LATENTBOT: REPLACING SOURCE CODE WITH ACTIVATIONS IN PROGRAM EQUILIBRIUM

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents increasingly interact in settings where cooperation depends not only on observed actions but on anticipating how other agents will behave. Classical program equilibrium solves this by letting agents read one another's source code, but an LLM's weights do not reveal the strategy it will follow in a given interaction. We introduce LATENTBOT, which replaces source-code inspection with activation inspection: it extracts an opponent's pre-action hidden state and uses lightweight probes to estimate both the opponent's immediate action and its counterfactual reaction to cooperation or defection. These forecasts are combined with interaction history and short-horizon utility estimates and passed to the acting LLM, which retains the final decision rather than following a fixed strategy. Across six games from COOPEVAL, LATENTBOT converges to sustained mutual cooperation in self-play while adapting to competitive, suspicious, and reciprocal opponents, and plays competitively in zero-sum Matching Pennies. Probes recover an opponent's counterfactual next action with 93.6% accuracy before it is emitted. The signal transfers across model families: a 128-dimensional linear alignment from Mistral into Qwen's latent space preserves a frozen Qwen probe's predictions without any Mistral game labels. A behaviorally matched surrogate extends the method to an output-only commercial model. Internal activations therefore offer a coordination channel between independently trained agents beyond what their actions and messages alone provide.

1 INTRODUCTION

In many strategic interactions, individually rational behavior leads to collectively suboptimal outcomes even when mutually beneficial cooperation is possible. Classical work in algorithmic game theory addresses this problem through *program equilibrium* (Tennenholtz, 2004; Oesterheld, 2019; Cooper et al., 2025; LaVictoire et al., 2014), where agents condition their actions on program executed by their opponent. In particular, agents such as FairBot and PrudentBot use access to opponent's source code along with logical reasoning to establish whether cooperation will be reciprocated. This lets agents cooperate with compatible counterparts without cooperating unconditionally.

Recent work extends these ideas to LLM agents by making the strategy explicit: agents generate, exchange, and inspect executable programmatic policies, preserving source-code transparency (Sistla & Kleiman-Weiner, 2025; Lin et al., 2025). Many LLM agents, however, act directly through neural inference, with no separately declared policy; their behavior depends on the prompt, history, and current computation, and is usually inferred from repeated observation, opponent models, or opponent shaping. This raises a natural question:

> *Can an LLM agent infer another agent's strategic program directly from its internal computation, rather than from its source code or repeated behavioral observations?*

We investigate this through Activation-Based Program Equilibrium, a neural analogue that replaces program access with internal activations. It retains the core idea of conditioning on an opponent's computation, without claiming classical equilibrium guarantees. We hypothesize that, for neural agents, strategically relevant computation is dynamically represented in activation space. Rather

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.

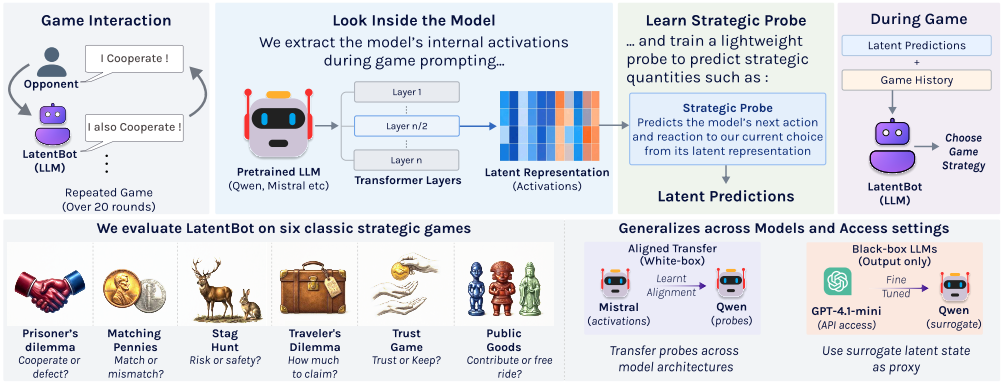


Figure 1: LATENTBOT uses internal LLM activations and lightweight probes to predict strategic responses and choose actions in repeated games. We evaluate it across six games and test transfer across both white-box and black-box model settings.

than proving from source code that another agent will cooperate, an agent decodes the opponent's latent strategic commitment from its hidden states, cooperating when that state indicates reciprocity and retaining the ability to defect otherwise. The approach is motivated by evidence that language models internally represent future decisions before generating them, and that such information is recoverable with lightweight probes (Hanna & Ameisen, 2026). We therefore treat intermediate activations as a probabilistic *latent commitment* to the strategy currently being computed.

A major difficulty is that representation spaces differ across architectures, preventing direct probe transfer. We address this using recent findings on representational convergence. The Platonic Representation Hypothesis suggests that sufficiently capable models learn structurally similar representations of the underlying data-generating process despite differences in architecture, objective, and modality (Huh et al., 2024), and related work on model stitching (Chen et al., 2025; Bansal et al., 2021) shows that comparatively simple mappings can relate internal representations across networks. We investigate if this structure lets a strategic intent decoder transfer between LLM families after learning only a lightweight representation mapping. The resulting agent, LATENTBOT, is designed to exhibit selective cooperation: two compatible LATENTBOT agents should recognize one another's cooperative intent and achieve mutually beneficial outcomes, while remaining robust when facing unknown or potentially defecting opponents. This distinguishes it from a COOPERATEBOT, whose willingness to cooperate makes it vulnerable to exploitation. Our main contributions are:

- We introduce **Activation-Based Program Equilibrium**, a neural analogue of program equilibrium in which agents reason about an opponent using its internal activations rather than its source code, and instantiate it as LATENTBOT.
- We show that a probe on an opponent's pre-action hidden state predicts both its current action and its counterfactual reaction to each candidate move, and that supplying these forecasts to the acting LLM produces selective rather than unconditional cooperation across six COOPEVAL games (Tewolde et al., 2026).
- We show that strategic intent representations transfer across LLM families: a 128-dimensional linear alignment from Mistral into Qwen's latent space preserves the predictions of a frozen Qwen probe without any Mistral game labels.
- We extend the mechanism to output-only opponents through a behaviorally matched surrogate, and evaluate it against a commercial API-based model.

2 RELATED WORK

Program equilibrium and cooperative AI. LaVictoire et al. (2014) show that agents reasoning about one another's source code via Löb's theorem can cooperate in the one-shot Prisoner's Dilemma without being exploited, and Cooper et al. (2025) characterize the equilibria reachable when agents instead simulate each other's programs. Recent work brings this to LLM agents that generate, share,

and inspect explicit programmatic strategies (Sistla & Kleiman-Weiner, 2025; Lin et al., 2025). LATENTBOT targets the complementary setting where LLM acts through neural inference and its weights don't expose its explicit strategy in an interaction; we keep the idea of conditioning on evidence about opponent's computation but use its intermediate activations as that evidence.

LLM agents in repeated games. Akata et al. (2025) show that LLMs play repeated games systematically but often suboptimally, coordinating unreliably and failing to forgive, and Sinishaw et al. (2026) find that self-play agents don't consistently reach equilibrium. We adopt COOPEVAL (Tewolde et al., 2026), a benchmark of cooperation-sustaining mechanisms across social dilemmas, as our evaluation suite. These studies characterize behavior from the outside; we ask whether the internal states generating it can be read and used by another agent.

Opponent modeling and shaping. Existing approaches infer the opponent from its outputs: explicit natural-language opponent models refined through feedback (Yu et al., 2025), structural causal models of opponent behavior (Cao et al., 2026), predictive models of strategic behavior (Za et al., 2026), and opponent shaping, which deliberately influences a co-player's learning (Segura et al., 2025). All need behavioral evidence accumulated over many rounds; LATENTBOT instead reads a pre-action representation, available before any action is observed and without modifying opponents.

Probing and aligning strategic representations. Our approach requires strategic intent to be linearly accessible in hidden states. Hanna & Ameisen (2026) show that models represent future tokens and plans before emitting them, increasingly with scale; Ferraz et al. (2026) probe internal correlates of strategic choice; and Das et al. (2026) show that activations reveal evaluation awareness absent from outputs. Black-box extraction can recover hidden dimensions and final-layer weights from API outputs (Carlini et al., 2024), whereas we seek behaviorally useful representations for online opponent reasoning. We turn such probing from an offline interpretability finding into a decision-making mechanism consumed by a second agent during play. Applying a probe across model families further requires related representation spaces; work on representational convergence and alignment (Huh et al., 2024; Chen et al., 2025; Jha et al., 2025; Park et al., 2024) suggests independently trained models converge toward shared structure, which we test by reusing a frozen decoder through a learned low-rank map.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

We consider a repeated game $\mathcal{G}$ between two language-model agents A and B over rounds $t = 1, \ldots, T$, with action spaces $\mathcal{A}^A, \mathcal{A}^B$ and payoffs $u^A, u^B : \mathcal{A}^A \times \mathcal{A}^B \to \mathbb{R}$ (e.g., $\mathcal{A}^A = \mathcal{A}^B = \{C, D\}$ in the Prisoner's Dilemma; the formulation is not restricted to binary or symmetric games). Before acting at round t, an agent observes the public history $h_t = (a_1^A, a_1^B, \ldots, a_{t-1}^A, a_{t-1}^B)$ together with a textual game description; $x(h_t)$ denotes the complete model input, and agent i induces a stochastic policy $\pi_i(a \mid h_t) = P(a_t^i = a \mid x(h_t))$. This behavioral view discards the internal computation by which the model reaches its decision.

Latent representation of strategic state. For the model M_i underlying agent i, we define its latent strategic representation at layer ℓ as

$$z_t^{i,\ell} = \phi_\ell(M_i, x(h_t)) \in \mathbb{R}^{d_\ell}, \tag{1}$$

where ϕ_ℓ specifies the layer and token position of extraction; we write z_t^i when ℓ is clear. Our central premise is that z_t^i carries information about the agent's strategic state not captured by its most recent action: two histories may yield the same action while inducing different representations and, consequently, different reactions to subsequent play.

3.2 LATENTBOT

LATENTBOT treats an opponent's intermediate representations as an additional source of information for strategic decision making, combining the public history with predictions derived from the

opponent's latent state at each round as shown in Figure 1. The latent component is not itself a game-playing policy; it estimates how the opponent is likely to behave under each candidate action, and these estimates are exposed to the acting language model when it produces its final decision. The pipeline has three stages: activation extraction, action decoding, and counterfactual evaluation.

Activation extraction. For opponent model M_B, we pass the complete textual game context $x(h_t)$ through the model and record hidden state $z_t^B = \phi_\ell(M_B, x(h_t))$ at an intermediate transformer layer, using final input position before action generation. z_t^B is therefore a pre-action representation of the opponent's strategic computation rather than an embedding of the generated action. Because raw state is high-dimensional relative to the number of interaction samples, we project it with a PCA map $\Pi : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_p}$ fitted on training activations only and held fixed at evaluation, giving $\tilde{z}_t^B = \Pi(z_t^B)$; the native experiments retain $d_p = 64$ components and read layer 14 of Qwen3.5-9B. The procedure doesn't modify the opponent model and requires no gradient updates to its parameters.

Action decoder. We first test whether the extracted representation contains information about the opponent's immediate behavioral decision. Given a latent representation $\tilde{z}_t^B$, an action decoder D predicts a probability distribution over the opponent's available actions,

$$D(\tilde{z}_t^B) = \hat{P}\big(a_t^B \mid h_t\big)\,, \tag{2}$$

which is binary for prisoner's dilemma and categorical in general. The decoder is trained independently of game reward; its only objective is to recover the opponent's generated action from its pre-action hidden state, so successful decoding indicates that the eventual strategic choice is already represented internally before the action is emitted. For each game, activations and actions are collected from sampled interaction states, with 25% reserved for probe evaluation. The decoder is an interpretation mechanism for the opponent's latent state, not a replacement policy for LATENTBOT.

Counterfactual reaction prediction. Predicting the opponent's current action is insufficient for strategic reasoning: LATENTBOT must estimate how the opponent would react to the action it is considering. Let $\hat{b}_t = \arg\max_{b \in \mathcal{A}^B} \hat{P}(a_t^B = b \mid h_t)$ be the decoded current action. For tractability, we condition each counterfactual roll-out on this modal estimate rather than marginalizing over the full current-action posterior. Importantly, $\hat{b}_t$ is used only to construct the current round's counterfactual contexts; it is not carried forward as an observed action in subsequent rounds. For every candidate action $a \in \mathcal{A}^A$, we construct the action-conditioned history $h_{t+1}^{(a,\hat{b}_t)} = h_t \oplus a \oplus \hat{b}_t$, evaluate the opponent model under this hypothetical context, and project the result,

$$z_{t+1}^{B,(a,\hat{b}_t)} = \Pi\Big(\phi_\ell\left(M_B, x\left(h_{t+1}^{(a,\hat{b}_t)}\right)\right)\Big)\,. \tag{3}$$

Applying the action decoder to this representation gives

$$\hat{P}\left(a_{t+1}^B = b \mid h_t, a_t^A = a, a_t^B = \hat{b}_t\right) = D\left(z_{t+1}^{B,(a,\hat{b}_t)}\right), \tag{4}$$

and repeating this for every $a \in \mathcal{A}^A$ produces an action-conditioned forecast of the opponent's behavior. Resulting probabilities are conditional on current public history and hypothesized completed round. Thus, an incorrect $\hat{b}_t$ can affect the counterfactual forecasts made within round t, but it is not inserted into the persistent interaction history and therefore does not induce multi-round prediction-error accumulation. Note that each intervention is rendered as a completed public round and passed through the opponent model again, rather than being simulated by editing an unchanged activation.

Strategic action selection. Each candidate action is also assigned its immediate expected payoff under the decoded current-action distribution, $V(a \mid h_t) = \sum_{b \in \mathcal{A}^B} \hat{P}\left(a_t^B = b \mid h_t\right) u^A(a, b)$; the native pipeline uses no discounted continuation term. A purely myopic strategy could directly choose $a_t^A = \arg\max_{a \in \mathcal{A}^A} V(a \mid h_t)$. LATENTBOT instead separates opponent prediction from policy generation: the action-conditioned predictions, the history, and the game description are supplied to the acting language model, which produces the final action under this augmented context. The latent models estimate how the opponent is likely to behave; the language model remains responsible for integrating that evidence and choosing the action. This lets LATENTBOT distinguish opponents whose observable histories look similar but whose internal computations imply different responses.

LATENTBOT at round t. **(1)** Run the opponent model on the public history and read its pre-action hidden state. **(2)** Decode the opponent's most likely current action $\hat{b}_t$. **(3)** For each own action a, render the hypothetical completed round $h_t \oplus a \oplus \hat{b}_t$. **(4)** Re-run the opponent on each hypothetical history and decode its next-round reaction. **(5)** Give the history, game description, payoff estimates, and all forecasts to the acting LLM, which outputs the action. **(6)** Append the *observed* actions (never $\hat{b}_t$) to the history and repeat.

Figure 2: Round-level LATENTBOT decision procedure.

3.3 CROSS-MODEL LATENT TRANSFER

A decoder trained on one model's activations cannot be applied directly to another model family, whose hidden states may differ in dimension, orientation, and layer-wise organization. We therefore learn a map from the latent space of a target model M_T into that of the source model M_S on which the decoder was trained.

Paired representation collection. For shared interaction contexts $x(h_n)$ we extract $z_n^S = \phi_{\ell_S}(M_S, x(h_n))$ and $z_n^T = \phi_{\ell_T}(M_T, x(h_n))$, giving $\mathcal{D}_{\text{align}} = \{(z_n^T, z_n^S)\}_{n=1}^N$. Each pair describes both models' computation under an identical strategic context, so no strategic action labels are needed for the target model.

Representation alignment. We standardize each model's activations and independently project them into r-dimensional PCA spaces via transforms Π_T^{align} and Π_S^{align}, writing $\tilde{z}_n^T = \Pi_T^{\text{align}}(z_n^T)$ and $\tilde{z}_n^S = \Pi_S^{\text{align}}(z_n^S)$, and fit a ridge-regularized linear map $R_\psi : \mathbb{R}^r \to \mathbb{R}^r$ with weights W_ψ,

$$\psi^* = \arg\min_\psi \sum_{n=1}^N \big\| R_\psi(\tilde{z}_n^T) - \tilde{z}_n^S \big\|_2^2 + \lambda \|W_\psi\|_F^2, \qquad A_\psi(z^T) = \big(\Pi_S^{\text{align}}\big)^{-1}\big(R_\psi(\Pi_T^{\text{align}}(z^T))\big), \tag{5}$$

so A_ψ returns vectors in source model's native activation space. With Qwen3.5-9B as source and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) as target (both 4096-dimensional), validation search over layer pairs, $r \in \{4, 8, 16, 32, 64, 128\}$, and ridge strengths selects r=128, mapping Mistral layer 17 into Qwen layer 14. The paired states are split $70/15/15$ into trajectory-disjoint training, validation, and test sets.

Frozen-decoder transfer. The source decoder is kept fixed, and target activations are interpreted as $\hat{P}_T(a) = D_S(A_\psi(z^T))$, where D_S applies its own independently fitted 64-dimensional PCA before classification. The same substitution is inserted into the counterfactual pipeline, so that only the alignment map is learned for the target model. This distinction is central. If the target model were given game-action labels and a new decoder were trained on its activations, success would establish only that the target also contains decodable strategic information. Preserving the source decoder and learning only a representation map instead tests whether strategically relevant structure transfers across independently trained model families, as representational convergence (Huh et al., 2024) would predict.

3.4 BLACK-BOX LATENTBOT

Methods above assume access to opponent's intermediate activations, which is unavailable for many deployed agents. We therefore also consider a black-box extension in which an accessible model M_S is fine-tuned to imitate the queried behavior of an inaccessible opponent $M_B^\star$, and LATENTBOT reads the surrogate's activations $z_t^S = \phi_\ell(M_S, x(h_t))$ as proxy for the computation it cannot observe.

Because the target's activations are unavailable, the representational correspondence cannot be verified, and the surrogate's latent state is treated only as an approximation of the target's strategic computation. The three access regimes differ only in how the representation supplied to the latent pipeline is obtained,

$$z_t^{\text{LB}} = \begin{cases} z_t^B, & \text{native white-box access,} \\ A_\psi(z_t^T), & \text{cross-model latent transfer,} \\ z_t^S, & \text{black-box surrogate access.} \end{cases} \tag{6}$$

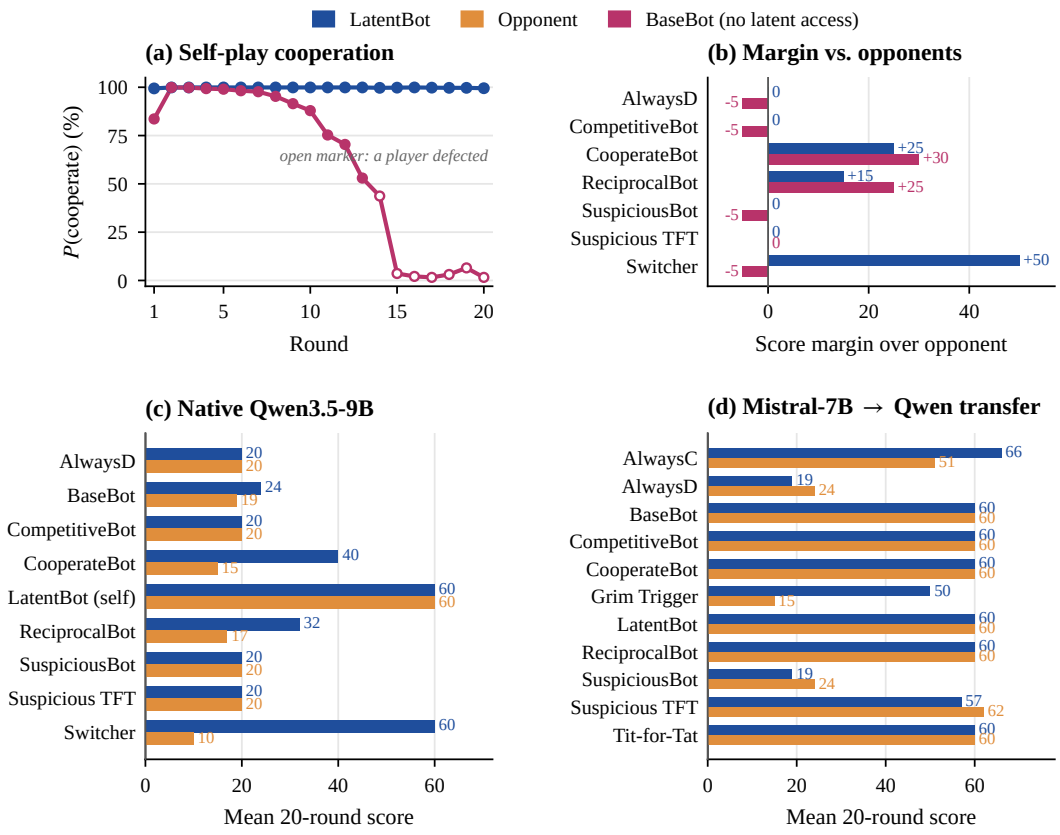


Figure 3: Prisoner's Dilemma, 20-round matches, means over five seeds. (a) Per-round probability of cooperation in self-play: LATENTBOT stays near 1 throughout, whereas BaseBot collapses after round 14. (b) Score margin over each opponent for BaseBot and LATENTBOT. (c) Total scores of native Qwen3.5-9B LATENTBOT. (d) Total scores after aligning Mistral-7B-Instruct-v0.3 activations to Qwen's latent space with a frozen Qwen probe.

Downstream prediction objective is unchanged. Native LATENTBOT accesses the target model's activation state directly; cross-model transfer preserves activation access through latent-space alignment; black-box variant instead relies on behaviorally matched surrogate as a proxy for latent state, rather than the target model's internal representation. Appendix B gives full construction, including the behavioral imitation objective and the procedure used to extract surrogate latent representations. Surrogate fidelity is evaluated on held-out trajectories before deployment; experiments reported here use the latent surrogate directly without an additional inference-time fallback mechanism.

4 EXPERIMENTS

We evaluate whether information extracted from an opponent's hidden state can support effective decision making in repeated multi-agent games. Experiments address four questions: (i) whether the opponent's latent representation predicts its behavior, (ii) whether LATENTBOT can cooperate selectively rather than following a fixed cooperative or competitive strategy, (iii) whether learned representation can be transferred between language-model families, and (iv) whether policy remains appropriate in final round, where short-term loss recovery is not possible through future interactions.

4.1 EXPERIMENTAL SETUP

Baseline. To isolate contributions of latent opponent modeling from the capabilities of underlying LLM, we compare LATENTBOT with BaseBot, otherwise identical agent that acts using interaction history but does not access opponent activations or counterfactual reaction predictions. Figure 3(a,b)

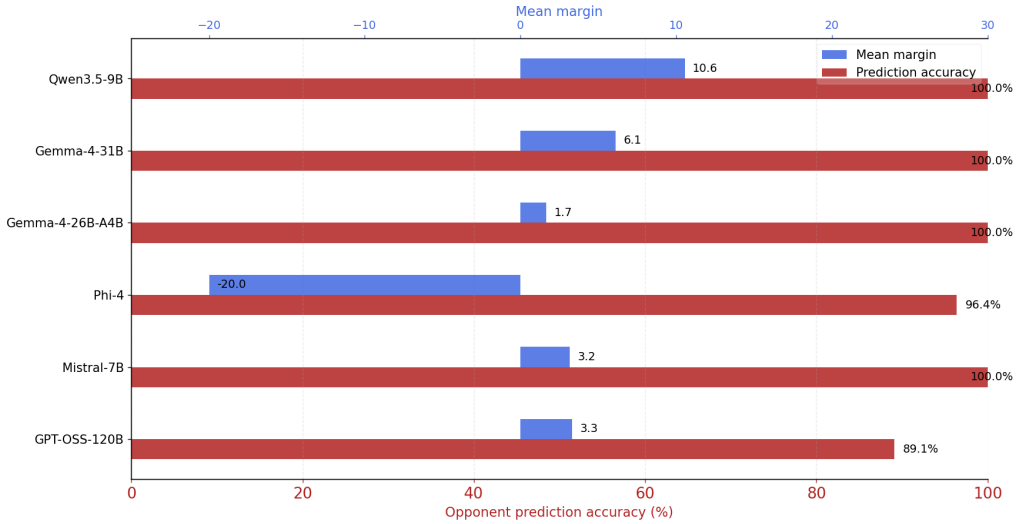


Figure 4: LATENTBOT generalizes across multiple open-weight language models, maintaining high opponent-action prediction accuracy (red) across backbones. Game margin (blue) is positive for most models, while Phi-4 shows negative score margin despite high prediction accuracy, suggesting that downstream decision quality also depends on the underlying model's game-playing ability.

shows that BaseBot fails to sustain cooperation in self-play and is susceptible to many opponent strategies, while LATENTBOT maintains stable cooperation and adapts effectively across opponents.

Games and opponents. The evaluation contains six games from COOPEVAL (Tewolde et al., 2026): Prisoner's Dilemma, Matching Pennies, Stag Hunt, Traveler's Dilemma, Trust, and Public Goods. Matches last 20 rounds and are repeated with five seeds. Opponent suites include fixed, reciprocal, conditional, exploitative, switching/noisy, baseline, and LATENTBOT strategies. In three-player Public Goods, the opponent bar shows the mean score of the two opponents. Unless stated otherwise, the plots report total reward over 20-round match, averaged across seeds. We use Qwen3.5-9B (Yang et al., 2025) as the primary backbone for the full evaluation, while additional open-weight models (Agarwal et al., 2025; Abdin et al., 2024; Team et al., 2026) are evaluated to test backbone generalization (Figure 4). The Qwen configuration uses layer 14, 64 principal components, histories of at most 10 rounds, and action temperature 1.0; probes are fit on 220 sampled states per game (240 for Traveler's Dilemma and Public Goods), with 25% held out.

Probe quality. On the held-out Prisoner's Dilemma data, the probe predicts opponent's current action with 100% accuracy on all samples. Its prediction of opponent's counterfactual reaction reaches 93.6% accuracy on all test samples ($F_1 = 0.936$). This shows a strong behavioral signal before game evaluations; the game evaluations below test if that signal is useful to the acting agent.

Cross-model transfer. On the held-out alignment test split (Section 3) the selected rank-128 map obtains 0.878 decision agreement with the source probe, a correlation of 0.773 between predicted defection probabilities, and a mean absolute probability error of 0.103. The transferred agent is evaluated without retraining the Qwen probe.

4.2 REPEATED-GAME RESULTS

Prisoner's Dilemma. Figure 3(c) shows that the native Qwen LATENTBOT is selectively cooperative. In self-play it reaches the mutually cooperative total of 60:60, while against AlwaysD, CompetitiveBot, SuspiciousBot, and Suspicious Tit-for-Tat it avoids exploitation and ties at 20:20. It outperforms BaseBot (24:19), CooperateBot (40:15), ReciprocalBot (32:17), and the regime-switching opponent (60:10). Consequently, the native agent is never outscored in this selected suite, but it does not obtain this protection by always defecting: self-play converges to mutual cooperation, with

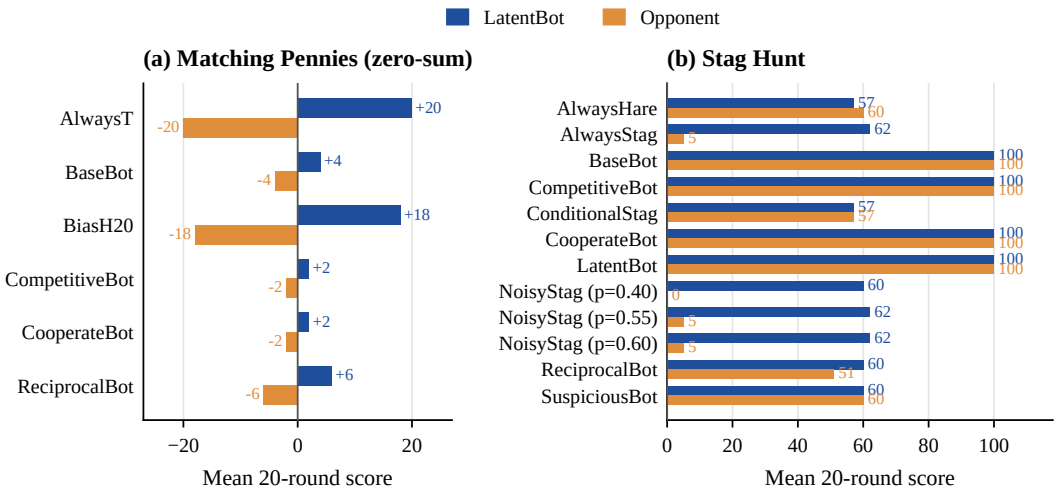


Figure 5: Mean 20-round scores (five seeds) for (a) Matching Pennies, which measures competitive exploitation, and (b) Stag Hunt, which measures coordination and risk-sensitive adaptation.

a mutual-cooperation rate of 1.0 in every one of the five seeds. The transferred Mistral policy preserves much of the cooperative behavior of the native system (Figure 3(d)). It ties at 60:60 with BaseBot, CompetitiveBot, CooperateBot, LatentBot, ReciprocalBot, and Tit-for-Tat; it also leads AlwaysC by 66:51 and Grim Trigger by 50:15. However, transfer is not uniformly protective. It loses 19:24 to AlwaysD and SuspiciousBot, and 57:62 to Suspicious Tit-for-Tat. These failures are small relative to the match length, but they show that representation alignment preserves the broad policy structure more reliably than its exact defensive boundary.

Matching Pennies and Stag Hunt. Matching Pennies tests whether the method can exploit predictable behavior when cooperation is neither required nor possible. In Figure 5(a), LATENTBOT wins against every opponent: 20:−20 against AlwaysT, 18:−18 against BiasH20, 6:−6 against ReciprocalBot, 4:−4 against BaseBot, and 2:−2 against both CooperateBot and CompetitiveBot. Zero-sum totals confirm the advantage is competitive, and not due to jointly higher rewards.

Stag Hunt tests coordination under risk. Figure 5(b) shows maximum mutual-stag totals of 100:100 with BaseBot, CompetitiveBot, CooperateBot, and LATENTBOT. Against ConditionalStag and SuspiciousBot, the policy settles on safe outcome and ties at 57:57 and 60:60, respectively. It obtains large positive margins against AlwaysStag and noisy-stag strategies, while its only loss is the small 57:60 deficit against AlwaysHare. The result therefore demonstrates opponent-sensitive shifts between efficient and risk-dominant equilibria, though AlwaysHare shows that adaptation is imperfect.

Traveler's Dilemma and Trust. In Traveler's Dilemma, LATENTBOT reaches the maximum displayed total of 100 with eight of the nine opponent types, and those opponents also receive 100. The exception is CompetitiveBot, against which LATENTBOT scores 94 while the opponent scores 102. Figure 6(a) therefore supports strong high-claim coordination across a varied suite, while also exposing one strategy that extracts a modest advantage.

The Trust results exhibit a sharper separation between cooperative and exploitative opponents. LATENTBOT achieves 200:200 with AlwaysGive, BaseBot, CooperateBot, LATENTBOT, ReciprocalBot, and Reciprocator. It is outscored by Exploiter (108:148), Switcher (132:172), and SuspiciousBot (76:96). As shown in Figure 6(b), the agent creates maximal mutual value with cooperative partners, but its early willingness to trust produces a measurable cost against opponents designed to exploit or withdraw cooperation.

Public Goods. Figure 6(c) reports the three-player Public Goods experiment. LATENTBOT and the average opponent each obtain the maximum displayed 20-round total of 30 for nine in ten opponent compositions, including reciprocal, competitive, and LATENTBOT compositions. Against

SuspiciousBot, LATENTBOT scores 19 and the opponents average 21.5. Near-uniform cooperative outcomes show that LATENTBOT can coordinate in multiple-agent settings as well, while the SuspiciousBot result again identifies cautious or adversarial partners as the principal failure mode.

4.3 FINAL-ROUND EVALUATION

Full-match totals mix early exploration with steady-state play, so we also examine round 20 alone, where no future retaliation or reward can justify a locally poor action; terminal behavior is identical across all five seeds (Appendix D). LATENTBOT wins every terminal round of Matching Pennies, and in Prisoner's Dilemma it cooperates in round 20 only against another LATENTBOT, with every other pairing ending in mutual defection. In Stag Hunt, Public Goods, and Trust it keeps cooperating with reciprocating partners but switches to the safe action against suspicious or exploitative ones (in Trust it keeps against Exploiter, Switcher, and SuspiciousBot despite earlier losses), and every Traveler's Dilemma match ends at the highest claim. Because the acting LLM, rather than an argmax over the immediate payoff estimates, chooses the action, terminal cooperation with LATENTBOT persists even though defection maximizes one-step payoff.

4.4 BLACK-BOX EXPERIMENTS

We finally test an output-only opponent: GPT-4.1 mini (`gpt-4.1-mini-2025-04-14`) playing a 20-round Prisoner's Dilemma under seven private instructions (Always Cooperate, Always Defect, Tit-for-Tat, Suspicious Tit-for-Tat, Grim Trigger, a Switcher that defects from round 11, and an API LATENTBOT). A single prompt-conditioned Qwen3.5-9B surrogate is LoRA-fine-tuned (Hu et al., 2022) on 3,248 current and counterfactual API action completions, frozen, and probed at layer 14. On held-out trajectories the surrogate imitates current and counterfactual actions with 98.0% and 93.2% accuracy, and the latent current-action and reaction probes reach 99.1% and 95.0% (prevalence baselines 53.9% and 54.3%); in 35 fresh live matches the current-action probe classifies every realized API action correctly. The resulting policy is selective and never outscored: it cooperates at rate 0.95 with the four primarily cooperative profiles, 0 against Always Defect, 0.50 against the Switcher, and 0.90 against the API LATENTBOT (e.g., winning 62–57 against Always Cooperate and 60–55 against the API LATENTBOT, tying Always Defect at 20–20). Unlike the native pipeline, this decision rule includes a discounted continuation term ($\gamma = 0.95$) that is dropped in round 20, where LATENTBOT therefore defects in every condition, showing that its earlier cooperation is horizon-sensitive. Appendices B and C give the full protocol and results.

4.5 LIMITATIONS

Across the six games the central result is selective rather than unconditional cooperation, but several losses delimit the claim: against AlwaysHare in Stag Hunt, CompetitiveBot in Traveler's Dilemma, SuspiciousBot in Public Goods, and exploitative or switching opponents in Trust, where early trust is costly. Mistral-to-Qwen transfer preserves self-play cooperation but introduces small losses against several defect-oriented Prisoner's Dilemma opponents. Higher probe accuracy alone does not establish that every behavioral improvement is causally attributable to the probed direction. These cases motivate stronger calibration, earlier detection of exploitation, and broader cross-model validation.

5 CONCLUSION

We introduced **Activation-Based Program Equilibrium** and its instantiation LATENTBOT, an LLM agent that conditions its play on an opponent's pre-action hidden state in addition to action history rather than on its source code. A lightweight probe recovers the opponent's current action and its counterfactual reaction to each candidate move, and these forecasts are supplied to the acting language model, which retains responsibility for the final decision. Across six COOPEVAL games, LATENTBOT sustains mutual cooperation in self-play while adapting to competitive and suspicious opponents, distinguishing it from an unconditional COOPERATEBOT; a low-rank alignment map lets a frozen Qwen probe interpret Mistral activations without Mistral game labels; and a behaviorally matched surrogate extends the mechanism to an output-only commercial model. Internal activations may therefore serve the role that source code plays in program equilibrium, exposing computation unavailable from an agent's actions or messages alone.

Ethics Statement

This work studies cooperation and exploitation between artificial agents in abstract economic games. No human subjects, personal data, or sensitive datasets were involved, and all opponents are either scripted strategies or language models under our own instruction. Two considerations nevertheless deserve comment. First, reading an opponent's activations to forecast its behavior is a dual-use capability: the same mechanism that sustains cooperation can be used to anticipate and exploit a counterpart, as our Matching Pennies and terminal-round results make explicit. We report these cases rather than only the cooperative ones. Second, the native and transfer settings presuppose that a counterpart's internal states are available, which is an assumption about deployment governance rather than a technical guarantee; we regard mutual, consensual activation access between cooperating agents as the intended setting, and we do not propose covert extraction of activations from unwilling systems.

Reproducibility Statement

Section 3 specifies the full pipeline, including the extraction layer, the token position, the PCA dimensionality, and the frozen-decoder transfer procedure. Section 4 reports the primary experimental configurations, including the six games, 20-round horizon, five seeds, probe sample counts, train/test splits, action temperature, and the main model settings used in our experiments. Appendix B gives the black-box construction and Appendix C the data-collection protocol, LoRA hyperparameters, probe-fitting procedure, and decision rule, including the discount factor and the explicit Q-value expressions.

AI Use Statement

In this work, we used generative AI tools for coding assistance and for editing and polishing prose. Language models are also objects of study in this work: they are used as experimental agents, as described in Sections 3 and 4. We did not use generative AI tools to generate research ideas, to conduct the literature search, or to produce experimental results without verification. All AI-assisted code was reviewed and tested by the authors, and all reported numbers were produced by executed experiments rather than by a language model. We have reviewed all AI-assisted text and take responsibility for the final content of this work, including its text, claims, and artifacts.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Elif Akata, Lion Schulz, Julian Coda-Forno, Seong Joon Oh, Matthias Bethge, and Eric Schulz. Playing repeated games with large language models. *Nature Human Behaviour*, 9(7):1380–1390, 2025.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

Shiyue Cao, Pei Xu, Likun Yang, Lei Cui, Xiaotang Chen, and Kaiqi Huang. SOM: Structured opponent modeling for LLM-based agents via structural causal model. *arXiv preprint arXiv:2605.07301*, 2026.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing part of a production language model. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=VE3yWXt3KB.

Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring linear features across language models with model stitching. *Advances in Neural Information Processing Systems*, 38: 48531–48563, 2025.

Emery Cooper, Caspar Oesterheld, and Vincent Conitzer. Characterising simulation-based program equilibria. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 13735–13744, 2025.

Aritra Das, Jaee Ponde, Mihir More, and Debayan Gupta. You can't escape your own activations: Evaluation awareness and multi-agent monitoring. *arXiv preprint arXiv:2609.03035*, 2026.

Vinícius Ferraz, Leon Houf, and Enrico Ferrea. The internal anatomy of strategic choice in large language models. *arXiv preprint arXiv:2609.07478*, 2026.

Michael Hanna and Emmanuel Ameisen. Latent planning emerges with scale. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H0B7pDTT0M.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The Platonic representation hypothesis. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Rishi Jha, Collin Zhang, Vitaly Shmatikov, and John Morris. Harnessing the universal geometry of embeddings. *Advances in Neural Information Processing Systems*, 38:45963–45987, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Patrick LaVictoire, Benja Fallenstein, Eliezer Yudkowsky, Mihaly Barasz, Paul Christiano, and Marcello Herreshoff. Program equilibrium in the Prisoner's Dilemma via Löb's theorem. In *AAAI workshop on multiagent interaction without prior coordination*, 2014.

Yue Lin, Shuhui Zhu, Wenhao Li, Ang Li, Dan Qiao, Pascal Poupart, Hongyuan Zha, and Baoxiang Wang. Policy-conditioned policies for multi-agent task solving. *arXiv preprint arXiv:2512.21024*, 2025.

Caspar Oesterheld. Robust program equilibrium. *Theory and Decision*, 86(1):143–159, 2019.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Marta Emili Garcia Segura, Stephen Hailes, and Mirco Musolesi. Opponent shaping in LLM agents. *arXiv preprint arXiv:2510.08255*, 2025.

Deborah Sinishaw, Qile Zhu, Edwin Meriaux, and Gregory Dudek. Do LLMs beat Nash? testing decentralized coordination in self-play multi-agent games. *arXiv preprint arXiv:2608.12547*, 2026.

Swadesh Sistla and Max Kleiman-Weiner. Evaluating LLMs in open-source games. *Advances in Neural Information Processing Systems*, 38:104032–104063, 2025.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Moshe Tennenholtz. Program equilibrium. *Games and Economic Behavior*, 49(2):363–373, 2004.

Emanuel Tewolde, Xiao Zhang, David Guzman Piedrahita, Vincent Conitzer, and Zhijing Jin. Coopeval: Benchmarking cooperation-sustaining mechanisms and LLM agents in social dilemmas. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=369qOr0ZnJ.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

XiaoPeng Yu, Wanpeng Zhang, and Zongqing Lu. LLM-based explicit models of opponents for multi-agent games. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 892–911, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.41. URL https://aclanthology.org/2025.naacl-long.41/.

Jennifer Za, Aristeidis Panos, and Jan Cuhel. Towards predictive models of strategic behaviour in large language model agents. In *ICML Workshop on Technical AI Governance Research*, 2026.

A Additional Game Results

This appendix contains the score panels for Traveler's Dilemma, Trust, and Public Goods discussed in Section 4.2.

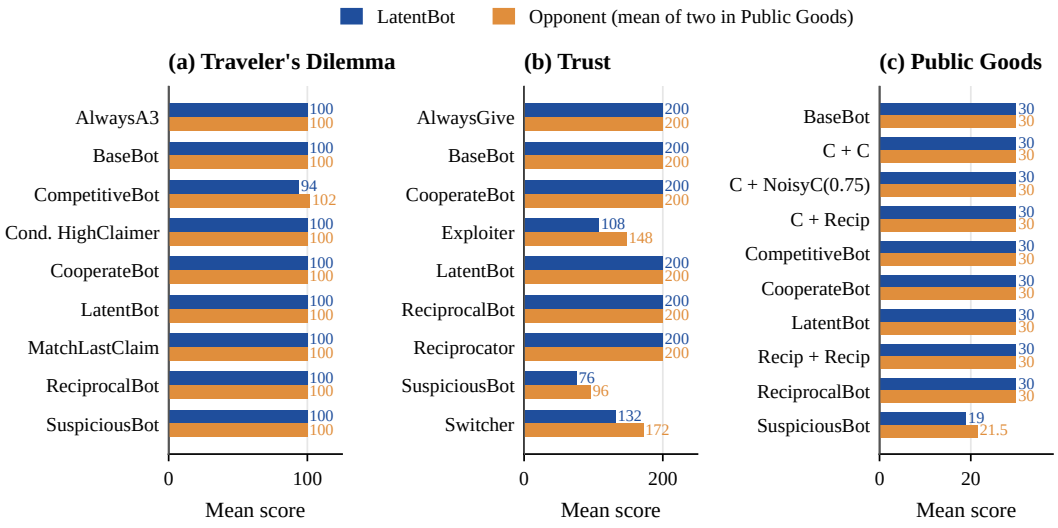


Figure 6: Mean 20-round scores (five seeds) for (a) Traveler's Dilemma, (b) Trust, and (c) Public Goods. In the three-player Public Goods game the opponent bar is the mean of the two other players.

B Black-Box LatentBot: Full Construction

This appendix expands Section 3.4.

Black-box access setting. Let $M_B^\star$ denote an inaccessible opponent model. The acting agent can supply an interaction context $x(h_t)$ and observe the resulting action $a_t^{B,\star} \sim M_B^\star(x(h_t))$, but cannot observe any internal representation $z_t^{B,\star}$. The native pipeline therefore cannot evaluate the opponent's internal computation directly, and we instead construct an accessible surrogate that approximates its behavior.

Behavioral surrogate. We query the black-box agent with a collection of diverse interaction histories $\mathcal{H} = \{h_1, \ldots, h_N\}$ and record its actions, giving $\mathcal{D}_{\mathrm{BB}} = \{(h_n, a_n^{B,\star})\}_{n=1}^N$. An accessible white-box model M_S is then adapted so that $\pi_S(a \mid h) \approx \pi_{B^\star}(a \mid h)$ by minimizing a behavioral imitation objective,

$$\theta_S^* = \arg\min_{\theta_S} \sum_{n=1}^{N} \mathcal{L}_{\mathrm{beh}}\left(\pi_S(\cdot \mid h_n), a_n^{B,\star}\right), \tag{7}$$

where $\mathcal{L}_{\mathrm{beh}}$ measures disagreement between the surrogate's predictions and the observed black-box behavior. The objective is behavioral fidelity rather than exact reproduction of the black-box model's internal computation.

Surrogate latent representation. Once a sufficiently accurate surrogate has been obtained, LatentBot extracts $z_t^S = \phi_\ell(M_S, x(h_t))$ and treats it as a proxy for strategic information that cannot be observed directly. The current-action prediction is first used to obtain $\hat{b}_t = \arg\max_{b \in \mathcal{A}^B} \hat{P}_S(a_t^B = b \mid h_t)$. For a candidate action a, LatentBot then constructs the com-

pleted hypothetical history $h_{t+1}^{(a,\hat{b}_t)} = h_t \oplus a \oplus \hat{b}_t$, evaluates $z_{t+1}^{S,(a,\hat{b}_t)} = \phi_\ell\Big(M_S, x\Big(h_{t+1}^{(a,\hat{b}_t)}\Big)\Big)$, and applies the standard latent reaction pipeline $z_{t+1}^{S,(a,\hat{b}_t)} \to \hat{z}_{t+1}^{S,(a,\hat{b}_t)} \to \hat{P}_S\Big(a_{t+1}^B \mid h_t, a, \hat{b}_t\Big)$.

Calibration against the black-box agent. The reliability of the approximation depends on how closely the surrogate reproduces the opponent's conditional behavior, which we evaluate on held-out interaction histories not used for adaptation. Let $q(h_t) \in [0,1]$ estimate surrogate reliability at h_t. When fidelity is high, greater weight can be placed on the latent prediction; when it is low, the latent forecast can be combined with a purely behavioral opponent model (this fallback is described for completeness and is not used in the reported experiments),

$$\hat{P}_{\mathrm{BB}} = \lambda_t \hat{P}_{\mathrm{latent}} + (1-\lambda_t)\hat{P}_{\mathrm{behavioral}}, \qquad \lambda_t \in [0,1], \tag{8}$$

where λ_t represents confidence in the surrogate's latent prediction. This fallback matters because even a surrogate that matches the target's actions with high accuracy may implement a different internal computation, so the surrogate latent state should be read as a proxy for the target's strategic computation rather than an estimate of its exact hidden activation.

C Black-Box Experiments: Protocol and Additional Results

This appendix expands Section 4.4. All reported opponent actions, including the counterfactual training labels, are API outputs; the reference strategy implementations in the repository are used only after play, to audit whether those outputs follow their instructions.

Opponent suite. The game is a 20-round simultaneous Prisoner's Dilemma with payoffs $(C,C) = (3,3)$, $(C,D) = (0,5)$, $(D,C) = (5,0)$, and $(D,D) = (1,1)$. The API temperature is zero and every response is constrained to a JSON action in $\{C, D\}$. The private instruction, game rules, role, horizon, and public history are provided to the API opponent on every stateless request. The same message contents are provided to the local surrogate, although GPT and Qwen necessarily use their own chat-template serializations. Thus "black box" denotes output-only access to the target model, not a setting in which the target's instruction is hidden.

Data collection and surrogate training. For each opponent instruction we collect 8 trajectories for surrogate fine-tuning, 4 disjoint trajectories for fitting probes, and 4 further disjoint trajectories for testing them; each trajectory contains 20 rounds. At a non-terminal state h_t we first query the API for its simultaneous current action B_t. Holding this sampled B_t fixed, we complete two intervened histories, $(h_t, A_t = C, B_t)$ and $(h_t, A_t = D, B_t)$, and query the API separately for B_{t+1} in each branch. Trajectories, rather than individual prefixes, are assigned to the three splits, preventing prefixes from one roll-out from crossing split boundaries. We fine-tune one shared, prompt-conditioned Qwen3.5-9B surrogate across all seven instructions. The model is loaded in 4-bit NF4 precision with bfloat16 computation and trained for two epochs using a rank-16 LoRA adapter with $\alpha = 32$ and dropout 0.05. Completion-only cross-entropy supervises the API's JSON action while masking every prompt token. The fine-tuning set contains 3,248 current and counterfactual action completions. A single adapter is shared by all profiles; opponent identity is represented only through the instruction in the prompt.

Latent probes. After fine-tuning, the surrogate is frozen. We extract the last prompt-token hidden state at transformer layer 14 and fit two logistic probes, each using standardization followed by 32-component PCA. The current-action probe estimates $p_t = P(B_t = D \mid h_t)$, whereas the reaction probe estimates

$$r_t(a) = P(B_{t+1} = D \mid h_t, A_t = a, B_t), \qquad a \in \{C, D\}, \tag{9}$$

from the hidden state of the corresponding completed counterfactual branch. The implementation does not append a C/D indicator to an unchanged activation: each intervention is rendered as a completed public round and passed through Qwen again. The probe-training split contains 560 current-action examples and 1,064 reaction examples, with equal-sized held-out test sets.

Decision rule and evaluation. Evaluation uses fresh 20-round API matches for five seeds per instruction, for 35 matches and 700 evaluation actions in total. On each non-terminal round the current API move is unavailable when Player A must commit, so the implementation uses the surrogate's pre-action choice $\hat{B}_t$ to complete both live branches; it never substitutes the API action revealed after the round. With discount $\gamma = 0.95$, the evidence passed to the policy includes p_t, both reaction forecasts, and the one-step utility estimates

$$Q_t(C) = 3(1 - p_t) + \gamma\big(5 - 4r_t(C)\big), \tag{10}$$

$$Q_t(D) = 5(1 - p_t) + p_t + \gamma\big(5 - 4r_t(D)\big). \tag{11}$$

The opponent-modeling LoRA adapter is then disabled, and the base Qwen policy receives the history and this evidence. Its action is the argmax of the sequence log-likelihoods of the two JSON completions. On round 20 the reaction branches and discounted continuation terms are omitted, so only immediate expected payoff remains.

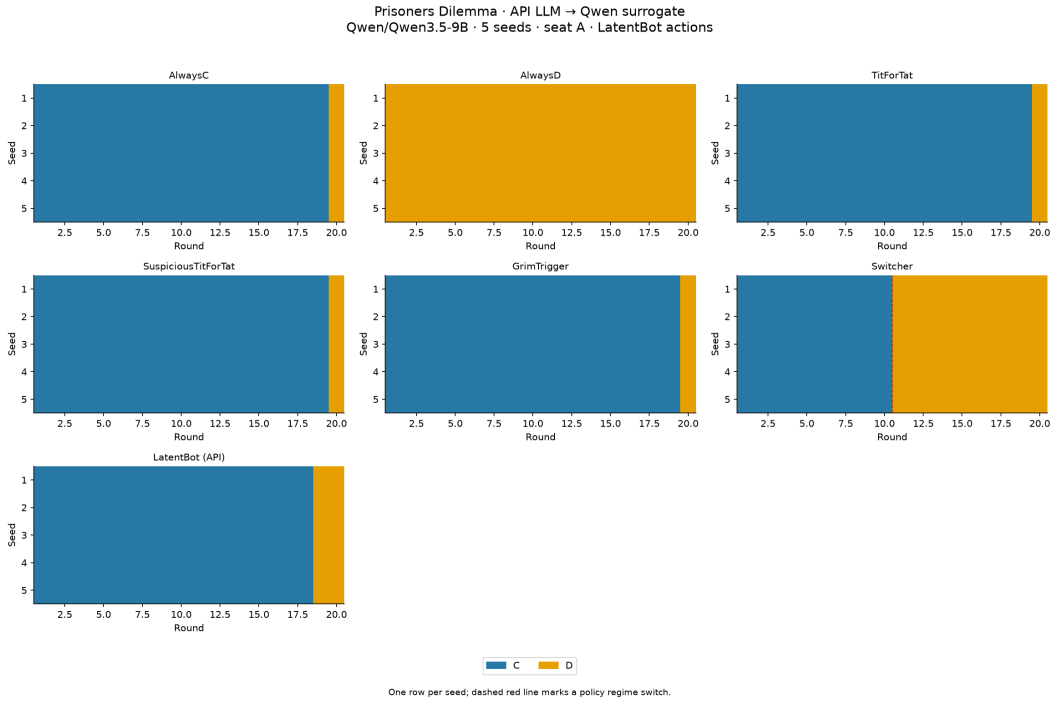


Figure 7: LATENTBOT actions across 20-round black-box Prisoner's Dilemma matches. Each row is one evaluation seed; blue denotes cooperation and orange denotes defection. The dashed line marks the Switcher's instructed regime change.

Adaptive action selection. Figure 7 shows the realized LATENTBOT action in every round and seed; the trajectories are identical across the five seeds. LATENTBOT cooperates through round 19 and defects only in the final round against Always Cooperate, Tit-for-Tat, Suspicious Tit-for-Tat, and Grim Trigger. It defects throughout against Always Defect, matches the Switcher's change exactly by cooperating in rounds 1–10 and defecting in rounds 11–20, and cooperates for 18 rounds before defecting in the two final rounds against the API LATENTBOT. The aggregate statistics in Figure 8 confirm that the policy is selective rather than uniformly cooperative or defective. Against the Switcher, both players cooperate for the first half and defect for the second, yielding mutual-cooperation and LATENTBOT-cooperation rates of 0.50.

Payoffs. Figure 9 reports total match scores: LATENTBOT wins 62–57 against Always Cooperate, Tit-for-Tat, and Grim Trigger, ties Always Defect at 20–20, Suspicious Tit-for-Tat at 59–59, and the

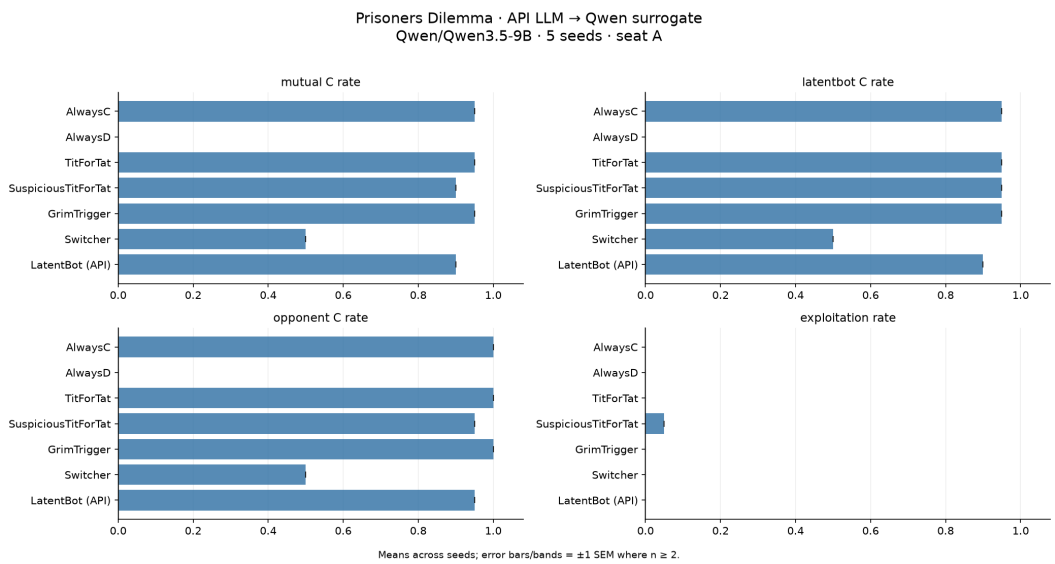


Figure 8: Behavioral statistics by API opponent instruction. Bars are means over five seeds and error bars are ± 1 standard error. Exploitation is a round in which LATENTBOT cooperates while its opponent defects.

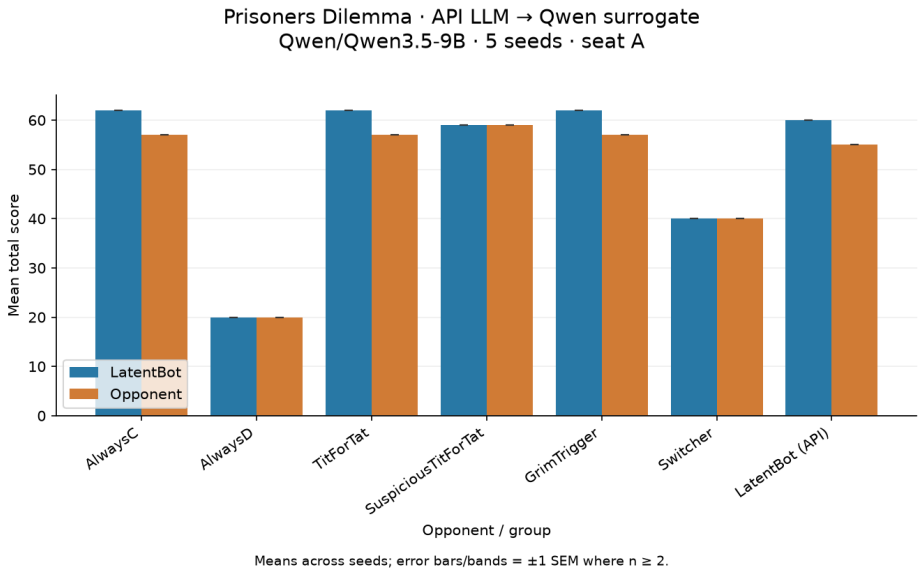


Figure 9: Mean 20-round total score for LATENTBOT and each API LLM opponent. Error bars show ± 1 standard error across five seeds.

Switcher at 40–40, and wins 60–55 against the API LATENTBOT. The 62–57 outcomes combine 19 mutually cooperative rounds with a profitable final-round defection, and the 40–40 Switcher result comes from ten rounds of mutual cooperation followed by ten of mutual defection.

Live forecast accuracy. The current-action probe classifies every realized API action correctly in the live matches for all seven profiles (Figures 10 and 11). This is a thresholded accuracy at $p_t = 0.5$, not a claim that every probability is exactly zero or one: the mean probability assigned to the realized action ranges from 0.986 for the API LATENTBOT to effectively 1.0 for the simplest fixed profiles. These are fresh trajectories but use the same seven instruction classes seen during

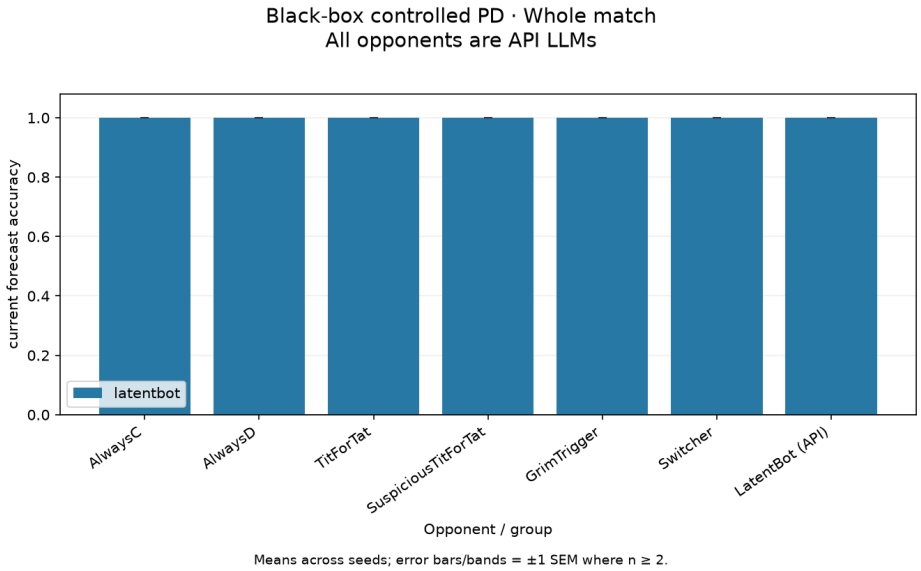


Figure 10: Current-action forecast accuracy over all rounds of the live evaluation. Accuracy is computed against the API LLM's realized action and averaged across five seeds.

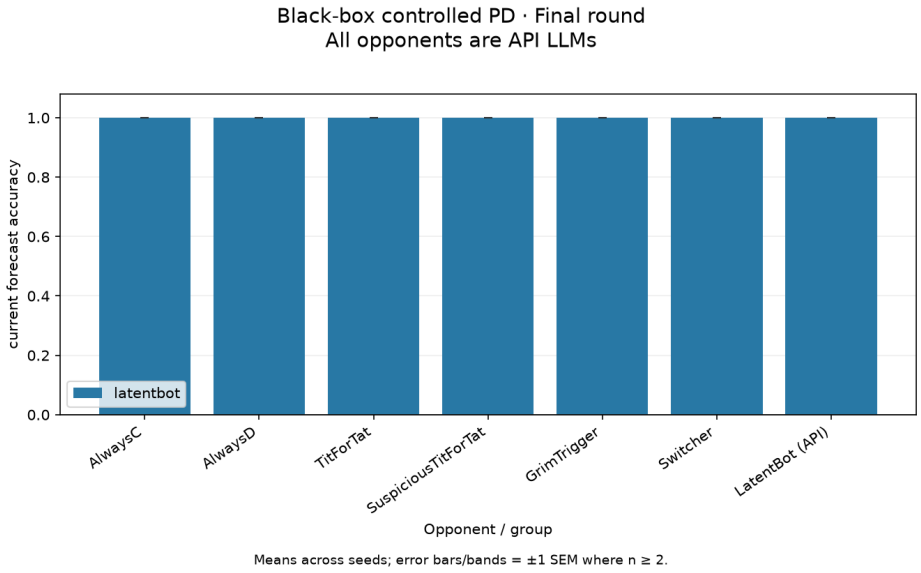


Figure 11: Current-action forecast accuracy on the final round only. Every final API action is classified correctly in all five seeds and all seven opponent conditions.

collection, so they test within-profile generalization rather than generalization to unseen opponent instructions. Accuracy remains 1.0 when evaluation is restricted to round 20 (Figure 11), which matters because the acting policy must distinguish predicting the opponent from valuing future retaliation: the current-action forecast is still used, but the reaction forecast is correctly excluded when no next round exists.

Finite-horizon behavior. Because defection strictly maximizes immediate Prisoner's Dilemma payoff for either predicted opponent action, removing the continuation term makes D the appropriate terminal action. LATENTBOT defects in round 20 in every condition. Figure 12 shows the resulting payoff: it receives 5 when the opponent cooperates (Always Cooperate, Tit-for-Tat, Suspicious Tit-

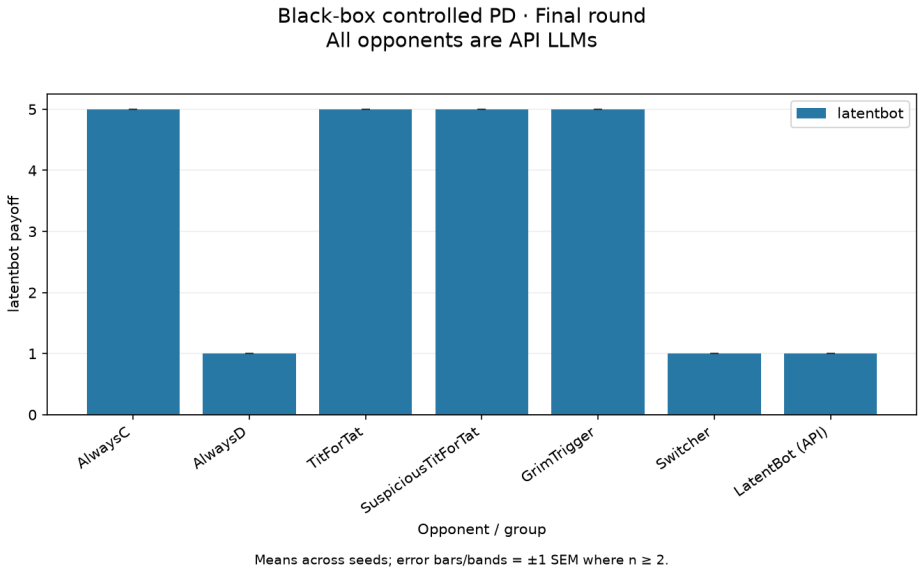


Figure 12: LATENTBOT's payoff in round 20. The continuation value is disabled on this round, and LATENTBOT defects against every opponent.

for-Tat, and Grim Trigger) and 1 when the opponent defects (Always Defect, Switcher, and the API LATENTBOT).

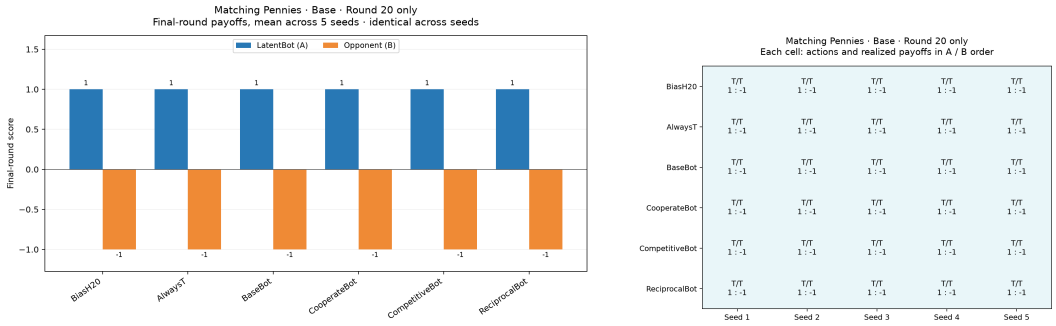


Figure 13: Matching Pennies at round 20. Left: mean final-round payoff. Right: action/payoff profile for every opponent and seed.

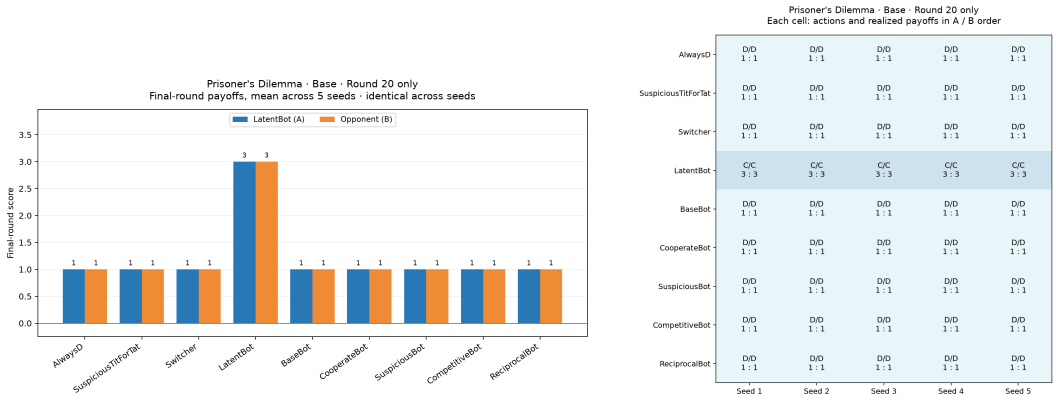


Figure 14: Prisoner's Dilemma at round 20. Mutual cooperation appears only in LATENTBOT self-play; all other displayed terminal profiles are mutual defection.

D FINAL-ROUND EVALUATION: PER-GAME DETAIL

This appendix expands Section 4.3. In Figures 13–18, the score panels show the mean final-round payoff over five seeds, while the outcome grids show the action and payoff realized for every opponent–seed pair.

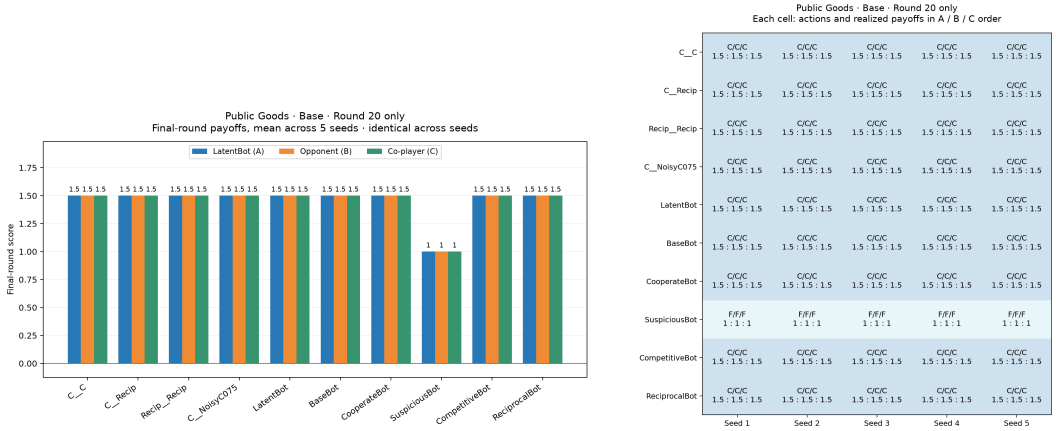


Figure 15: Public Goods at round 20. Left: mean payoff for LATENTBOT and both opponents. Right: three-player action/payoff profiles across five seeds.

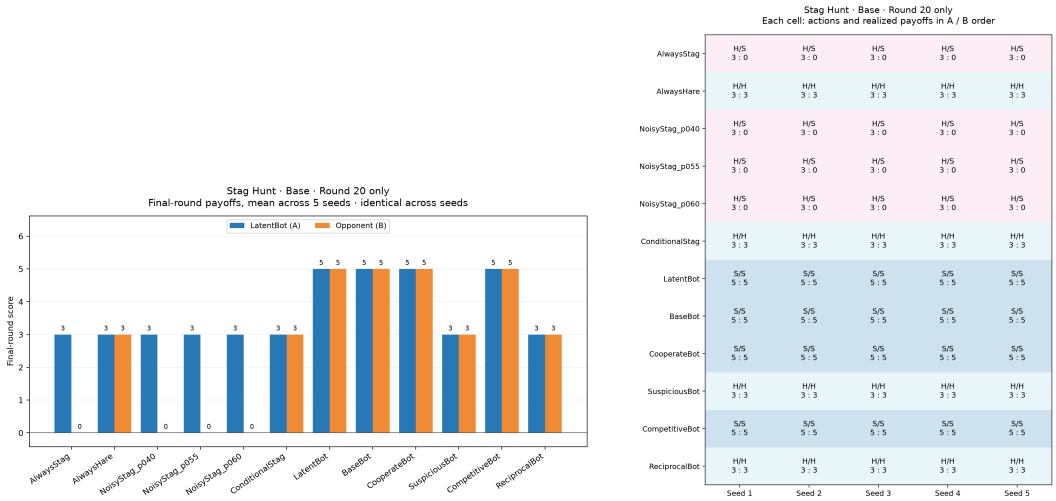


Figure 16: Stag Hunt at round 20. Efficient stag coordination and safe hare play are selected according to the opponent.

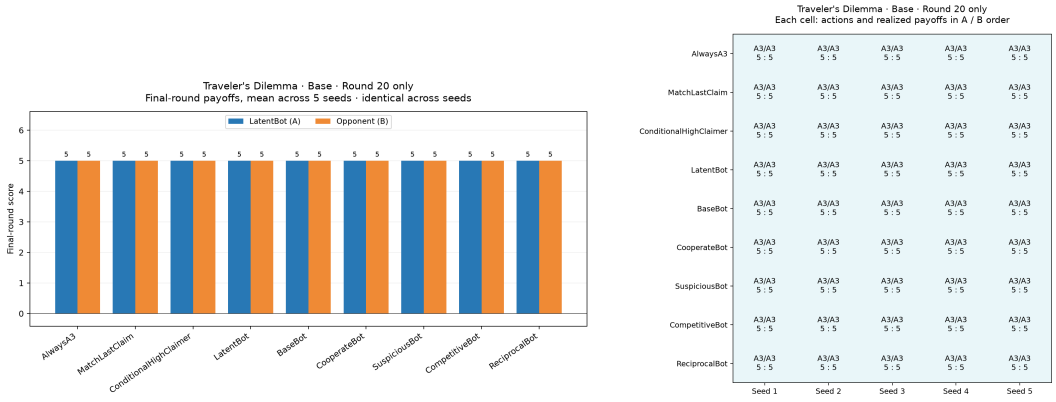


Figure 17: Traveler's Dilemma at round 20. All displayed matches converge to the highest-claim action and an equal payoff of 5:5.

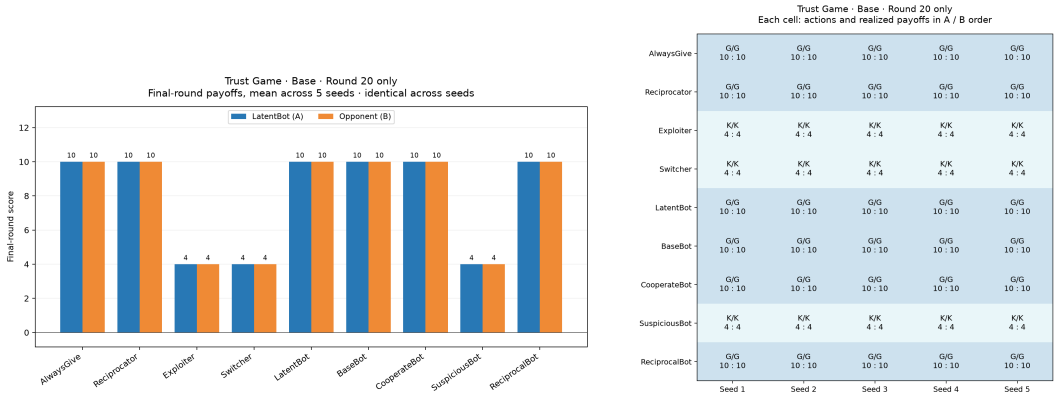


Figure 18: Trust at round 20. The agent gives to cooperative partners and keeps against exploitative, switching, or suspicious partners.