FROM ONE MONITOR TO ONE JOINT: TRACTABLE PROBABILISTIC INFERENCE FOR AGENT SAFETY WITH TENSOR NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

A safety monitor typically estimates the risk of a completed agent trajectory, but runtime safety must reason as the available evidence changes: observations may be missing, actions assessed before execution, or future violations anticipated from a prefix. Existing methods can answer these queries individually, but do not generally ensure that their answers are probabilistically consistent. The missing ingredient is a shared joint distribution. We therefore propose *TTMonitor*, which extends a black-box completed-trajectory monitor into a joint distribution over trajectories and safety outcomes, so that different runtime queries become conditionals of a unified probabilistic model. TTMonitor combines an estimated trajectory distribution with monitor distillation and represents the resulting joint as a tensor train, enabling tractable marginalization over various types of evidence. Experiments on InjecAgent and AgentDojo show that TTMonitor maintains high fidelity to the original monitor while ensuring cross-query consistency. Exact tensor-network contraction further avoids the accuracy–latency tradeoff of completion sampling and remains practical relative to direct neural prediction.

1 INTRODUCTION

A safety monitor answers a question: given an agent trajectory, how likely is it to be unsafe? (Yuan et al., 2024; Chen et al., 2025). At runtime, however, the available evidence may be incomplete, a proposed tool call may need to be assessed before execution, and the controller may need to anticipate violations before the trajectory is complete. Thus, it needs to further ask: What is the risk when part of the trajectory is unavailable? What if the agent takes a proposed action next? What is the risk over all possible futures from the current state? For an agent, these questions are not edge cases. They are raised precisely when a runtime safety decision needs to be made.

Recent work has begun to address these runtime questions under different evidence settings. For example, sequential monitors detect harmful intent that emerges across multiple instructions or actions (Chen et al., 2025; 2026); ToolSafe evaluates proposed tool invocations before execution (Mou et al., 2026); and ProbGuard estimates the risk of future safety violations from an evolving execution trace (Wang et al., 2026). It is worth noting that these risk estimates are inherently related: they describe the same safety outcome under different evidence, yet this relationship is not explicitly modeled by existing approaches. As a result, their predictions are not generally guaranteed to be probabilistically consistent.

Why does the probabilistic consistency matter? Figure 1 gives a simple example. Consider a monitor that assigns an unsafe-outcome probability of 0.20 given the trajectory observed so far. The next event is not yet observed and can take one of two possible values (A or B). Once it is revealed, the corresponding runtime queries assign risks of 0.60 and 0.75, respectively. Each of these three predictions may appear reasonable on its own. Together, however, they are impossible. Under a single joint distribution,

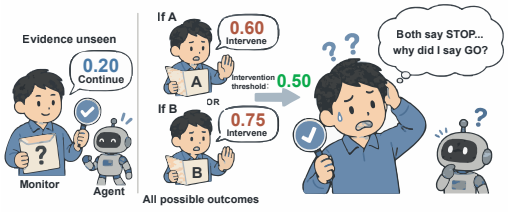

Figure 1: Inconsistency in a safety monitor.

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 unresolved risk must be an average (between 0.60 and 0.75 in the example) over its possible resolutions, and therefore cannot be smaller than both (e.g., 0.20). This inconsistency is not merely mathematical; it can change the runtime decision. With an intervention threshold of 0.5, the monitor says to continue now, even though every possible outcome of the missing event would trigger intervention. Thus, individually reasonable safety predictions can lead to *contradictory* runtime decisions when they are not grounded in a unified probabilistic model.

The missing object is not another predictor, but a probability law over complete trajectories and safety outcomes. With such a joint, the questions above become different inference operations on the same model: marginalizing missing events, conditioning on proposed actions, or averaging over possible futures. The runtime query changes; the underlying probability model does not.

We therefore introduce *TTMonitor*, which extends a black-box completed-trajectory monitor to a joint distribution over trajectories and safety outcomes. Runtime questions become different conditionals of the same model, ensuring cross-query consistency by construction. To make this joint tractable while retaining the behavior of the original monitor, TTMonitor combines an estimated trajectory distribution with query-aware monitor distillation and represents the resulting joint as a tensor train (TT) (Oseledets, 2011). The TT representation enables tractable marginalization, while query-aware distillation improves monitor fidelity under partial evidence. Our contributions are threefold.

- We formulate heterogeneous runtime safety queries as different conditionals of a single joint distribution over trajectories and safety outcomes, providing cross-query consistency by construction across diverse forms of runtime evidence.
- We introduce *TTMonitor*, which combines an estimated trajectory distribution with query-aware monitor distillation and represents the resulting joint as a tensor train, enabling tractable marginalization over possible completions.
- We evaluate the proposed framework on InjecAgent and AgentDojo, showing that it maintains high fidelity to the original monitor while ensuring cross-query consistency. Exact tensor-network contraction further avoids the accuracy–latency tradeoff of completion sampling and remains practical relative to direct neural prediction.

Notation. In the rest of the paper, we use $p(\cdot \mid \cdot)$ to denote conditional probability, and write $[n] := \{1, \dots, n\}$. For an index set O, x_O denotes the entries of x indexed by O. We write $x \models e$ when x is compatible with evidence e. We use $\mathbf{1}$ to denote an all-ones vector of dimension clear from context, and $\mathbb{I}[\cdot]$ for the indicator function.

2 From Monitoring to Runtime Probabilistic Inference

In this section, we formalize runtime safety queries probabilistically and identify three requirements for answering them within a joint model: monitor fidelity, a trajectory distribution, and tractable inference.

Let $x_{1:T} \in [d]^T$ be a finite-length trajectory of agent events and $y \in \{0, 1\}$ its safety label, where $y = 1$ indicates an unsafe trajectory. In this work, we take as given a calibrated completed-trajectory teacher monitor $c_\theta(y \mid x, T)$, which is frozen and treated as a black box accessible only through its soft-label outputs. At execution step k, the available evidence may include an observed history, hidden events, proposed actions, and an unresolved suffix. Thus, we consider four runtime queries:

$$q_{\text{done}} = p(y = 1 \mid x_{1:T}, T), \qquad q_{\text{miss}} = p(y = 1 \mid x_O, T), \tag{1}$$

$$q_{\text{cand}} = p(y = 1 \mid x_{1:k}, a, T \geq k + 1), \qquad q_{\text{future}} = p(y = 1 \mid x_{1:k}, T \geq k), \tag{2}$$

where O denotes the observed positions and a a proposed event. As shown in equations, these queries respectively condition on a complete trajectory, marginalize missing events, condition on a candidate action, and marginalize unresolved future continuations, as different conditionals of a single joint distribution over trajectories and safety outcomes.

Extend, do not replace. Learning a joint model from scratch is not sufficient. An existing monitor may already have undergone model selection and calibration, with its risk scores used for downstream safety decisions. Training a new joint model from scratch would relearn this predictor and

could therefore influence its established behavior. The joint should instead preserve the teacher on completed trajectories while extending it to probabilistic inference under runtime evidence.

A trajectory distribution is necessary. The completed-trajectory monitor specifies safety risk for each trajectory, but does not determine how compatible trajectories should be weighted under partial evidence. In general,

$$p(y = 1 \mid e) = \sum_{x \models e} p(y = 1 \mid x)\, p(x \mid e),$$

so runtime risk depends on both the monitor outputs and the relative probability of compatible trajectories. Defining the joint therefore additionally requires a label-independent trajectory distribution (denoted by $b(x)$ throughout the paper) learned from source trajectories.

Runtime inference must be tractable. A joint distribution makes the runtime queries well defined, but answering them during execution must also be computationally practical. Queries may be issued repeatedly along a trajectory, and candidate actions may need to be assessed before execution. For missing or future evidence, directly marginalizing over all compatible completions requires exponentially many terms in the unresolved horizon. Monte Carlo estimation avoids this exhaustive computation, but introduces sampling cost and estimation error at query time. In addition, the safety layer may operate separately from a remotely served agent model and cannot always rely on graphical processing unit (GPU) acceleration. Runtime inference should therefore remain practical on commodity central processing units (CPUs). Tractable exact marginalization is thus a key computational requirement for the joint model.

3 A Tractable Joint for Runtime Probabilistic Inference

Following the guidance in Section 2, we now construct *TTMonitor*, which extends the completed-trajectory monitor into a tractable joint model.

3.1 A Joint Model and Its Tractability Challenge

For a completed trajectory x, let $w_{\theta,y}(x) > 0$ denote the soft score assigned by the teacher monitor c_θ to label y . Let $b(x)$ denote the trajectory distribution, i.e., a probability distribution over complete trajectories. These two sources of information naturally define the joint:

$$p(x, y) \propto b(x)^\beta w_{\theta,y}(x). \tag{3}$$

Note that we impose $\beta \geq 0$ in (3) to control the relative influence of trajectory likelihood when marginalizing unresolved evidence. When $\beta = 1$, the trajectory likelihood enters the joint without tempering; smaller β flattens its relative mass and reduces reliance on the trajectory model. Thus, $w_{\theta,y}(x)$ determines how safety varies across completed trajectories, while $b(x)^\beta$ determines how those trajectories are weighted when evidence is incomplete.

For partial evidence e, the corresponding runtime conditional is thus obtained by marginalizing over all compatible completions:

$$p(y \mid e) \propto \sum_{x \models e} b(x)^\beta w_{\theta,y}(x). \tag{4}$$

Tractability challenge. Eq. (4) defines the desired probabilistic semantics for runtime queries, but not yet a practical runtime model. In general, evaluating the joint requires summing $b(x)^\beta w_{\theta,y}(x)$ over trajectories. Particularly, the partition function sums over all trajectories and labels, while a runtime query restricts the same computation to trajectories compatible with the observed evidence. The number of summation terms can grow *exponentially* with the unresolved horizon. Thus, even if $b(x)$ and $w_{\theta,y}(x)$ are individually easy to evaluate, their product need not admit tractable marginalization. Our goal in the next step is to represent these two factors in compatible structured forms for which both normalization and evidence-conditioned marginalization can be computed efficiently.

3.2 Learning from Trajectories and Monitor Outputs

To make the joint model tractable, we replace the two generic factors $b(x)$ and $w_{\theta,y}(x)$ in Eq. (3) with *tensor-train (TT)* (Oseledets, 2011).

Modeling trajectory probabilities with a TT. First, we model the trajectory distribution with the *first-order Markov* model. In more detail, let n_{ij} denote the number of transitions from event i to event j in the source data and let $N_i = \sum_j n_{ij}$. With additive smoothing $\alpha > 0$, we define

$$P_\eta(j \mid i) = \frac{n_{ij} + \alpha}{N_i + d\alpha}. \tag{5}$$

The resulting trajectory distribution and its tempered form are

$$b(x_{1:T}) = p(x_1) \prod_{t=2}^{T} P(x_t \mid x_{t-1}), \qquad b(x_{1:T})^\beta = p(x_1)^\beta \prod_{t=2}^{T} P(x_t \mid x_{t-1})^\beta. \tag{6}$$

Note that the Markov factorization admits an exact TT representation with shared local tensors. For each event $x \in [d]$, define

$$B_x(i,j) = P(x \mid i)^\beta \mathbf{1}[j = x], \qquad \ell_x(i) = p(x)^\beta \mathbb{I}[i = x], \quad i,j \in [d]. \tag{7}$$

Then

$$b(x_{1:T})^\beta = \ell_{x_1}^\top B_{x_2} \cdots B_{x_T} \mathbf{1}. \tag{8}$$

We can see from Eq. (8) that the tempered trajectory distribution admits an exact TT representation with ranks at most d.

Distilling the teacher monitor into tensor networks. Next, we distill the teacher monitor into a nonnegative TT with shared local tensors:

$$s(y \mid x, T) = \ell^\top A_{x_1} \cdots A_{x_T} r_y, \qquad A_x \mathbf{1} = \mathbf{1}, \quad \ell^\top \mathbf{1} = 1, \qquad r_0 + r_1 = \mathbf{1}. \tag{9}$$

Here, $\ell \in \mathbb{R}_+^r$, each $A_x \in \mathbb{R}_+^{r \times r}$, and $r_y \in \mathbb{R}_+^r$. The constraints in (9) ensure that $\sum_y s(y \mid x, T) = 1$ for every trajectory x and length T. In practice, we enforce these constraints by applying a softmax to ℓ, a row-wise softmax to each A_x, and a label-wise softmax across $(r_0(i), r_1(i))$ for each latent index i. The shared local tensor A therefore has dimension $r \times d \times r$, with one event slice A_x for each $x \in [d]$; the same slices are reused at every trajectory position and for both labels. The right boundary r_y selects the output label. Note that the TT sequence models (also known as uMPS) and their finite-state interpretations are well established (Miller et al., 2021).

Training on the queries the student will answer. We train the TT student in Eq. (9) by distilling the frozen teacher from its completed-trajectory soft outputs. Let $w(x)$ and $s(x)$ denote the teacher and student scores of the unsafe label for simplicity, h the binary cross-entropy (BCE) loss, and $\mathcal{D}_{\text{src}}$ the empirical source-trajectory distribution. Beyond completed trajectories, we consider F runtime query families during training. For each $f \in [F]$, let $\mathcal{D}_f$ denote its distribution over evidence e, and define $\mathcal{D}_{\text{query}} := \frac{1}{F} \sum_{f=1}^{F} \mathcal{D}_f$. The *query-aware* distillation objectives are

$$\begin{aligned}
L_{\text{trace}} &= \mathbb{E}_{x \sim \mathcal{D}_{\text{src}}} h\big(w(x), s(x)\big), \\
L_{\text{comp}} &= \mathbb{E}_{e \sim \mathcal{D}_{\text{query}}} \mathbb{E}_{x \sim b(\cdot \mid e)} h\big(w(x), s(x)\big), \\
L_{\text{post}} &= \mathbb{E}_{e \sim \mathcal{D}_{\text{query}}} h\left(\mathbb{E}_{x \sim b(\cdot \mid e)} w(x), \mathbb{E}_{x \sim b(\cdot \mid e)} s(x)\right), \\
L_{\text{dual}} &= \frac{1}{2}\left(L_{\text{comp}} + L_{\text{post}}\right), \qquad L = \frac{L_{\text{trace}} + F L_{\text{dual}}}{F + 1}.
\end{aligned} \tag{10}$$

Here, L_{trace} preserves completed-trajectory behavior, while L_{comp} and L_{post} match the teacher at the completion and posterior levels, respectively. The factor F gives each uniformly mixed query family the same weight as L_{trace}. Their average L_{dual} therefore targets fidelity both before and after marginalization. In practice, we approximate these expectations using sampled completions from the fitted trajectory law conditioned on the evidence e.

3.3 Runtime Inference as Tensor Contraction

Coupling the two TTs. We next combine the trajectory TT in Eq. (8) with the distilled safety TT in Eq. (9) into a unified and tractable form. Because the *TT format is closed under pointwise multiplication* (Cichocki et al., 2016), the product between Eqs. (8) and (9) is again representable as a TT. For positions after the first event, define the coupled local tensors

$$C_x = B_x \otimes A_x, \qquad R_y = \mathbf{1} \otimes r_y, \tag{11}$$

where $\otimes$ denotes the Kronecker product. The first event additionally incorporates the initial trajectory probability through ℓ_x in Eq. (8). We therefore define the first-site joint vector

$$H_x^\top = \ell_x^\top \otimes (\ell^\top A_x). \tag{12}$$

Combining these terms gives the unnormalized joint weight

$$\widetilde{Q}(x_{1:T}, y \mid T) = H_{x_1}^\top C_{x_2} \cdots C_{x_T} R_y = b(x_{1:T})^\beta s(y \mid x_{1:T}, T). \tag{13}$$

The resulting joint TT has the ranks at most $d \times r$. Finally, the normalized joint distribution is then

$$Q(x_{1:T}, y \mid T) = \frac{H_{x_1}^\top C_{x_2} \cdots C_{x_T} R_y}{H^\top S^{T-1}(R_0 + R_1)}, \tag{14}$$

where $S := \sum_{x \in [d]} C_x$, and $H^\top := \sum_{x \in [d]} H_x^\top$. Note that the denominator is the partition function of the joint. Instead of explicitly summing over all d^T trajectories, in (14) it is computed by sequential tensor contractions.

Four queries become four contractions. Once the joint model (14) is constructed, runtime queries differ only in how tensors are contracted. An observed event selects the corresponding tensor slice, whereas an unresolved event is marginalized by summing over the corresponding index. The four runtime queries therefore reduce to different contractions of the same joint:

Query	Evidence	Label-mass contraction	Probabilistic operation
Completed	$x_{1:T}$	$H_{x_1}^\top C_{x_2} \cdots C_{x_T} R_y$	condition on all events
Missing	x_O	select observed slices; sum hidden slices	marginalize hidden events
Candidate	$x_{1:k}, a$	$H_{x_1}^\top C_{x_2} \cdots C_{x_k} C_a S^{T-k-1} R_y$	condition on a; marginalize future
Future	$x_{1:k}$	$H_{x_1}^\top C_{x_2} \cdots C_{x_k} S^{T-k} R_y$	marginalize future

For missing evidence, the first-site tensor is $H_{x_1}^\top$ if x_1 is observed and $H^\top = \sum_x H_x^\top$ otherwise. At each subsequent position $t \geq 2$, the contraction uses C_{x_t} if x_t is observed and $S = \sum_x C_x$ otherwise. Thus, arbitrary missingness is handled by locally switching between selection and marginalization without changing the joint model. For any evidence pattern e, therefore, the corresponding contraction yields an unnormalized mass $Z_y(e)$ for each label y. Normalizing these masses gives the runtime risk

$$Q(y = 1 \mid e) = \frac{Z_1(e)}{Z_0(e) + Z_1(e)}. \tag{15}$$

We can see that the runtime query changes the contraction, not the learned joint.

Contracting over unknown futures. The contractions above assume a fixed terminal length. For future and candidate queries, however, the endpoint may itself be unresolved. We therefore marginalize not only over future events, but also over the length of the unresolved suffix.

We model the suffix length by geometric stopping. After each event, execution continues with probability $\gamma \in (0, 1)$ and terminates with probability $1 - \gamma$. Thus, a suffix of length $m \geq 0$ receives prior weight $(1 - \gamma)\gamma^m$. Recall that $S = \sum_x C_x$ marginalizes one unresolved event. Consequently, $S^m R_y$ marginalizes over all event sequences of length m before reaching the label boundary R_y. Marginalizing also over unknown suffix lengths gives the *future boundary*

$$G_y := (1 - \gamma) \sum_{m=0}^{\infty} \gamma^m S^m R_y. \tag{16}$$

Table 1: Runtime inference complexity of TTMonitor. The missing-evidence time complexity is a worst-case bound. The future boundary is precomputed once and reused across queries; N_{fp} denotes the number of fixed-point iterations required to reach the prescribed numerical tolerance. Overall model storage is $O(d^2 + dr^2)$; the table reports additional working storage.

Query	Contraction	Time	Working storage
Completed	$H_{x_1}^\top C_{x_2} \cdots C_{x_T} R_y$	$O(Tr^2)$	$O(r)$
Missing	select / marginalize	$O\big(T(d^2 r + dr^2)\big)$	$O(dr)$
Future, fixed horizon	$h_k^\top S^{T-k} R_y$	$O\big(kr^2 + (T-k)(d^2 r + dr^2)\big)$	$O(dr)$
Future, unknown horizon	$h_k^\top G_y$	$O(kr^2)$	$O(dr)$
Candidate	$\gamma h_k^\top C_a G_y$	$O(kr^2 + r^2)$	$O(dr)$
Candidate, cached prefix	$\gamma h_k^\top C_a G_y$	$O(r^2)$	$O(dr)$
Future-boundary setup	fixed-point iteration	$O\big(N_{\mathrm{fp}}(d^2 r + dr^2)\big)$	$O(dr)$

Hence, G_y summarizes all finite continuations into a single boundary vector: the term $m = 0$ corresponds to terminating immediately, while $m \geq 1$ accounts for increasingly long unresolved suffixes. Note that the same quantity admits a recursive form. Separating the $m = 0$ term in Eq. (16) gives

$$G_y = (1-\gamma) R_y + \gamma S G_y. \tag{17}$$

This fixed-point form allows the infinite sum to be evaluated without enumerating suffix lengths. The following proposition tells when the future boundary is well defined and the fixed-point iteration converges.

Proposition 1 (**Future-boundary convergence**). *Let $\rho(S)$ denote the spectral radius of S. If $\gamma\rho(S) < 1$, then the series in Eq.* (16) *converges, and its sum G_y is the unique solution to Eq.* (17)*, with*

$$G_y = (1-\gamma)(I - \gamma S)^{-1} R_y. \tag{18}$$

Proof. Since $\rho(\gamma S) < 1$, the Neumann series converges and $\sum_{m \geq 0} (\gamma S)^m = (I - \gamma S)^{-1}$, yielding the stated unique solution. □

Proposition 1 ensures that contracting over future continuations of unbounded length yields a finite and uniquely defined boundary whenever $\gamma\rho(S) < 1$.

Therefore, for an observed prefix $x_{1:k}$, define the joint prefix message

$$h_k^\top := H_{x_1}^\top C_{x_2} \cdots C_{x_k}. \tag{19}$$

The future and candidate label masses are then

$$Z_y^{\mathrm{future}} = h_k^\top G_y, \qquad Z_y^{\mathrm{cand}}(a) = \gamma\, h_k^\top C_a G_y. \tag{20}$$

We remark that, for a candidate query, C_a first selects a. The factor γ is shared across labels and thus cancels in the conditional risk in Eq. (15).

We also remark that, for fixed model parameters and γ, we compute each future boundary G_y once by iterating $G_y^{(n+1)} = (1-\gamma) R_y + \gamma S G_y^{(n)}$ to a prescribed numerical tolerance and *reuse* the resulting boundary across runtime queries. Although Proposition 1 gives the closed-form solution $(1-\gamma)(I - \gamma S)^{-1} R_y$, the iteration avoids explicitly forming or inverting the $(dr) \times (dr)$ transfer matrix S. Instead, each product $S G_y^{(n)}$ is evaluated directly from $S = \sum_x B_x \otimes A_x$ using the factors B_x and A_x. At runtime, candidate queries sharing the same observed prefix also reuse the prefix message h_k, requiring only the candidate-specific contraction $h_k^\top C_a G_y$. Thus, unknown-horizon inference accounts for all finite future continuations without enumerating individual completions or imposing a finite suffix horizon. Table 1 summarizes the resulting runtime inference complexities.

4 Experiments

In this section, we use numerical results to show that TTMonitor can faithfully extend a completed-trajectory safety monitor to diverse runtime queries while maintaining cross-query consistency. In addition, we show that the introduced query-aware distillation loss improves fidelity to the teacher monitor, while tensor-network contraction enables exact and efficient inference.

Table 2: **Prediction, monitor fidelity, and cross-query consistency on InjecAgent.** We report Brier score for prediction, MAE to the teacher for fidelity, and joint repair and incompatibility rate (10^{-3} tolerance) for consistency. Prediction and fidelity show means $\pm$ 95% confidence interval (CI) half-widths; consistency shows 95% bootstrap CIs. Lower is better.

		Prediction ↓		Fidelity ↓		Consistency ↓	
Method	Type	Seen	Held-out	Complete	Query	Repair	Incomp. (%)
Teacher[‡]	Direct	.060±.002	.055±.003	–	–	.005 [.000,.012]	3.75 [0.00,8.33]
Pooled logistic	Direct	.037±.002	.029±.003	.012±.001	.099±.020	.024 [.008,.043]	15.00 [7.69,23.08]
Pooled GRU	Direct	**.034±.002**	**.021±.002**	**.004±.001**	.086±.018	.001 [.000,.004]	2.92 [0.00,7.41]
Pooled Trans.	Direct	**.034±.002**	**.021±.002**	.005±.001	.088±.019	.002 [.000,.005]	2.92 [0.00,6.83]
Markov joint	Joint	.051±.002	.044±.003	.038±.003	.104±.024	$< 10^{-12}$	0.00
HMM joint	Joint	.062±.004	.059±.008	.040±.006	.138±.033	$< 10^{-12}$	0.00
Ours	Joint	.049±.002	.031±.002	**.004±.001**	**.007±.003**	$< 10^{-12}$	0.00

[‡]Teacher compatibility is evaluated on its direct partial-evidence outputs, not on a runtime posterior constructed from its completed-trajectory predictions. For explicit joint models, repair below 10^{-12} reflects numerical precision; zero incompatibility denotes no detected violation in the audit.

Table 3: **Ablation of the query-aware distillation objective.** Query-aware supervision substantially improves runtime fidelity, while joint compatibility is preserved across objectives.

Training objective	Complete MAE ↓	Query MAE ↓	Repair ↓	Incomp. (%) ↓
Trace only	.004	.028	$< 10^{-12}$	0.00
Trace + completion	.004	.010	$< 10^{-12}$	0.00
Trace + posterior	.004	.011	$< 10^{-12}$	0.00
Dual (ours)	.004	**.007**	$< 10^{-12}$	0.00

Protocol. We evaluate on InjecAgent (Zhan et al., 2024) and AgentDojo (Debenedetti et al., 2024). Teachers are trained with trajectory-level safety labels and frozen before distillation; students access only their soft outputs on completed sequences. We train/select on four query families and evaluate generalization on seven held-out families for InjecAgent and four for AgentDojo. The main experiments use GRU and Transformer teachers, respectively, and AgentDojo additionally evaluates a GRU teacher to test sensitivity to teacher architecture. Model selection and calibration use validation data only; training repetitions and full experimental details are given in Appendix A.

4.1 Prediction, Fidelity, and Consistency

We compare against two classes of baselines in the experiment. *Direct predictors*, including a pooled logistic model, GRU, and Transformer, map each observed evidence pattern directly to a safety score without constructing an explicit joint distribution. *Joint baselines*, including a Markov model and an HMM, instead define explicit probability distributions over trajectories and safety outcomes, providing consistency controls without monitor distillation.

We evaluate three complementary properties: *prediction* against observed safety outcomes, *monitor fidelity* to the teacher, and *cross-query consistency*. Prediction is measured by the Brier score, and fidelity by mean absolute error (MAE) to the corresponding teacher target.

We assess cross-query consistency through joint compatibility. Starting from a partially observed trajectory, we select two missing positions and progressively reveal their values to form a group of related queries. We test whether all predictions in the group can be interpreted as conditionals of a single joint distribution over admissible completions. *Joint repair* measures the minimum maximum absolute change required to restore compatibility, averaged across query groups. *Incompatibility rate* reports the fraction of groups that remain incompatible at tolerance 10^{-3}. Lower is better for all metrics. Formal definitions are provided in Appendix B.2.

Table 2 illustrates the experimental results, revealing a clear trade-off between predictive accuracy, monitor fidelity, and cross-query consistency. Direct predictors generally achieve lower Brier scores

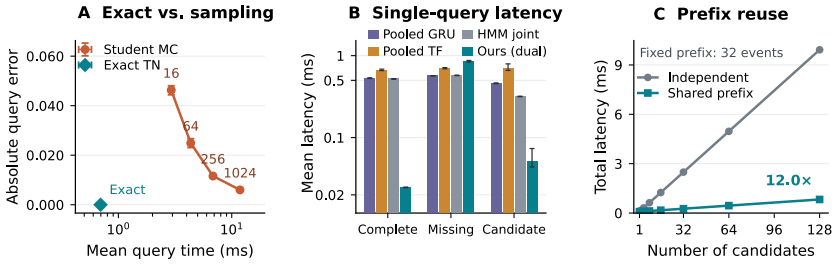

Figure 2: **Efficiency of exact runtime inference.** **(A)** Exact contraction versus Monte Carlo (MC) completion sampling within the same distilled joint; MC error is measured relative to exact inference. **(B)** Warm single-query CPU latency across query types. **(C)** Candidate-query latency when reusing a shared prefix message (prefix length 32). All timings use one CPU thread with warm inference; setup costs are reported separately in Appendix D.

than explicit joint models, although ours performs best among the joint models and improves over the original teacher. However, the pooled GRU and Transformer deviate significantly from the teacher on runtime queries, whereas ours maintains strong fidelity, allowing it to extend rather than replace the existing monitor. The consistency metrics show a complementary structural difference: the pooled logistic model requires .024 joint repair and remains incompatible on 15.00% of tested query groups, while all explicit joint models show only numerical-precision repair and no detected inconsistency. Thus, TTMonitor combines the cross-query consistency of a joint model with fidelity to the existing monitor.

4.2 Query-Aware Distillation Improves Runtime Fidelity

Table 3 shows that the trivial completed-trajectory distillation (i.e., trace only) alone is insufficient for faithful runtime inference. Although trace-only training already matches the teacher well on completed trajectories (MAE $\approx$ .004), its query-level deviation is .028. Adding completion- or posterior-level supervision reduces this deviation to .010 and .011, respectively, while combining both yields the lowest Query MAE of .007. Thus, the improvement does not come from better imitation on completed trajectories, but from explicitly training the student on the conditional quantities required by runtime queries. Importantly, all variants require only numerical-precision joint repair and show no detected incompatibility. This separates the roles of architecture and training: the joint construction provides cross-query compatibility, while query-aware distillation determines how faithfully its runtime conditionals reproduce those induced by the teacher monitor.

4.3 Runtime Inference Cost

Figure 2 illustrates the computational benefits of TTMonitor on *CPU* (AMD® EPYC 7742). Panel A shows that exact contraction because of the tractability is faster than Monte Carlo (MC) sampling while avoiding its approximation error. Panel B shows competitive single-query latency compared with other baselines. Finally, Panel C focuses specifically on candidate queries, illustrating that candidate queries sharing the same prefix in TTMonitor can reuse intermediate contractions, significantly reducing runtime as the number of candidates increases.

4.4 Transfer to AgentDojo

We finally test whether the main conclusions from InjecAgent persist on a different benchmark and different teacher architectures. On AgentDojo, we use a Transformer teacher for the main comparison and repeat the extension with independently trained GRU teachers.

Table 4 reproduces the main pattern observed on InjecAgent. TTMonitor preserves competitive held-out prediction while achieving significantly better teacher fidelity than the Markov and HMM joints and preserving cross-query compatibility. The same pattern persists with independently trained GRU teachers, although the achievable fidelity depends on the teacher being approximated. Together, these results show that the separation between prediction, fidelity, and consistency is not specific to InjecAgent or a single teacher architecture.

Table 4: **AgentDojo transfer results.**

Method	Seen ↓	Held-out ↓	Complete MAE ↓	Query MAE ↓	Repair ↓	Incomp. (%) ↓
Pooled logistic	.062±.006	.096±.007	.081±.007	.125±.033	$< .001$	1.79
Pooled GRU	.047±.006	.068±.006	.072±.013	.101±.039	$< .001$	0.00
Pooled Transformer	.056±.006	.072±.006	.073±.010	.097±.032	$< .001$	0.00
Independent heads	.059±.006	.076±.006	.074±.005	.100±.026	.017	35.71
Markov joint	.102±.010	.097±.009	.194±.010	.196±.041	$< 10^{-12}$	0.00
HMM joint	.124±.018	.116±.016	.176±.015	.175±.039	$< 10^{-12}$	0.00
Transformer teacher	.058±.006	.076±.008	–	–	$< .001$	0.00
TTMonitor (Transformer)	.057±.007	.069±.006	**.063±.011**	**.074±.031**	$< 10^{-12}$	0.00
GRU teacher	.050±.006	.075±.007	–	–	$< .001$	.60
TTMonitor (GRU)	.057±.006	.070±.006	**.048±.011**	**.055±.026**	$< 10^{-12}$	0.00

5 Related Work

Runtime agent safety and monitoring. Existing benchmarks expose failures in multi-step tool use and prompt-injection settings (Ruan et al., 2024; Andriushchenko et al., 2025; Kumar et al., 2025; Debenedetti et al., 2024; Zhan et al., 2024), while runtime monitors score trajectories or intermediate behaviors (Yuan et al., 2024; Chen et al., 2026; Zhan et al., 2026). Other systems connect such signals to runtime enforcement (Mou et al., 2026; Liu et al., 2026; Hossain et al., 2026; Ng et al., 2026), and ProbGuard introduces probabilistic state modeling through a symbolic-state DTMC (Wang et al., 2026). Our focus is complementary: given a selected completed-trajectory monitor, TTMonitor constructs a single probabilistic extension whose conditionals answer heterogeneous runtime queries while remaining faithful to the original monitor. A distinct ProbGuard studies token-level probability distributions (Huang et al., 2026).

Tractable probabilistic inference. HMMs, weighted automata, nonnegative tensor trains (also known as matrix-product states), Born machines, and probabilistic circuits provide structured representations supporting exact marginalization (Han et al., 2018; Glasser et al., 2019; Adhikary et al., 2021; Vergari et al., 2021; Miller et al., 2021; Gens & Domingos, 2013; Chan, 2017; Leland & Choi, 2026). Recent probabilistic circuits further connect tractable inference with reasoning or other tasks (Papez et al., 2024; Kang et al., 2024; Gala et al., 2024; Zhang et al., 2025; Loconte et al., 2026). Our goal is not to introduce another tractable model class, but to use tractability to extend a frozen black-box monitor from completed trajectories to partial, candidate, and prospective evidence. The resulting joint gives these queries a shared probabilistic semantics, while tensor-network contraction makes the required conditioning and marginalization tractable.

6 Concluding Remarks: What Does a Tractable Probabilistic Extension Buy?

In this paper, we introduced *TTMonitor*, a tensor-network-based probabilistic model that extends an existing AI-safety monitor to tractable runtime inference. Our results show that TTMonitor maintains strong monitor fidelity while providing cross-query consistency and exact marginalization. But what does tractable probabilistic modeling actually buy? The answer lies in the trade-off. Not necessarily better prediction: probabilistic consistency constrains the admissible predictions, while tractability further restricts model expressivity. Thus, what it buys instead is a unified probabilistic semantics: completed, missing, candidate, and future-risk queries become conditionals of the same tractable joint with *probabilistic consistency*.

These trade-offs also suggest where to go next. Tensor networks are natural for sequential trajectories, but are not the only tractable representation; extending the framework to probabilistic circuits may better accommodate other dependency structures. Meanwhile, the algebraic structure of tensor networks opens theoretical questions on approximation and perturbation error. In summary, the goal of this paper is not to tell how to replace every safety monitor with a tensor network, but to provide a principled probabilistic framework when one monitor must answer many related runtime queries consistently.

AI USE STATEMENT

Generative AI tools were used throughout the development of this paper in iterative collaboration with the authors, including conceptual development, mathematical derivations, experimental design, implementation and testing, result interpretation, literature work, writing and editing, and figure construction. All parts of the paper involved alternating human and AI contributions; no part was independently completed by AI. The authors reviewed and verified the resulting code, proofs, citations, claims, results, and artifacts, and take full responsibility for the final manuscript and associated artifacts.

ETHICS STATEMENT

This study uses public benchmarks for prompt-injection research. Raw trajectories may contain untrusted or sensitive-looking text and are kept local; released results contain only aggregate metrics, hashes, and structural schemas. No instruction contained in a trajectory is executed, and no external model API is called during evaluation. The proposed monitor is observational and may fail under distribution shift; deployment should therefore combine it with appropriate authorization, sandboxing, provenance, or information-flow controls.

REPRODUCIBILITY STATEMENT

Appendices A–D provide complete proofs, data and split provenance, preprocessing details, hyperparameters, statistical procedures, and implementation and environment details. The anonymous supplement contains the implementation and an offline dependency lock, and all reported results and figures are generated from machine-readable experiment outputs.

REFERENCES

Sandesh Adhikary, Siddarth Srinivasan, Jacob Miller, Guillaume Rabusseau, and Byron Boots. Quantum tensor networks, stochastic processes, and weighted automata. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 2080–2088, 2021. URL https://proceedings.mlr.press/v130/adhikary21a.html.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c493d23af93118975cdbc32cbe7323f5-Abstract-Conference.html.

Hei Chan. Incorporating uncertain evidence into arithmetic circuits representing probability distributions. In *Proceedings of the 3rd International Workshop on Advanced Methodologies for Bayesian Networks*, volume 73 of *Proceedings of Machine Learning Research*, pp. 105–116, 2017. URL https://proceedings.mlr.press/v73/chan17a.html.

Yueh-Han Chen, Nitish Joshi, Yulin Chen, He He, and Rico Angell. Monitoring LLM agents for sequentially contextual harm. In *ICLR Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=LC0XQ6ufbr.

Yueh-Han Chen, Nitish Joshi, Yulin Chen, Maksym Andriushchenko, Rico Angell, and He He. Monitoring decomposition attacks with lightweight sequential monitors. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2985777c8f2152664946ed30deb77291-Abstract-Conference.html.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4-5):249–429, 2016.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.13352.

Gennaro Gala, Cassio de Campos, Antonio Vergari, and Erik Quaeghebeur. Scaling continuous latent variable models as probabilistic integral circuits. *Advances in Neural Information Processing Systems*, 37:10959–10987, 2024.

Robert Gens and Pedro Domingos. Learning the structure of sum-product networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 873–880, 2013. URL https://proceedings.mlr.press/v28/gens13.html.

Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and J. Ignacio Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.neurips.cc/paper/8429-expressive-power-of-tensor-network-factorizations-for-probabilistic-modeling.

Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018. doi: 10.1103/PhysRevX.8.031012.

Elias Hossain, Md Mehedi Hasan Nipu, Tasfia Nuzhat Ornee, Rajib Rana, and Niloofar Yousefi. NEXUS: Structured runtime safety for tool-using LLM agents. *arXiv preprint arXiv:2607.19356*, 2026. URL https://arxiv.org/abs/2607.19356.

Xinzhe Huang, Biwu Yao, Kedong Xiu, Mengnan Zhao, Di Wang, Puning Zhao, and Tianhang Zheng. ProbGuard: Calibrated safety risk estimation from LLM output distributions. *arXiv preprint arXiv:2608.10621*, 2026. URL https://arxiv.org/abs/2608.10621.

Mintong Kang, Nezihe Merve Gürel, Linyi Li, and Bo Li. COLEP: Certifiably robust learning-reasoning conformal prediction via probabilistic circuits. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3ab868033387edbdd775b8edd78ed056-Abstract-Conference.html.

Priyanshu Kumar, Elaine Lau, Saranya Vijayakumar, Tu Trinh, Elaine Chang, Vaughn Robinson, Shuyan Zhou, Matt Fredrikson, Sean Hendryx, Summer Yue, and Zifan Wang. Aligned LLMs are not aligned browser agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/42f92b78a6695f60db0cd38b54d57a41-Abstract-Conference.html.

John Leland and YooJung Choi. On the hardness of approximating distributions with tractable probabilistic models. *Advances in Neural Information Processing Systems*, 38:42977–43000, 2026.

Hailin Liu, Eugene Ilyushin, Jie Ni, and Min Zhu. SafeAgent: A runtime protection architecture for agentic systems. *arXiv preprint arXiv:2604.17562*, 2026. URL https://arxiv.org/abs/2604.17562.

Lorenzo Loconte, Adrián Javaloy, and Antonio Vergari. How to square tensor networks and circuits without squaring them. In *International Conference on Learning Representations*, volume 2026, pp. 77505–77550, 2026.

Jacob Miller, Guillaume Rabusseau, and John Terilla. Tensor networks for probabilistic sequence modeling. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3079–3087, 2021. URL https://proceedings.mlr.press/v130/miller21a.html.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. ToolSafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37125–37153.

Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1850. URL https://aclanthology.org/2026.findings-acl.1850/.

Albus W. Ng, Yi Han, Jusheng Zhang, and Wenhao Wang. Agent safety should be a runtime contract. *arXiv preprint arXiv:2608.11274*, 2026. URL https://arxiv.org/abs/2608.11274.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Milan Papez, Martin Rektoris, Vaclav Smidl, and Tomáš Pevný. Sum-product-set networks: Deep tractable models for tree-structured graphs. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/7b64c47dcb067efd6be5eee854c14835-Abstract-Conference.html.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/7274ed909a312d4d869cc328ad1c5f04-Abstract-Conference.html.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, 34:13189–13201, 2021.

Haoyu Wang, Christopher M. Poskitt, Jiali Wei, and Jun Sun. ProbGuard: Proactive runtime monitoring for LLM agent safety via probabilistic prediction. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering*, 2026. URL https://arxiv.org/abs/2508.00500. To appear.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-Judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1467–1490. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.79. URL https://aclanthology.org/2024.findings-emnlp.79/.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Zhuoxin Zhan, Akbar Rafiey, Avery Ma, Leila Pishdad, and Layla El Asri. StepJack: Benchmarking computer-use agent safety against multi-step indirect prompt injection. *arXiv preprint arXiv:2608.06477*, 2026. URL https://arxiv.org/abs/2608.06477.

Honghua Zhang, Meihua Dang, Benjie Wang, Stefano Ermon, Nanyun Peng, and Guy Van den Broeck. Scaling probabilistic circuits via monarch matrices. *arXiv preprint arXiv:2506.12383*, 2025.

A Experimental Details

The experiments separate three questions: whether a model predicts the recorded safety outcome, whether it remains faithful to a selected teacher, and whether related runtime answers admit a common probabilistic explanation. This section specifies the data, query interfaces, and training procedures. Appendix B defines the corresponding measurements.

A.1 Data sources, splits, and preprocessing

We use published agent execution records rather than generating new agent interactions. An example consists of a discrete event sequence and the benchmark's recorded binary unsafe-outcome label. Missing-evidence and prospective queries retain that trajectory-level label.

Table 5: Data used in the reported experiments. The last column is the number of scenario/task groups used for clustered evaluation statistics. All vocabularies are fitted on source-fit records only.

Dataset	Source fit	Validation	Evaluation	Alphabet d	Eval. groups
InjecAgent	48,676	16,220	17,409	246	445
AgentDojo	6,935	1,035	3,145	391	629

InjecAgent. The data come from the official result archive pinned at repository revision `f19c9f2`. The preprocessing retains prompted configurations with the complete four-member archive layout and removes audited duplicates. This yields 57 configurations and 120,156 records before label filtering. We retain the 82,305 records whose official evaluation is `succ` or `unsucc`; the former defines $y = 1$. Invalid or unparsed outcomes are excluded.

All runs of the same scenario are assigned to one split. We hash the scenario identifier with a fixed salt and assign the resulting buckets to source fit, validation, or evaluation. The resulting split is scenario-disjoint. Canonicalization retains the user instruction, available reasoning, tool calls, tool observations, and subsequent response events. Benchmark text is parsed as data and is never executed.

AgentDojo. We use published AgentDojo v0.1.35 runs at repository revision `a75aba7`. Source-fit and validation data are obtained from the archived source configuration pool using task-group hashing, while evaluation uses the established development configuration pool. Evaluation model configurations therefore differ from those used for source fitting.

The loader reads the archived `important_instructions` attack runs, discards records that cannot be parsed into nonempty event sequences, and uses the archived security outcome as y. Message bodies and tool calls become separate events.

A.2 Event representations and available information

InjecAgent uses 230 source-fitted structural symbols, including an unknown symbol, and 16 text-risk summary symbols. A structural descriptor combines event role, event kind, a coarse function category, text-length bin, and, for actions, an argument-shape descriptor. Rare source-fit descriptors map to the unknown symbol.

The summary is obtained from a TF–IDF logistic classifier fitted to decision-prefix text and source-fit labels. Source-fit score quantiles define 16 bins, and one summary symbol is inserted after the designated decision prefix. Its position therefore varies across records.

If any prefix event used to compute the summary is hidden, the summary is also hidden. All compared InjecAgent models use the same representation. AgentDojo instead uses a source-fitted structural encoder with 391 symbols and no supervised risk-summary symbol. The frozen teachers operate on these discrete event sequences.

A.3 RUNTIME QUERY FAMILIES

Completed queries reveal all events. A fixed-length missing query preserves the original positions and length while hiding specified event values. Direct predictors receive a dedicated mask symbol at hidden positions; joint models marginalize them. Query-family masks are shared across methods.

Table 6: Runtime-query families. Seen families are used for query-aware training and shared calibration. Held-out families are excluded from these steps, except for the explicitly query-supervised independent-head control.

Dataset	Seen	Held-out
InjecAgent	Candidate; MCAR 20%; block 20%; action redaction	MCAR 40%; block 40%; observation redaction; reasoning redaction; two-block 30%; alternating interior; retained-prefix 60%
AgentDojo	MCAR 20%; block 20%; suffix 20%; single hidden event	MCAR 40%; block 40%; suffix 40%; alternating events

InjecAgent masks. MCAR hides events independently at the indicated rate. Block masking hides a consecutive segment, while action, reasoning, and observation redaction hide the corresponding event types. Two-block masking hides approximately 30% of events in two segments; alternating-interior masking hides alternating non-endpoint events. The retained-prefix family preserves the first $\lceil 0.6T \rceil$ events and hides the remainder. These masks are defined on the original event sequence and then augmented with the summary-visibility rule above.

The candidate workload uses the decision-prefix interface and marginalizes the unresolved suffix with $\gamma = 0.9$. Its target is the recorded final outcome. Candidate conditioning is observational rather than an estimate of the causal effect of forcing an action.

AgentDojo masks. The first event remains observed. MCAR masks the remaining positions independently. Block and suffix masks hide approximately the specified fraction of the sequence. The single-event family hides one uniformly chosen noninitial event, and alternating masking hides alternating noninitial positions. All eight families have a fixed endpoint.

A.4 TEACHERS AND TRAJECTORY LAWS

The InjecAgent teacher is one frozen width-64 GRU. AgentDojo uses width-64 Transformer and GRU teachers trained independently with seeds 0, 1, and 2; the Transformer has two encoder layers and four attention heads.

All teachers use trajectory-level safety labels and 20% independent mask augmentation during training. They are frozen before distillation. Distillation accesses only their probabilities on completed sequences, including sampled completions; it does not access hidden states or gradients. Direct masked teacher outputs are distinct from the teacher-induced runtime posterior defined in Appendix B.1.

The trajectory model uses source-fit initial and transition counts with additive smoothing. All reported main results use $\beta = 1$ and final smoothing parameter $\alpha = 0.01$, selected by validation conditional completion likelihood. InjecAgent additionally imposes the exactly-one-summary support described in Appendix C.3; AgentDojo does not.

A.5 SHARED-SAMPLE DISTILLATION

Let $h(u, v) = -u \log v - (1-u) \log(1-v)$, and abbreviate the teacher and student unsafe probabilities by $w(x)$ and $s(x)$. Each source trajectory is assigned in a balanced randomized design to one of the four seen query families. For evidence e_i, we cache $N = 8$ completed sequences $x_{i1}, \ldots, x_{iN}$ sampled from the training trajectory law conditioned on e_i, together with the corresponding teacher

probabilities. The empirical query losses are

$$\widehat{\mathcal{L}}_{\mathrm{comp}} = \frac{1}{nN} \sum_{i=1}^{n} \sum_{j=1}^{N} h\big(w(x_{ij}), s(x_{ij})\big), \tag{21}$$

$$\widehat{\mathcal{L}}_{\mathrm{post}} = \frac{1}{n} \sum_{i=1}^{n} h\left(\frac{1}{N} \sum_{j} w(x_{ij}), \frac{1}{N} \sum_{j} s(x_{ij}) \right). \tag{22}$$

The trace loss averages $h(w(x_i), s(x_i))$ on source trajectories. Trace-only uses that loss alone. The completion and posterior variants use $(\widehat{\mathcal{L}}_{\mathrm{trace}} + 4\widehat{\mathcal{L}}_{\mathrm{comp/post}})/5$. Dual uses the mean of the two query losses in this expression. The factor four corresponds to the four seen query families.

For a fixed seed, the loss variants reuse the same evidence, completed samples, and teacher targets. The teacher and student means in Eq. (22) use the same completed samples. The eight-sample posterior loss is therefore a finite-sample approximation to the population objective.

Fixed-length completions are sampled by forward filtering and backward sampling under the fitted Markov law, including the support state when applicable; they need not occur in the source dataset. InjecAgent candidate training uses geometric continuations. A negligible tail beyond length 256 is truncated when constructing the training cache; runtime future contraction does not truncate the suffix horizon.

InjecAgent selection. We search TT ranks $\{16, 32, 64\}$ and learning rates $\{0.001, 0.003, 0.01\}$ for the distillation objectives using validation data for early stopping and model selection. Held-out query families are not used for selection. The final BCE fits use three independent seeds. Trace-only selects rank 32, while completion, posterior, and dual select rank 16; all select learning rate 0.01.

AgentDojo selection. For each teacher, we search ranks $\{16, 32\}$ and learning rates $\{0.003, 0.01\}$ for the four BCE objectives. Early stopping and model selection use validation data. All four objectives select rank 32 and learning rate 0.01. The GRU-teacher experiment repeats the search for trace-only and dual.

A.6 Baseline training and calibration

Pooled predictors use one output head and label supervision on completed trajectories plus the four seen runtime views. Missing positions remain explicit masks. Optional query-type tokens distinguish completed, fixed-length missing, and, on InjecAgent, candidate inputs.

Logistic predictors. Pooled logistic uses TF–IDF discrete-token features with validation-selected regularization and optional query-type tokens. Independent heads fit one predictor and calibrator for each query family and one for complete trajectories. AgentDojo therefore has nine heads. Its held-out-family cell includes supervision on those families and is not an unseen-query generalization result.

Pooled GRU and Transformer. We select hidden width, learning rate, and optional query-type tokens using seen-family validation data. The GRU has one recurrent layer; the Transformer has two layers, four attention heads, dropout 0.1, and masked mean pooling. Optimization uses AdamW with early stopping. Final neural baselines are evaluated across three independent fits.

Classical joint models. The Markov baseline fits class-conditional first-order trajectory models and a class prior. The HMM baseline fits class-conditional hidden Markov models, with the number of states and smoothing selected on validation data. These baselines use their own learned trajectory marginals; on InjecAgent they obey the same legal support and geometric-length convention as TTMonitor. They are label-supervised joint models rather than students of the teacher.

Shared calibration. Prediction uses a validation-selected scalar log-odds offset,

$$\widehat{q}_{\delta}(e) = \mathrm{sigmoid}(\mathrm{logit}\, q(e) + \delta)\,. \tag{23}$$

One offset is fitted jointly on the four seen families with equal family weight and reused on held-out families. Independent heads use query-specific calibration. Fidelity and compatibility use native, unshifted outputs. No test label, Query MAE, or compatibility score is used for model selection.

Training repetitions. On InjecAgent, the teacher is fixed, while each displayed TT student, pooled neural predictor, and selected HMM is evaluated across three independent fits; logistic and Markov baselines are deterministic. On AgentDojo, each teacher architecture is trained with three seeds and paired with the corresponding student seed. Metrics are averaged across replicas after evaluation rather than from an ensemble of their predictions.

B Evaluation Metrics and Statistical Procedures

B.1 Prediction and monitor fidelity

For query family f, with evidence e_{if} and final label y_i, we compute

$$\mathrm{Brier}_f = \frac{1}{n_f}\sum_{i=1}^{n_f}\big(\widehat{q}_\delta(e_{if}) - y_i\big)^2. \tag{24}$$

Seen and Held-out are equal-family averages over their respective family sets.

Complete fidelity is the mean absolute difference between a model's native completed prediction and its designated frozen teacher:

$$\mathrm{MAE}_{\mathrm{complete}} = \frac{1}{n}\sum_i \left|q(y=1 \mid x_i, T_i) - w(x_i)\right|. \tag{25}$$

For runtime fidelity, the reference is the teacher-induced posterior under the common trajectory law μ:

$$t(e) = \mathbb{E}_{X\sim\mu(\cdot|e)}[w(X)], \qquad \widehat{t}(e) = \frac{1}{M}\sum_{j=1}^{M} w(X_j), \quad M = 8192. \tag{26}$$

Query MAE averages $|q(e) - \widehat{t}(e)|$ equally across query families. The reference is evaluated on fixed length-stratified subsets rather than all evaluation queries.

All methods on InjecAgent use the same GRU reference. On AgentDojo, the main comparison uses the corresponding Transformer teacher, while the GRU extension uses its own teacher. The Monte Carlo reference draws are shared across compared methods and are independent of the training caches.

B.2 Query Consistency

The used query group. Choose a fixed-length parent evidence pattern e with at least two hidden positions, denoted A and B, while keeping other missing positions unresolved. The group contains the parent risk $q_\varnothing$, all single-position refinements $q_{a\bullet}$ and $q_{\bullet b}$, and all admissible two-position refinements q_{ab}.

An explaining joint distribution is represented by

$$\begin{aligned} &m_{ab} = p(A=a, B=b \mid e) > 0, \qquad \sum_{(a,b)\in\mathcal{V}} m_{ab} = 1, \\ &u_{ab} = p(A=a, B=b, Y=1 \mid e), \qquad 0 \le u_{ab} \le m_{ab}. \end{aligned} \tag{27}$$

Its risks are

$$\begin{aligned} \pi_\varnothing &= \sum_{a,b} u_{ab}, & \pi_{a\bullet} &= \frac{\sum_b u_{ab}}{\sum_b m_{ab}}, \\ \pi_{\bullet b} &= \frac{\sum_a u_{ab}}{\sum_a m_{ab}}, & \pi_{ab} &= \frac{u_{ab}}{m_{ab}}. \end{aligned} \tag{28}$$

Sums run over admissible cells. Positive support ensures that every queried conditional is well defined. Completion probabilities are optimized freely: the audit asks whether some positive-support joint can explain the predictions.

Joint repair. For group g, collect the risks into vectors q_g and $\pi(m, u)$. The minimum repair is

$$J_g = \inf_{(m,u) \text{ satisfying (27)}} \|\pi(m, u) - q_g\|_\infty, \qquad \text{JointRepair} = \frac{1}{G}\sum_{g=1}^{G} J_g. \tag{29}$$

Thus, J_g is the smallest uniform perturbation of the group's predictions required to admit a common joint.

Incompatibility rate. Let $\mathcal{F}_g(\tau)$ be the set of positive-support joints satisfying $|\pi_i - q_{g,i}| \le \tau$ for every answer i in group g. We report

$$\text{Incompatibility}(\tau) = \frac{100}{G}\sum_{g=1}^{G} \mathbf{1}\{\mathcal{F}_g(\tau) = \varnothing\}, \qquad \tau = 10^{-3}. \tag{30}$$

This is a percentage of query groups, not a label-error rate.

Linear feasibility and positive support. For a fixed perturbation budget δ, compatibility can be checked by linear feasibility. For any queried event E_i, with total mass $m(E_i)$ and unsafe mass $u(E_i)$,

$$\max(0, q_i - \delta)m(E_i) \le u(E_i) \le \min(1, q_i + \delta)m(E_i). \tag{31}$$

We solve the resulting linear programs numerically and recompute all conditional risks from returned masses before accepting a feasible solution.

For explicit joint models, independently contracted positive cell masses provide an additional compatibility witness. Their residual repair values are at numerical precision and should not be interpreted as rankings among joint architectures.

Using a fixed seed, we choose eight parents per eligible fixed-length query family without using model predictions or labels for selection. InjecAgent contains 80 groups from 69 scenarios across ten families; AgentDojo contains 56 groups from 38 task groups across seven families. InjecAgent retains its legal-summary support.

A zero detected rate means that no incompatibility was found on the audited groups at the specified tolerance; it is not a global compatibility certificate for a direct predictor. Positive incompatibility likewise measures disagreement with a common-joint interpretation rather than downstream safety harm.

B.3 Confidence intervals

Prediction, fidelity, and compatibility use 2,000 bootstrap resamples of scenario clusters on InjecAgent or task-group clusters on AgentDojo. All records and queries from a sampled cluster are retained together. Prediction and Query MAE preserve equal-family aggregation, while compatibility averages the sampled audit groups directly. For stochastic methods, training replicas are additionally resampled, and AgentDojo fidelity preserves teacher–student pairing.

We report percentile 95% bootstrap intervals. When the main text displays "mean $\pm$ half-width," the half-width is half the percentile-interval width and need not be symmetric around the mean. The intervals condition on the selected models, audited queries, and Monte Carlo reference.

C Joint Construction and Exact Inference

This section gives the algebra behind the construction and the conditions needed for unknown-length inference. We use the main text's A_x, B_x, C_x, H_x, R_y, and G_y throughout. Statements of exactness concern inference in the fitted joint, not equality between the student and its teacher.

C.1 Normalization and coupling

The student is a conditional distribution. Nonnegative factors and the row-stochastic constraints imply

$$\sum_y s(y \mid x, T) = \ell^\top A_{x_1} \cdots A_{x_T}(r_0 + r_1) = \ell^\top \mathbf{1} = 1. \tag{32}$$

Softmax parameterization enforces these constraints throughout training. The parameterization contains $r + dr^2 + 2r$ trainable logits.

Product representation. Since B_x has only column x nonzero,

$$\ell_{x_1}^\top B_{x_2} \cdots B_{x_T} \mathbf{1} = p(x_1)^\beta \prod_{t=2}^{T} P(x_t \mid x_{t-1})^\beta = b(x)^\beta. \tag{33}$$

The mixed-product identity gives

$$\begin{aligned} H_{x_1}^\top C_{x_2} \cdots C_{x_T} R_y &= \big(\ell_{x_1}^\top B_{x_2} \cdots B_{x_T} \mathbf{1}\big)\big(\ell^\top A_{x_1} \cdots A_{x_T} r_y\big) \\ &= b(x)^\beta s(y \mid x, T). \end{aligned} \tag{34}$$

Thus C_x has bond dimension at most dr and introduces no additional learned parameters.

Summing Eq. (34) over events and labels yields the partition function in the main text. Conditioning on a positive-mass completed trajectory cancels the common trajectory factor:

$$Q(y \mid x, T) = s(y \mid x, T). \tag{35}$$

The joint therefore exactly retains the student; agreement with the frozen teacher is determined by distillation.

C.2 Evidence contractions and refinement identities

Let $\mathcal{X}_t(e)$ denote the permitted events at position t and define

$$H_e^\top = \sum_{x \in \mathcal{X}_1(e)} H_x^\top, \qquad D_t(e) = \sum_{x \in \mathcal{X}_t(e)} C_x \quad (t \geq 2). \tag{36}$$

Distributivity gives

$$Z_y(e) = H_e^\top D_2(e) \cdots D_T(e) R_y = \sum_{x \models e} b(x)^\beta s(y \mid x, T). \tag{37}$$

The posterior is $Z_1/(Z_0 + Z_1)$ for positive-mass evidence.

If $\{e_a\}_a$ partitions e into disjoint refinements,

$$Q(Y = 1 \mid e) = \sum_a Q(e_a \mid e) Q(Y = 1 \mid e_a). \tag{38}$$

The weights are induced by the same joint. Markov and HMM joints satisfy the same identity; compatibility is a property of the joint representation, not of tensor trains specifically.

C.3 Legal support without changing the TT framework

InjecAgent additionally restricts valid trajectories to those containing exactly one risk-summary event. Let $\sigma(x)$ indicate such an event. We augment the trajectory state with a count $c \in \{0, 1\}$:

$$\overline{C}_x\big((c, i, u), (c', j, v)\big) = P(x \mid i)^\beta A_x(u, v) \mathbf{1}[j = x] \mathbf{1}[c' = c + \sigma(x) \leq 1]. \tag{39}$$

The first-site vector initializes $c = \sigma(x_1)$, while the terminal vector retains only states with $c = 1$. Consequently, contraction sums exactly over trajectories containing one summary event.

The count doubles the formal bond dimension but leaves the factorized asymptotic complexity unchanged. If the summary is hidden, its position is marginalized; the true hidden insertion position is not supplied to inference. AgentDojo uses the unaugmented construction.

C.4 Unknown lengths and the future boundary

For $\beta = 1$, we define the length-aware joint by

$$Q(T, x, y) = (1-\gamma)\gamma^{T-1} b(x \mid T) s(y \mid x, T), \qquad T \geq 1. \tag{40}$$

Since $b(\cdot \mid T)$ and $s(\cdot \mid x, T)$ are normalized, this defines a normalized joint over finite trajectories. For an observed prefix, marginalizing a suffix of unknown length gives

$$G_y = (1-\gamma) \sum_{m=0}^{\infty} \gamma^m S^m R_y. \tag{41}$$

Candidate evidence contributes one additional transition C_a; its common factor γ cancels when normalizing across labels.

Proof of Proposition 1. Let $M = \gamma S$. If $\rho(M) < 1$, the Neumann series converges,

$$\sum_{m=0}^{\infty} M^m = (I - M)^{-1},$$

and therefore

$$G_y = (1-\gamma)(I - \gamma S)^{-1} R_y.$$

This is the unique solution of

$$G_y = (1-\gamma) R_y + \gamma S G_y.$$

For the reported setting $\beta = 1$, S is stochastic or substochastic, so $\rho(S) \leq 1$ and $0 < \gamma < 1$ is sufficient for convergence.

In implementation, G_y is computed once by fixed-point iteration and cached for subsequent runtime queries; no finite suffix horizon is imposed.

C.5 Factorized implementation and complexity

The implementation does not materialize B_x, C_x, or a dense $dr \times dr$ matrix S. Reshape a joint row message as $M(i, u)$. A missing-event update factors as

$$N(x, u) = \sum_i M(i, u) P(x \mid i)^{\beta}, \qquad M'(x, v) = \sum_u N(x, u) A_x(u, v). \tag{42}$$

The two operations cost $O(d^2 r)$ and $O(dr^2)$ and require $O(dr)$ working storage. Consecutive observed updates cost $O(r^2)$.

Thus a worst-case missing query costs $O(T(d^2 r + dr^2))$, while a completed query costs $O(Tr^2)$. Future-boundary construction costs $O(N_{\text{fp}}(d^2 r + dr^2))$. Once the boundary is cached, a fully observed prefix costs $O(kr^2)$ and each additional candidate costs $O(r^2)$, reducing K candidate evaluations from $O(Kkr^2)$ to $O(kr^2 + Kr^2)$.

Model storage is $O(d^2 + dr^2)$, with an additional $O(dr)$ working memory for messages and boundaries. This storage bound exploits the factored transition structure rather than a generic dense TT of bond dimension dr.

D Runtime Inference Cost

The three panels of the main text's efficiency figure evaluate sampling error within the distilled joint, warm single-query latency, and reuse across candidate queries. All use the final InjecAgent trajectory law and three rank-16 dual TTMonitor students.

D.1 Hardware and timing protocol

Measurements use an AMD EPYC 7742 CPU with one process and one numerical thread; no GPU is used. Neural predictors use float32 arithmetic and joint contractions use float64.

All inference paths are warmed before measurement and evaluated over five interleaved timing rounds. Inputs are pre-encoded. Timings exclude text encoding, checkpoint loading, and training. Model-static caches are allowed, but complete-answer memoization is not.

D.2 Panel A: contraction versus completion sampling

The workload contains 32 length-stratified queries from each of four families: candidate, MCAR 20%, block 40%, and retained-prefix 60%. Exact inference contracts the student joint, while Monte Carlo samples from the same joint's conditional trajectory distribution and averages the student's completed predictions.

Budgets are 16, 64, 256, and 1,024 completions. Absolute sampling error is measured against exact contraction of the same student joint.

Table 7: Same-joint inference. Values are means with 95% interval endpoints. Exact contraction is the numerical reference.

Inference	Query latency (ms)	Absolute sampling error
Exact TTMonitor	0.696 [0.690, 0.703]	0 (reference)
Student MC, 16	2.926 [2.889, 2.976]	0.046270 [0.044377, 0.048037]
Student MC, 64	4.349 [4.285, 4.465]	0.024892 [0.023074, 0.026680]
Student MC, 256	6.829 [6.759, 6.954]	0.011583 [0.011063, 0.012141]
Student MC, 1,024	11.820 [11.684, 12.099]	0.005992 [0.005761, 0.006254]

D.3 Panel B: single-query native interfaces

We use 64 length-stratified queries from each of 12 families: complete, candidate, and ten missing-evidence families. Missing latency is the equal-family mean over the latter ten. Three checkpoints each are used for pooled GRU, pooled Transformer, HMM, and TTMonitor.

Table 8: Warm single-query latency in milliseconds, mean [95% CI].

Method	Complete	Missing	Candidate
Pooled GRU	.530 [.526,.537]	.567 [.566,.569]	.458 [.454,.465]
Pooled Transformer	.666 [.657,.685]	.699 [.691,.715]	.695 [.661,.794]
HMM joint	.520 [.516,.524]	.575 [.571,.579]	.317 [.315,.320]
TTMonitor	.025 [.024,.025]	.849 [.838,.875]	.052 [.044,.074]

D.4 Panel C: shared-prefix candidate evaluation

The workload fixes prefix length at $k = 32$ and varies $K \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. Sixteen valid prefixes are sampled from the source-fitted trajectory law. For each prefix, event alternatives are selected using source transition probabilities without outcome labels or monitor predictions.

Both implementations use the same cached future boundary. Independent evaluation recomputes the prefix for every alternative, while shared evaluation computes it once per candidate set. At $k = 32, K = 128$, their mean latencies are 9.921 ms and 0.826 ms, respectively, corresponding to a $12.01\times$ reduction.

D.5 Initialization and future-boundary construction

Table 9 separates checkpoint loading, static factor preparation, and future-boundary construction. Fixed-length complete and missing queries do not require a future boundary. Once constructed, the boundary is reused while the student, trajectory law, and continuation parameter remain fixed.

Reproducibility. Experiments use NumPy 1.26.4, SciPy 1.13.1, scikit-learn 1.5.2, and PyTorch 2.0.1+cpu; compatibility optimization uses SciPy's HiGHS interface. Dataset versions, preprocessing, splits, model selection, randomization, and evaluation procedures are specified in Appendix A and Appendix B.

Table 9: One-time setup costs in milliseconds, mean [95% CI].

Method	Checkpoint loading	Static preparation	Future boundary
Pooled GRU	1.414 [1.266,1.732]	.002 [.001,.004]	—
Pooled Transformer	4.408 [4.211,4.729]	.001 [.001,.002]	—
HMM joint	.368 [.341,.409]	.061 [.053,.079]	12.742 [9.913,14.773]
TTMonitor	1.351 [1.181,1.588]	.405 [.373,.445]	247.480 [244.921,252.348]