Continual, interactive, causal agents

Nando de Freitas and Pedro Ortega

June 7, 2026

Abstract

Modern language-model agents are usually built by stacking separate training regimes: pretraining, mid-training, supervised fine-tuning, preference modeling, rejection sampling, reinforcement learning, reasoning-specific tuning, self-distillation, and deployment-time patches. This is intelligence by design and engineering, as opposed to emergent intelligence. The multi-stage recipe is a research local minimum, but it has produced powerful systems. It has no single semantics for an interaction transcript: user messages, tool outputs, demonstrations, model actions, verifier judgements, and corrections are often treated as if they were the same kind of evidence. This paper studies a simpler alternative: a continual, causal interaction stream. The central rule is that world-written tokens are evidence, whereas self-written agent tokens are interventions. In LLM fine-tuning this rule becomes a loss mask: keep the agent's own attempts in the context, but remove them from the supervised target. In a small, reproducible STEM reasoning experiment, this interventional stream agent reaches held-out solve accuracy, which is comparable to that of ReST, GRPO, and SFT with oracle corrections. The result is not a claim of benchmark dominance; it is a proof of viability for a single continual learning agent that can use interaction, causality, feedback, and corrections to achieve purposeful and useful behaviour.

Introduction

Large language models are no longer only passive next-token predictors. They answer questions, call tools, browse documents, write code, invoke APIs, use verifiers, and receive corrections in multi-turn loops. Systems such as WebGPT, ReAct, Toolformer, Gorilla, and Voyager make this agentic pattern explicit: a model acts, the world responds, and the next decision depends on the history [7, 11, 21, 26, 27]. Yet the dominant training story for such systems is still a sequence of loosely connected stages. First we pretrain on web data; then we continue or mid-train on selected domains; then we run supervised fine-tuning on instructions; then we collect preferences, fit reward models, use RLHF or rejection sampling, add reasoning-specific procedures, and finally distill the improved behavior back into another model [2, 4, 6, 10, 22]. The pipeline works, but it is conceptually fragmented: many datasets, many objectives, many interfaces, and many opportunities for a handoff to silently change the meaning of the data.

Figure 1 summarizes the motivation. The left side is the familiar "Frankenstein" post-training stack. It can be highly effective, but each stage requires a new interpretation of what counts as a target: a web token, an instruction answer, a preference label, a sample selected by a verifier, a reward, a chain-of-thought trace, or a distilled completion. The right side asks whether we can instead keep one representation throughout deployment and training: an interaction stream containing questions, demonstrations, agent attempts, verifier outputs, tool observations, oracle corrections, and ordinary environment feedback. In such a stream, a teacher solution and an agent solution can look identical as strings. The difference is provenance. A teacher solution is world-written evidence. The agent's own solution is an action.

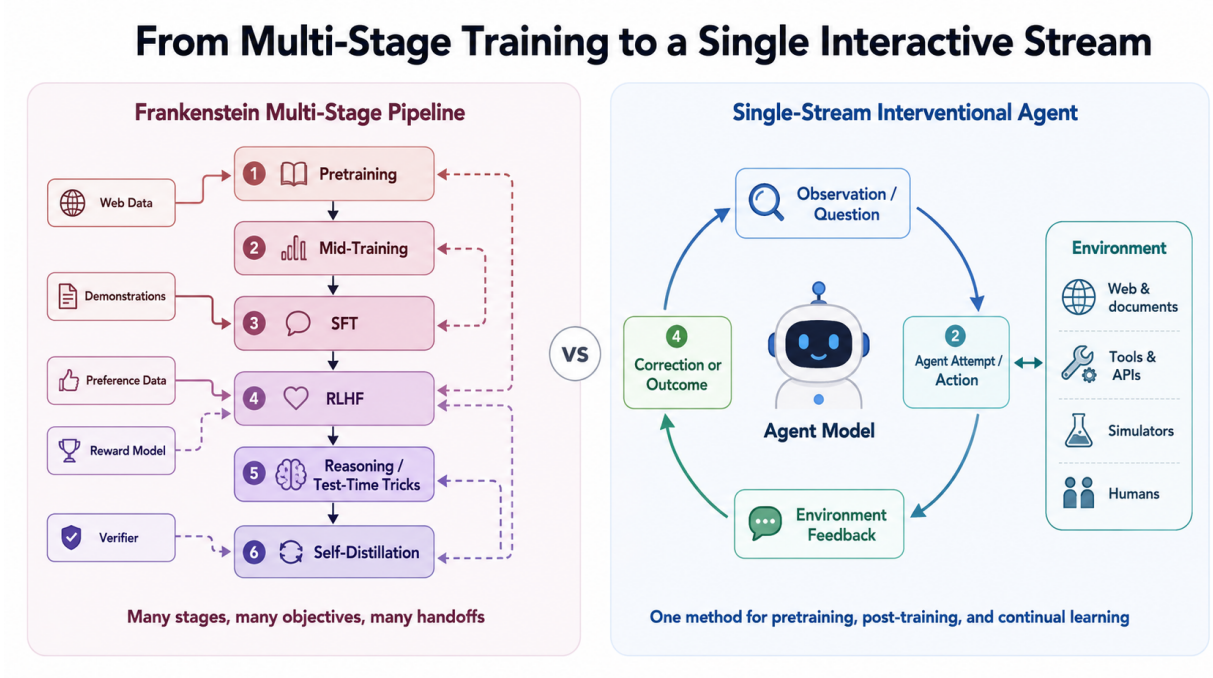

Figure 1: The field of AI is at at steep local minimum: The left-hand pipeline treats pretraining, mid-training, SFT, RLHF, reasoning tricks, and self-distillation as separate engineered stages with separate data interfaces. The right-hand picture is the alternative studied here: a single interventional stream in which the agent receives an observation or question, acts, observes environment feedback, and learns from externally grounded corrections or outcomes.

This provenance distinction is exactly Pearl's distinction between conditioning and intervention [12, 13]. Observing an action-like event a gives the conditional distribution $\mathbb{P}(o \mid a)$: it asks what tends to be true in records where that event occurred. Setting the action ourselves gives $\mathbb{P}(o \mid \mathrm{do}\,(a))$: it asks what follows after the action mechanism has been overwritten. For a deployed agent, its own turns belong to the second category. The agent should condition on what it did when predicting later world responses, but it should not update as if its own attempt were independent evidence about the world. In a causal language-model loss, the translation is simple: keep self-authored tokens in the input prefix, but mask them out of the target labels. World-written tokens - user messages, teacher demonstrations, tool outputs, verifier verdicts, and oracle corrections - remain supervised evidence.

The point matters because modern post-training is already interactive imitation. Behavior cloning and learning from demonstrations have a long history in control and robotics [1, 16, 20]. Their classic failure mode is covariate shift: after a learner makes a mistake, it reaches states that were rare under the teacher, and future errors compound [18, 19]. LLM agents face an analogous problem at the token level. If we fine-tune on transcripts without marking who wrote which tokens, the model can learn from its own failed completions as if the world had endorsed them. This is the same self-confirmation pathology studied in causal sequence-model accounts of delusion [9], and it is related to empirical failures such as sycophancy and model collapse under repeated self-training [14, 23, 24].

The alternative developed here is a continual, interactive, causal (CIC) objective. It is close in spirit to interactive imitation and to Ortega's interactional account of agency, in which purposeful

behavior can be learned from the structure of an interaction history rather than from a primitive scalar reward [8]. It is also consistent with the causal view of decision making in bandits and reinforcement learning: an arm being pulled by someone else is evidence, whereas pulling the arm ourselves is an intervention [3, 28]. The contribution here is deliberately practical. We apply the same causal accounting to the ordinary token-level supervised loss used for LLM fine-tuning.

The experiment is designed to isolate the idea. Starting from the same one-epoch SFT seed of a Qwen2.5-0.5B model [17], we compare SFT, ReST, GRPO, SFT+oracle-corrections, an observational stream agent, and an interventional stream agent on the same synthetic STEM reasoning tasks. Every method uses the same verifier and the same held-out solve-accuracy metric. The key ablation is especially instructive. SFT+oracle-corrections and the interventional stream receive the same 400 oracle correction targets and the same 17,924 supervised correction tokens. The difference is that SFT+oracle-corrections learns a direct map $x \mapsto y^\star$, whereas the interventional stream learns a repair map $(x, \mathrm{do}(\widehat{y}), v) \mapsto y^\star$: it sees the failed attempt and the verifier verdict as context, but it does not imitate the failed attempt. This extra repair state is what the prediction lemma later formalizes, and it is what the results show: 85% held-out solve accuracy for the interventional stream versus 81% for SFT+oracle-corrections.

The comparison with the observational stream is the causal lesson in its sharpest form. The observational and interventional stream agents see the same 785 rollouts, the same questions, the same first attempts, the same verifier judgements, and the same 400 oracle corrections. The observational stream supervises 51,121 token positions and reaches 61% solve accuracy. The interventional stream supervises only 17,924 token positions and reaches 85%. More supervised tokens are worse when many of those tokens are the learner's own mistakes. The useful data are not just the corrections; they are the corrections in the right causal context.

This paper makes four contributions.

1. It gives a tutorial derivation of why an agent's own actions are interventions rather than observations, building on do-calculus and structural causal models [5, 12, 13, 15, 25].
2. It converts that derivation into a token-level objective for continual interactive causal agents: supervise world-written evidence, mask self-authored actions, and condition on both.
3. It provides a reproducible notebook-style experiment covering dataset construction, stream construction, ReST, GRPO, SFT+oracle-corrections, observational stream training, interventional stream training, and solve-accuracy evaluation.
4. It shows that stream agents are a viable alternative to existing verifier-based recipes: in this run, the interventional stream is competitive with ReST, stronger than GRPO, and more robust than observational stream training while using a simpler supervised objective.

The claim is not that this small STEM experiment settles post-training. It does not. The claim is more basic: the intervention/evidence distinction gives a single semantics for interaction data. That semantics can apply before deployment, during post-training, and in continual learning after the agent is already acting in the world. Instead of treating pretraining, SFT, RLHF, reasoning, and self-distillation as unrelated rituals, a stream agent asks one question at every token: who wrote this, and should it be evidence or context?

1 Causal interactive agents

1.1 Bayes under intervention

Let $p \in \mathcal{P}$ be a latent program, hypothesis, schema, or LLM expert that can affect both the chosen action and the resulting observation. Observationally, the joint distribution factors as

$$\mathbb{P}(p, a, o) = \mathbb{P}(p)\,\mathbb{P}(a \mid p)\,\mathbb{P}(o \mid p, a).$$

If we condition on the event (observation) $a = a^\star$, the Bayes posterior is:

$$\mathbb{P}(p \mid a^\star, o) = \frac{\mathbb{P}(p)\mathbb{P}(a^\star \mid p)\mathbb{P}(o \mid p, a^\star)}{\sum_{p' \in \mathcal{P}} \mathbb{P}(p')\mathbb{P}(a^\star \mid p')\mathbb{P}(o \mid p', a^\star)}.$$

Under an intervention, however, the mechanism $\mathbb{P}(a \mid p)$ is removed and replaced by a point mass at the imposed value:

$$\mathbb{P}_{\mathrm{do}(a^\star)}(p, a, o) = \mathbb{P}(p)\,\delta_{a^\star}(a)\,\mathbb{P}(o \mid p, a),$$

where

$$\delta_{a^\star}(a) = \begin{cases} 1, & a = a^\star, \\ 0, & a \neq a^\star. \end{cases}$$

The factor $\mathbb{P}(a^\star \mid p)$ has changed because the agent chose one specific value: $a = a^\star$. If a conditional is computed by normalising this interventional joint, the delta collapses the action sum:

$$\begin{aligned} \mathbb{P}(o \mid \mathrm{do}\,(a^\star)) &= \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} \mathbb{P}(p)\delta_{a^\star}(a)\mathbb{P}(o \mid p, a) \\ &= \sum_{p \in \mathcal{P}} \mathbb{P}(p)\mathbb{P}(o \mid p, a^\star). \end{aligned}$$

When an outcome arrives, the posterior updates through the outcome channel:

$$\mathbb{P}(p \mid \mathrm{do}\,(a^\star), o) = \frac{\mathbb{P}(p)\mathbb{P}(o \mid p, a^\star)}{\sum_{p' \in \mathcal{P}} \mathbb{P}(p')\mathbb{P}(o \mid p', a^\star)}.$$

The action is recorded in the conditioning variables for predicting consequences, but its probability as an action is not used as evidence.

1.2 LLM training

To extend the ideas of Bayesian updates and causal updates to LLMs, we instantiate two fresh copies of a base LLM model, $p_{\theta^{\mathrm{obs}}}$ and $p_{\theta^{\mathrm{do}}}$, and fine-tune each on the *same* data, with the *same* optimizer. The only difference is which tokens contribute to the loss:

- **Observational SFT** ($p_{\theta^{\mathrm{obs}}}$): Every "dialogue" token is supervised, including the agent's own turns. This corresponds to using the observational likelihood.
- **Interventional SFT** ($p_{\theta^{\mathrm{do}}}$): Only the world-written tokens (world turns and structural prefixes) contribute to the loss. The agent's own turns are masked. This corresponds to the interventional likelihood.

Both models still *see* the agent's turns in the input context. They are only excluded from the gradient.

Let θ denote the model parameters, h a tokenized dialogue with token sequence $z_1, \ldots, z_T$, and $\gamma_i \in \{0, 1\}$ the gate indicating who wrote slot i ($\gamma_i = 1$ for the agent, $\gamma_i = 0$ for the world). The two losses are:

$$\mathcal{L}^{\text{obs}}(\theta) = -\frac{1}{T} \sum_{i=1}^{T} \log p_\theta(z_i \mid z_{<i}, c), \tag{1}$$

$$\mathcal{L}^{\text{do}}(\theta) = -\frac{1}{|\{i : \gamma_i = 0\}|} \sum_{i:\gamma_i=0} \log p_\theta(o_i \mid z_{<i}, c). \tag{2}$$

Equation (1) is the standard causal-LM loss, namely the negative log of the observational likelihood for the dialogue. Equation (2) drops the action-channel factors from the sum, leaving only world-channel factors.

1.3 Interventional vs SFT learning in a stream with the same teacher

In interventional learning, the agent actions are recorded in the conditioning variables for predicting consequences. SFT could use the same teacher as an interventional agent and receive the same feedback. Their losses would be identical, but an interventional agent conditions on its own actions as well. Intevential is a self-improvement method.

To see this, let us consider a small example with the following history:

$$t_1, a_1, t_2, a_2, t_3, x_3, t_4, x_4, \ldots$$

where t stands for a teacher turn, a an agent turn, and x some environmental turn. Suppose that t_4 is square-integrable. Define the coarse and rich information sets

$$\mathcal{G}_0 = \sigma(t_1, t_2, t_3), \qquad \mathcal{G}_1 = \sigma(t_1, a_1, t_2, a_2, t_3, x_3).$$

Then $\mathcal{G}_0 \subseteq \mathcal{G}_1$. Let

$$m_0 = \mathbb{E}[t_4 \mid \mathcal{G}_0], \qquad m_1 = \mathbb{E}[t_4 \mid \mathcal{G}_1].$$

Thus m_0 is the SFT predictor that uses only teaching turns (e.g. answers to questions) t_1, t_2, t_3, while the interventional agent m_1 also uses its previous actions and any other data a_1, a_2, x_3.

Prediction lemma

For squared-error prediction, the richer predictor is weakly better:

$$\mathbb{E}[(t_4 - m_1)^2] \leq \mathbb{E}[(t_4 - m_0)^2].$$

More precisely,

$$\mathbb{E}[(t_4 - m_0)^2] = \mathbb{E}[(t_4 - m_1)^2] + \mathbb{E}[(m_1 - m_0)^2].$$

The inequality is strict exactly when the extra variables a_1, a_2, x_3 change the conditional mean of t_4 with positive probability.

Proof. By the tower property,

$$\mathbb{E}[m_1 \mid \mathcal{G}_0] = \mathbb{E}[\mathbb{E}[t_4 \mid \mathcal{G}_1] \mid \mathcal{G}_0] = \mathbb{E}[t_4 \mid \mathcal{G}_0] = m_0.$$

Also $\mathbb{E}[t_4 - m_1 \mid \mathcal{G}_1] = 0$. Since $m_1 - m_0$ is $\mathcal{G}_1$-measurable, the cross term below vanishes:

$$\begin{aligned}\mathbb{E}[(t_4 - m_1)(m_1 - m_0)] &= \mathbb{E}[\mathbb{E}[(t_4 - m_1)(m_1 - m_0) \mid \mathcal{G}_1]] \\ &= \mathbb{E}[(m_1 - m_0)\mathbb{E}[t_4 - m_1 \mid \mathcal{G}_1]] \\ &= 0.\end{aligned}$$

Therefore

$$\begin{aligned}\mathbb{E}[(t_4 - m_0)^2] &= \mathbb{E}[(t_4 - m_1 + m_1 - m_0)^2] \\ &= \mathbb{E}[(t_4 - m_1)^2] + \mathbb{E}[(m_1 - m_0)^2],\end{aligned}$$

which proves the claim.

Of course, for the gap to be meaningful, the previous agent self-actions or side information have to carry useful information. We illustrate this next with a simple question and answer experiment.

2 Experiment

Suppose we already have a small SFT policy, a deterministic verifier, and a stream of new quantitative problems. Can we use the agent's own attempts, together with verifier feedback and oracle corrections, as a training signal without turning failed attempts into imitation targets?

This matters for continual learning. In deployment, pre-training and post-training are not really separate regimes. An assistant continually receives new prompts, calls tools, produces answers, observes tool outputs, and receives corrections. A reward-based post-training method can be useful, but it introduces a new interface: scalar rewards, policy ratios, KL penalties, replay buffers, selection rules, and extra hyperparameters. The interventional stream view offers a single rule that can operate both before and after deployment: world-written tokens are evidence, self-written tokens are interventions, and the learner may condition on its own interventions without treating them as targets. In that sense, stream agents are a candidate for one continual-learning recipe that works for both pre-training-style imitation and post-training-style correction.

ReST and GRPO are two common offline and online ways to exploit a verifier. ReST samples many candidate completions, keeps the completions that pass the verifier, and then runs another SFT pass on those selected samples. GRPO keeps the online sampling loop, converts verifier scores within a group into relative advantages, and applies a policy-gradient update against a reference policy. Both are sensible recipes. The stream-agent alternative is deliberately more direct. It records an interaction stream

$$x \quad \longrightarrow \quad \widehat{y} \quad \longrightarrow \quad v \quad \longrightarrow \quad y^\star, \tag{3}$$

where x is the problem, $\widehat{y}$ is the agent's sampled solution, v is the verifier's judgement, and $y^\star$ is the oracle correction when the attempt is wrong. The causal rule is the same as before: $\widehat{y}$ is an action, so the agent should read it as $\mathrm{do}(\widehat{y})$. The verifier and oracle are world-written observations, so they are evidence.

This experiment puts SFT, ReST, GRPO, an observational stream agent, an interventional stream agent, and an SFT+oracle-corrections baseline into one reproducible harness. All trainable methods use the same base model, the same synthetic STEM generator, and the same deterministic numeric verifier. The headline comparison is held-out solve accuracy on $\mathcal{D}_{\text{test}}$. The intended lesson is not that stream agents get a stronger verifier. They get the same verifier. The intended lesson is that

verifier/correction transcripts can be used as ordinary supervised data once provenance is marked correctly. The observational stream supervises the agent's proposal tokens. The interventional stream keeps those proposal tokens in context but masks them from the loss.

2.1 Dataset and verifier

Each dataset row is an atomic quantitative exercise with a known formula, a numeric answer, a unit when appropriate, and a tolerance. We write one row as

$$x_j = (q_j, y^\star{}_j, \alpha_j, u_j, \epsilon_j^{\text{rel}}, \epsilon_j^{\text{abs}}, d_j, s_j), \tag{4}$$

where q_j is the natural-language question, $y^\star{}_j$ is the worked oracle solution, α_j is the target numeric answer, u_j is the target unit, d_j is the domain, and s_j is the skill. The experiment uses three domains (physics, chemistry, and materials) and ten skills: Ohm's law voltage, Ohm's law current, electric power, density, specific heat, molarity, pH, kinetic energy, engineering stress, and first-order Bragg diffraction.

The inspected run uses

$$|\mathcal{D}_{\text{SFT}}| = 200, \qquad |\mathcal{D}_{\text{agent}}| = 200, \qquad |\mathcal{D}_{\text{test}}| = 100. \tag{5}$$

The set $\mathcal{D}_{\text{SFT}}$ supplies clean teacher solutions for the initial SFT model. The set $\mathcal{D}_{\text{agent}}$ is reserved for ReST, GRPO, and stream rollouts. The set $\mathcal{D}_{\text{test}}$ is held out for evaluation.

A completion y is graded by a deterministic numeric verifier. The verifier first tries to extract the number after a `Final answer:` marker. If that marker is missing, it falls back to the last plausible answer number in the completion. The solve verdict is

$$\text{Solve}(x_j, y) = \mathbf{1}\Big\{|\widehat{\alpha}(y) - \alpha_j| \le \max\{\epsilon_j^{\text{abs}}, \epsilon_j^{\text{rel}}|\alpha_j|\}\Big\}, \tag{6}$$

with an additional unit check when the model explicitly supplies a unit. We use solve accuracy because it is a common metric for SFT, ReST, GRPO, and the stream agents.

Representative SFT teacher row

```
Question:
A strong acid solution has [H+] = 1.0e-04 M. What is the pH?
Proposed solution:
pH = -log10([H+]) = -log10(1.0e-04) = 4.000. Final answer: 4.000.
```

This is ordinary supervised data. The question and protocol prefix are context. The oracle solution is the supervised continuation.

2.2 The single stream approach

We create a stream that starts from the same SFT-1 single-epoch checkpoint. Without loss of generality, the assumption is the agent has started observing the world passively. We do this for simplicity of implementation only.

In the stream, we enable the agent to interact. For instance, imagine an agent browsing the web. When it finds a question with a solution, it attempts to answer the question. If it gets it correctly, it continues. If it gets it incorrectly, it gets told what the correct solution is because the solution (oracle) is available in this case. The only requirement is an evaluator, but it is well known in Computer Science that building an evaluator is far easier than building a solving agent.

When the agent is correct, the data looks like the following:

Correct first attempts: what the two stream agents see

```
Interventional: correct first attempt, but DO NOT train on self-actions (MASK them)
[MASK] Question:
[MASK] 12.5 g of solute (molar mass 98.0 g/mol) is dissolved to make 1.0 L of solution. What
    is the molarity?
[MASK] Proposed solution:
[MASK] n = 12.5/98.0 = 0.125 mol; M = n/V = 0.125 M. Final answer: 0.125 M.
[MASK] <eos>

Observational: the same correct rollout is an imitation target (no MASK for solution)
[MASK] Question:
[MASK] 12.5 g of solute (molar mass 98.0 g/mol) is dissolved to make 1.0 L of solution. What
    is the molarity?
[MASK] Proposed solution:
[LOSS] n = 12.5/98.0 = 0.125 mol; M = n/V = 0.125 M. Final answer: 0.125 M.
[LOSS] <eos>
```

If the agent's attempt fails, the oracle provides the correct solution as follows:

Stream when the agent is correct

```
Interventional: wrong first attempt
[MASK] Question:
[MASK] 10.0 g of solute (molar mass 40.0 g/mol) is dissolved to make 0.25 L of solution. What
     is the molarity?
[MASK] Proposed solution:
[MASK] n = 10.0/40.0 = 0.250 mol; M = n/V = 0.250 M. Final answer: 0.250 M.
[MASK] <eos>
[MASK] Proposed solution:
[LOSS] n = 10.0/40.0 = 0.250 mol; M = n/V = 1.000 M. Final answer: 1.000 M.
[LOSS] <eos>

Observational: the same wrong rollout
[MASK] Question:
[MASK] 10.0 g of solute (molar mass 40.0 g/mol) is dissolved to make 0.25 L of solution. What
     is the molarity?
[MASK] Proposed solution:
[LOSS] n = 10.0/40.0 = 0.250 mol; M = n/V = 0.250 M. Final answer: 0.250 M.
[LOSS] <eos>
[MASK] Proposed solution:
[LOSS] n = 10.0/40.0 = 0.250 mol; M = n/V = 1.000 M. Final answer: 1.000 M.
[LOSS] <eos>
```

The visible transcript is almost the same in both cases. The difference is provenance. In observational training, the first solution is a supervised target because it appeared in the transcript. In interventional training, the same first solution is an intervention by the learner and is therefore masked. This is the one-line distinction between $\mathbb{P}(o \mid a)$ and $\mathbb{P}(o \mid \mathrm{do}\,(a))$ translated into a causal-LM label tensor.

2.3 Methods

All trainable policies are LoRA adapters on top of the same Qwen2.5-0.5B causal language model. The plain evaluation prompt is

```
Question:
<question>

Proposed solution:
```

The methods differ only in how they turn verifier information into training data.

Base. The base model is evaluated without fine-tuning. It sees none of the synthetic teacher, rollout, or correction data.

SFT. The SFT seed trains on the 200 clean teacher rows in $\mathcal{D}_{\text{SFT}}$. For a prompt x_j and oracle solution $y^\star{}_j$, the loss is

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{(x_j, y^\star{}_j)\in\mathcal{D}_{\text{SFT}}} \log p_\theta(y^\star{}_j \mid x_j). \tag{7}$$

We train for only one epoch on purpose.

SFT + ReST. ReST starts from the SFT seed. For each sampled problem it draws a group of candidate answers

$$\widehat{y}_{jk} \sim p_{\theta_{\text{SFT}}}(\cdot \mid x_j), \qquad k = 1, \ldots, K, \tag{8}$$

with $K = 8$. It grades them with the numeric verifier, keeps verifier-passing parseable candidates, deduplicates, and in this run selects at most one clean candidate per prompt group. The selected set is

$$\mathcal{D}_{\text{ReST}} = \{(x_j, \widehat{y}_{jk}) : \text{grade}(x_j, \widehat{y}_{jk}) = 1\}. \tag{9}$$

The notebook samples 400 candidate completions in 50 groups, selects 30 verifier-correct candidates, adds 200 SFT replay teacher examples, and trains on 230 supervised pairs. ReST therefore learns only from sampled successes. Wrong attempts are rejected rather than used as repair contexts in this simple implementation.

SFT + GRPO. GRPO also starts from the SFT seed, but keeps the sampled group rather than throwing away the failed samples. For each group it converts verifier rewards r_{jk} into standardized group-relative advantages

$$A_{jk} = \frac{r_{jk} - \frac{1}{K}\sum_{\ell=1}^{K} r_{j\ell}}{\sqrt{\frac{1}{K}\sum_{\ell=1}^{K}(r_{j\ell} - \bar{r}_j)^2 + 10^{-6}}}, \tag{10}$$

and applies a clipped policy-gradient update against the old policy, with a KL penalty to the SFT reference model:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\sum_{j,k} \min\{\rho_{jk} A_{jk}, \text{clip}(\rho_{jk}, 1-\epsilon, 1+\epsilon) A_{jk}\} + \beta\, \widehat{D_{\text{KL}}}(p_\theta \,\|\, p_{\theta_{\text{ref}}}). \tag{11}$$

In this simple low-scale experiment, GRPO samples 400 completions in 50 groups. Twenty-four of the groups are flat, meaning the group gives no useful relative preference signal. The optimizer performs 26 non-flat update steps.

SFT+oracle-corrections. This baseline starts from SFT 1 and uses the oracle corrections from failed SFT-1 rollouts, but converts them back into ordinary direct-SFT rows:

$$\mathcal{L}_{\text{oracle}}(\theta) = -\sum_{j:\, v_j=0} \log p_\theta(y^\star{}_j \mid x_j). \tag{12}$$

It sees the same 400 oracle correction targets that the interventional stream uses. The difference is that the wrong first attempt $\widehat{y}_j$ is not in the conditioning context. In the saved output those 400 correction rows correspond to 70 unique correction questions from $\mathcal{D}_{\text{agent}}$.

Observational stream agent. The observational stream agent starts from SFT 1 and trains on every row in the stream rollout file. The notebook first asks the frozen SFT-1 model to attempt items from $\mathcal{D}_{\text{agent}}$ until it has collected 400 wrong attempts. This takes 785 stream turns. The observational objective treats the first attempt as an ordinary observation and therefore supervises it:

$$\mathcal{L}^{\text{obs}}_{\text{stream}}(\theta) = -\sum_{j=1}^{785} \log p_\theta(\widehat{y}_j \mid x_j) - \sum_{j:\, v_j=0} \log p_\theta(y^\star{}_j \mid x_j, \widehat{y}_j). \tag{13}$$

Correct first attempts produce a supervised first-attempt target. Wrong first attempts produce two supervised targets: the wrong first attempt and then the oracle correction. Operationally, the observational dataset supervises 51,121 of 81,131 token positions (63.0%).

Interventional stream agent. The interventional stream agent starts from the same SFT-1 checkpoint and uses the same 785 stream turns. It also conditions on the same first attempts. The only causal difference is the label mask: the first attempt is in the input, but its labels are set to the ignore value. A correct first attempt therefore gives no gradient. A wrong first attempt creates a repair example in which the model sees the first attempt and then learns the oracle correction:

$$\mathcal{L}^{\text{do}}_{\text{stream}}(\theta) = -\sum_{j:\, v_j=0} \log p_\theta(y^\star{}_j \mid x_j, \text{do}\,(\widehat{y}_j)\,, v_j). \tag{14}$$

The supervised positions are exactly the correction positions: 17,924 of 81,131 tokens (22.1%). This is why the comparison with observational training is a clean causal ablation. Both stream agents see the same questions, the same first attempts, and the same corrections. The observational model imitates the sampled attempts. The interventional model learns only from the world-written correction conditioned on those attempts.

Figure 2 makes this difference visible in token space. The interventional stream and SFT+oracle-corrections have the same number of loss-bearing correction tokens, but the interventional stream has many more masked context tokens because it preserves the first attempt in the transcript. The observational stream has still more loss-bearing tokens because it trains on both correct and wrong first attempts, and as we will seen in the results this is the wrong thing to do. More data in this case does not mean better performance. Figure 3 shows how the SFT-1 rollout policy does on the different questions making up this small dataset.

2.4 Solve-accuracy comparison

Figure 4 summarizes the held-out solve accuracy on 100 test problems. The base model solves 42% of the questions. The one-epoch SFT seed raises this to 63%. ReST reaches 84%, and GRPO reaches 75%. The observational stream agent falls to 61%, despite receiving more supervised tokens

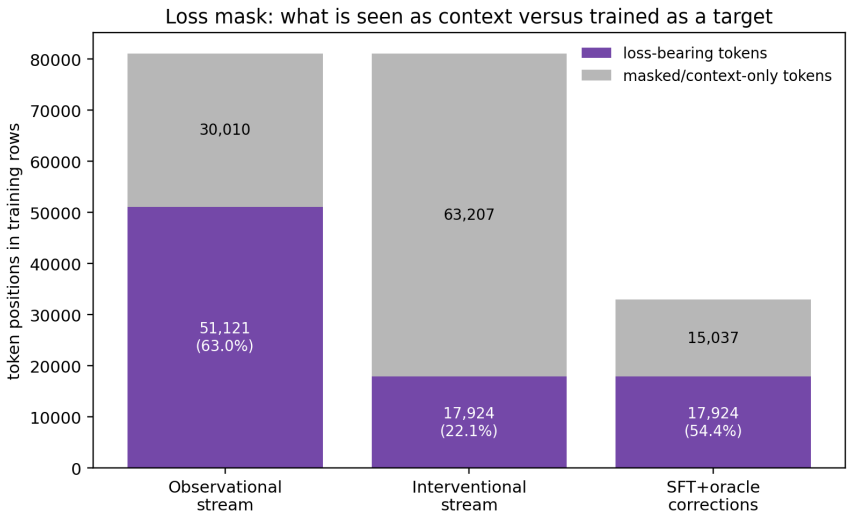

Figure 2: Loss-bearing versus context-only token positions in the stream-format datasets. The crucial comparison is interventional stream versus SFT+oracle-corrections: both supervise 17,924 correction tokens, but only the interventional stream keeps the first-attempt context while masking it from the loss.

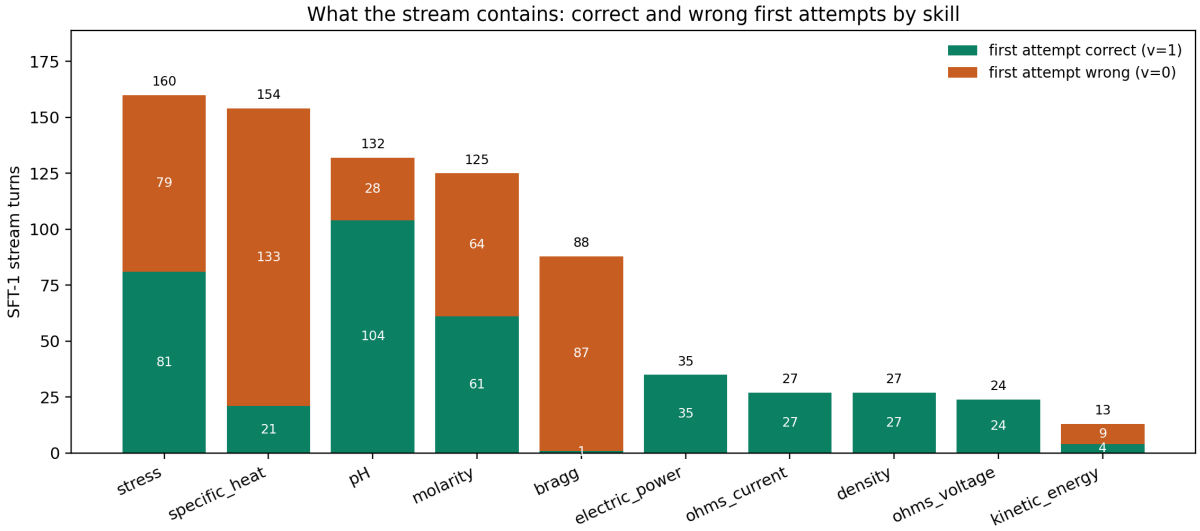

Figure 3: Composition of the SFT-1 rollout stream used by both stream agents. The notebook samples from $\mathcal{D}_{\text{agent}}$ until it has 400 wrong first attempts; in the inspected run this required 785 turns and also produced 385 correct first attempts. Correct first attempts are plentiful, but interventional training keeps them as context rather than targets.

than the interventional stream agent. The interventional stream agent reaches 85%, the best solve accuracy in the inspected run.

The strongest comparison is not just the top-line 85% versus 84%. It is that the interventional stream objective reaches this performance with ordinary supervised learning over a causal transcript. There is no rejection threshold to choose, no policy ratio, no clipped advantage, and no KL coefficient

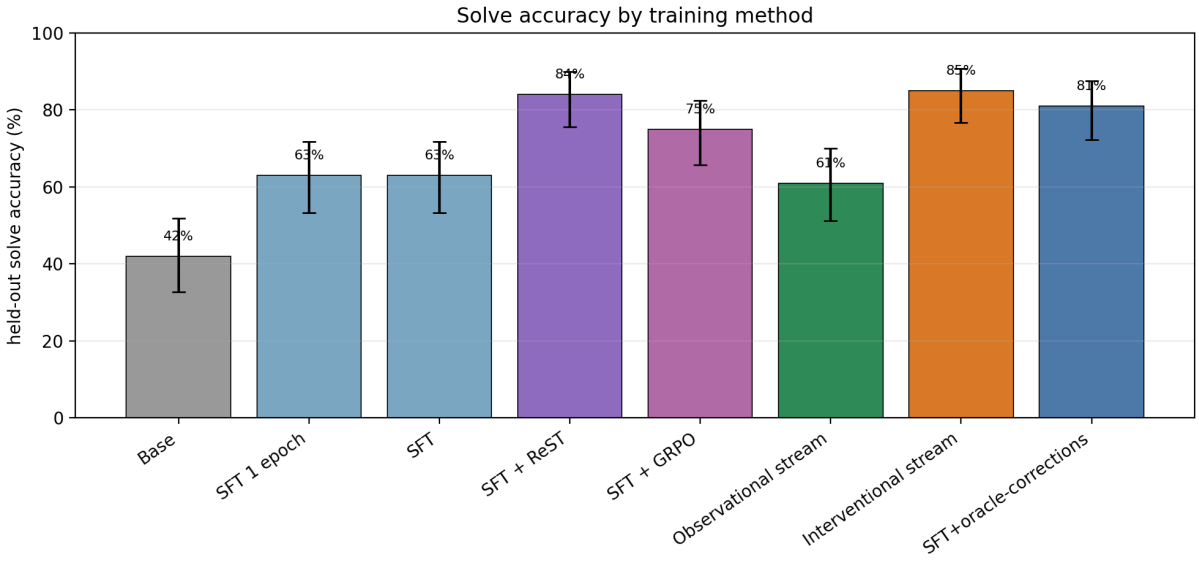

Figure 4: Solve accuracy by training method. Error bars are Wilson confidence intervals for $n = 100$ held-out questions.

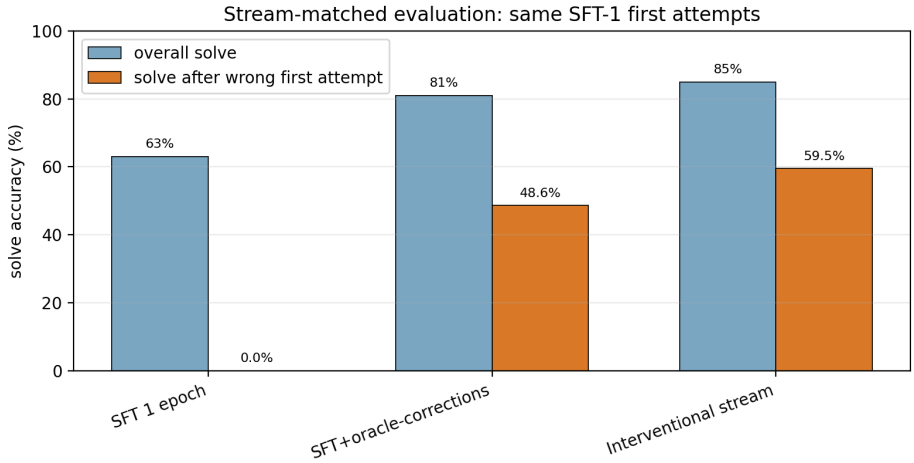

Figure 5: Stream-matched comparison. The first-attempt stream is generated by the frozen SFT-1 policy for all three rows. Interventional training improves the cases where the first attempt was wrong because it preserves the repair state instead of collapsing the correction back into an ordinary SFT example.

to tune. The method's price is different: It needs a correction or informative world response, not merely a scalar reward.

2.5 Why interventional beats SFT+oracle-corrections with the same correction targets

The comparison with SFT+oracle-corrections is the most educational one. Both methods start from SFT 1. Both receive the same 400 oracle correction targets, with the same 17,924 supervised

correction tokens. Yet the interventional stream reaches 85% solve accuracy, while SFT+oracle-corrections reaches 81%. On the 37 held-out examples where the frozen SFT-1 first attempt is wrong, interventional solves 59.46% of the repair cases, while SFT+oracle-corrections solves 48.65%.

The results of Figure 5 agree with the theory provided by our Prediction Lemma. SFT+oracle-corrections learns a direct map $x \mapsto y^\star$. The interventional stream learns a repair map $(x, \mathrm{do}\,(\widehat{y})\,, v) \mapsto y^\star$. The second map has the information needed to diagnose the first error.

This also clarifies why interventional does better than observational stream training in the same notebook. Observational training has the repair context, but it also trains on the wrong attempt as a target. Interventional training has the repair context without turning the wrong attempt into evidence. That is the causal advantage.

2.6 Analysis

The results teach four lessons.

1. **Stream agents are viable verifier-based post-training methods.** The interventional stream agent reaches 85% solve accuracy, compared with 84% for ReST and 75% for GRPO in this run. The intervals overlap with ReST, so the correct claim is not strict dominance. The correct claim is that a causal stream objective belongs in the same toolkit.
2. **The intervention mask matters when the stream contains bad actions.** Observational and interventional stream training use the same 785 rollouts and the same 400 corrections. Observational supervises 51,121 token positions and reaches 61%. Interventional supervises only 17,924 correction positions and reaches 85%. More supervised tokens are not better if many of those tokens are the learner's own mistakes.
3. **Correction context matters.** SFT+oracle-corrections receives the same oracle targets as interventional stream training, but it discards the first attempt. The stream model keeps the first attempt in context and can learn a conditional repair map.
4. **Verifier quality and world-side data design still matter.** The interventional loss removes self-action likelihood factors, but it cannot make poor world evidence good. ReST does very well on molarity in this run because it selects many verifier-passing sampled answers. The stream method needs the world to supply informative corrections, not merely noisy feedback.

The broader lesson is methodological. ReST, GRPO, and stream agents all start from the same practical ingredients: a policy, a verifier, and new tasks. ReST turns the verifier into a selection filter. GRPO turns it into a reward. The stream agent turns it into an observation in an interaction history. That last move is closest to the causal account: Actions stay in the history, but only world-written observations update the model.

3 Discussion and the need for further research

A few things, in order of generality.

1. The distinction between $\mathbb{P}(o \mid a)$ and $\mathbb{P}(o \mid \mathrm{do}\,(a))$ is not just relevant to formal Bayesian agents with discrete hypothesis classes. It survives the translation to LLM (and Omni) fine-tuning, where it becomes the distinction between supervising agent tokens and masking them.

2. Pipelines that train on conversational transcripts as if every token were evidence are committing self-confirmation errors, with measurable consequences for what the model says afterwards. *This is not hypothetical: standard SFT recipes for chat models, web-agent traces, and tool-use logs frequently train on the agent's own outputs without distinction.*

3. The fix is essentially free. Masking agent tokens from the loss requires no extra data, no extra compute, no architectural change, and no auxiliary objective. The one-line change recovers the intervention/evidence distinction at training time.

4. World-side data design matters because the intervention rule is necessary but not sufficient. We believe this is only the beginning of the road toward a principled approach to post-train agentic LLMs.

Some caveats are due. The experiment is still small. The STEM stream experiment is a single run with a tiny 0.5B base model, 200 SFT items, 200 agent items, and 100 held-out questions. The solve-accuracy intervals for ReST and the interventional stream overlap, so the result should be read as evidence of viability and competitiveness rather than a final benchmark claim. Results would likely vary with hyper-parameter search. This experiment and research note should be taken as a starting point for more investigation into intelligence emergence, as opposed to intelligence design in the current multi-stage approach.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] Yuntao Bai, Andy Jones, Kamal Ndousse, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[3] Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. In *NeurIPS*, 2015.

[4] Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

[5] A. Philip Dawid. Influence diagrams for causal modelling and inference. *International Statistical Review*, 70(2):161–189, 2002.

[6] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[7] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2022.

[8] Pedro A. Ortega. Universal artificial intelligence as imitation. https://www.adaptiveagents.org/_media/universal-ai-as-imitation.pdf, 2026. Under review; accessed 2026-05-05.

[9] Pedro A. Ortega, Markus Kunesch, Grégoire Delétang, et al. Shaking the foundations: Delusions in sequence models for interaction and control. *arXiv preprint arXiv:2110.10819*, 2021.

[10] Long Ouyang, Jeff Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[11] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

[12] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

[13] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2 edition, 2009.

[14] Ethan Perez, Sam Ringer, Kamile Lukosiute, et al. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*, 2022.

[15] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

[16] Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *NeurIPS*, 1989.

[17] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[18] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

[19] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[20] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[21] 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 *NeurIPS*, 2023.

[22] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[23] Mrinank Sharma, Meg Tong, Tomasz Korbak, et al. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*, 2023.

[24] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024.

[25] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, 2 edition, 2000.

[26] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

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

[28] Junzhe Zhang and Elias Bareinboim. Designing optimal dynamic treatment regimes: A causal reinforcement learning approach. In *ICML*, 2020.