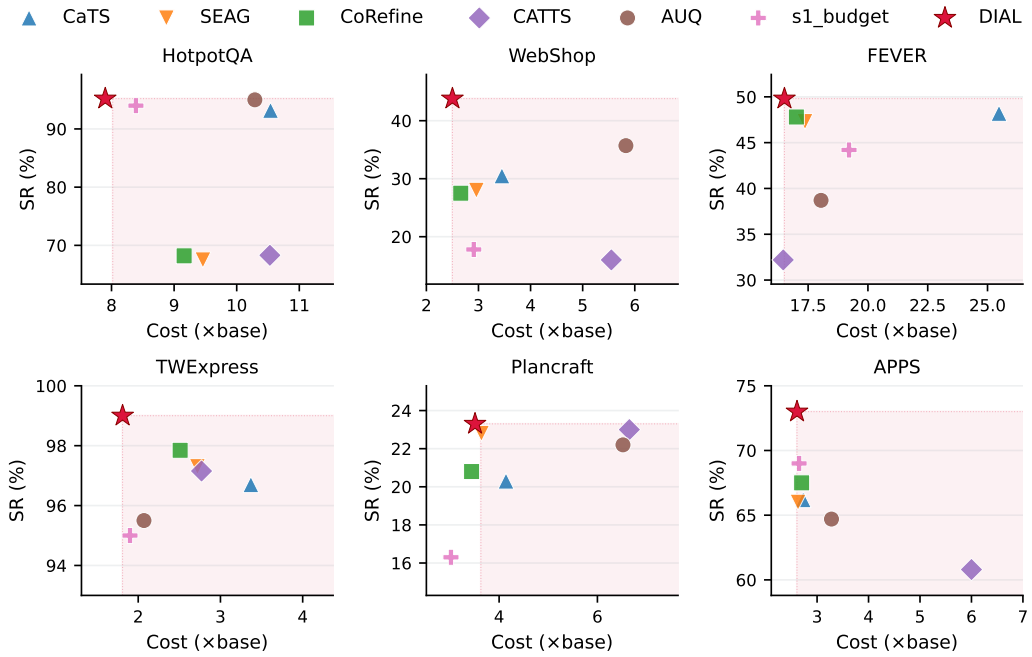


Figure 7: SR vs. Cost (×base) across all 6 environments on Qwen3-4B. Shaded region: dominated by DIAL (lower SR *and* higher cost). Full numerical results across three backbones are in Table 2.

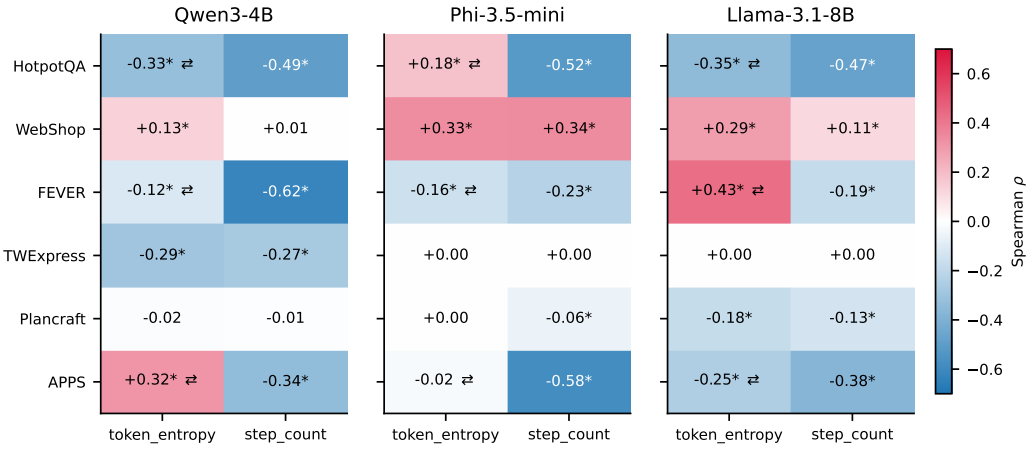


Figure 8: Multi-backbone signal direction heatmap. * marks cells significant at $p<0.05$. Cells marked with ⇄ are *sign-flip cells*: the same (environment, signal) pair yields a significantly positive ρ on at least one backbone and a significantly negative ρ on another, i.e. the signal–utility direction is reversed across model families. Entropy direction is model-dependent; structural signals (`step_count`) are more stable.

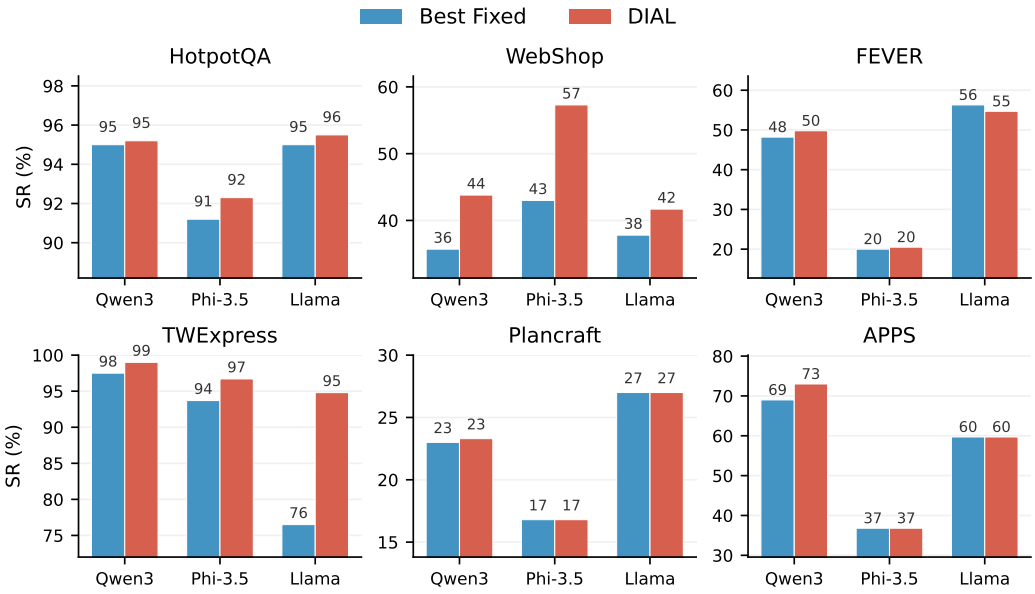


Figure 9: Cross-backbone comparison: DIAL (red) vs. best fixed-direction baseline (blue) across 3 backbones. Fixed baselines show large SR variance across backbones; DIAL adapts robustly.

G.7 Evaluation on a Newer Backbone: Llama 4 Maverick

To check whether DIAL's advantage persists on a more recent model than the three main backbones, we evaluate Llama 4 Maverick on WebShop and APPS. The agent setup, optimizer, prompts, decoding configuration, and evaluation protocol are identical to the main experiments. We compare against three adaptive baselines that cover calibrated confidence, verbalized uncertainty, and fixed budget allocation, together with the two reference bounds. Table 23 reports the results. This evaluation tests DIAL's SR–cost trade-off; the direction-reversal analysis of §3 remains based on the three main backbones.

Table 23: Results on Llama 4 Maverick. SR is mean ± std over seeds; Tokens is the mean per episode; Cost is normalized by base_only.

Env	Method	SR (%)	Tokens	Cost (×base)
WebShop	CaTS	40.5 ± 2.2	32,196	3.61
	AUQ	42.3 ± 1.9	37,787	4.24
	s1_budget	17.8 ± 9.6	18,844	2.11
	DIAL	$\mathbf{47.2 \pm 3.7}$	**14,780**	**1.66**
	base_only	27.2 ± 1.4	8,912	1.00
	always_trigger	53.0 ± 5.1	45,111	5.06
APPS	CaTS	76.2 ± 0.8	5,488	3.69
	AUQ	74.7 ± 0.3	7,154	4.81
	s1_budget	79.0 ± 1.3	5,588	3.76
	DIAL	$\mathbf{82.3 \pm 2.6}$	5,488	3.69
	base_only	71.7 ± 0.6	1,486	1.00
	always_trigger	87.7 ± 2.8	7,074	4.76

DIAL achieves the highest SR among adaptive methods in both environments. On WebShop it also has the lowest cost, and it comes within 5.8 points of always_trigger while using about one third of its tokens. On APPS it exceeds CaTS by 6.1 points at the same token cost and s1_budget by 3.3 points at slightly lower cost, closing roughly two thirds of the gap between base_only and always_trigger.

These results also give a within-family comparison across model generations. With Llama-3.1-8B, DIAL reaches 41.7% SR at 1.68× on WebShop and 59.7% at 4.29× on APPS (Table 2). With Llama 4 Maverick the corresponding numbers are 47.2% at 1.66× and 82.3% at 3.69×. The SR–cost trade-off therefore remains favorable across both generations, with the larger gain on APPS. A systematic longitudinal study would compare additional checkpoints from one family under matched conditions and separate model-generation effects from post-training effects; we leave this to future work.

G.8 Environment-Adaptive Trigger Behavior

Beyond direction and feature selection, DIAL's gating magnitude also adapts to each environment without explicit programming. Figure 10 shows the per-step trigger rate across all 6 environments, ordered by rollout headroom Δ. In rollout-safe TWExpress ($\Delta{=}+31.8$), DIAL triggers aggressively (RR = 73%), consistent with the high headroom. In rollout-harmful Plancraft ($\Delta{=}-7.0$), the trigger rate decays from 49% at step 0 to $<$20% in later steps, as the gate learns that late-step rollouts are increasingly harmful. WebShop and APPS show moderate, stable trigger rates. This full behavioral range emerges entirely from the per-environment signed weights fitted in §4.3: DIAL never receives an explicit headroom estimate, because signal direction already encodes whether the environment rewards or penalizes additional computation.

H Broader Impact

This work addresses computational efficiency in LLM agent deployment. By reducing unnecessary optimizer invocations, DIAL lowers the energy and cost footprint of test-time compute without sacrificing task performance. The prescriptive framework (characterize information structure before designing gates) may help practitioners avoid deploying fixed-direction methods that actively

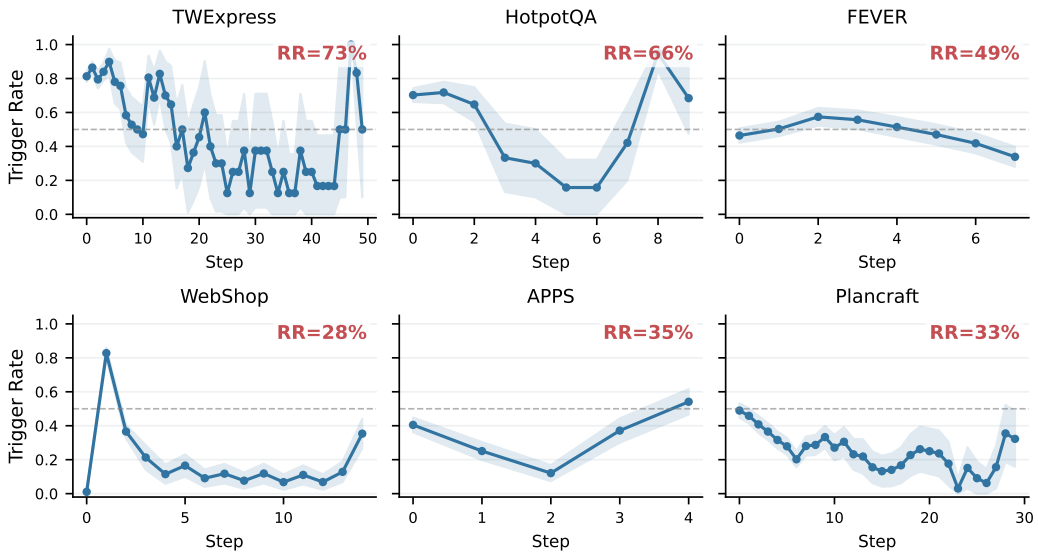


Figure 10: Environment-adaptive trigger behavior. Panels ordered by rollout headroom Δ. Solid line: per-step trigger rate with 95% CI band. Dashed gray: 50% reference. RR: overall trigger rate. DIAL's trigger rate ranges from 73% (TWExpress) to $<$20% (late-step Plancraft), emerging from direction learning alone.

harm performance in mismatched environments. We do not foresee direct negative societal impacts beyond those inherent to the LLM agents themselves.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state three contributions. First, we empirically document the signal-to-utility reversal across environments and backbones. Second, we provide a two-source (Type I and Type D) explanation grounded in Simpson's paradox, together with a necessity result (Proposition 1). Third, we propose the DIAL method. Each contribution is supported by a corresponding section in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 explicitly discusses limitations, including the weakest result on FEVER (high absolute cost due to value concentration in early steps and the limits of random exploration), the one-time per (environment, backbone) exploration cost, and the scope of evaluation (six environments, three backbones).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The two-source model (Section 5.4) and Proposition 1 state all assumptions in their formal statement; a complete proof of Proposition 1 is provided in the appendix, with a proof sketch and intuition included in the main text.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 fully describes the DIAL algorithm; Section 5 specifies the six environments, three backbones, baselines, and evaluation protocol; the appendix provides hyperparameters, prompts, data splits, and exploration details. The code is publicly released at `https://github.com/zmli6/DIAL`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We publicly release our code at `https://github.com/zmli6/DIAL`, including training scripts, evaluation pipelines, configuration files, and a README with reproduction instructions for all six environments. All datasets used (HotpotQA, APPS, WebShop, FEVER, TextWorld Express, Plancraft) are publicly available; preparation steps are documented in the repository.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3.1 reports the core experimental setup (environments, backbones, baselines, splits). Appendix C and Appendix D provide full hyperparameters, optimizer choices, exploration budgets, and the rationale for selection.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All main results are averaged over three random seeds, and we report standard deviations across seeds in the corresponding tables and figures. The reported error bars capture variability from random seeds, which control exploration sampling and gate initialization.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments are conducted on NVIDIA A100 (40GB) GPUs. Appendix D.5 reports the compute breakdown (model serving, exploration rollouts, and gate training) and an estimate of the total A100 GPU-hours used, including preliminary runs not reported in the main paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics. The research uses publicly available datasets and open-source LLMs, and involves no human subjects.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Adaptive compute reduces inference cost and energy consumption of LLM agents, with positive sustainability implications. Potential negative impacts are limited and indirect: more efficient agents could lower the cost barrier to deploying autonomous LLM systems in high-stakes settings; we view this as an application-level concern rather than a property of the gating mechanism itself.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any new pre-trained models, generative models, or scraped datasets. The released artifact is gating code and configurations operating on top of publicly available LLMs and benchmarks; it carries no incremental misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (HotpotQA, APPS, WebShop, FEVER, TextWorld Express, Plancraft) and model backbones (Qwen3, Llama-3.1, Phi-3.5) are cited in the paper, and Appendix D.6 lists each asset together with its source and license. We use each asset within the terms of its license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The DIAL implementation, training scripts, and configuration files are publicly released at `https://github.com/zmli6/DIAL` with a README documenting installation, data preparation, training, and evaluation across all six environments.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so no IRB approval is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs (Qwen3, Llama-3.1, Phi-3.5) appear in this paper only as the agent backbones being studied, which is a standard usage in LLM-agent research. No LLM is used as an original or non-standard component of the methodology, such as a judge, label generator, or synthetic data source. LLM usage for writing assistance is outside the scope of this declaration.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.