BRANCHCAST: PLANNING OVER NAMED TOOL OUTCOMES BEFORE LATENT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A timeout and a silent-wrong completion may be neighbors in a learned latent space, yet a planner should retry one and reject the other. BranchCast preserves this decision-relevant distinction by predicting a named tool outcome—success, invalid call, retry/timeout, or silent error—with recoverability and severe-failure probability before generating within-outcome latent variation. We test the complete interface with a crossed comparison that fixes the encoder, proposal and scoring networks, beam search, horizon, four generator evaluations, and a 20–22 ms active-transition latency band. Across five paired seeds and 2,250 held-out WebArena, non-weather ToolBench, and code-execution decisions, replacing implicit-branch diffusion with BranchCast raises success by +6.5 points (95% CI [+5.4,+7.7]) and lowers severe failure by -6.2 points (95% CI [-7.5,-5.0]). The success gain remains +5.0 points when severity reranking is disabled and grows from +2.6 to +10.8 points across training-defined branch-load quartiles. The interface also changes model selection: rectified flow leads diffusion by 4.0 points behind an implicit interface, whereas diffusion leads same-scaffold rectified flow by +0.9 points once both expose named outcomes. Outcome structure is therefore part of the world-model contract with search, not merely an auxiliary prediction target.

1 INTRODUCTION

Consider two imagined continuations of the same tool call. Their latent residuals can be equally plausible, but one encodes a timeout with budget left and the other a syntactically valid wrong answer marked complete. Beam search should retry the former and prune the latter. Malformed calls create a third case in which fluent next-state prediction offers no protection against an unrecoverable action. Such branches recur in browser, API, and code agents (Schick et al., 2023; Yao et al., 2023; Nakano et al., 2021); collapsing them before search can place an invalid future above a recoverable one.

Most learned world models expose a generated state, reward, or generic termination signal to planning. Multimodal residuals can represent several futures, but an unnamed component does not tell beam search which tool consequence was sampled or whether recovery remains possible. This leaves the transition interface and residual generator entangled: a strong density model may still hide the variable that determines the next action.

BranchCast turns tool consequences into a planner-facing transition variable. Given an observation–action pair, it predicts one of four mutually exclusive outcomes—success, invalid call, retry/timeout, or silent error—plus branch-conditioned recoverability and severe-failure probability. The planner receives this tuple before a four-evaluation conditional residual generates variation inside the selected outcome. Validators provide labels during training and judge realized episodes during evaluation; no validator output enters planning.

Our crossed interface–generator protocol measures the resulting decision effect. It combines implicit or explicit branches with diffusion or rectified-flow residuals while fixing the encoder, proposal and task-scoring networks, search algorithm, horizon, generator-call budget, and active-transition latency band. The explicit-versus-implicit contrast changes the *complete outcome interface*—supervision, categorical sampling, conditional generation, and planner exposure—whereas the severity-off intervention changes only risk reranking. The crossed residual cells then reveal whether a generator comparison transfers across interfaces.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The answer is consequential. On five paired seeds and 2,250 held-out branch-heavy decisions, BranchCast moves diffusion success from 43.0 to 49.5 and severe failure from 21.3 to 15.1. Gains occur in WebArena, non-weather ToolBench, and code execution, persist without severity reranking, and rise from +2.6 to +10.8 points across branch-load quartiles. Moreover, the diffusion–flow ordering reverses: flow leads by 4.0 points with implicit branches, but diffusion leads by +0.9 points with explicit branches. Generator quality is therefore conditional on what the transition model makes legible to search.

Contributions. The central contribution is a named-outcome transition factorization that makes tool consequences available before search ranks a future. A crossed, four-evaluation protocol separates the complete interface effect from generator operating points and severity reranking. Five-seed evidence across three suites then shows improved decisions, an interface-dependent diffusion–flow ordering, and a monotone response to branch load.

2 Related Work and Technical Positioning

World-model interfaces. Recurrent and latent world models compress dynamics for planning (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; 2025); MDNs, ensembles, and discrete latents represent multimodality or epistemic variation (Bishop, 1994; Lakshminarayanan et al., 2017; Micheli et al., 2023). A sampled component can influence the next state without carrying evaluator-aligned semantics or remaining observable when search ranks candidates. BranchCast differs along both axes: its branch is supervised with named tool outcomes, and its probability, recoverability, and severity remain planner-visible before residual generation. Thus the branch is a decision interface rather than an unnamed density component or auxiliary post-generation classifier.

Generative dynamics and planning. Diffusion and score models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) support trajectory synthesis and world modeling (Janner et al., 2022; Ajay et al., 2023; Ding et al., 2024; Rigter et al., 2024); flow matching and rectified flow provide continuous-time alternatives (Lipman et al., 2023; Liu et al., 2023). Learned planners such as MuZero and TD-MPC likewise emphasize decision-aligned dynamics (Schrittwieser et al., 2020; Hansen et al., 2022). Prior operating points typically choose representation, interface, and generator together. Our crossed cells instead place diffusion and rectified flow behind each of two branch interfaces with a shared planner and four generator evaluations, making their interaction directly measurable.

Tool failures, risk, and calibration. WebArena, Mind2Web, WebVoyager, WorkArena, AgentBench, and BrowserGym expose invalid actions, stochastic retries, and silent failures (Zhou et al., 2024; Deng et al., 2023; He et al., 2024; Drouin et al., 2024; Liu et al., 2024; de Chezelles et al., 2025). Calibration separates confidence from accuracy (Guo et al., 2017; Minderer et al., 2021), while risk-aware systems use forecasted failure to guide computer-using agents (Chen et al., 2026) or evaluate optimism in world models (Yang et al., 2026); LeFlow targets efficient latent generation (Huang et al., 2026). BranchCast unifies four validator-observable consequences as a training target, exposes the predicted distribution to search, and uses a severity-off intervention to distinguish outcome factorization from risk-score reranking.

3 BranchCast and Commit-Before-Generate Modeling

3.1 Planner-Facing Outcome Factorization

For observation o_t and action a_t, a frozen encoder produces $\mathbf{z}_t = \phi(o_t, a_t)$. Let s_{t+1} denote the adapter's fixed-dimensional next-state representation rather than raw webpage or execution text. BranchCast factors the transition through outcome branch b_t and recoverability v_t:

$$p_\theta(s_{t+1} \mid \mathbf{z}_t) = \sum_{b=1}^{B} \sum_{v \in \{0,1\}} p(b_t = b \mid \mathbf{z}_t) p(v_t = v \mid b_t = b, \mathbf{z}_t) p(s_{t+1} \mid b_t = b, v_t = v, \mathbf{z}_t), \quad (1)$$

where $B = 4$ denotes `success`, `invalid_call`, `retry/timeout`, and `silent_error`. Search receives the branch distribution, branch-conditioned recoverability, and severe-failure probability $\hat{e}_t \in [0, 1]$ before a residual is produced; the final factor represents variation within the

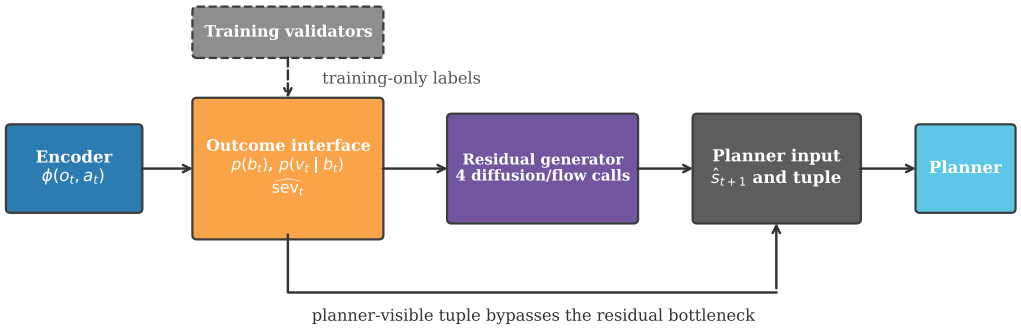

Figure 1: BranchCast exposes a predicted outcome tuple before within-branch generation. Benchmark validators provide training-only targets (dashed path); the deployment interface contains branch probability, recoverability, severe-failure probability, and the imagined state. Diffusion and rectified flow occupy the same four-evaluation generator boundary.

selected outcome. This commit-before-generate order preserves incompatible continuations as named hypotheses when the planner ranks trajectories and leaves the residual generator replaceable.

A two-layer MLP predicts $p_\theta(b_t \mid \mathbf{z}_t)$; two branch-conditioned heads predict $p_\theta(v_t \mid b_t, \mathbf{z}_t)$ and $p_\theta(e_t = 1 \mid b_t, v_t, \mathbf{z}_t)$. A sampled rollout uses $\hat{e}_t = p_\theta(e_t = 1 \mid b_t, v_t, \mathbf{z}_t)$; planner-independent severe-event AUPRC uses its pre-sampling marginal $\bar{e}_t = \sum_{b,v} p_\theta(b \mid \mathbf{z}_t) p_\theta(v \mid b, \mathbf{z}_t) p_\theta(e_t = 1 \mid b, v, \mathbf{z}_t)$. The suite adapter maps realized and imagined states into the same scorer and proposal input space, while g_{decode} maps the latent update and sampled (b_t, v_t) back to that representation. BranchCast models the residual $\boldsymbol{\varepsilon}_t$ with

$$q(\boldsymbol{\varepsilon}_t^{(k)} \mid \boldsymbol{\varepsilon}_t^{(0)}) = \mathcal{N}\left(\sqrt{\bar{\alpha}_k}\boldsymbol{\varepsilon}_t^{(0)}, (1 - \bar{\alpha}_k)\mathbf{I}\right), \tag{2}$$

$$\epsilon_\theta(\boldsymbol{\varepsilon}_t^{(k)}, k, \mathbf{z}_t, b_t, v_t) \approx \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{3}$$

and decodes $\hat{s}_{t+1} = g_{\text{decode}}(\mathbf{z}_t + \hat{\boldsymbol{\varepsilon}}_t, b_t, v_t)$. The residual target is the difference between the adapter-encoded realized next state and $\mathbf{z}_t$; every residual family uses this target and decoder. Progressive distillation followed by consistency fine-tuning reduces 50 denoising steps to four (Salimans & Ho, 2022; Song et al., 2023).

3.2 Training Targets and Planner Contract

The joint objective is

$$\mathcal{L} = \mathcal{L}_{\text{branch}} + \lambda_v \mathcal{L}_{\text{recover}} + \lambda_e \mathcal{L}_{\text{severity}} + \lambda_\epsilon \mathcal{L}_{\text{diff}} + \lambda_c \mathcal{L}_{\text{cal}}, \tag{4}$$

with $\lambda_v = \lambda_e = \lambda_\epsilon = 1.0$ and $\lambda_c = 0.1$. The first loss is four-way cross-entropy; $\mathcal{L}_{\text{recover}}$ and $\mathcal{L}_{\text{severity}}$ are binary cross-entropies; $\mathcal{L}_{\text{diff}}$ is denoising MSE; and $\mathcal{L}_{\text{cal}}$ is the multiclass Brier score. A transition is recoverable when it is nonterminal and the remaining action budget permits another attempt. The severity label is one exactly for an unrecoverable invalid call, an exhausted retry, or a silent-wrong terminal completion (Section 4). Implicit-interface models retain identically weighted recoverability and severity losses, but predict both scalars directly from $\mathbf{z}_t$.

HTTP status codes, exception traces, timeout logs, and offline correctness checks assign mutually exclusive branch targets. The precedence is: a completed output rejected only by the offline correctness check is `silent_error`; a request rejected before a valid tool response or ending in a terminal exception is `invalid_call`; a timeout or retry signal with no terminal output is `retry/timeout`; all validator-accepted transitions are `success`. This decision order resolves traces containing more than one diagnostic signal. These signals supervise training and label evaluation outcomes; none is available during planning.

The frozen proposal model is trained once on successful training trajectories with next-action cross-entropy. For suite j, its adapter serializes the real observation and execution history, the partial

action sequence, and the current decoded imagined state into the same input fields used at training time. The proposal emits four ranked actions; its parameters and top-4 rule are shared, while action identities may differ because imagined states differ across transition models. A separate frozen scorer $q_\psi(\hat{s}_t) \in [0, 1]$, trained by binary cross-entropy on successful versus unsuccessful training prefixes, assigns progress to each decoded state. At search depth d, every prefix contains d score terms: a prefix that terminates earlier is deterministically padded with absorbing copies of its terminal state and severity, without further proposal or transition calls. We rank prefixes by

$$S_d(\tau) = \frac{1}{d} \sum_{t=1}^{d} \left[q_\psi(\hat{s}_t) - \lambda_{\text{sev}} \hat{e}_t \right], \qquad \lambda_{\text{sev}} = 0.30. \tag{5}$$

The common denominator prevents terminal frequency from changing the relative scaling of task and severity terms. Search never compares scores across suites, so no cross-suite normalization enters action selection; validators label only realized evaluation outcomes.

Starting from the real observation, each active beam node is adapter-encoded with each proposed action. Four rollout replicas use common random numbers keyed by seed, task, depth, proposal rank, and replica. Explicit variants sample (b_t, v_t), generate a conditional residual, decode $\hat{s}_{t+1}$, and feed its adapter serialization to the next proposal step; implicit variants generate an unconditional residual and direct scalar predictions using the same keys. Equation 5 retains the best four prefixes for three depths. Absorbing terminal prefixes remain eligible but are not model-expanded; ties use lower proposal rank and replica index. The executed action is the root of the final best prefix. Thus the proposal/scoring machinery and random coupling are shared, not the model-dependent deeper action identities.

Both rectified-flow cells use the same straight-line flow-matching target (Liu et al., 2023), four-layer width-512 residual backbone, progressive solver-distillation schedule, and four explicit Euler evaluations. Branch+Rectified-Flow adds only the explicit-interface inputs and heads used by BranchCast; implicit flow predicts the two scalar heads from $\mathbf{z}_t$. The explicit model has 34.7 M trainable transition parameters and runs in 21.4 ms per active imagined transition. Parameter counts are reported because the protocol matches architecture depth, planner, calls, and latency rather than total parameters.

Branch+Gaussian uses the same explicit interface and decoder. Its residual head predicts a diagonal Gaussian mean and log variance, is trained by Gaussian negative log likelihood, and draws four paired samples per candidate so that the number of residual forward evaluations matches the four-step diffusion and flow variants.

3.3 Crossed Interface–Generator Contrasts

For metric M, branch representation b, and residual family r, we define

$$\Delta^{r}_{\text{branch}} = M(\text{explicit branch}, r) - M(\text{implicit branch}, r), \tag{6}$$

$$\Delta^{b}_{\text{residual}} = M(b, \text{diffusion}) - M(b, \text{flow}), \tag{7}$$

$$\Delta_{\text{severity}} = M(\text{severity on}) - M(\text{severity off}). \tag{8}$$

Positive success differences and negative severe-failure differences indicate improvement. Every contrast uses paired tasks under the same encoder, planner, horizon, beam width, checkpoint rule, four-call residual budget, and active-transition latency band. The first is the complete explicit-interface effect: branch supervision, categorical sampling, residual conditioning, and planner exposure form one intervention. The second compares the implemented diffusion and flow operating points within a fixed interface, and the third changes only the severity term in Equation 5.

4 Experimental Design

The evaluation asks four questions: Does the complete named-outcome interface improve selected actions across domains? Does its effect survive removal of severity reranking? Does training-defined branch load predict its value? Does the diffusion–flow ordering transfer across interfaces?

Table 1: Crossed intervention design. Diffusion and rectified flow form the four-call interface–generator core; Gaussian and severity-off rows are targeted controls.

Component	Specification
Fixed inputs	Held-out task pairs, suite adapters, proposal/scorer networks, decoder space, beam width 4, horizon 3, random-number keys, four calls per active transition, and 20–22 ms call band.
Crossed core	Implicit/explicit branch interface $\times$ four-step diffusion/rectified-flow residual.
Mechanism controls	Explicit branch+Gaussian; severity scoring on/off in Equation 5.
Primary outputs	Paired changes in episode success and severe failure, with branch macro-F1, severe-event AUPRC, and branch ECE as planner-independent checks.
Regime output	Complete-interface response to training-defined branch load; model-level equivalence on schema-stable slices.

Suites and fixed splits. Each seed evaluates the same 100 held-out WebArena task IDs (Zhou et al., 2024), 200 non-weather ToolBench API-task IDs (Qin et al., 2024), and 150 evaluation-only code-problem IDs from `openai/human-eval` (Chen et al., 2021). Training and validation trajectories are partitioned by task ID and episode before labels are constructed; no test transition selects a checkpoint or inclusion stratum. The experimental unit is a seed–task decision: 450 per seed and 2,250 over five paired seeds. A seed–task family enters the primary manifest when its corresponding training partition contains at least two outcome classes and either branch entropy exceeds 0.35 or severe-outcome prevalence exceeds 10%. Consequently, the same task can occupy different load quartiles across training seeds, yielding the 563/563/562/562 decision supports reported below. Weather APIs are reserved for the low-branch analysis. We use WebArena v0.2.0, the ToolBench 2023-09-29 ToolEval update, `openai/human-eval`, and `microsoft/deberta-v3-large` under their stated Apache-2.0 or MIT licenses and provider terms.

Models and optimization. All variants use a frozen DeBERTa-v3-large encoder (He et al., 2023) over DOM or response text, tool arguments, schema hints, and the latest five execution steps. The transition backbone is a four-layer Transformer (Vaswani et al., 2017) with hidden size 512, eight heads, and 35.2 M trainable parameters for BranchCast. Training uses Adam (Kingma & Ba, 2015), learning rate 3×10^{-4}, batch size 256, cosine decay, and 100K updates on eight A100 80GB GPUs with CUDA 12.4. Failure trajectories are oversampled $5\times$. Diffusion and flow teachers follow the same $50 \to 16 \to 8 \to 4$ progressive evaluation schedule; diffusion additionally receives consistency fine-tuning at four steps.

Implicit-branch diffusion removes the categorical loss, sampled branch token, branch-conditioned residual input, and planner-visible branch distribution, while retaining direct recoverability and severity heads with the losses in Equation 4. Its contrast with BranchCast therefore measures the complete explicit factorization, not visibility alone. The implicit and explicit flow cells use the common rectified-flow procedure described above and differ by the same interface modules; Branch+Gaussian retains the explicit interface and changes the residual distribution. Component deletions retrain under the common schedule: no-recoverability removes its head and loss, no-calibration sets $\lambda_c = 0$, and no-oversampling restores empirical transition frequencies. The 34.8 M diffusion and 28.7 M flow counts disclose a capacity difference between the implemented generators; $\Delta^b_{\text{residual}}$ is defined at their fixed four-call operating points. RSSM, MDN-8, Ensemble-5, and ten-step Flow-Match provide additional three-seed operating-point context.

Metrics and aggregation. Success is an episode accepted by the benchmark validator. Severe failure is an unrecoverable invalid transition, exhausted retry loop, or silent-wrong terminal completion; the last executes but fails hidden correctness tests. Rates are micro-averaged over paired episodes within suite and seed and then over the displayed denominators. In 10,000 hierarchical paired-bootstrap replicates, we resample five training seeds and then unique held-out tasks within suite, retaining paired outcomes and clustering repeated rollouts by task. ECE uses 15 equal-width bins. Macro-F1 uses the fixed transition labels, and severe-event AUPRC uses the branch-marginalized $\bar{e}_t$ defined above.

Table 2: Four-call comparison on the branch-heavy manifest over five paired seeds. Each suite entry is success / severe-failure rate (%). Pooled uncertainty is a paired-bootstrap 95% CI half-width; raw counts cover 2,250 decisions per method. Latency is per active imagined transition.

Variant	WebArena	ToolBench	Code
Implicit-branch diffusion	41.0/24.2	44.5/21.8	42.4/18.7
Branch+Gaussian	42.8/23.0	46.7/19.9	44.5/17.2
Implicit-branch rectified flow	44.2/21.8	48.6/18.2	46.8/16.1
Branch+Rectified-Flow	46.0/19.2	49.9/15.9	48.7/13.5
BranchCast (explicit diffusion)	46.4/18.8	50.8/15.0	49.9/12.8

Variant	Pooled success / severe (%)	Raw success / severe
Implicit-branch diffusion	43.0±1.6 / 21.3±1.7	968/479 of 2250
Branch+Gaussian	45.1±1.4 / 19.7±1.5	1015/443 of 2250
Implicit-branch rectified flow	47.0±1.3 / 18.3±1.4	1058/412 of 2250
Branch+Rectified-Flow	48.6±1.1 / 15.8±1.2	1094/356 of 2250
BranchCast (explicit diffusion)	49.5±1.0 / 15.1±1.1	1114/340 of 2250

Variant	Active transition (ms)	Residual budget	Parameters (M)
Implicit-branch diffusion	20.7	4 denoiser calls	34.8
Branch+Gaussian	21.0	4 forward calls	31.9
Implicit-branch rectified flow	20.0	4 Euler calls	28.7
Branch+Rectified-Flow	21.4	4 Euler calls	34.7
BranchCast (explicit diffusion)	21.0	4 denoiser calls	35.2

Checkpoint and latency protocol. For every family, one checkpoint is chosen before test evaluation by the lowest validation objective averaged across suites; test success never selects sampler depth. Latency is the median for one active imagined transition at batch size one on a single A100 80GB GPU, after 200 warm-ups and 1,000 synchronized repetitions per seed. This measurement matches the four-call generator boundary. Beam width, horizon, proposal count, and severity weight remain fixed.

Appendix G summarizes the evaluated failure capabilities, trust boundary, comparators, and supporting measurements.

5 Results: Named Outcomes Change Search

We first quantify the complete interface effect and its interaction with the generator, then separate severity reranking and locate the effect by branch load.

5.1 Complete Interface Gain and Conditional Generator Ranking

Table 2 shows the predicted interaction. Under diffusion, the complete explicit interface gives $\Delta^{\text{diffusion}}_{\text{branch}} = +6.5$ success points (95% CI [+5.4,+7.7]) and -6.2 severe-failure points (95% CI [-7.5,-5.0]). The success / severe effects are +5.4 ([+3.7,+7.1]) / -5.4 ([-7.0,-3.8]) on WebArena, +6.3 ([+5.0,+7.6]) / -6.8 ([-8.1,-5.4]) on ToolBench, and +7.5 ([+5.9,+9.0]) / -5.9 ([-7.3,-4.5]) on code execution. Under rectified flow, the same interface moves success by +1.6 points and severe failure by -2.5 points.

The residual comparison changes sign. With an implicit interface, diffusion trails rectified flow by 4.0 success points (95% CI [+3.1,+4.9]) and incurs +3.0 more severe failures (95% CI [+2.0,+4.1]). With the explicit interface, BranchCast leads same-scaffold rectified flow by +0.9 success points (95% CI [+0.2,+1.7]) and -0.7 severe-failure points (95% CI [-1.3,-0.1]). The difference-in-differences is +4.9 for success (95% CI [+3.6,+6.2]) and -3.7 for severe failure (95% CI [-5.1,-2.4]). The result is an interface–generator interaction: neither residual family has an interface-independent ranking on this manifest.

Table 3: Mechanism evidence on the same 2,250 paired decisions. Severity-off rows have paired intervals in text; component deletions are descriptive point estimates. Diagnostics are measured before search.

Intervention	Success (%)	Severe (%)
BranchCast	49.5	15.1
Implicit-branch diffusion	43.0	21.3
No recoverability head	46.8	18.1
No calibration loss	48.6	16.0
No failure oversampling	46.0	18.9
Branch+Gaussian	45.1	19.7
Branch+Rectified-Flow	48.6	15.8
BranchCast, severity off	47.8	17.2
Implicit-branch diffusion, severity off	42.8	22.0

Branch-explicit residual	Macro-F1	Severe AUPRC	Branch ECE
Gaussian	0.748	0.644	0.049
Rectified flow	0.774	0.681	0.041
BranchCast diffusion	0.781	0.692	0.039

5.2 Most of the Interface Effect Survives Severity Removal

Severity-aware reranking contributes +1.7 success points (95% CI [+0.8,+2.6]) and -2.1 severe-failure points ([-3.0,-1.2]) for BranchCast. With severity disabled for both models, the complete interface still contributes +5.0 success points ([+3.9,+6.1]) and -4.8 severe-failure points ([-6.0,-3.6]). Most of the interface effect therefore reaches search through outcome sampling, conditioning, and recoverability rather than severity reranking alone.

The pre-search diagnostics agree with the downstream order. Relative to same-scaffold rectified flow, BranchCast changes macro-F1 by +0.007 ([+0.002,+0.013]), severe-event AUPRC by +0.011 ([+0.004,+0.019]), and ECE by -0.002 ([-0.004,-0.001]). Descriptively, deleting recoverability supervision, failure oversampling, or calibration changes success by -2.7, -3.5, and -0.9 points, respectively. These point estimates serve as sensitivity checks; paired intervals support the severity and interface conclusions above.

5.3 Branch Load Locates Interface Value

Branch load is fixed before model scoring as $L = 0.6H(b)/\log 4 + 0.4\min(p_{\text{fail}}/0.30, 1)$, where $H(b) = -\sum_{k=1}^{4} \hat{p}_k \log \hat{p}_k$ uses outcome frequencies from each seed's training partition. The 0.6/0.4 weighting prioritizes branch diversity while retaining failure prevalence. Cutoffs 0.22, 0.39, and 0.58 define descriptive Q1–Q4 strata over seed–task decisions, with supports 563, 563, 562, and 562.

From Q1 to Q4, success effects are +2.6 ([+1.2,+4.0]), +5.0 ([+3.5,+6.4]), +7.6 ([+6.0,+9.2]), and +10.8 ([+9.0,+12.6]); severe-failure effects are -2.1, -4.8, -7.4, and -10.5, with intervals in Figure 2 and Table 8. This monotone response ties decision gains to competing outcomes rather than benchmark identity alone.

On the separate schema-stable slices, a predeclared ± 2.0-point margin contains every paired BranchCast difference against Branch+Rectified-Flow and RSSM. ToolBench weather rates are 71.0/71.3/71.8; WebArena direct-navigation rates are 75.8/76.1/75.5. The corresponding intervals appear in Table 7. These comparisons establish model-level equivalence in two low-ambiguity settings; the interface-specific regime evidence is the Q1–Q4 contrast above.

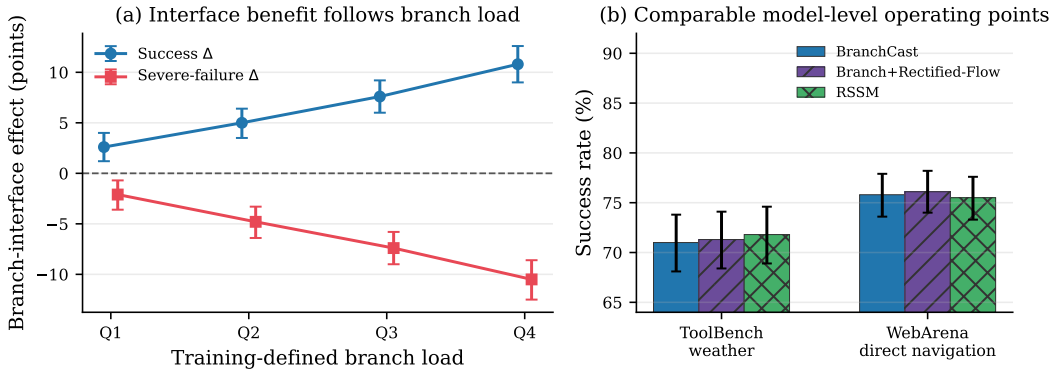

Figure 2: Where outcome structure matters. Left: the complete BranchCast-minus-implicit-diffusion effect grows with training-defined branch load; negative severe-failure changes are favorable. Right: model-level equivalence among BranchCast, branch-explicit rectified flow, and RSSM on schema-stable slices.

5.4 Sampler-Depth Context

The crossed study fixes four evaluations: BranchCast takes 21 ms per active transition and Branch+Rectified-Flow 21.4 ms. Separate depth runs show the calibration–latency trade-off: 50- and eight-step BranchCast reach ECE 0.028 and 0.031 at 145 and 38 ms, versus 0.039 at four steps. Paired eight-minus-50 success differences are +0.9 ([-0.3,+2.0]) on WebArena, +0.7 ([-0.2,+1.6]) on ToolBench, and +0.6 ([-0.5,+1.7]) on code execution. These auxiliary runs contextualize sampler depth; the interface–generator claim comes only from the fixed four-call crossed cells. Appendix F reports all operating points.

6 Implications

BranchCast's largest measured contribution is the +6.5-point complete-interface effect under diffusion; the explicit-interface diffusion–flow contrast is +0.9 points. Severity-off search and planner-independent branch metrics support the same hierarchy. Treating an end-to-end world model as one operating point would hide how the transition interface converts a generated future into a selected action.

The branch-load response supplies an actionable regime map: the interface effect grows from +2.6 points in Q1 to +10.8 in Q4. The separate equivalence slices show that BranchCast, branch-explicit rectified flow, and RSSM are interchangeable at the declared margin when schemas are stable; they do not estimate the implicit-to-explicit interface effect. Together, the analyses direct extra outcome modeling toward tasks where competing consequences can reorder a short plan.

Scope. The evidence covers four validator-derived outcomes, a three-step width-four beam, and branch-heavy browser, API, and code decisions. The +6.5-point estimate belongs to the complete interface—supervision, sampling, conditioning, and planner exposure together—while the severity intervention isolates reranking. The timing control matches active-transition calls and latency, whereas terminal-dependent root-decision latency can vary; the paired search policy and absorbing score treatment keep the decision comparison fixed. These boundaries make the protocol reusable for other outcome taxonomies and planners without asserting unchanged effect sizes.

7 Conclusion

BranchCast makes named tool outcomes available when search can still act on them. Across branch-heavy browser, API, and code decisions, its complete interface raises success by +6.5 points, lowers severe failure by -6.2 points, persists without severity reranking, and grows with branch load. The

diffusion–flow ordering moves from -4.0 to +0.9 success points across interfaces. World-model selection should therefore evaluate the outcome contract and generator together.

AI USE STATEMENT

Generative-AI assistance was used during manuscript preparation, including editorial refinement and formatting and consistency checks.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2211.15657.

Christopher M. Bishop. Mixture density networks. Technical Report NCRG/94/004, Aston University, 1994. URL https://publications.aston.ac.uk/id/eprint/373/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Yurun Chen, Zeyi Liao, Ping Yin, Taotao Xie, Keting Yin, and Shengyu Zhang. SafePred: A predictive guardrail for computer-using agents via world models. *arXiv preprint arXiv:2602.01725*, 2026. URL https://arxiv.org/abs/2602.01725.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023. URL https://arxiv.org/abs/2306.06070.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning. *arXiv preprint arXiv:2402.03570*, 2024. URL https://arxiv.org/abs/2402.03570.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, volume 70, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2451–2463, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, volume 97, pp. 2555–2565, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, volume 162, pp. 8387–8406, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, 2024. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://arxiv.org/abs/2006.11239.

Hsiang-Wei Huang, Jianxu Shangguan, Junbin Lu, and Jenq-Neng Hwang. LeFlow: Generative latent flow planning for world models. *arXiv preprint arXiv:2608.24855*, 2026. URL https://arxiv.org/abs/2608.24855.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, volume 162, pp. 9902–9915, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://openreview.net/forum?id=8gmWwjFyLj.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6402–6413, 2017. URL https://arxiv.org/abs/1612.01474.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.00588.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15682–15694, 2021. URL https://arxiv.org/abs/2106.07998.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.16789.

Marc Rigter, Jun Yamada, and Ingmar Posner. World models via policy-guided trajectory diffusion. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2312.08533.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://arxiv.org/abs/2302.04761.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038/s41586-020-03051-4.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, volume 37, pp. 2256–2265, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, volume 202, pp. 32211–32252, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://arxiv.org/abs/1706.03762.

Tianzhuo Yang, Zihan Shen, Zirui Mi, Zhaoyi Zhang, Jiayi Zhou, Jiaming Ji, Juntao Dai, Jiawei Chen, Boyuan Chen, and Yaodong Yang. MiraBench: Evaluating action-conditioned reliability in robotic world models. *arXiv preprint arXiv:2605.29360*, 2026. URL https://arxiv.org/abs/2605.29360.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zlsj9akpaa.

A Training and Comparator Details

The branch, recoverability, and severity predictors are two-layer MLPs with hidden dimension 512 and GELU activations. The residual backbone uses four Transformer layers, eight attention heads, hidden dimension 512, and sinusoidal timestep embeddings. Full BranchCast training takes approximately 18 hours on 8 NVIDIA A100 80GB GPUs. Distillation follows interface conditioning so every sampler depth preserves identical planner-visible outputs.

Both flow cells use the same latent residual target. For data residual ε_1 and Gaussian source ε_0, they sample $u \sim \mathrm{Uniform}(0,1)$, construct $\varepsilon_u = (1-u)\varepsilon_0 + u\varepsilon_1$, and regress velocity $\varepsilon_1 - \varepsilon_0$ before progressive solver distillation to four Euler intervals. The explicit cell additionally receives (b_t, v_t) and the branch losses. Hidden width, attention pattern, training transitions, planner, and validator boundary remain fixed.

B Branch Prediction Support and Confusion

The fixed planner-independent transition set selects one designated decision-point transition from each primary episode and contains 1180 success, 402 invalid-call, 337 retry/timeout, and 331 silent-error labels. Table 4 reports BranchCast predictions; rows are benchmark-side labels and columns are model predictions.

Table 4: Four-way BranchCast confusion matrix on one designated decision-point transition from each of 2,250 held-out primary episodes.

Actual / predicted	Success	Invalid	Retry	Silent
Success	1028	50	42	60
Invalid call	39	311	35	17
Retry/timeout	32	37	252	16
Silent error	42	20	18	251

The confusion matrix yields macro-F1 after classwise averaging. Severe-event AUPRC treats invalid call, exhausted retry/timeout, and silent error as positive outcomes and evaluates the branch-marginalized $\bar{e}_t$.

C Raw Primary Counts

Table 5: Per-suite raw success / severe counts over five paired seeds. Denominators are 500 WebArena, 1,000 ToolBench, and 750 code-execution decisions. The final column gives silent-wrong code counts.

Variant	WebArena	ToolBench	Code	Code silent
Implicit-branch diffusion	205/121	445/218	318/140	73
Branch+Gaussian	214/115	467/199	334/129	65
Implicit-branch rectified flow (FMD)	221/109	486/182	351/121	58
Branch+Rectified-Flow	230/96	499/159	365/101	52
BranchCast	232/94	508/150	374/96	49

Table 6: Failure-category decomposition for the strongest four-call comparator and BranchCast. Counts are invalid / retry for WebArena and ToolBench, and unrecoverable execution / silent-wrong for code. Each pair sums to Table 5.

Variant	WebArena	ToolBench	Code execution
Branch+Rectified-Flow	51/45	88/71	49/52
BranchCast	48/46	82/68	47/49

D Low-Branch Boundary Counts

The ToolBench weather slice has 100 distinct episodes and WebArena direct navigation has 150. Each receives ten stochastic rollout replicates across five paired seeds. Cluster bootstrapping resamples episodes and preserves within-episode outcomes. Because all displayed methods except RSSM use the explicit interface, this appendix supports model-level equivalence rather than an implicit-to-explicit contrast.

Table 7: Low-branch success rates, cluster-bootstrap marginal 95% intervals, and raw successful rollouts. Paired-difference intervals for the equivalence test are reported in Section 5.

Model	ToolBench weather	WebArena direct navigation
BranchCast	71.0% [68.1,73.8], 710/1000	75.8% [73.6,77.9], 1137/1500
Branch+Rectified-Flow	71.3% [68.4,74.1], 713/1000	76.1% [74.0,78.2], 1142/1500
RSSM	71.8% [68.9,74.6], 718/1000	75.5% [73.3,77.6], 1133/1500

E Branch-Interface Effects by Load and Seed

Table 8 applies the complete explicit-interface estimand to 2,250 seed–task decisions, comparing BranchCast with implicit-branch diffusion. Quartiles are assigned separately within each training seed from training-partition statistics; supports therefore count decisions, not unique task IDs. Their weighted effects recover the pooled +6.5 success and -6.2 severe-failure changes after rounding.

Table 8: Complete explicit-interface effect within diffusion, by branch-load stratum. Intervals use the hierarchical paired bootstrap.

Stratum	Support	Success effect (95% CI)	Severe effect (95% CI)
Q1	563	+2.6 [+1.2,+4.0]	-2.1 [-3.6,-0.7]
Q2	563	+5.0 [+3.5,+6.4]	-4.8 [-6.4,-3.3]
Q3	562	+7.6 [+6.0,+9.2]	-7.4 [-9.0,-5.8]
Q4	562	+10.8 [+9.0,+12.6]	-10.5 [-12.5,-8.6]

For deployment-level selection, Table 9 compares BranchCast with the strongest implicit-branch four-step system, Flow-Match-Distilled, on a separate balanced set of 3200 held-out transition decisions. This contrast jointly reflects interface and residual changes and is therefore reported separately from the within-diffusion interface estimand in Figure 2.

All five seeds move both endpoints in the same favorable direction. The smallest/largest success effects are +5.8/+6.9 points, while severe-failure reductions range from -5.7 to -6.6 points; the pooled bootstrap remains the inferential summary.

Table 9: Deployed BranchCast-minus-Flow-Match-Distilled response by branch load. Each stratum contains 800 held-out decisions.

Stratum	Success effect (95% CI)	Severe effect (95% CI)
Q1	+0.4 [-0.5,+1.2]	-0.3 [-1.0,+0.4]
Q2	+2.1 [+1.1,+3.0]	-1.7 [-2.6,-0.8]
Q3	+4.6 [+3.3,+5.8]	-3.5 [-4.7,-2.3]
Q4	+6.9 [+5.3,+8.4]	-6.1 [-7.6,-4.8]

Table 10: Seed-level paired BranchCast-minus-implicit-diffusion effects on the primary manifest.

Training seed	Success effect	Severe-failure effect
1	+5.8	-5.7
2	+6.3	-6.0
3	+6.7	-6.4
4	+6.9	-6.6
5	+6.8	-6.3

F Operating-Point Context

Tables 11 and 12 separate the five-seed crossed test from a three-seed full-suite sweep. The latter includes ToolBench weather and is reported to contextualize conventional baselines and sampler depth, not to replace the paired estimands.

Table 11: Operating points from the three-seed full-suite sweep. Values are success / severe-failure rates (%); ToolBench includes weather. These checkpoint-selection measurements use the full-suite split and three-seed average, whereas Table 2 uses the five-seed crossed-interface manifest.

Model	WebArena	ToolBench full	Code execution
RSSM	34.2 / 28.4	41.5 / 22.1	38.7 / 19.4
MDN-8	38.1 / 24.1	44.8 / 19.3	42.1 / 16.7
Ensemble-5	39.7 / 21.7	46.2 / 17.8	43.5 / 15.2
Flow-Match	40.3 / 22.9	45.1 / 18.5	43.9 / 15.8
BranchCast, 50-step	47.2 / 16.3	53.1 / 13.2	50.8 / 11.6
BranchCast, 8-step	48.1 / 17.1	53.8 / 13.9	51.4 / 12.1
BranchCast, 4-step	46.5 / 18.8	52.4 / 14.7	49.9 / 12.8

Longer BranchCast samplers improve ECE from 0.039 to 0.031 and 0.028 while increasing active-transition latency from 21 to 38 and 145 ms. The four-call rows define the main comparison; the remaining rows show the broader accuracy–calibration–cost landscape.

Table 12: Active-transition latency, failure-branch ECE, and trainable transition parameters. Counts exclude DeBERTa-v3-large.

Model	Latency (ms)	ECE	Parameters (M)
RSSM	12	0.089	24.1
MDN-8	18	0.072	31.4
Ensemble-5	62	0.058	120.5
Flow-Match	22	0.064	28.7
Branch+Rectified-Flow	21.4	0.041	34.7
BranchCast, 50-step	145	0.028	35.2
BranchCast, 8-step	38	0.031	35.2
BranchCast, 4-step	21	0.039	35.2

G Capability and Measurement Matrix

Table 13 connects each protected outcome to its failure capability, trust boundary, comparator, held-out condition, and measurement. The evaluated capabilities comprise stochastic and tool-induced failures across held-out tasks with varying schemas and providers.

Table 13: Reliability evidence matrix. The protected object is the planner-selected episode outcome in every row; validators are labeling and evaluation instruments outside the planner boundary.

Outcome	Failure capability and trust boundary	Comparator and held-out setting	Property, utility, observed failure, and evidence
Invalid call	Malformed or semantically rejected action; validator is evaluation-only	Branch+Rectified-Flow on held-out APIs and schemas	Avoid unrecoverable rejection at fixed task utility; success, severe failure, 21 ms latency, and category counts in Tables 2, 4, and 6
Retry exhaustion	Stochastic timeout or repeated recoverable failure; timeout logs are training/evaluation-only	Branch+Rectified-Flow across held-out providers and tasks	Rank recovery paths before budget exhaustion; episode success and retry counts in Tables 2 and 6
Silent-wrong completion	Executable but incorrect output; hidden tests remain outside planning	Branch+Rectified-Flow on held-out code problems	Avoid accepting a wrong terminal state; success, severe, and silent counts in Tables 2, 5, and 6
Branch misordering	Stochastic or tool-induced assignment of a benign branch to a damaging transition	Flow-Match-Distilled and Branch+Rectified-Flow on paired tasks	Preserve outcome order under fixed four-call active-transition cost; treatment effects and macro-F1 in Table 3 and Figure 2
Schema-stable operation	Benign call with low branch entropy; validators remain evaluation-only	Branch+Rectified-Flow and RSSM on two low-branch slices	Preserve utility at low overhead; practical equivalence in Figure 2 and Table 7

The matrix separates three realized failure types from the cross-cutting branch-misordering mechanism. Invalid calls, retry exhaustion, and silent-wrong completions each retain suite-native evidence, while the interaction and load analyses test whether one shared outcome interface changes their ordering inside search. The final row records the complementary low-branch boundary rather than extending the primary branch-heavy claim.