FIRA: DEPLOYMENT-TIME RESCUE OF FROZEN RL AGENTS VIA A LOGICAL FIREWALL

Anonymous authors
Paper under double-blind review

ABSTRACT

Frozen reinforcement-learning policies can fail at deployment in narrow, recognizable ways, yet retraining them may be impractical. We study *deployment-time rescue*: correcting a frozen policy with a separately trained intervention layer, without modifying the policy or adapting during deployment. Training such a layer offline exposes an *offline-to-live rescue gap*: models with similar offline accuracy can behave very differently in closed loop, and running the rule-based supervision procedure as a live controller does not predict how well the learned models will perform. We introduce FIRA (*Firewalled Intervention via Relational Atoms*), which represents scenes as ego-centric graphs of affordance-tagged entities and predicts atoms describing threats, resources, and progress. A *logical firewall* blocks action-loss gradients through the predicted-atom pathway and permits the rescue action to override the frozen policy only when an intervention atom fires. FIRA is trained on fixed offline scenes labeled from environment rules, without reward optimization, rescue-policy rollouts, or deployment-time adaptation. Under identical supervision, matched flat-MLP and behavior-cloning baselines attain high offline accuracy but transfer less consistently to closed-loop control. Across three Atari games, two frozen-policy families, and two ViZDoom scenarios, FIRA rescues novel hazards, resource mismanagement, reactive-control failures, unseen threats, and health depletion. On Frostbite, FIRA raises the frozen policy's score from 126 to 2,527 and advances beyond the first level in 19 of 20 episodes, compared with 0 of 20 without rescue.

1 INTRODUCTION

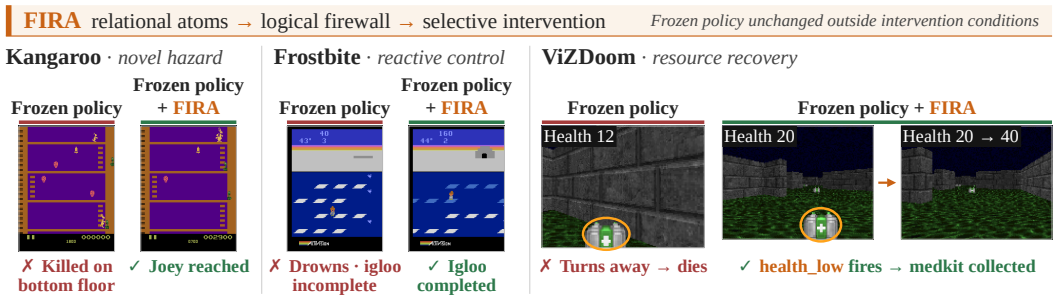


Figure 1: **FIRA rescues frozen policies across distinct deployment failures in 3 different games.** Representative rollouts contrast frozen policies with their FIRA-wrapped counterparts under a novel hazard in Kangaroo, reactive-control failure in Frostbite, and resource neglect in ViZDoom. FIRA enables the agent to reach the joey, complete the Frostbite igloo, and collect a needed medkit, while leaving the frozen policy in control outside intervention conditions. ViZDoom panels come from a fresh rollout of the reported configuration, with health values overlaid from its simulator-state log.

Reinforcement-learning policies trained from pixels can match human play on their training distribution yet fail in narrow, recognizable ways at deployment: an enemy with an unfamiliar sprite, a resource the policy never learned to budget, or a lethal move that must be corrected within a few

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.

frames (Song et al., 2024; Delfosse et al., 2025). Retraining for each such failure is often impractical, particularly when failures surface only after deployment. Fortunately, many share a common structure: a condition in the current scene signals an impending failure, and a targeted corrective action is needed only while that condition holds. We study how to learn a shared rescue mechanism that performs such interventions without modifying the frozen policy (Fig. 1), and what makes that mechanism transfer reliably from offline supervision to live closed-loop control.

The offline-to-live gap. The natural approach is to label rescue situations offline and fit a model to them. Doing so exposes the central difficulty: *fitting the labels well does not ensure successful rescue*. Rescue models from three architectures, each trained repeatedly on identical offline supervision, attain similar offline accuracy; once deployed in closed loop, however, some obtain near-zero reward while others achieve the highest observed reward (Fig. 4). This echoes a known result for imitation learning: a policy trained on expert states drifts to states the expert never visited, so offline accuracy understates closed-loop error (Ross et al., 2011; Codevilla et al., 2018). Two features make the rescue setting different. The learned model acts on only a minority of frames and hands control back to a frozen policy, so its errors are interleaved with, and partly corrected by, another policy's behaviour. And the gap persists even when the supervising labeler is run live: across tasks it scores far below or far above the models trained on its supervision, so its own score does not predict which learned model will transfer. Closed-loop reliability must therefore be designed and evaluated explicitly; nearby approaches either replace the frozen policy or are too slow for frame-level control (Sec. 2).

Our FIRA approach. A reliable rescue layer needs three properties: a scene representation that abstracts away irrelevant appearance, an explicit rule governing when the learned rescue action may take control, and supervision for what to do that requires no live learning. **FIRA** (*Firewalled Intervention via Relational Atoms*) provides each. It represents the scene as an ego-centric graph of entities with affordance tags and geometry rather than sprite appearance, allowing a novel-looking object to retain a familiar behavioral role. From this graph, it predicts symbolic *atoms* describing threats, resources, and progress. Designated *intervention atoms* encode recognizable failure conditions derived from the environment's termination and progress rules, and a deterministic gate permits the rescue action to replace the frozen policy only while one of these atoms fires. Training uses offline scenes labeled from environment rules, without reward or online adaptation. Failures outside the specified atom vocabulary remain under the frozen policy's control rather than triggering an unsupported intervention.

Overall, our contributions are: *(i)* we identify and measure an *offline-to-live rescue gap*, the analogue of covariate shift in imitation learning for a rescue layer that acts on a minority of frames, showing under matched supervision that offline predictive accuracy is insufficient to determine closed-loop rescue performance; *(ii)* we introduce a *logical firewall* that blocks action-loss gradients through the predicted atoms during training and authorizes deployment-time overrides only through intervention atoms; and *(iii)* we demonstrate reliable selective rescue across three Atari games, two frozen-policy families, and two ViZDoom scenarios spanning novel hazards, resource management, reactive control, unseen threats, and health depletion. On Frostbite, where success requires precisely timed transitions between moving ice floes, FIRA raises the frozen policy's score from 126 to 2,527 and advances beyond the first level in 19 of 20 episodes, compared with 0 of 20 without rescue.

2 Related Work

Neuro-symbolic reinforcement learning. Neuro-symbolic RL combines neural perception with structured reasoning through differentiable or probabilistic logic (Manhaeve et al., 2018; Cunnington et al., 2023; Yang et al., 2023), decision trees (Delfosse et al., 2024b; Fuhrer et al., 2025; Marton et al., 2025), logic policies (Jiang & Luo, 2019; Kimura et al., 2021; Delfosse et al., 2023; Sha et al., 2024), programs (Verma et al., 2018; Trivedi et al., 2021; Cao et al., 2022; Kohler et al., 2024; Wüst et al., 2024), symbolic equations (Luo et al., 2024), and fused neural-symbolic architectures (Emunds et al., 2025; Shindo et al., 2025). These approaches primarily learn or distill the task policy itself, often using entity identities supplied by an object extractor such as OCAtari (Delfosse et al., 2024a). In contrast, FIRA preserves an independently trained frozen policy and learns a separate rescue layer over affordance tags and geometry. Its symbolic atoms determine when that layer may intervene, rather than serving as a replacement policy.

Vision-language agents and policy adaptation. VLM-based agents (Liu et al., 2023; Hurst et al., 2024) support open-ended game play (Zhu et al., 2023; Wang et al., 2024) and interaction with 3D environments through generated code or actions (Wu et al., 2023; Tan et al., 2025); language models have also been used to guide policy pretraining (Du et al., 2023). These keep a large model in the decision loop; FIRA runs a 3.2 ms graph reasoner and uses a VLM only as an optional tagger (App. E). Visual distribution shift degrades pixel policies (Yuan et al., 2023); test-time adaptation (Iwasawa & Matsuo, 2021; Wang et al., 2021) and meta-learning (Finn et al., 2017) respond by updating or adapting the deployed policy. FIRA leaves both the frozen policy and the rescue model unchanged during deployment.

Safety shields and runtime intervention. Constrained MDPs and shielding methods (Alshiekh et al., 2018; Bethell et al., 2025; Debot et al., 2025) restrict actions using costs, safety specifications, learned monitors, or environment models. FIRA shares their separation between an acting policy and an intervention mechanism, but addresses a broader deployment-time rescue setting: its atoms cover not only immediate hazards, but also resource depletion and stalled task progress. Moreover, the logical firewall governs a learned corrective controller trained entirely offline, while leaving the original policy untouched. Switching control between a primary controller and a backup under a monitored condition is the Simplex architecture (Seto et al., 1998), and correcting a fixed policy with a separately learned component is residual policy learning (Silver et al., 2018; Johannink et al., 2019). FIRA differs from both: it replaces the frozen policy's action rather than adding a residual, it switches on learned predictions of rule preconditions rather than on a verified safety envelope, and it is trained entirely offline.

Offline supervision construction. Counterfactual augmentation uses model-based rollouts (Pitis et al., 2020), or influence-aware augmentation (Armengol Urpí et al., 2024). Affordance learning (Borja-Diaz et al., 2022) produces dense action-possibility maps, while privileged distillation (Chen et al., 2020; Cai et al., 2024) requires a teacher or expert with privileged access to the target environment. We instead vary intervention-relevant factors in structured scenes and derive atom and corrective-action labels from transition, termination, and progress rules.

3 FIRA: Firewalled Intervention via Relational Atoms

Let $\pi_\theta(a \mid o_t)$ be a frozen policy acting on its original observation o_t, and let x_t denote a structured description of the same scene. FIRA trains a separate rescue policy $\pi_\phi(a \mid x_t, g)$, conditioned on a goal g, and substitutes its proposed action only when a designated atom predicts that intervention is required. The parameters θ of the frozen policy are never modified. As shown in Fig. 2, FIRA has three parts: **(1) scene input**, an object-centric observation with an affordance tag for each entity; **(2) relational reasoning**, an ego-centric graph processed by an edge-aware star-topology reasoner with atom and action heads, where predicted atoms enter the action head through a stop-gradient; and **(3) rescue**, a deterministic gate that executes the rescue action only when an intervention atom fires.

3.1 Part 1 of FIRA: Scene Input

At each frame, the reasoner receives the entities in the scene with their positions, sizes, velocities, and one affordance tag from a small vocabulary (`dangerous`, `collectible`, `traversable`, or `goal`), together with ego-state variables such as oxygen, ammunition, and the level timer when supplied by the environment. Object-centric state of this form is available for Atari through OCAtari (Delfosse et al., 2024a) and is used by neuro-symbolic policies such as BlendRL (Shindo et al., 2025). FIRA retains only geometry, state variables, and canonical affordance tags; the frozen policy continues to use its original observation space and architecture.

A task adapter maps environment objects to structural roles and affordance tags without exposing their sprite identities to the reasoner. Consequently, a novel sprite can remain familiar to FIRA when assigned an existing affordance, even though its appearance is novel to the pixel policy. This input is privileged relative to the frozen pixel policy, and the comparison should be read as such. Obtaining the tags from pixels is a separate perception problem: App. E replaces the object interface with an open-vocabulary detector fine-tuned on Kangaroo frames and a VLM tagger, and rescue remains effective for sprites whose tags are stable across frames.

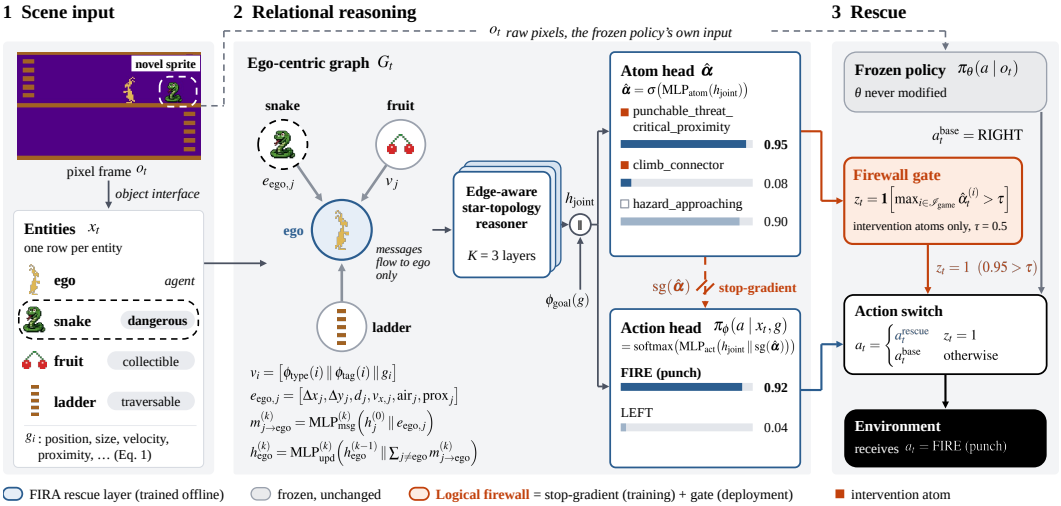


Figure 2: **The FIRA architecture.** (1) The object interface supplies entities with affordance tags; the frozen policy keeps its own input (here, pixels). (2) An edge-aware star-topology reasoner over an ego-centric graph feeds atom and action heads; predicted atoms enter the action head through a stop-gradient. (3) A deterministic gate lets the rescue action replace the frozen policy's action only when an intervention atom exceeds τ. The stop-gradient and gate form the logical firewall. A novel snake fires `punchable_threat_critical_proximity` ($0.95 > \tau$), so FIRE replaces RIGHT.

3.2 Part 2 of FIRA: Relational Reasoning

Ego-centric graph. Each frame becomes a graph $G_t = (\mathcal{V}_t, \mathcal{E}_t)$ containing the agent as an *ego* node and the remaining entities as its neighbors. Node i carries the feature vector

$$v_i = [\phi_{\text{type}}(i) \,\|\, \phi_{\text{tag}}(i) \,\|\, g_i]\,, \tag{1}$$

where ϕ_{type} embeds a canonical structural role (ego, platform, connector, hazard, collectible, or goal), ϕ_{tag} embeds the affordance tag, and g_i contains the entity's geometry: position, size, velocity, airborne status, ego-relative displacement, distance, and inverse proximity. Neither embedding identifies the original sprite.

Each ego–neighbor edge carries

$$e_{\text{ego},j} = \left[\Delta x_j, \Delta y_j, d_j, v_{x,j}, \text{air}_j, \text{prox}_j\right], \tag{2}$$

which describes how entity j relates to the ego.

Edge-aware star-topology reasoner. Neighbors send ego-relative, edge-conditioned messages to the ego, and only the ego representation is updated. This star topology keeps inference linear in the number of entities (App. F.2). Let $h^{(0)}_i$ be a projection of v_i. At each layer $k = 1, \ldots, K$, the reasoner computes

$$m^{(k)}_{j\to\text{ego}} = \text{MLP}^{(k)}_{\text{msg}}\left(h^{(0)}_j \,\|\, e_{\text{ego},j}\right), \qquad j \neq \text{ego}. \tag{3}$$

$$h^{(k)}_{\text{ego}} = \text{MLP}^{(k)}_{\text{upd}}\left(h^{(k-1)}_{\text{ego}} \,\|\, \sum\nolimits_{j\neq\text{ego}} m^{(k)}_{j\to\text{ego}}\right). \tag{4}$$

The neighbor representations remain fixed while the ego accumulates relational information across layers. After $K = 3$ layers, the ego representation is concatenated with a goal embedding:

$$h_{\text{joint}} = h^{(K)}_{\text{ego}} \,\|\, \phi_{\text{goal}}(g). \tag{5}$$

An atom head predicts the per-game predicates $\hat{\boldsymbol{\alpha}} \in [0,1]^{|\mathcal{A}|}$; App. D lists the intervention atoms:

$$\hat{\boldsymbol{\alpha}} = \sigma(\text{MLP}_{\text{atom}}\,(h_{\text{joint}}))\,. \tag{6}$$

The action head receives both h_{joint} and the predicted atoms, but gradients from the action loss are stopped at the atom predictions; sg denotes this stop-gradient:

$$\pi_\phi(a \mid x_t, g) = \text{softmax}(\text{MLP}_{\text{act}}\left(h_{\text{joint}} \,\|\, \text{sg}\left(\hat{\boldsymbol{\alpha}}\right)\right)). \tag{7}$$

The action head can therefore condition on predicted atom values, but its loss cannot modify those predictions through this connection. The shared encoder still receives gradients from the atom and action objectives through their respective heads.

The logical firewall. Two mechanisms keep the learned rescue action subordinate to the atoms. *During training*, the stop-gradient in Eq. 7 blocks action-loss gradients through the predicted atoms. The atom head is trained only by atom supervision, while the shared encoder receives both objectives. *At deployment*, the deterministic gate in Sec. 3.3 permits an override only when a designated intervention atom fires. The stop-gradient protects what the gate relies on. When the action loss can reach concept predictions, those predictions absorb action-relevant information they are not meant to encode, a failure known as concept leakage (Mahinpei et al., 2021; Margeloiu et al., 2021). For a rescue layer this matters more than for a classifier: the learned action executes only when an intervention atom fires, so the gate is only as trustworthy as the atoms' meaning, and the stop-gradient keeps the action loss from turning the atoms into latent features for the action; since the shared encoder still receives that loss, it limits leakage rather than excluding it. The training-time half resembles the sequential training regime of concept-bottleneck models (Koh et al., 2020; Delfosse et al., 2024b); the addition here is a deployment gate restricted to a designated subset of those concepts, so that the rescue layer acts only on frames where one of them holds. The logical firewall combines blocking action-loss gradients through predicted atoms with execution-time gating; Sec. 4.4 isolates the contribution of each mechanism.

Training. FIRA is trained on fixed offline instances $(x, g, \boldsymbol{\alpha}^\star, a^\star)$, where $\boldsymbol{\alpha}^\star$ and $a^\star$ are target atoms and corrective actions. We construct these instances from recorded or simulated gameplay scenes by varying intervention-relevant scene factors and labeling the resulting scenes with environment transition, termination, and progress rules; Sec. 4.1 describes this procedure and its task-specific specifications. Training uses class-weighted cross-entropy for actions and positive-weighted binary cross-entropy for atoms:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda \mathcal{L}_{\text{atom}}. \tag{8}$$

Optimization uses only this fixed dataset: no reward supervision, rescue-policy rollouts, or deployment-time adaptation are used.

3.3 Part 3 of FIRA: Rescue

At deployment, a deterministic gate decides whether the rescue action replaces the frozen policy's action. A small per-game subset $\mathcal{I}_{\text{game}} \subseteq \mathcal{A}$, called the *intervention atoms*, is authorized to trigger an override; all remaining atoms provide context to the action head. The gate fires when any intervention atom exceeds a fixed threshold τ:

$$z_t = \mathbf{1}\left[\max_{i \in \mathcal{I}_{\text{game}}} \hat{\alpha}_t^{(i)} > \tau\right]. \tag{9}$$

Let a_t^{base} and a_t^{rescue} denote the actions proposed by the frozen and rescue policies. The environment receives a_t^{rescue} when $z_t = 1$ and a_t^{base} otherwise.

The authorized set contains threat-proximity, resource (`oxygen_critical`, `ammo_low`), and commitment atoms (`climb_connector`); the latter authorize a navigation action for the whole multi-frame commitment. We use $\tau = 0.5$ throughout; a Kangaroo sweep produces unchanged conclusions over $\tau \in [0.3, 0.7]$ (App. F.1). Sec. 4.1 specifies how the intervention atoms and their labels are derived for each environment.

4 Empirical Evaluation

We evaluate whether FIRA transfers offline supervision into reliable deployment-time rescue, whether both parts of the logical firewall matter, and whether its interventions address the intended failures.

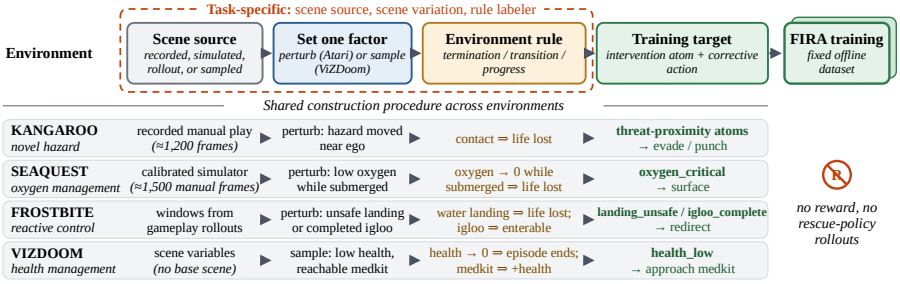


Figure 3: **Offline supervision construction.** Per task, scenes with a varied intervention-relevant factor are labeled by environment rules with target atoms and corrective actions. Frostbite's rollouts come from the frozen policy and the labeler, on data seeds disjoint from evaluation; no reward or rescue-policy rollouts are used.

4.1 Experimental Setup

Offline supervision, per-task cost, and calibration. For each environment, we vary failure-relevant factors in structured scenes and label them with its transition, termination, and progress rules (Fig. 3; App. C). The reasoner, objectives, gate, and hyperparameters are shared; per task we write a scene adapter, a scene source, and a labeler of a few prioritized rules that adds one or two atoms (Table 16). The scene sources are not part of the method, and no training scene comes from an evaluation-seed rollout. Each new intervention atom writes the precondition of the environment rule it guards with a margin from game constants, checked manually against the labeler live. To test whether this over-fits, half of the evaluation seeds were added only after every such choice was fixed; on those held-out seeds alone, every gated rescue of Table 2 holds, with Seaquest combat reward again level (App. P).

Protocol. The reasoner is deployed either *standalone*, controlling every frame, or *gated*, overriding the frozen policy only when an intervention atom fires (Sec. 3.3). Learned rescue models receive the structured input of Sec. 3.1; frozen pixel policies see only pixels. We report reward, intervention rate (IR), task-specific success, and offline action and atom accuracy. A training seed fixes the train/validation split (baselines: also initialization and data order); an evaluation seed fixes an environment realization. Unless stated otherwise, we train five checkpoints and evaluate each on twenty seeds (ViZDoom: twenty episodes), reporting mean $\pm$ std; the external baselines of Table 4 use their own five seeds. Environments are JAXAtari (Delfosse et al., 2026) and ViZDoom (Wydmuch et al., 2019). The FIRA reasoner adds only 3.2 ms per decision; measurement details and empirical runtime scaling with graph size are reported in App. F. Full protocol details are provided in App. B.

4.2 High Offline Accuracy Does Not Ensure Live Rescue, Nor Does the Labeler's Live Score

We train three rescue models on identical supervision and training seeds: FIRA; a *flat MLP reasoner*, an MLP over the nearest entities with the same atom head, stop-gradient, and gate; and *behavior cloning*, the same flat encoder trained on actions only. All fit their supervision to 90–100 %, and how well that predicts live reward varies by task (Fig. 4a; rank correlation 0.55). On ViZDoom health gathering the orders agree (FIRA $>$ flat MLP $>$ behavior cloning, offline and live). Kangaroo shows the clearest mismatch: two flat-MLP checkpoints 0.3 accuracy points apart score 32 K and 3 K, and every FIRA checkpoint transfers. Episodes cluster by checkpoint. Nor is the labeler's live score a guide (Fig. 4b): it lands on every side of the learned models, from 1.8 K against 45 K standalone on Kangaroo to 256 K against 101 K standalone on Seaquest resource management.

When does learning outperform the labeler? Not always. Where the precondition is one state variable and the rescue is one or two actions (oxygen, a landing gap, ammunition), the rules suffice live: the labeler lands within the spread of the learned models gated on Frostbite, Seaquest combat, and ViZDoom defend-the-center, and is the best controller we ran on Seaquest resource management. The difference appears where rescue spans many frames: the labeler's rules take a scene *bound* once

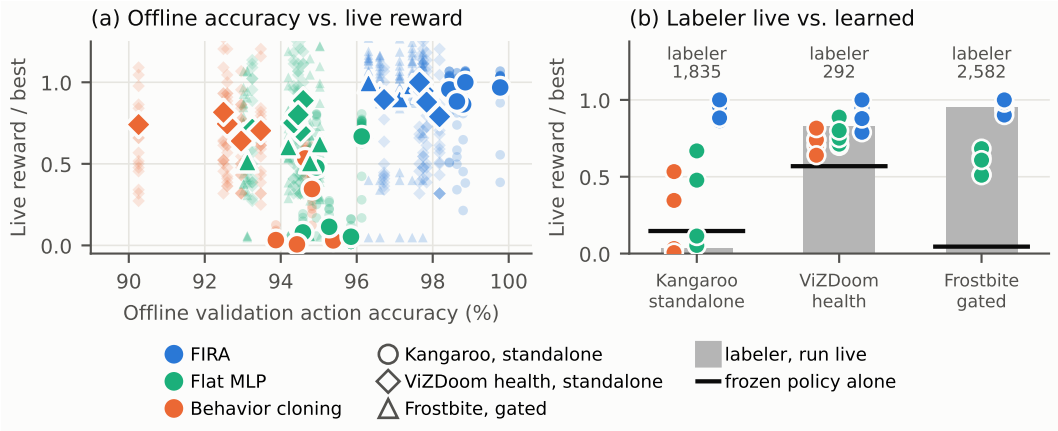


Figure 4: **High offline accuracy does not ensure live rescue, and the labeler's live score does not predict it.** *(a)* Every checkpoint of every learned rescue model in Kangaroo (standalone; circles), ViZDoom health gathering (standalone; diamonds), and Frostbite (gated; triangles); large markers are checkpoint means, faint points single seeds, reward normalized per setting. All sit at 90–100 % offline accuracy, yet the relation to live reward ranges from matching orders (ViZDoom) to a clear mismatch (Kangaroo). *(b)* The labeler run live (grey bar) against the learned checkpoints (dots) and the frozen policy alone (line). Gated on Frostbite, where the rescue is two rules, it lands within the spread of the learned models; standalone on ViZDoom health gathering, it sits between FIRA and the flat MLP; standalone on Kangaroo, it scores 1.8 K while reasoners trained on its labels score 45 K.

Table 1: **Which rescue models transfer to closed loop?** Cells list each model's checkpoints, each a mean reward over evaluation seeds (thousands; spread in App. Q). *Transfer*: checkpoints reaching at least half of the best reward of any checkpoint of any model in that game.

	Kangaroo (standalone)		**Seaquest** (standalone)		**Frostbite** (gated)	
Model	Reward per checkpoint ↑	Transfer ↑	Reward per checkpoint ↑	Transfer ↑	Reward per checkpoint ↑	Transfer ↑
Behavior cloning	1 / 2 / 17 / 0 / 26	1/5	0 / 9 / 3 / 1 / 1	0/5	not evaluated (no atom head)	
Flat MLP	32 / 4 / 3 / 23 / 6	1/5	10 / 85 / 0 / 8 / 20	1/5	1.69 / 1.40 / 1.85 / 1.38 / 1.65	5/5
FIRA	**47 / 46 / 42 / 49 / 43**	**5/5**	**99 / 89 / 79 / 48 / 101**	**4/5**	**2.53 / 2.55 / 2.44 / 2.66 / 2.71**	**5/5**

per clip (which ladder is the target), whereas a live controller must rebind every frame, and each mismatch is a fixed point (App. L); we read the reasoner's success as having learned that binding. The claim is not that learning beats rules, but that it transfers reliably in both kinds of setting.

Table 1 reports every checkpoint of the Atari comparisons (the 3D numbers are in App. J). Across the three games, FIRA transfers on 14 of 15 checkpoints, the flat MLP on 7 of 15, and behavior cloning on 1 of 10. The flat MLP's best checkpoint reaches 65–85 % of FIRA's best in each game, but in Kangaroo and Seaquest four of its five checkpoints fail to transfer; FIRA's one miss is a Seaquest checkpoint at 48 K, just below the transfer threshold. The best baseline checkpoint can approach FIRA; the distribution over checkpoints does not. Because the flat MLP shares the firewall, this isolates the relational encoder; the ablations (Sec. 4.4) isolate the firewall's parts. The ranking holds at a 30 % or 70 % threshold (FIRA 15 or 14 of 15, flat MLP 8 or 1, behavior cloning 2 or 0 of 10).

4.3 Gated Rescue Works Across Games, Policies, and 3D

Table 2 evaluates FIRA gated around frozen PPO+CNN and BlendRL policies across novel hazards, resource management, reactive control, unseen threats, and health depletion; it significantly improves the targeted failure in every Atari setting and in ViZDoom health gathering ($p < 0.01$; App. O). On Frostbite, gating raises reward from 126 to 2,527 and completes the first level in 19 of 20 episodes, although the reasoner alone scores only 30. In Seaquest, it eliminates all 47 oxygen deaths in combat and, under resource management, cuts BlendRL's oxygen deaths from 34 to 1 and raises its reward from 62,852 to 81,614 while overriding only 2.9 % of frames. On Kangaroo, PPO+CNN rises from 4,397 to 26,658 and BlendRL from 1,175 to 10,370; with tags produced from pixels by a detector

Table 2: **Gated rescue.** Frozen base policy alone and with FIRA gated; one pre-declared checkpoint per row (ViZDoom: std over checkpoints). *Success*: Kangaroo, levels completed; Seaquest, episodes without oxygen death (combat) or divers (resource); Frostbite, first level completed; ViZDoom, steps survived (reward: kills) or time-cap episodes (reward: steps). Kangaroo averages three novel sprites; Seaquest overrides also require oxygen below 0.2.

			Reward ↑		Success ↑		
Game	**Failure mode**	**Base policy**	Base	+ FIRA	Base	+ FIRA	**IR (%)**
Kangaroo	Novel hazard + stall	PPO+CNN	4,397 ± 714	**26,658 ± 2,612**	0.0 ± 0.0	**8.2 ± 1.1**	44.4 ± 0.5
	Novel hazard	BlendRL	1,175 ± 628	**10,370 ± 3,508**	0.3 ± 0.3	**5.3 ± 1.9**	18.2 ± 1.3
Seaquest	Combat	PPO+CNN	1,249 ± 220	**1,345 ± 475**	0.00 ± 0.00	**1.00 ± 0.00**	5.6 ± 1.6
	Resource mgmt.	BlendRL	62,852 ± 46,197	**81,614 ± 29,531**	96 ± 52	**119 ± 31**	2.9 ± 0.7
Frostbite	Reactive control	PPO+CNN	126 ± 14	**2,527 ± 849**	0.00 ± 0.00	**0.95 ± 0.22**	23.3 ± 5.1
ViZDoom (3D)	Unseen threats, ammo	PPO+CNN	9.81 ± 0.31	**10.61 ± 0.72**	184 ± 5	**216 ± 13**	66.9 ± 2.1
	Health depletion	PPO+CNN	200 ± 28	**268 ± 26**	0.03 ± 0.02	**0.16 ± 0.05**	23.7 ± 1.7

and a VLM instead of the object interface, the same pipeline still raises PPO+CNN from 3,833 to 9,500 on the snake and chicken variants (App. E). A hand-written time-to-contact shield over the same input stays at the frozen policy's level (4,393 against 4,397 on Kangaroo). Details: App. G.

3D. In ViZDoom *defend-the-center*, the reasoner sees threats outside the pixel policy's 90° field of view and gates on `unseen_threat_close` and `ammo_low`. Gating raises kills from 9.81 to 10.61, extends survival from 184 to 216 steps, and reduces rear hits by 62 %; App. J gives the atom-wise analysis. In *health gathering*, where an acid floor drains health, `health_low` plays the role of Seaquest's oxygen atom. Gating raises survival from 200 to 268 steps and timeout completion from 3 % to 16 %. Standalone, FIRA reaches 317 steps against 292 for the live labeler (App. J).

4.4 What the Logical Firewall Contributes

Gradient detachment. Table 3 pools the five training runs of each configuration. The full model reaches 45.4 K and transfers on all five runs; each ablation transfers on four of five. Removing edge features lowers reward to 28.4 K. Removing the atom head removes the atoms required for gated deployment and averages 34.2 K: four runs reach 35–49 K, while one collapses to 1.6 K despite 98.8 % offline action accuracy. The non-detached model averages 32.9 K, also with one collapsed run. Across these Kangaroo runs, only the full model transfers on every run (per-run values in App. Q), though the evidence is from one game and five runs per configuration.

Table 3: **Ablations.** Kangaroo, standalone; reward pooled over each configuration's five training runs.

	Live	Offline (%)	
Configuration	**Reward ↑**	**Action ↑**	**Atom ↑**
FIRA	**45.4 ± 9.1**	98.9	97.0
w/o edge features	28.4 ± 12.6	95.9	90.6
w/o atom head	34.2 ± 19.0	96.9	n.a.
Non-detached head	32.9 ± 19.0	93.9	97.2

The override gate. Gated, the reasoner controls 3–67 % of frames (App. N). On Kangaroo's snake variant, the gated checkpoint reaches 29,920 while overriding 44.5 % of frames and the standalone checkpoint 47,090 acting on every frame, against 1,250 for the base policy: the gate trades some reward for leaving the base policy in control of most frames. Frostbite is the complementary regime: the gated combination outperforms both controllers alone (2,527 against 126 and 30).

Standalone against policies trained live. Trained only offline, the reasoner also outperforms four policies trained in the environment on Kangaroo's default level (Table 4): 3.7× the reward of the strongest (BlendRL), triple its goal completions, and the only method with lives left at timeout.

Table 4: **Kangaroo, standalone, against policies trained live.** Default level. Baselines are trained in the environment, FIRA offline. Goal comp.: rescued joeys per episode. Latency: forward pass.

Model	Goal comp. ↑	Lives left ↑	Reward ↑	Latency (ms) ↓
NUDGE (Delfosse et al., 2023)	0.0 ± 0.0	0.0 ± 0.0	700 ± 82	20.0
NLRL (Jiang & Luo, 2019)	0.0 ± 0.0	0.0 ± 0.0	700 ± 163	30.0
PPO+CNN (Schulman et al., 2017)	3.0 ± 0.8	0.0 ± 0.0	7,133 ± 1,958	**2.1**
BlendRL (Shindo et al., 2025)	7.3 ± 0.4	0.0 ± 0.0	12,200 ± 424	40.0
FIRA	**22.3 ± 4.5**	**2.4 ± 1.2**	**45,436 ± 9,108**	3.2

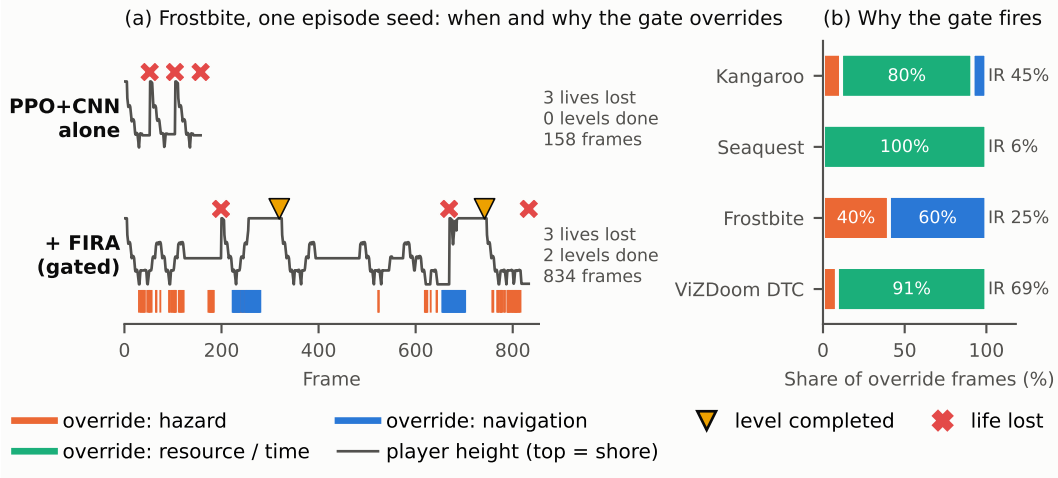


Figure 5: **What the interventions are.** *(a)* One Frostbite episode seed, PPO+CNN alone (top) and with FIRA gated on `landing_unsafe` and `igloo_complete` (bottom). Grey: player height; ticks: overrides coloured by atom family. Alone, the policy loses all three lives within 160 frames; gated, the landing atom fires only before jumps that would miss the floe (orange), the igloo atom steers home (blue), and the agent completes two levels. *(b)* Share of override frames by atom family over all gated evaluation episodes of each game, and the override rate pooled over frames (Table 2 averages per seed).

4.5 The Interventions Are Readable, and the Intervention Atoms Are Accurate

Every override names the atom that triggered it, so interventions can be read from the traces (Fig. 5a). Across games, the triggering families match the failure each game was chosen for (Fig. 5b): landing and igloo atoms in Frostbite, oxygen in Seaquest, ammo and unseen threats in ViZDoom, and in Kangaroo mostly `time_critical`, which breaks the frozen policy's loop of punching in place for reward. Replaying 299 Kangaroo interventions with the base policy's action instead changes the outcome in 293 (App. K). Trigger atoms can be far more accurate than contextual ones: in Seaquest, `oxygen_critical` reaches precision 0.96 and recall 0.99 against the labeler, while contextual atoms, which cannot trigger an override, fall as low as 0.04.

5 Conclusion

We introduced FIRA, a deployment-time rescue layer for frozen RL agents whose logical firewall keeps interpretable atoms free of action-loss gradients and permits an override only when an intervention atom fires. Across three Atari games, two frozen-policy families, and two ViZDoom scenarios, it transfers offline supervision into selective closed-loop intervention: under matched supervision it transfers on 14 of 15 Atari checkpoints, against 7 of 15 for the flat MLP and 1 of 10 for behavior cloning, and gated it significantly improves the targeted failure in every Atari setting and in ViZDoom health gathering. Neither offline accuracy nor the labeler's live score predicts this reliability: the offline-to-live rescue gap. Extending grounding and failure discovery beyond an object-centric scene and a fixed failure vocabulary is future work (limitations in App. A).

Ethics Statement

FIRA studies deployment-time recovery for frozen RL agents that meet conditions outside their training distribution; the intended benefit is that a fixed policy can be augmented with an inspectable rescue layer instead of being retrained. This work is a proof of concept in Atari-style environments and two 3D scenarios, and is not a safety solution for real systems: it assumes an object-centric scene with affordance tags and depends on labeler-generated supervision, so errors in tagging or atom design can cause missed or inappropriate interventions. Rescue layers of this kind can also create a false sense of safety if deployed without independent validation, monitoring, and domain-specific risk analysis.

AI Use Statement

We used LLM-based assistants for editing text and as coding assistants for routine engineering: experiment and evaluation scripts, the analysis scripts that aggregate logged runs into tables and figures, and code cleanup. The method, the offline supervision construction, the experimental design, and the interpretation of results are the authors' own. The authors reviewed all text and code and take full responsibility for the content of the paper.

Reproducibility Statement

The supplementary code contains the scripts that construct the offline datasets, train every model and ablation, run every evaluation protocol, and regenerate Tables 1–4, Figs. 4, 5, and 6, and the appendix tables of App. E, G, H, J, and L from the run directories the evaluation scripts write. Frozen-policy checkpoints, recorded manual-play frames, trained FIRA checkpoints, and logged runs are not included. Architecture, training data, and hyperparameters are given in App. B; offline supervision construction in App. C; intervention atoms in App. D; and per-checkpoint results and evaluation-seed variation in App. Q.

References

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Núria Armengol Urpí, Marco Bagatella, Marin Vlastelica, and Georg Martius. Causal action influence aware counterfactual data augmentation. In *International Conference on Machine Learning*, volume 235, pp. 1709–1729, 2024.

Daniel Bethell, Simos Gerasimou, Radu Calinescu, and Calum Imrie. Safe reinforcement learning in black-box environments via adaptive shielding. In *European Conference on Artificial Intelligence*, pp. 2450–2457, 2025.

Jessica Borja-Diaz, Oier Mees, Gabriel Kalweit, Lukas Hermann, Joschka Boedecker, and Wolfram Burgard. Affordance learning from play for sample-efficient policy learning. In *International Conference on Robotics and Automation*. IEEE, 2022.

Yang Cai, Xiangyu Liu, Argyris Oikonomou, and Kaiqing Zhang. Provable partially observable reinforcement learning with privileged information. In *Advances in Neural Information Processing Systems*, 2024.

Yushi Cao, Zhiming Li, Tianpei Yang, Hao Zhang, Yan Zheng, Yi Li, Jianye Hao, and Yang Liu. GALOIS: Boosting deep reinforcement learning via generalizable logic synthesis. In *Advances in Neural Information Processing Systems*, 2022.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pp. 66–75. PMLR, 2020.

Felipe Codevilla, Antonio M. López, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *European Conference on Computer Vision*, pp. 246–262, 2018. doi: 10.1007/978-3-030-01267-0_15.

Daniel Cunnington, Mark Law, Jorge Lobo, and Alessandra Russo. FFNSL: feed-forward neural-symbolic learner. *Machine Learning*, 112(2), 2023.

David Debot, Gabriele Venturato, Giuseppe Marra, and Luc De Raedt. Neurosymbolic reinforcement learning: Playing MiniHack with probabilistic logic shields. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 29631–29633, 2025.

Quentin Delfosse, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. Interpretable and explainable logical policies via neurally guided symbolic abstraction. In *Advances in Neural Information Processing Systems*, 2023.

Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. OCAtari: Object-centric Atari 2600 reinforcement learning environments. In *Reinforcement Learning Conference*, volume 1, pp. 400–449, 2024a.

Quentin Delfosse, Sebastian Sztwiertnia, Mark Rothermel, Wolfgang Stammer, and Kristian Kersting. Interpretable concept bottlenecks to align reinforcement learning agents. In *Advances in Neural Information Processing Systems*, 2024b.

Quentin Delfosse, Jannis Blüml, Fabian Tatai, Théo Vincent, Bjarne Gregori, Elisabeth Dillies, Jan Peters, Constantin Rothkopf, and Kristian Kersting. Deep reinforcement learning agents are not even close to human intelligence. *arXiv preprint arXiv:2505.21731*, 2025.

Quentin Delfosse, Raban Emunds, Paul Seitz, Sebastian Wette, Jannis Blüml, and Kristian Kersting. JAXAtari: A high-performance framework for reasoning agents. https://github.com/k4ntz/JAXAtari/, 2026.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, 2023.

Raban Emunds, Jannis Blüml, Quentin Delfosse, and Kristian Kersting. Interpretable reinforcement learning via meta-policy guidance. In *Eighteenth European Workshop on Reinforcement Learning*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Benjamin Fuhrer, Chen Tessler, and Gal Dalal. Gradient boosting reinforcement learning. In *International Conference on Machine Learning*, pp. 17960–17985, 2025.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, 2021.

Zhengyao Jiang and Shan Luo. Neural logic reinforcement learning. In *International Conference on Machine Learning*, 2019.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *International Conference on Robotics and Automation*, pp. 6023–6029, 2019.

Daiki Kimura, Masaki Ono, Subhajit Chaudhury, Ryosuke Kohita, Akifumi Wachi, Don Joven Agravante, Michiaki Tatsubori, Asim Munawar, and Alexander Gray. Neuro-symbolic reinforcement learning with first-order logic. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348, 2020.

Hector Kohler, Quentin Delfosse, Riad Akrour, Kristian Kersting, and Philippe Preux. Interpretable and editable programmatic tree policies for reinforcement learning. In *European Workshop on Reinforcement Learning*, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pp. 38–55. Springer, 2024.

Lirui Luo, Guoxi Zhang, Hongming Xu, Yaodong Yang, Cong Fang, and Qing Li. End-to-end neuro-symbolic reinforcement learning with textual explanations. In *International Conference on Machine Learning*, 2024.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*, 2021.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, 2018.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do concept bottleneck models learn as intended? *arXiv preprint arXiv:2105.04289*, 2021.

Sascha Marton, Tim Grams, Florian Vogt, Stefan Lüdtke, Christian Bartelt, and Heiner Stuckenschmidt. Mitigating information loss in tree-based reinforcement learning via direct optimization. In *International Conference on Learning Representations*, 2025.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In *Advances in Neural Information Processing Systems*, 2020.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Danbing Seto, Bruce Krogh, Lui Sha, and Alongkrit Chutinan. The Simplex architecture for safe online control system upgrades. In *American Control Conference*, volume 6, pp. 3504–3508, 1998.

Jingyuan Sha, Hikaru Shindo, Quentin Delfosse, Kristian Kersting, and Devendra Singh Dhami. EXPIL: Explanatory predicate invention for learning in games. *arXiv preprint arXiv:2406.06107*, 2024.

Hikaru Shindo, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. BlendRL: A framework for merging symbolic and neural policy learning. In *International Conference on Learning Representations*, 2025.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Wonil Song, Hyesong Choi, Kwanghoon Sohn, and Dongbo Min. A simple framework for generalization in visual RL under dynamic scene perturbations. In *Advances in Neural Information Processing Systems*, 2024.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control. In *International Conference on Machine Learning*, 2025.

Dweep Trivedi, Jesse Zhang, Shao-Hua Sun, and Joseph J Lim. Learning to synthesize programs as interpretable and generalizable policies. In *Advances in Neural Information Processing Systems*, 2021.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2018.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Russ Salakhutdinov, Amos Azaria, Tom M. Mitchell, and Yuanzhi Li. SPRING: studying papers and reasoning to play games. In *Advances in Neural Information Processing Systems*, 2023.

Antonia Wüst, Wolfgang Stammer, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. Pix2Code: Learning to compose neural visual concepts as programs. In *Conference on Uncertainty in Artificial Intelligence*, 2024.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. ViZDoom competitions: Playing Doom from pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019.

Wen-Chi Yang, Giuseppe Marra, Gavin Rens, and Luc De Raedt. Safe reinforcement learning via probabilistic logic shields. In *International Joint Conference on Artificial Intelligence*, 2023.

Zhecheng Yuan, Sizhe Yang, Pu Hua, Can Chang, Kaizhe Hu, and Huazhe Xu. RL-ViGen: A reinforcement learning benchmark for visual generalization. In *Advances in Neural Information Processing Systems*, 2023.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the Minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.

A LIMITATIONS

Per-task labeler and fixed failure vocabulary. Each task needs its own labeler of prioritized rules, and each new task has added a few atoms derived from the environment's rules (App. M). Writing them takes knowledge of the task. The atoms are fixed before training, so FIRA rescues only the failures they name; finding failures that no atom names is not addressed here.

When the rules alone suffice. Where the precondition is one state variable and the rescue is one or two actions, the labeler run live is about as good as the learned models (Sec. 4.2); the learned reasoner matters most where a rescue spans many frames.

Scene input and evaluation scope. The reasoner reads entities with affordance tags from the environment's object interface, which is privileged relative to the frozen pixel policy. The end-to-end version that produces the tags from pixels (App. E) is tested only on Kangaroo, on three novel sprites, and over three evaluation seeds; its rescue depends on each entity's tag staying stable across frames. All experiments are in games (three Atari games and two ViZDoom scenarios, with two frozen-policy families); we have not tested FIRA on robots or other physical systems.

Calibration on evaluation seeds. Some atoms, margins, and runtime conditions were set with live runs on at most evaluation seeds 0–9 (Frostbite on separate development seeds). The held-out seeds of App. P were added only after these choices were fixed, so they rule out over-fitting to particular seeds, but not to the environments themselves.

B FIRA IMPLEMENTATION DETAILS

The code for all experiments is provided in the supplementary material; its `README.md` describes how to construct the offline datasets, train every model, run each evaluation protocol, and regenerate the tables and figures listed in the Reproducibility Statement.

B.1 REASONER ARCHITECTURE

The reasoner consumes a structured ego-centric graph and emits both atom predictions and an action distribution. All MLPs are 2-layer with ReLU activations and hidden dimension 128; no dropout is used. Embeddings are $\phi_{\text{type}} \in \mathbb{R}^{16}$ for node type, $\phi_{\text{tag}} \in \mathbb{R}^{16}$ for affordance tag, and $\phi_{\text{goal}} \in \mathbb{R}^{32}$ for the goal token. Geometry and edge features are normalized, and their size depends on the task adapter: for Kangaroo, $g_i \in \mathbb{R}^{10}$ and $e_{ij} \in \mathbb{R}^{6}$, so $v_i \in \mathbb{R}^{42}$; the other tasks use 8–14 geometry and 6–7 edge dimensions (an ego-centric polar layout in ViZDoom), and an ego-state variable of Sec. 3.1, such as the level timer or igloo progress, adds one dimension. The flat-MLP and behavior-cloning baselines receive identical inputs, so these features do not favor FIRA. The node feature v_i is projected to $\mathbb{R}^{128}$ before message passing; after $K = 3$ star-topology message-passing layers, the ego representation is concatenated with the goal embedding to form $h_{\text{joint}} \in \mathbb{R}^{160}$. The atom head outputs sigmoid probabilities over $|\mathcal{A}|$ atoms (per-game atom sets in App. D); the action head's input is h_{joint} concatenated with the detached atom probabilities, and outputs a softmax distribution over the per-game action space. Total parameter count is approximately 0.4M per game.

B.2 TRAINING DATA AND HYPERPARAMETERS

FIRA is trained on the fixed offline datasets described in Sec. 4.1. Per-game scene counts and train/validation splits are summarized in Table 5; clips are split $80/20$ with the training seed, and the validation split is used only for checkpoint selection and learning-rate scheduling. Action weights are computed from training-label frequencies and atom positive weights from per-atom firing rates. The same hyperparameters are used across all five tasks (Table 6).

B.3 BASELINES

PPO+CNN (Schulman et al., 2017) is the CleanRL (Huang et al., 2022) implementation with default hyperparameters and minor adaptations for JAXAtari (total timesteps 1×10^7, learning rate 2.5×10^{-4},

Table 5: Offline training data for the Atari games (Kangaroo and Frostbite: all checkpoints, with 24,000 timer frames added for the gated Kangaroo checkpoint; Seaquest: the gated checkpoint, while the standalone checkpoints use 39,872 frames with corrected labels): labeled frames, the 80/20 split by clip, and action classes (Kangaroo: NOOP never labeled). Each scenario carries an action and atom vector for each goal token; atom vocabularies are in App. D.

Game	Scene source	Labeled frames	Train	Val	Action classes
Kangaroo	perturbed manual play	226,630	181,455	45,175	6
Seaquest	simulator (calibrated on manual play)	37,248	29,792	7,456	5
Frostbite	rollout windows and synthetic scenes	219,960	174,964	44,996	5

Table 6: Training hyperparameters for FIRA.

Hyperparameter	Value
Optimizer	Adam
Initial learning rate	1×10^{-3}
LR scheduler	ReduceLROnPlateau (factor 0.5, patience 7 epochs)
Batch size	512
Epochs	up to 150; the checkpoint with the best validation score is kept
Early stopping	patience 10 epochs (min. improvement 0.005)
Validation score	$0.5 \times$ action accuracy $+ 0.5 \times$ atom Jaccard
Atom-loss weight λ	5.0
Action loss	Class-weighted cross-entropy
Atom loss	Per-atom positive-weighted BCE
Goal tokens	every scene labeled and trained under each goal in $\mathcal{G}$

$\gamma = 0.99$, GAE $\lambda = 0.95$). **BlendRL** (Shindo et al., 2025) uses the official implementation with default hyperparameters. **NUDGE** (Delfosse et al., 2023) and **NLRL** (Jiang & Luo, 2019) use the re-implementations in the BlendRL repository with the original logical policies; NLRL was not originally tested on Atari, so we wrote logical policies for it; these baseline policies are not part of the supplementary code and will be released with the final code. All training and evaluation ran on a single NVIDIA A100 GPU (80 GB).

C Offline Supervision Details

Each training example is $(s, g, a_g, \boldsymbol{\alpha}_g)$: a structured scene, goal token, corrective action, and atom vector. Atari object-centric scenes come from recorded manual play (Kangaroo), a simulator calibrated on manual play (Seaquest), and, for Frostbite, windows of rollouts by the frozen PPO+CNN policy and by the labeler run live (data seeds disjoint from evaluation) plus synthetic scenes. The labeler-steered rollouts only supply scenes that the frozen policy rarely reaches (e.g., after the igloo is complete); every scene is labeled by the same rules, and the learned rescue model is never run during data collection. We vary intervention-relevant entity positions, motion, and state variables within the environment's valid ranges. ViZDoom scenes are sampled directly from the corresponding structured variables. Invalid configurations, such as overlapping entities or placements outside the game geometry, are re-sampled or discarded before labeling.

A deterministic rule binding maps each valid scene and goal to its atom vector and corrective action. The rules are derived from environment transition, termination, and progress conditions; examples include oxygen depletion, landing in water, and health reaching zero. Goal-conditioned labels are retained when the same scene admits different corrective actions. This process uses no scalar reward, learned teacher, or rollout of the learned rescue policy. Dataset sizes are reported in Table 5, intervention atoms in Table 7, and the provenance of newly introduced atoms in Table 16. Complete transformations and rule bindings are included in the released code.

D INTERVENTION ATOM SETS

Table 7 lists the atoms authorized to trigger an override. Other predicted atoms provide relational and goal context to the action head but cannot activate the gate. Complete per-game vocabularies are defined in the per-task labelers, scenario catalogs, and configurations of the supplementary code (`fira/games/`).

Table 7: Intervention atoms used by the deployment gate, and examples of context atoms that the same checkpoint predicts but that cannot open the gate.

Task	Intervention atoms	Context atoms (examples)
Kangaroo	hazard proximity and evasion, connector commitment, time critical (full list below)	hazard approaching, collectible present, multiple hazards present
Seaquest	oxygen critical	threat above/below/left/right, firable threat close
Frostbite	landing unsafe, igloo complete	obstacle close (left/right), obstacle in target row
ViZDoom: defend-the-center	unseen threat close, ammo low	threat approaching, threat critical proximity, no threat visible
ViZDoom: health gathering	health low	static threat proximity (poison ahead), aligned to target, no collectible visible

E END-TO-END VERSION WITH AUTOMATIC TAGGING

The main text supplies the reasoner's input, entities with positions and affordance tags, from the environment's object interface. In the end-to-end version that input is produced from pixels: an open-vocabulary detector (GroundingDINO (Liu et al., 2024), Swin-T backbone, fine-tuned on Kangaroo frames) proposes boxes, a constant-velocity Kalman tracker carries them between detections at the control rate, and each new track's crop is sent once to a VLM (GPT-4o (Hurst et al., 2024)) that returns one affordance tag from the reasoner's vocabulary; tags are cached per track and re-queried only on track loss. The detector and VLM run off the control loop; the reasoner and gate are unchanged. Table 8 reports detection and tagging quality on Kangaroo frames with novel sprites, none of which appear in the offline training data.

Table 8: Detection and tagging quality on Kangaroo frames with novel sprites (three evaluation seeds). Detection F1 and recall against emulator boxes (IoU ≥ 0.5); tagging accuracy per detected crop.

Sprite	Detection F1	Detection recall	Tagging acc. (%)
Snake	84.6	83.9	94.8
Chicken	81.8	81.3	93.9
Chain	72.8	76.3	77.3
Rope	76.7	82.9	93.9
Mean	**79.0**	**81.1**	**90.0**

Gated rescue from pixels. Table 9 runs the gated Kangaroo rescue with this pipeline in place of the object interface (PPO+CNN, standalone checkpoint gated, snake and chicken variants, 3 evaluation seeds, limited by the cost of live VLM calls).

Table 9: Gated rescue with tags produced from pixels, Kangaroo novel hazards (snake and chicken), PPO+CNN; mean $\pm$ std over 3 evaluation seeds.

	Reward ↑		Levels completed ↑	
Scene input	Base	+ FIRA	Base	+ FIRA
Tags from pixels	$3{,}833 \pm 1{,}222$	$\mathbf{9{,}500 \pm 1{,}572}$	0.0 ± 0.0	0.0 ± 0.0

What the rescue depends on. With the automatic entity list, the snake variant reproduces the object-interface result of the same checkpoint (10,800 with the standalone checkpoint gated; the 29,920 of Table 11 is the timer checkpoint and is not the comparison) and the chicken variant most of it ($8{,}200 \pm 3{,}143$). What matters is not per-crop tagging accuracy but whether the tag of one physical entity is stable across frames, since an atom must hold for the gate to act. On the rope, whose sprite fragments into several boxes, temporal consensus over each track's tags raises the gated reward from

267 ± 115 to $5{,}333 \pm 1{,}604$ with the detector and tagger unchanged. The tagging prompt is in the released code.

F Robustness and Scalability Diagnostics

F.1 Override-Gate Threshold

The gate fires when an intervention atom's probability exceeds $\tau = 0.5$. Sweeping τ from 0.3 to 0.7 without retraining (Kangaroo, gated PPO+CNN, standalone checkpoint, 3 evaluation seeds) leaves the override rate at $19.9 \pm 0.6\%$ and the reward at $10{,}733 \pm 115$ at every value: the atom head's outputs are close to binary (97% of probabilities outside $[0.1, 0.9]$), so τ is not a sensitive hyperparameter.

F.2 Inference Latency Scaling

The reasoner's star topology is designed for $\mathcal{O}(N)$ message passing in the number of on-screen entities N: only the ego node aggregates messages from neighbors, avoiding the $\mathcal{O}(N^2)$ cost of all-pairs attention. We measure the isolated reasoner forward pass as N varies from 4 to 128, in a microbenchmark of $1{,}000$ repetitions on an A100; the 3.2 ms of Table 4 is the per-frame reasoner call timed inside the evaluation loop.

Table 10: Reasoner inference latency as a function of entity count N. Mean and 99th-percentile (*p99*) latency in milliseconds, averaged over 1,000 forward passes per setting on a single GPU.

N **entities**	**Mean (ms)**	**Std**	**p99 (ms)**
4	1.61	0.09	1.79
12	1.60	0.14	1.78
16	1.60	0.10	1.76
32	1.61	0.02	1.78
64	1.63	0.14	1.81
128	1.62	0.02	1.70

Mean and p99 latency are flat over the range; the cost is fixed GPU overhead rather than message passing, as the $\mathcal{O}(N)$ star topology intends.

G Cross-Game Rescue: Absolute Performance Breakdown

This appendix expands Table 2 with per-sprite rows and the affordance-gap family (Kangaroo: gated checkpoint, 20 evaluation seeds; per-seed values are written by `analysis/perseed_kangaroo.py` in the supplementary code). The rope is not a failure of PPO+CNN, whose bare policy already climbs it. With BlendRL as the base policy the affordance gap is not rescued (10 ± 45 alone and 180 ± 223 with FIRA, averaged over the two connectors): its logic policy has no rule for a connector it does not recognize, jumps in place at the start position, and loses each life within about 230 frames to arriving hazards, before any connector or timer atom can fire.

Hand-written shield. We additionally compare against RuleShield-TTC, a deterministic shield that uses hand-written distance and time-to-contact conditions over the same structured scene input. Across the three Kangaroo hazard variants, it obtains 4 393 mean reward, compared with 4 397 for the frozen policy and 26 658 with FIRA (per-variant results in Table 11). Thus, access to structured state and a conventional hazard-proximity shield alone does not explain the rescue gains. The two rule baselines answer different questions: RuleShield-TTC is a generic hazard shield written without the task's labeler, whereas the labeler of Sec. 4.2 is the rule set that produced FIRA's supervision, run live.

Table 11: **Gated rescue, absolute values.** Frozen base policy, base + RuleShield-TTC (hand-written time-to-contact and distance rules, gated, override rate 5–7%), and base + FIRA (gated), with FIRA's override rate; PPO+CNN unless marked; 20 evaluation seeds per cell.

Game	Failure mode	Base reward	Base + RuleShield-TTC	Base + FIRA	IR (%)
Kangaroo	Novel ground hazard (snake)	$1{,}250 \pm 1{,}076$	$3{,}140 \pm 1{,}738$	$\mathbf{29{,}920 \pm 448}$	44.5 ± 0.3
	Novel ground hazard (chicken)	$4{,}990 \pm 1{,}282$	$4{,}950 \pm 1{,}044$	$\mathbf{28{,}840 \pm 3{,}004}$	44.7 ± 0.3
	Novel airborne hazard (wasp)	$6{,}950 \pm 1{,}589$	$5{,}090 \pm 1{,}089$	$\mathbf{21{,}215 \pm 7{,}476}$	43.9 ± 1.3
	Affordance gap (chain)	$8{,}260 \pm 1{,}620$	$6{,}080 \pm 1{,}734$	$\mathbf{27{,}340 \pm 5{,}218}$	44.5 ± 1.1
	Affordance gap (rope)	$10{,}530 \pm 412$	—	$\mathbf{24{,}255 \pm 8{,}325}$	44.5 ± 2.1
Seaquest	Combat (PPO+CNN)	$1{,}249 \pm 220$	—	$\mathbf{1{,}345 \pm 475}$	5.6 ± 1.6
	Resource management (BlendRL)	$62{,}852 \pm 46{,}197$	—	$\mathbf{81{,}614 \pm 29{,}531}$	2.9 ± 0.7
Frostbite	Reactive control (PPO+CNN)	126 ± 14	—	$\mathbf{2{,}527 \pm 849}$	23.3 ± 5.1

H Seaquest Resource-Management Setting

In both Seaquest settings the gate fires on `oxygen_critical` only while the true oxygen fraction is below 0.2, the range in which the rule can end a life within the atom's margin; this second condition is fixed and shared by the labeler-as-rescue runs of Fig. 4b. Two further pass-through conditions are fixed in the Seaquest runs: the gate stays closed while the ego is at the surface (`at_surface`), where oxygen refills without any action, and during the engine's respawn window, whose sentinel state values lie outside the training range. The Seaquest resource-management rows of Table 2 disable the hostile enemies (sharks, submarines). Default Seaquest reward is dominated by kills, which both base policies were trained to chase and which mask the oxygen-versus-diver decision the rescue layer targets; with enemies removed, reward comes from the diver-collection and surfacing loop, whose pickup, oxygen-decay, and surfacing-bonus mechanics are unchanged. The six-diver surfacing bonus makes rewards large (10^4–10^5), so divers collected per episode is the more direct success measure and both are reported. PPO+CNN scores only from kills and is therefore not a rescue target in this setting (Table 12); BlendRL keeps its diver-seeking behaviour and mismanages oxygen, which is the failure FIRA is evaluated against.

Table 12: Seaquest with enemies disabled, 20 evaluation seeds (PPO+CNN alone: 6). Divers/ep: divers picked up per episode; IR: fraction of frames overridden.

Configuration	Reward ↑	Divers / ep ↑	IR (%)
PPO+CNN alone	0 ± 0	1.5 ± 1.2	0.0
BlendRL alone	$62{,}852 \pm 46{,}197$	96 ± 52	0.0
BlendRL + FIRA, gated	$\mathbf{81{,}614 \pm 29{,}531}$	$\mathbf{119 \pm 31}$	2.9 ± 0.7
FIRA standalone	$98{,}967 \pm 10{,}656$	—	100.0

The labeler of Sec. 4.2, run standalone in this setting with the same oxygen condition, scores $256{,}120 \pm 8{,}832$ over 20 evaluation seeds, above every learned checkpoint of Table 1: with enemies disabled the rescue is one rule, and the rules suffice.

I Seaquest Mechanism Analysis

Death causes. Every lost life in the combat setting (PPO+CNN, default enemies, 20 evaluation seeds, 60 lives per arm) is attributed from the emulator state on the frame of loss: *oxygen* (counter reached zero while submerged), *empty-tank surfacing* (the submarine surfaced with no divers, which the game penalizes with a life), or *collision*. The bare policy loses 47 of 60 lives to oxygen; with `oxygen_critical` as the intervention atom, oxygen deaths fall to 0 and the remaining losses are collisions during combat, which no oxygen rule addresses. Reward in this setting moves with the death mode ($1{,}249 \rightarrow 1{,}345$), so the death-cause breakdown, not reward, is the informative metric here.

J VIZDOOM: 3D EXISTENCE PROOF

Scenario. *defend-the-center*: the agent stands at the center of a circular arena and enemies approach from any direction; the first-person view covers $\pm 45°$, so a pixel policy cannot see threats behind it. This is the failure mode a rescue layer with a scene graph can address and a pixel policy structurally cannot.

What is reused and what is new. The reasoner, training loop, dataset layer, override gate, action alphabet, and affordance taxonomy are reused unchanged. New per-game code is an adapter, a labeler with 13 atoms instantiated from the Atari families and one new atom, `unseen_threat_close` (an enemy outside the field of view within engagement range), and the offline scene sampler. As for the Atari games, where constants are read from the emulator, the labeler's game constants (arena radius, maximum closing speed) are measured once from the environment with a scripted probe; no policy, reward, or learned model is involved.

Offline and live. Five training seeds reach $97.2 \pm 0.3\%$ action accuracy and $98.5 \pm 0.2\%$ atom Jaccard offline. Live, each arm is 5 seeds $\times$ 20 episodes around the same frozen PPO policy. *Rear-hit rate* is the number of health-loss events per 100 decision steps whose nearest threat on the previous frame was outside the field of view. With the rear-threat atom alone, damage from unseen threats halves ($2.06 \rightarrow 1.07$ per 100 steps) while kills and survival are unchanged, because every episode still ends by death once the pistol is empty. Table 13 adds the resource atom. *Ammo out* is the fraction of episodes in which the agent empties its 26 rounds before dying.

Table 13: ViZDoom, gated PPO+CNN, 5 training seeds $\times$ 20 episodes, mean $\pm$ std over seeds; same checkpoints, threshold, and base policy, only the intervention atoms differ. Rear-hit: health-loss events per 100 steps whose nearest threat was outside the field of view; ammo out: fraction of episodes in which the pistol is emptied before death.

Intervention atoms	Kills ↑	Survival (steps) ↑	Rear-hit ↓	Ammo out ↓	IR (%)
none (PPO+CNN alone)	9.81 ± 0.31	184 ± 5	2.06	0.99	0
`unseen_threat_close`	9.91 ± 0.53	188 ± 10	1.07	0.89	27
`ammo_low`	$\mathbf{10.91 \pm 0.38}$	$\mathbf{219 \pm 8}$	**0.78**	0.86	64
`unseen_threat_close`, `ammo_low`	10.61 ± 0.72	216 ± 13	**0.78**	0.88	67
labeler as rescue layer, both atoms	10.65 ± 0.37	218 ± 6	0.82	0.83	67

The ammo atom accounts for the gain in kills and survival, and either atom reduces unseen damage; the ammo atom alone scores slightly higher kills than the two combined, within their spread. All gated arms use the efficiency goal, under which the reasoner fires only within conserving range.

Health gathering. The second scenario, *health gathering supreme*, is a maze whose floor drains 8 health every 8 decision steps; medkits restore about 20 and poison vials remove 30; the episode ends at zero health or after 525 steps. The failure is resource depletion in 3D. The new atom, `health_low`, is the precondition of the health-reaches-zero rule with a margin: the health that remains lasts fewer steps than reaching the nearest visible medkit takes. The labeler has five rules and nine atoms, eight from existing families (`static_threat_critical_proximity` for a poison vial ahead, alignment and collection atoms). Offline construction samples 30,000 scenes labeled under two goals; five training seeds reach $97.6 \pm 0.5\%$ action accuracy and $97.9 \pm 0.4\%$ atom Jaccard. Table 14 reports every arm, including the flat MLP and behavior cloning trained on the same supervision (offline action accuracy 94.2% and 92.4% against 97.6% for FIRA): standalone, FIRA outlives both; gated, the frozen policy acts on about 76% of frames and the flat MLP comes close to FIRA.

K COUNTERFACTUAL INTERVENTION REPLAY

For 299 unique interventions on held-out Kangaroo runs, the episode is replayed from the intervention frame with the base policy's action instead of the override. In 293 cases the replayed episode loses a life or fails to reach the state the rescued episode reaches within the same horizon; in 6 it does not (the intervention was unnecessary but harmless). Over the gated Kangaroo

Table 14: ViZDoom health gathering, 5 training seeds $\times$ 20 episodes, mean $\pm$ std over seeds. Survival: decision steps (cap 525); timeout: episodes reaching the cap; medkits: collected per episode.

Arm	Survival (steps) ↑	Timeout (%) ↑	Medkits ↑	IR (%)
PPO+CNN alone	200 ± 28	3.0 ± 2.4	6.7 ± 1.8	0
gated, `health_low`	$\mathbf{268 \pm 26}$	$\mathbf{16.0 \pm 4.9}$	10.6 ± 1.7	24
gated, `health_low` + `static_threat_critical_proximity`	264 ± 23	10.0 ± 7.1	9.5 ± 1.5	23
FIRA standalone	317 ± 25	14.0 ± 7.3	$\mathbf{14.2 \pm 1.9}$	100
labeler standalone	292 ± 23	7.0 ± 4.0	12.7 ± 1.7	100
flat MLP standalone	271 ± 24	2.0 ± 2.4	10.8 ± 1.6	100
flat MLP gated, `health_low`	259 ± 16	9.0 ± 3.7	10.0 ± 1.0	23
behavior cloning standalone	257 ± 20	2.0 ± 4.0	9.9 ± 1.2	100

evaluation episodes of Fig. 5b, overrides are triggered by the timer atom (80.5%), hazard atoms (11.1%; `forced_self_defense`, `punchable_threat_critical_proximity`), and connector atoms (8.5%; `climb_connector`); none by an atom outside the intervention set.

Why the timer atom dominates in Kangaroo. The frozen PPO+CNN Kangaroo policy learns a reward shortcut: it stays on a low floor punching the enemies that keep arriving, which earns points but never reaches the joey, until the level timer expires. `time_critical` is the precondition of that timer rule, like `oxygen_critical` for Seaquest's oxygen rule; when it fires, the rescue action breaks the loop and climbs toward the goal, and the frozen policy resumes once the ego is making progress. The hazard and connector atoms handle the novel sprite and the climb itself. Its gated gains therefore combine two failures, the novel hazard and the stalled progress (Table 2, "Novel hazard + stall"; shares in Fig. 5b). BlendRL does not stall, and its Kangaroo row is the novel hazard alone.

Resource atoms hold for as long as the resource is low. In Kangaroo most overrides come from `time_critical`, the precondition of the level-timer rule, and in ViZDoom from `ammo_low`; both stay true until the level is done or the ammunition is spent, and these persistent conditions account for the sustained overrides in both games.

L The Labeler Run Live: Ports of the Scene Binding

The Kangaroo labeler's rules take bound arguments (the target connector, the platform band the ego occupies, the nearest hazard). During offline labeling, the construction binds the connector once per clip from the clip's first frame and holds it, and tracks the band across the clip. A live controller must rebind from each frame, and each mismatch with the pipeline's binding leaves the labeler at a fixed point; porting the binding step by step removes one fixed point and exposes the next (Table 15, 10 evaluation seeds per row). With the full binding the labeler climbs, and then loses its lives to a hazard its own proximity rule handles with the wrong action. The reasoner trained on the same labels (Table 1) needs none of these ports and passes that floor.

Table 15: Kangaroo, labeler run live as a standalone controller, 10 evaluation seeds per row (20 for the final port, the number quoted in Sec. 4.2). Each row adds one port of the labeling pipeline's scene binding to the live wrapper. *Frames*: mean episode length; *top floor*: seeds reaching the top floor; *dominant atom*: the non-goal atom on for the most frames.

Live binding	**Reward**	**Frames**	**Top floor**	**Dominant atom**
Nearest ladders by distance (adapter order)	0 ± 0	804	0/10	`climb_connector`
+ ladders of the ego's platform band (as in labeling)	0 ± 0	3,877	0/10	`left_connector_close`
+ hold the ladder being climbed (the clip's fixed connector)	0 ± 0	3,881	0/10	`left_connector_close`
+ band geometry from the held ladder (dismount at the lip)	$1{,}835 \pm 1{,}251$	1,031	6/20	`move_to_right_connector`

M Atom Derivation

Table 16 lists the new atoms of each task and the environment rule each is the precondition of.

Table 16: New atoms per task and the environment rule each comes from.

Task	New atoms ← environment rule
Kangaroo	first game, all six families defined here; `time_critical` ← level timer expires
Seaquest	`oxygen_critical` ← oxygen reaches zero; `no_diver_in_tank` ← surfacing with no divers
Frostbite	`landing_unsafe` ← landing in water; `igloo_complete` ← level advance
ViZDoom: defend-the-center	`unseen_threat_close` ← contact from outside the field of view
ViZDoom: health gathering	`health_low` ← health reaches zero

Worked example: Frostbite. The environment ends a life when the player is in water (`in_water` $= \neg$`on_shore` $\wedge \neg$`on_ice` $\wedge \ldots$) and advances the level when the igloo is complete (`building_igloo_idx` $= 15$) and the player enters its door. The first step names these two rules; the second writes their preconditions over the perceived scene: `landing_unsafe` (the ice row the player will land on has no floe under the landing interval, with a 3-pixel margin) and `igloo_complete` (16 blocks placed). The third trains the reasoner and checks it against the labeler live on development seeds 0–4 (precision 100%, recall 98% on the intervention atom); on fresh seeds 15–34 the gated agent completes the first level in $19/20$ episodes against $0/20$ for the bare policy.

N Override Gate: Gated vs. Standalone

Table 17 compares, for one setting per game, the frozen policy alone with FIRA deployed gated and standalone; Sec. 4.4 discusses the trade-off between reward and leaving the frozen policy in control.

Table 17: **The override gate.** Each game uses its pre-declared checkpoint per deployment mode: the gated column is the checkpoint of Table 2 and the standalone column that of Table 1 (the same checkpoint for Frostbite); *gated* = override only when an intervention atom fires, *standalone* = the reasoner acts every frame; IR is the gated override rate. Kangaroo: snake variant in all three columns; Seaquest: resource management; Frostbite: seeds 15–34; 20 evaluation seeds each.

Setting	Policy alone	FIRA gated	FIRA standalone	IR gated (%)
Kangaroo, novel hazard	$1{,}250 \pm 1{,}076$	$29{,}920 \pm 448$	$\mathbf{47{,}090 \pm 8{,}599}$	44.5
Seaquest, resource mgmt. (BlendRL)	$62{,}852 \pm 46{,}197$	$81{,}614 \pm 29{,}531$	$\mathbf{98{,}967 \pm 10{,}656}$	2.9
Frostbite	126 ± 14	$\mathbf{2{,}527 \pm 849}$	30 ± 0	23.3

O Significance Tests for Gated Rescue

Table 18 tests each gated-rescue row of Table 2 for a difference between the frozen policy alone and with FIRA: Welch's t-test over evaluation seeds for the Atari rows (20 per arm), and, for ViZDoom, where episodes are nested within 5 checkpoints, Welch's t-test over checkpoint means (the Mann–Whitney test over pooled episodes, which ignores that nesting, is shown in parentheses for reference). In every Atari setting, the failure FIRA is designed to rescue improves significantly ($p < 0.01$): reward on Kangaroo, first-level completion on Frostbite, and oxygen deaths in both Seaquest settings. In ViZDoom health gathering, survival and time-cap episodes also improve significantly at the checkpoint level ($p < 0.01$ over five checkpoints); in defend-the-center the gain in kills is consistent in direction but not significant ($p = 0.09$). In Seaquest the rescue removes oxygen deaths almost entirely while reward, which is dominated by kills and dives, stays level or rises.

P Held-Out Evaluation Seeds

The atoms, margins, and runtime conditions were set with live runs on at most evaluation seeds 0–9: the Kangaroo timer atom on seeds 0–4, the Seaquest oxygen condition on seeds 0–5, the ViZDoom intervention sets on the reported checkpoints, and the Frostbite atoms on development seeds 0–4, disjoint from evaluation. Atari evaluation seeds 10–19 (Frostbite 25–34) were added after every atom, margin, and runtime condition was fixed and were not inspected beforehand. Table 19 repeats the gated rescue of Table 2 on these seeds alone.

Table 18: Significance of gated rescue, frozen policy alone vs. with FIRA.

Setting	**Metric**	**Alone**	**+FIRA**	p
Kangaroo, PPO+CNN	reward	4,397	26,658	$< 10^{-20}$
Kangaroo, BlendRL	reward	1,175	10,370	2×10^{-10}
Seaquest, combat	oxygen deaths per episode	2.4	0.0	3×10^{-12}
Seaquest, combat	reward	1,249	1,345	0.42
Seaquest, resource mgmt.	oxygen deaths (total)	34	1	8×10^{-6}
Seaquest, resource mgmt.	reward	62,852	81,614	0.14
Frostbite	reward	126	2,527	10^{-10}
Frostbite	first level completed	0/20	19/20	8×10^{-14}
ViZDoom, defend-the-center	kills	9.81	10.61	0.09 (6×10^{-6})
ViZDoom, health gathering	episodes reaching the time cap	3%	16%	3×10^{-3} (2×10^{-3})
ViZDoom, health gathering	survival steps	200	268	8×10^{-3} (10^{-4})

Table 19: Gated rescue on the held-out evaluation seeds only (10 seeds per row; same checkpoints and settings as Table 2).

		Reward ↑		**Success ↑**	
Setting	**Base policy**	Base	+FIRA	Base	+FIRA
Kangaroo, novel hazard	PPO+CNN	$4{,}380 \pm 755$	$\mathbf{27{,}080 \pm 2{,}249}$	0.0	**8.3** levels
Kangaroo, novel hazard	BlendRL	$1{,}080 \pm 519$	$\mathbf{9{,}803 \pm 3{,}846}$	0.2	**5.0** levels
Seaquest, combat	PPO+CNN	$1{,}258 \pm 188$	$1{,}224 \pm 412$	23 oxygen deaths	**0**
Seaquest, resource mgmt.	BlendRL	$66{,}384 \pm 49{,}500$	$\mathbf{72{,}501 \pm 31{,}847}$	99 divers	**111**
Frostbite, reactive control	PPO+CNN	130 ± 16	$\mathbf{2{,}613 \pm 649}$	0/10	**10/10**

Q Spread Over Evaluation Seeds and the Live Agreement Diagnostic

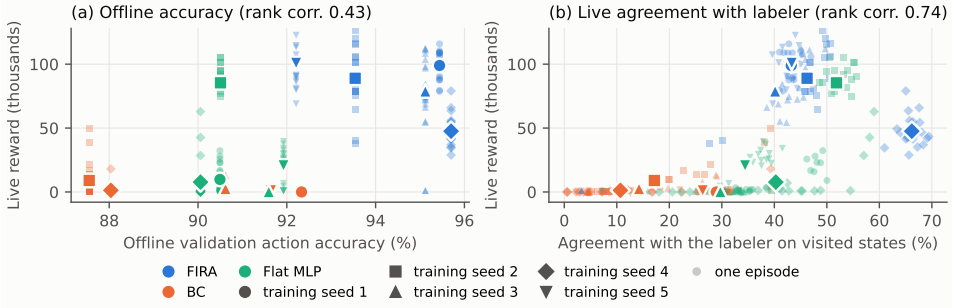


Figure 6: Seaquest, resource management, standalone; the fifteen checkpoints of the Seaquest column of Table 1 (five training seeds per model, twenty evaluation seeds each). (a) Offline validation action accuracy vs. live reward (rank correlation 0.43 over checkpoints). (b) Agreement of the deployed model with its labeler on the states it visits vs. live reward; a diagnostic measured after deployment (rank correlation 0.74 over checkpoints), not an offline predictor.

Tables 1 and 3 report means over evaluation seeds; the tables below add the standard deviation. The released analysis scripts write the per-seed values behind these tables as CSV files (`analysis/collect_per_seed.py`, `analysis/perseed_kangaroo.py`), and `scripts/evaluate.sh` lists the command behind each run.

Table 20: Table 1 with the spread over evaluation seeds: per training checkpoint, mean ± std of reward (thousands) and the number of evaluation seeds.

Model	Kangaroo	n	Seaquest	n	Frostbite	n
Behavior cloning	1±1 / 2±2 / 17±5 / 0±0 / 26±3	20	0± 0 / 9±14 / 3± 4 / 1± 4 / 1± 1	20	—	
Flat MLP	32± 8 / 4± 2 / 3± 2 / 23±10 / 6± 4	20	10±11 / 85±22 / 0± 1 / 8±17 / 20±13	20	1.69±0.92 / 1.40±0.82 / 1.85±0.57 / 1.38±0.88 / 1.65±0.99	20
FIRA	**47± 9 / 46± 8 / 42±10 / 49± 7 / 43±11**	20	**99±11 / 89±24 / 79±34 / 48±12 / 101±17**	20	**2.53±0.85 / 2.55±0.75 / 2.44±0.79 / 2.66±0.79 / 2.71±0.81**	20

Table 21: Table 3 per training run: live reward (mean ± std over 20 evaluation seeds, thousands) in training-seed order.

Configuration	Reward per run	Runs
FIRA	**47.1± 8.6 / 46.5± 8.1 / 42.0± 9.7 / 48.6± 7.2 / 42.9±10.6**	5
w/o edge features	39.6± 8.2 / 32.9± 9.8 / 28.9± 9.6 / 15.9± 9.4 / 24.7±12.7	5
w/o atom head	34.9±12.2 / 44.9± 1.1 / 40.1±15.1 / 49.4± 2.9 / 1.6± 1.5	5
Non-detached head	0.0± 0.0 / 36.5±13.4 / 48.8± 5.0 / 31.5± 5.3 / 47.7± 5.2	5