POKAITRAINER: SCALING EQUILIBRIUM SEARCH TO COMPETITIVE POKÉMON VGC

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision-time equilibrium search carried poker to superhuman play, but it has so far relied on tractable subgames: a handful of actions per decision, chance confined to card deals, one player moving at a time. Competitive Pokémon in its official doubles format (VGC) breaks all three assumptions at once. Both players act simultaneously from joint menus in the hundreds, each joint action resolves to hundreds of stochastic outcomes, and the opponent's reserves and stat allocations are hidden. No prior Pokémon agent performs equilibrium search, and whether it scales to this regime was open; we show that it does, and report what it took. PokaiEngine, our Rust battle engine, enumerates a joint action's full weighted outcome distribution in one pass, at $\sim$99% parity with Pokémon Showdown and a fraction of the cost of sampling it. PokaiTrainer adapts Student of Games to this scale and trains it by self-play over hundreds of human teams. Each decision is solved by counterfactual regret minimization as a Bayesian matrix game over public belief states, subgames grow under an explicit compute budget, and value targets are harvested from the interior of every solve and grounded by realized outcomes. The strength is in the search. The network's policy alone loses even to a shallow heuristic search. PokaiTrainer is, to our knowledge, the first VGC agent rated on the live Showdown ladder. Under open team sheets it wins 59% of 150 best-of-three sets against a human field averaging $\sim$1320 Elo, holds a 1350–1400 Elo band, and at its peak reached 1492 Elo, entering the format's top 500.

1 INTRODUCTION

Equilibrium search has produced landmark results in imperfect-information games, carrying poker from competitive to superhuman play (Zinkevich et al., 2007; Moravčík et al., 2017; Brown & Sandholm, 2018; 2019a). Its modern form, exemplified by ReBeL (Brown et al., 2020) and Student of Games (Schmid et al., 2023), solves a depth-limited subgame rooted at the current public belief state, with a learned value function at the leaves and self-play training. The recipe has so far relied on tractable subgames: poker offers a handful of actions per decision and confines chance to card deals, so ReBeL can enumerate its subgames exhaustively. Even Student of Games, growing its subgames incrementally, branches on a single player's move at a time. Equilibrium search also reached human level in Diplomacy (Gray et al., 2021; Meta Fundamental AI Research Diplomacy Team (FAIR) et al., 2022), whose turns are simultaneous, but the game is deterministic and the search one step deep over a few dozen sampled actions.

Competitive Pokémon Battling, in its official doubles format (VGC), breaks that tractability on every axis. Turns are simultaneous, with hundreds to thousands of joint actions, and each joint action resolves to hundreds of possible stochastic outcomes. Prior Pokémon agents sidestep this: whether heuristic or search-based, LLM-driven, or learned by imitation and RL (Lee & Togelius, 2017; Hu et al., 2024; Karten et al., 2025; Grigsby et al., 2025; Angliss et al., 2026), none performs equilibrium search at decision time (Section 6). Whether equilibrium search scales to this regime has been an open question, and we report what answering it took.

To handle the outcome stochasticity of Pokémon, we built **PokaiEngine**, a Rust battle engine that enumerates the chance outcomes of a joint action, returning its weighted outcome distribution in one pass. It reaches $\sim$99% behavioral parity with Pokémon Showdown (Luo & contributors, 2011),

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.

simulates single timelines $26\times$ faster, and recovers an outcome distribution more accurately than 256 Showdown replays at $1/500$th of their cost (Section 3).

We then train **PokaiTrainer**, a search agent in the Student of Games (Schmid et al., 2023) mold: CFR-based subgame solving over public belief states with value-network leaves, the subgame grown incrementally under an explicit compute budget, trained by self-play. Decisions are solved as simultaneous-move Bayesian matrix games with enumerated chance nodes; expansion is priced by compute cost rather than node count; hidden stat spreads are inferred by re-branching observed turns through the engine itself; and supervision is harvested from the interior of every solve, grounded by realized outcomes, and extended by hypothetical augmentation to positions self-play under-visits.

The trained agent wins 59% of 150 best-of-three sets on the live Showdown ladder against a human field averaging $\sim$1320 Elo, holding a 1350–1400 Elo band and at its peak entering the format's top 500 (Section 5.4). Search carries the strength, as the policy network alone loses even to a shallow material-heuristic search. The equilibrium makes the agent robust to an informed best response (Section 5.2). Training returns are now steeply diminishing, and the defect that persists is the value network's early-game optimism, which deeper solves mask rather than fix.

We evaluate under Open Team Sheets and leave closed sheets, where the opponent's unrevealed sets must be inferred, as future work. We claim no algorithmic novelty beyond the adaptations above. The contributions are the system, the empirical findings on what this scale demands, and a mapped frontier of what remains hard. We intend to release the engine and trainer with the live-ladder client removed, and to withhold checkpoints while their format is in play (Ethics Statement).

2 Background

2.1 Competitive Pokémon VGC

VGC is the official competitive Pokémon format: a **two-player zero-sum (2p0s)** game in which each player brings a team of 6 customized Pokémon. A battle opens with *team preview*: both teams are revealed, then each player secretly selects 4 of their 6, leading with two on the field and holding two in reserve. Each turn both players simultaneously choose a move or a switch for each on-field Pokémon; actions resolve in order of priority and speed, switches first, and moves deal damage, build indirect advantage (stat boosts, disruption, field effects), or pivot the user out mid-turn for a replacement. Fainted Pokémon are replaced from reserves at the end of the turn, and a player with none left loses. The following properties make this game challenging:

Simultaneous moves. Both players commit before either action resolves, so every turn is a matrix game with genuine rock–paper–scissors structure: *Protect* blocks attacks aimed at the user but wastes the action if the opponent targets elsewhere, and switching may dodge a predicted hit but invites a punish. Deterministic play is exploitable, so optimal play requires mixing.

Imperfect information. Each Pokémon is configured at team-building time: a species, an ability, a held item, four moves, and a stat allocation (a nature and a budget of effort values) that tunes its speed, power, and bulk. *Closed Team Sheets* hide everything beyond the six species; official tournament play and the Showdown best-of-three (Bo3) ladder we evaluate on use *Open Team Sheets* (OTS), which reveal everything but the stat allocations. OTS still leaves three sources of hidden information: the stat spreads, which shift speed order and knock-out thresholds; which two of the four non-leads were brought; and the opponent's live commitment on every simultaneous turn.

Stochasticity. Even a basic damaging move draws damage from 16 uniform multipliers and can miss, critically strike, or trigger a secondary effect. Up to four moves resolve per turn, compounding to the outcome counts of Table 1 even in the low-variance lines competitive play favors.

Mechanical density. Each format admits hundreds of species, moves, abilities, and items, plus a once-per-battle gimmick (Mega Evolution in Reg-MB). Simulation is solved (Luo & contributors, 2011), so the burden falls on learning, where interaction-specific knowledge competes with strategic knowledge for capacity, motivating search that computes it on demand.

	Chess / Go	HUNL poker	Scotland Yard	No-press Diplomacy	VGC doubles
Simultaneous actors	—	—	—	7 powers	2 players × 2 slots
Hidden information	—	$\sim 10^3$ hands	$\leq$199 stations	—	15 brings × $\sim 10^7$ spreads per Pokémon
Chance	—	card deals only	—	—	10^2–10^3 outcomes per joint action
Actions per decision	∼35 / ∼250	a few bet sizes	∼10	up to $\sim 10^{24}$ per power; a few dozen searched	$\sim 10^2$ per player, $\sim 10^4$ joint
Horizon	∼80 / ∼150 plies	$\leq$4 bet rounds	24 rounds	open-ended	5–20 turns

Table 1: VGC compared to the domains decision-time equilibrium search has mastered: the four of Student of Games (Schmid et al., 2023) and no-press Diplomacy (Gray et al., 2021). The $\sim 10^7$ spread grid matters less than its size suggests (Section 4.4).

2.2 Equilibrium Search in Imperfect-Information Games

The properties above make a VGC battle a two-player zero-sum extensive-form game with simultaneous moves, imperfect information, and chance. Strong human players adapt over a few games and beat the VGC-Bench agent (Angliss et al., 2026), and in our experience the four an agent brings is the first thing opponents read in a best-of-three. In this class a Nash equilibrium is unexploitable, so we target approximate equilibrium play with the toolkit developed in poker: counterfactual regret minimization (CFR) (Zinkevich et al., 2007; Bowling et al., 2015), combined with decision-time search that solves a *subgame* rooted at the current *public belief state* (PBS)—the public history plus each player's distribution over private states—(Brown & Sandholm, 2017; 2018; Moravčík et al., 2017; Brown & Sandholm, 2019a), cut off at a fixed depth by a learned *counterfactual value network* returning one leaf value per private state (Moravčík et al., 2017). ReBeL (Brown et al., 2020) closed the training loop, learning the value network from the subgames solved during self-play. Student of Games (SoG) (Schmid et al., 2023) extends the recipe to public trees too large to build up front—growing-tree CFR (GT-CFR) interleaves regret updates with expansions chosen by a walk mixing PUCT with CFR's average policy—and supplements main-line targets with re-solved leaf *queries*. As Table 1 shows, each of these branches on one player's move at a time; Section 4 is what the recipe needs when it does not.

3 The PokaiEngine

A standard Pokémon engine, both the games themselves and Showdown (Luo & contributors, 2011), advances a battle down a single timeline of chance rolls. What matters about a joint action for evaluating a decision, however, is its outcome distribution, which a one-path engine exposes only by replaying the turn many times, and since chance compounds across the moves of a turn, even moderately likely outcomes get missed at practical sample counts.

PokaiEngine, our Rust battle engine, instead supports chance-based forking natively. The same code runs in two modes. At each chance event, `Compute` mode rolls the RNG and continues down one timeline, while `Search` mode forks the state and carries every branch forward with its probability weight, as Figure 1 traces for one attack. Resolving a turn in `Search` mode therefore returns the joint action's full weighted outcome distribution in a single pass, with no sampling variance.[1]

3.1 Optimizations

Forked naively, a turn multiplies out fast (16 damage rolls, an accuracy, a critical-hit, and a secondary-effect roll per move, up to four moves). Two optimizations keep enumeration cheap (details in Appendix A.2). *Representing outcomes compactly*: branching that changes numbers without changing

[1] In the reported run, branches below a 2% probability floor are pruned and at most eight are kept per joint action, retaining 97% of the outcome mass on the probe of Table 2, and identical states are merged.

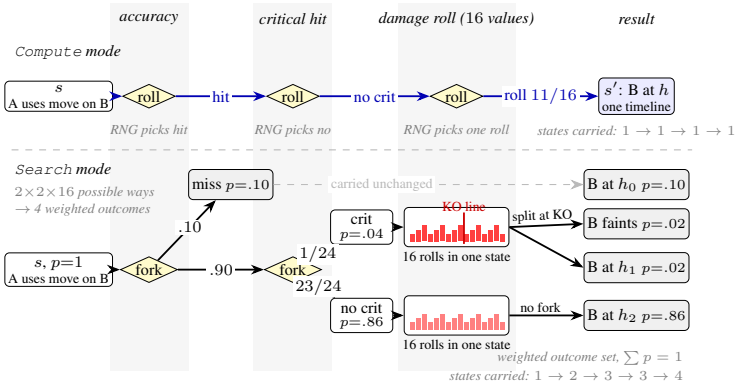

Figure 1: One attack's three chance events under the two engine modes. `Compute` rolls each event and continues a single timeline. `Search` forks the state and carries every branch, holding the 16 damage rolls inside one state as an HP distribution split only where it straddles a knock-out.

Method	Mean time (ms)	Mass covered	Mass misallocated
`Search`, exhaustive	9.9	100%	0% (reference)
`Search`, 32-branch cap	1.0	99.5%	0.7%
`Compute` / Showdown sampling, N=16	1.3 / 36.0	94.3% / 93.0%	10.0% / 10.2%
`Compute` / Showdown sampling, N=64	4.9 / 132.0	98.4% / 98.2%	4.8% / 4.8%
`Compute` / Showdown sampling, N=256	19.4 / 509.0	99.7% / 99.4%	2.8% / 2.6%

Table 2: Recovering the outcome distribution of one joint action: `Search`-mode enumeration versus sampling, averaged over 500 random openings with random joint move actions. Outcomes are distinct when they differ in any end-of-turn discrete features. Coverage is the reference outcome mass the method observed, and misallocation is the total-variation distance from the reference weights. A 32-branch cap is simultaneously faster and more accurate than sampling at any tested budget.

behavior stays inside a single state, as Figure 1 shows for HP, and hidden stat spreads can optionally ride the same way, assuming **stat spreads of different Pokémon are independent**. A fork is spent only where outcomes diverge in kind rather than degree, such as a damage roll that straddles a knock-out or an uncertain speed order. *Sharing work across branches*: the closely related forked states of a turn advance together, batched on the attributes move resolution can share.

3.2 Validation

We test 500 random openings with random joint actions, resolved exhaustively in `Search` mode as a reference distribution, then replayed 256 times by Showdown and `Compute` mode (Table 2).

Correctness: the exhaustive enumeration should contain every outcome, so any sampled output landing outside it is a bug. The first pass mismatched 6 of 500 turns and exposed four bugs, since fixed, and that first-pass $\sim$99% is the parity figure we quote (Appendix A.3). Two regression suites guard it, more than 600 synthetic scenarios and over 500 recorded human ladder battles replayed turn by turn.

Performance: as a plain simulator, `Compute` mode resolves a median turn in 0.08 ms against Showdown's 2.0 ms, a 26$\times$ gap (single Node process; neither side's time includes transport).

Coverage: enumeration yields a median of 9 and up to 18,176 weighted branches per joint action (Table 2), and sampling's $1/\sqrt{N}$ error is widest where it matters. On the branchiest probe (960 distinct outcomes), 256 samples miss 17% of the mass.

4 Method

Our method began as an adaptation of ReBeL (Brown et al., 2020), which under the pressure of VGC grew into something close to Student of Games (Schmid et al., 2023): CFR over public belief states

with a counterfactual value network at the leaves, the subgame grown incrementally under a compute budget, and the networks retrained from self-play.VGC adds two axes neither poker nor Go has, simultaneous joint actions and wide chance nodes, and is far more data-hungry per unit of search; our departures from SoG, summarized in Table 4, are adaptations to those facts, not a new algorithm.

4.1 Problem Formulation

An OTS VGC battle is a 2p0s game with simultaneous moves, imperfect information, and chance (Section 2.2). We write 1 for the seat being solved for and 2 for the opponent.

Worlds and beliefs. The *public state* s is everything derivable from the battle log (species and formes, HP in coarse buckets, status, stages, field). A *world* $w \in \mathcal{W}(s)$ is one hypothesis about which four of their six the opponent brought: $|\mathcal{W}(s_{\text{preview}})| = \binom{6}{4} = 15$ at team preview, dropping to $\leq \binom{4}{2} = 6$ once the lead pair is revealed. A *public belief state* $\beta = (s, b)$ pairs s with the solver's posterior $b \in \Delta(\mathcal{W}(s))$, seeded by the team-preview solve (Section 4.2). Each opponent Pokémon j additionally hides a stat spread ξ_j, carried not as extra worlds but inside every world as a mixture over a candidate set $\mathcal{S}_j$ with weights $q_j \in \Delta(\mathcal{S}_j)$ (Section 4.4). Information is one-sided: seat 1's configuration is treated as known to both players, conceding the opponent our hidden spreads and reserves. Conceding information makes the solved game harder for us, never easier, so σ_1 is conservative, though it never exploits what the opponent does not know.

Dynamics and strategies. At a decision node each seat picks $a_i \in \mathcal{A}_i$ simultaneously. The engine resolves the joint action $a = (a_1, a_2)$ in world w to an enumerated weighted batch of successors, $\mathcal{T}(s, w, a) = \{(s'_k, w'_k, p_k)\}_k$ with $\sum_k p_k = 1$. Seat 1 plays one strategy $\sigma_1 \in \Delta(\mathcal{A}_1)$; seat 2, whose private information is w, plays one per world, $\sigma_2^w \in \Delta(\mathcal{A}_2(w))$. The value network $v_\theta(\beta) \in \mathbb{R}^{|\mathcal{W}(s)|}$ returns one value per world, as in DeepStack's and SoG's counterfactual value networks (Moravčík et al., 2017; Schmid et al., 2023); $z \in \{-1, 0, +1\}$ is a realized game outcome.

4.2 Subgame Solving

We solve every decision the agent faces, and every decision node inside a subgame, as a Bayesian matrix game. Chance first draws the world $w \sim b$, which seat 2 observes and seat 1 does not; seat 1 plays σ_1, seat 2 plays σ_2^w, and chance resolves the joint action through $\mathcal{T}$. The per-world payoff is

$$U_w(a_1, a_2) = \sum_{(s', w', p) \in \mathcal{T}(s, w, a)} p\, V(\beta'), \qquad V(\beta_{\text{leaf}}) = \langle b,\, v_\theta(\beta_{\text{leaf}}) \rangle, \tag{1}$$

where β' carries the belief the strategies above it induce, and the node's value is

$$V(\beta) = \max_{\sigma_1} \min_{\{\sigma_2^w\}_w} \sum_w b(w) \sum_{a_1, a_2} \sigma_1(a_1)\, \sigma_2^w(a_2)\, U_w(a_1, a_2), \tag{2}$$

solved by alternating-update linear CFR (Brown & Sandholm, 2019b) with one regret table for seat 1 and one per world for seat 2; the average profile $\bar{\sigma}$ converges toward a Nash equilibrium of the subgame. The four decision types of a battle differ only in menus and in who holds private information (catalogued in Appendix B.2, sketched in Figure 4); the same solver runs all of them.

Team preview. Both rosters are public and nothing is hidden yet, so Equation (2) collapses to a plain matrix game over the $\binom{6}{4}\binom{4}{2} = 90$ (bring, leads) options per side, each cell opening the battle at a known world and solved as a turn node under the same budgeted expansion (Section 4.3). The solve is used twice: $\bar{\sigma}_1$ is sampled for our bring, and $\bar{\sigma}_2$'s bring marginal, blended with a corpus bring prior (Section 4.4) and conditioned on the revealed leads, becomes the initial belief b_0: this is where the hidden information of every later β is created.

Turn. A turn node's menus are the joint action choices of the two active slots (Section 2.1), on the order of 10^2 per side, so both menus are shortlisted by policy weighting before the matrix is solved. Our menu is cut to the top k_1 by the policy prior, with the root cap configurable up to the whole menu. The opponent's joint options are enumerated per world from that world's bench, aligned across worlds by identity, scored by the belief-weighted prior $\sum_w b(w)\, \pi_2^w(a_2)$, and capped at k_2, with slices of the cap reserved for switch and gimmick joints so those lines survive a low prior (Appendix B.2).

Switches. After end-of-turn faints both sides pick replacements blind over at most 2×2 menus per world. A pivot move instead pauses the turn with one seat choosing a replacement, and the chooser optimizes against a posterior that hypothesizes the other side's unseen committed actions, weighted by the pre-turn average profile and resolved from the paused state (Appendix B.2).

4.3 Subgame Expansion

The subgame is grown incrementally, interleaving expansion with regret updates as in GT-CFR (Schmid et al., 2023). We choose the continuation to expand next by SoG's selection rule adapted to the matrix game, a walk that mixes a PUCT argmax with the current average policy. From the root, sample $w \sim b$, then each seat's action from

$$\tfrac{1}{2}\,\mathrm{onehot}\Big(\arg\max_a\; Q_i(a) + c\,\pi_i(a)\,\frac{\sqrt{N}}{1+N_a}\Big) + \tfrac{1}{2}\,\bar{\sigma}_i, \tag{3}$$

with Q_i the seat's counterfactual action values, π_i the policy prior, and N, N_a visit counts, then sample the chance outcome from $\mathcal{T}$, descend, and admit the first unexpanded continuation.

Unlike GT-CFR, we budget compute rather than node count. A turn continuation costs 1 and a mid-turn or forced-switch decision $c_{\mathrm{sw}} \ll 1$, each scaled by belief width $|\mathcal{W}|^{\rho}$ since engine and network work grow with a node's worlds. Expansion halts when admitted cost reaches the budget B (or its wall-clock equivalent at deployment), and in practice it binds long before any depth would.

4.4 Hidden Stat Spreads

We model each opponent Pokémon's hidden stat spread as a mixture over a small candidate menu. The menu $\mathcal{S}_j$ and its prior q_j^0 come from a team corpus, conditioned on the nature that open sheets reveal, with a reserved slice of base-stat archetype spreads so an off-meta build stays representable. We rely on the engine for inference: after each turn the completed joint action is re-branched per world and per candidate, and a candidate is reweighted by the mass the engine's own transition puts on the observed public outcome and move order, so damage rolls and speed order discriminate through the exact rules with no hand-derived formula to keep in sync. Posteriors persist across turns and the games of a Bo3 series (Appendix C).

4.5 Encoding and Networks

We encode a position's publicly observable information as features, along with two belief summaries, the weighted quantiles of each opponent Pokémon's stat-spread mixture and the turn phase with ours and hypothesized opponent's queued commitments. One transformer trunk feeds both heads. The policy head reads out the prior π over a seat's joint two-slot menu, scoring the same joint actions the solver's matrices are built from, and the value head the per-world vector $v_\theta(\beta)$. Appendix B.3 gives the exact token structure, and Section 5.1 the sizes.

4.6 Training

We initialize our networks by Behavior Cloning on public Showdown replays, similar to VGC Bench (Angliss et al., 2026). Replays carry no world labels, so cloning trains the policy head and a *scalar* value head. The per-world head is minted from the clone by tiling the scalar head's output across bring slots, so every world opens at the clone's prediction and self-play differentiates them.

We then trained in rounds, where fleets of self-play games run the search at every decision, every solve is harvested into training rows, then the networks are retrained and the next round starts from them. The value target for world slot w of a row is:

$$y_w = \begin{cases} (1-\lambda)\,V_w + \lambda z & w = w^\star \text{ and the row is grounded} \\ V_w & \text{otherwise,} \end{cases} \tag{4}$$

where V is the solved backup and $w^\star$ the realized world. We have 3 kinds of rows we train with:

Grounding. Pure solver targets bootstraps through the leaf network, so a drifted network can confirm its own bias, while the realized outcome z is unbiased but noisy. Equation (4) mixes them on the slot of the world played. Counterfactual worlds and interior nodes keep pure solver targets.

Interior supervision. Each solve has already computed values for its interior nodes, so we also train on them: up to K per decision, sampled $\propto$ reach under $\bar{\sigma}$, each row weighted by its normalized reach and discounted by how much of its backup priced through the leaf network rather than terminating in-tree (sampling and weighting details in Appendix B.4).

Hypothetical augmentation. Self-play under-visits lopsided and unusual positions, so we manufacture them. A fixed number of times per game, the live battle is forked at a decision boundary, a small set of edits is applied (HP, status, field, stat stages, or a hidden stat spread redrawn from the full usage prior), and the edited game is played out by the same search, contributing rows grounded by its own outcome. This is similar in spirit to SoG's re-solved leaf queries.

5 Experiments

5.1 Setup

Format and teams. All results are in Pokémon Champions, VGC 2026 Regulation M-B doubles under OTS.[2] Self-play uses single game rewards, while online Showdown ladder plays best-of-three (Bo3) series.[3] Self-play draws teams from 663 unique public team pastes scraped from the community-maintained VGCPastes repository (VGCPastes, 2026). Evaluation uses a hand-picked pool of 16 teams spanning the game's primary archetypes ("eval16").

Agent. The networks instantiate Section 4.5 with a 256-wide, 6-layer, 8-head trunk (5.4M parameters). It is initialized by behavior cloning on 122,804 human ladder replays (2.56M decisions). Search follows Section 4: up to 15 worlds per solve, budgeted PUCT expansion, interior menus of 16×16 joint actions, the opponent's root menu capped at 16, and one hypothetical fork per game. The expansion budget B follows the ramp in Section 5.2; Appendix E.1 lists the remaining constants.

Evaluation protocol. An *eval16* cell plays two agents over ordered pairs of the 16 eval teams with a fixed seed schedule: 256 games (each matchup 1 game, 95% Wilson interval ±6.1pp), and for certain close calls pooled to 1024 games (each matchup 4 games, ±3.1pp). Checkpoints play argmax of $\bar{\sigma}$ so that they are compared at their means (eval opponents do not adapt, so a deterministic policy cannot be exploited there), while ladder play samples at $T=0.5$. Three fixed opponents recur: D1SW15, our strongest earlier-generation agent (Appendix E.2); GREEDY, a material-heuristic depth-one belief search; and π-ONLY, a network's policy head without search.

Compute. Self-play runs on ten cloud slices of 48 CPU game workers, each sharing one L40S inference server; training uses one A100-40G. The 32-round run below costs roughly 250 L40S-hours of self-play and $\sim$15 A100-hours of training; the ladder runs on one local RTX 4090.

5.2 The Training Run

The reported run is 32 rounds of the loop of Section 4, ramping B from 8 to 32 and games per round from 1k to 3k (schedule in Appendix E.1). By the matched-lineage read of Figure 2, the back half of the run bought $+5$pp over round 20.

Diminishing returns at $r=32$. Training progress has not stopped, but its price has collapsed. Eight further rounds at $B=16$ end at parity with round 32 head-to-head (50.8%, 1,024 games), and a $B=32$ continuation arm does no better (48.1%). A $B=64$ extension with deepened child solves, at $1.6\times$ the cost per round, peaks at 54.5% in its fourth round (1,024 games) but sits at 49–51% in rounds three, six, and nine. Round 32 remains the agent evaluated below.

The plateau leaves the network's early-game optimism in place. It shows on self-play roots, certified positions, and ladder solves, and neither further rounds nor deeper solves move it (Appendix E.6).

[2]The ability Illusion is too rare in VGC to support, so the agent forfeits the occasional game against it.

[3]Showdown format `gen9championsvgc2026regmbbo3`.

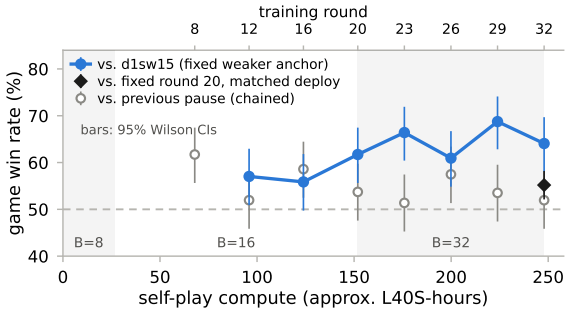

Figure 2: Checkpoint strength by three references: the fixed older agent D1SW15 (inflates as the lineage learns to exploit it), the previous pause checkpoint (leans high when chained), and fixed round 20 at matched deployment search (the read we use).

Deployment variant	Win% [95% CI]	p_{50}
budget B=128	50.8 [44.7, 56.8]	2
interior menus 24×24	**42.6** [36.7, 48.7]	2
B=64, own 32	53.7 [47.6, 59.8]	2
B=128, own 32	56.6 [50.5, 62.6]	3
B=128, own 64	52.0 [45.8, 58.0]	3
B=256, own 32	59.0 [52.9, 64.8]	3
B=128, own 32, iters 200	**63.7** [57.6, 69.3]	3
B=256, own 32, iters 200	59.6 [53.5, 65.5]	3

Table 3: Round-32 network, variant vs. native deployment (B=32, 100 child CFR iterations), eval16 cells. Own k: child menus widened to k; iters: child CFR iterations; p_{50}: median solve depth in turns.

Search carries the agent. GREEDY beats every raw policy head we trained (71% pooled) and a PPO self-play policy built with the VGC-Bench recipe (Angliss et al., 2026) at 60.5% (Appendix E.3). Every search agent beats its own policy head, and our round-one network under search beats that PPO policy 73–27. Self-play widens the lead, as the behavior-cloned parent wins only 19.5% against round 32 under the same search. Almost every training-side lever lands within an eval16 cell's noise (Appendix E.4). Compute and the encoder moved strength, not target construction, and the hidden-stat belief summary was worth +8–12pp (Appendix E.2).

Equilibrium and exploitability. Decoupled UCT (Shafiei et al., 2009) computes no equilibrium. Run in place of CFR at every solve, with the same network, tree, and menus and ten times the iterations, it plays the native agent at parity (51.7%, 2,048 games). An informed local best response (Lisý & Bowling, 2017) that sees the agent's strategy and the true world separates the two. It wins 83.6% against the decoupled-UCT mix and 49.6% against the CFR mix (Appendix E.5). Search carries the strength, and only an adapting opponent tests the equilibrium.

5.3 Scaling Search at Deployment

Each cell of Table 3 plays the round-32 checkpoint against itself, one side at the native training shape and one at a variant. Three regularities organize the table. *Budget alone saturates*: the PUCT walk descends through the product of two seat policies and a chance factor, so depth grows only logarithmically in budget, and the extra spend buys width the value network cannot use. *Width alone is harmful*: wider interior menus spread the fixed child iterations over a 2.25× cell area and dilute per-cell CFR convergence. *What pays is deepening and sharpening child solves*: budget times a wider *own* child menu converts spend into depth (solves reaching ≥3 turns: 18% → 56%), and doubling child CFR iterations on that shape is the largest single lever. The two stack, though the shared anchor hides it. The B=256 combination reads worse than its B=128 parent yet beats it head-to-head (56.6%), so we settle close calls by direct play. The winning shapes cost 4–6× the native wall clock.

5.4 Human Evaluation: Showdown Ladder

The final agent played two 75-set runs on the live Showdown Bo3 ladder, each from a fresh account and one set at a time, against a field averaging ∼1320 Elo (max 1620). One run used its native training shape (median 2.7 s per decision), the other the head-to-head winner of Table 3 (B=256, own 32, iters 200) under a window-filling time budget (median 23 s). The native run won **44–31 sets (58.7%)**, taking a fresh account from 1000 Elo to a peak of 1492, briefly inside the format's top 500. The scaled run won **45–30 sets (60.0%)**.The game records are 95–84 (53.1%) and 98–77 (56.0%), and the pooled sets are 89–61 (95% Wilson interval 51–67%), though a ladder's baseline is the field's rating, not 50%. The eval16 builds are mid-tier (median Bradley–Terry rank 217 of 648), and a third run with corpus-top teams won 56.3% of 142 sets (Appendix D), so stronger builds did not help.

The peak is not the level. The native run fell back out of the top 500 within ten sets, and across both runs the accounts kept reverting to a band of **1350–1400 Elo**, with GXE (Showdown's estimated win rate against a random ladder opponent) of 72–74, which we report as the agent's strength. The scaled shape, 59.6% against the native shape internally (Table 3), shows no noticeable ladder edge, +2.9pp of games over 354 and a lower GXE (71.9 vs. 73.7). Win rate does not fall across a set, and in 83 opt-in private challenges (52 on challengers' own teams) the agent won 64% of games (Appendix D).

The recorded solves show that the opponent model's shortlist contains the opponent's realized *joint* action on 64% of observable turns against 97% per-slot marginal coverage, so what the 16-joint menu misses is correlations between the two slots, not action vocabulary. Widening the menu at matched cost buys no strength (Appendix E.4). And the solved root value is better calibrated than the network that seeds it (Brier 0.171 vs. 0.214), sharpening from 0.22 on turns 1–3 to 0.10 from turn 8, the defect of Section 5.2 observed live (Appendix E.6).

6 Related Works

The equilibrium-search lineage itself is covered in Section 2.2. From the AlphaGo lineage (Silver et al., 2016; 2018; Schrittwieser et al., 2020) we borrow, as SoG did, the outer loop in which search produces training targets and retrained networks strengthen the next round of search.

Pokémon agents. Early singles agents were scripted or searched hand-built engines (Stone, 2010; Lee & Togelius, 2017). The closest system to ours is Foul Play (Mariglia & contributors, 2019), a singles ladder bot whose Rust engine also branches a move pair into weighted outcomes. It runs an independent MCTS per sampled opponent team with a handcrafted evaluation, whereas we solve an equilibrium over a belief state and train value networks by self-play. Our engine is also much more sophisticated, carrying stat and damage uncertainty as in-state distributions forked only where a condition depends on them, whereas Foul Play fixes one spread per search and uses expected values. Recent agents are LLM-driven or learned from data: PokeLLMon (Hu et al., 2024) prompts an LLM with battle state, PokeChamp (Karten et al., 2025) adds minimax lookahead over LLM-sampled actions, Metamon (Grigsby et al., 2025) reaches human-level singles ladder play by offline RL on human replays and self-play data, and VGC-Bench (Angliss et al., 2026) provides a doubles benchmark with behavior-cloning, RL, and LLM baselines. None computes equilibria at decision time: PokeChamp takes $\arg\max_a \min_b$ over sampled shortlists, Foul Play's MCTS selects each side's action by independent UCB1, and the rest act directly from a policy. All but VGC-Bench target singles, and none is rated on the doubles ladder.

Simultaneous-move games. DeepNash (Perolat et al., 2022) reached expert-level Stratego, a turn-based imperfect-information game with a hidden initial deployment, model-free via regularized Nash dynamics with no search at decision time; whether search pays its way in games of this shape is the question we take up. Simultaneous moves also break standard MCTS: decoupled UCT selection, Foul Play's rule, can converge to exploitable strategies (Shafiei et al., 2009), which SM-MCTS variants with regret-matching selection repair (Lisý et al., 2013; Tak et al., 2014). We treat simultaneity, hidden information, and chance by one mechanism, the CFR solve over belief subgames, and the concern resurfaces only in the expansion order (Section 4.3).

7 Limitations and Conclusion

Limitations. Under closed team sheets the hidden space grows from stat spreads to entire builds, and inferring those from revealed events is unsolved in our framework. Team construction is out of scope. The reported run, a single seed, cost about 250 L40S-hours (Section 5.1), two to three orders of magnitude below the milestones our method descends from (Brown et al., 2020; Perolat et al., 2022; Schmid et al., 2023). The diminishing returns of Section 5.2 describe this scale, not the recipe's ceiling. Early-game value optimism is the one defect training did not move. We suspect label noise, as a realized outcome is one draw from hundreds of branches per turn (Appendix E.6).

Conclusion. Equilibrium search does scale to simultaneous moves, wide chance nodes, and hidden information at once. With an enumerating engine and a budgeted solver, the recipe that mastered poker holds 1350–1400 Elo against human VGC players.

Ethics Statement

Play against humans. The ladder results of Section 5.4 come from games on the public Showdown ladder against anonymous opponents, who were not told that they faced a program. Between July and September 2026 the agent played under two thousand ladder games across development and evaluation, 681 of them in the three reported runs. We did not seek advance authorization for this play and relied on three grounds. First, Showdown's published rules place no restriction on automated play. Second, in a December 2025 policy review of ladder bots (Smogon Forums, 2025), Showdown's stated concern was that they skew the usage statistics behind Smogon's tiers (fan-community-regulated formats), which does not apply to VGC, whose rules are set by the game's publisher. That review closed in April 2026 with technical limits on ladder bots rather than a rule change, and our runs came after it. Third, prior agents were evaluated on the same Showdown ladder (Hu et al., 2024; Karten et al., 2025; Grigsby et al., 2025; Mariglia & contributors, 2019). Each reported run used a fresh account and played one set at a time. The agent kept within the format's clock (it never lost a game on time in those runs) and forfeited rather than stalled the games it could not play (Section 5.1). Opponents' usernames appear in our logs but are not reported, and will be removed from any released logs. Every participant in the private challenge matches of Appendix D chose to challenge the agent and received, after each game, a recap of the agent's solved policy and its estimate of theirs.

Contact with Showdown. In August 2026 we opened a support ticket with Showdown's staff, describing the agent and its ladder rating and asking to discuss releasing its code and models. Staff replied that they did not oppose the release but would likely revise the site's policy on bot accounts in the near future. The following week we emailed the staff address a draft of this paper, raising the concern that released code would make ladder bots easier to build and asking how they would prefer we release it, and we have not received a reply.

Release. Our main concern is proficient bots flooding the public ladder, so we intend to release PokaiEngine and PokaiTrainer with the live-ladder client removed and to withhold trained checkpoints while their format is in play. Behavior cloning used publicly archived replays, and the usernames in them are not used in training.

Reproducibility Statement

Every mechanic the engine implements is checked against Pokémon Showdown by the parity harness of Section 3 and Appendix A.3, and every number in Section 5 is computed from logged games by the evaluation scripts. The hyperparameters, staging schedule, and run scale are listed in Appendix E.1. We intend to release PokaiEngine and PokaiTrainer once the live-ladder client has been removed from the shipped build. Release is not conditioned on Showdown's approval, though staff have said they do not oppose it (Ethics Statement). Trained checkpoints will not be released while the format they were trained on is in play, and we will revisit this when the format rotates.

AI Use Statement

In this work we used generative AI tools (Claude, Anthropic) for the following tasks with required disclosure: implementing methods, designing and providing feedback on experiments and evaluation methodology, proposing and refining hypotheses, cleaning and reformatting the replay dataset, and interpreting results. In particular, AI served as a brainstorming partner in designing the equilibrium-search method, while the engine design is the authors' own. We did not use them to generate synthetic data or to develop theoretical models, and the work involves no mathematical proofs, translation, or qualitative data analysis. We also used them for tasks with recommended disclosure: writing code, suggesting experimental parameters, creating figures, drafting and editing the text, searching and summarizing related literature, formatting references, and brainstorming.

We have reviewed all AI-assisted work. Code was reviewed by the authors and is exercised by the engine's parity harness against Pokémon Showdown (Section 3) and the project's test suites. Every reported number was recomputed from logged game artifacts, and literature summaries and citations were checked against the original sources. All text was reviewed and edited by the authors. We take

responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Cameron Angliss, Jiaxun Cui, Jiaheng Hu, Arrasy Rahman, and Peter Stone. VGC-Bench: Towards mastering diverse team strategies in competitive Pokémon. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2026.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems 30*, 2017.

Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890, 2019a.

Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1829–1836, 2019b.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Advances in Neural Information Processing Systems 33*, 2020.

Jonathan Gray, Adam Lerer, Anton Bakhtin, and Noam Brown. Human-level performance in no-press Diplomacy via equilibrium search. In *International Conference on Learning Representations*, 2021.

Jake Grigsby, Yuqi Xie, Justin Sasek, Steven Zheng, and Yuke Zhu. Human-level competitive Pokémon via scalable offline reinforcement learning with transformers. In *Reinforcement Learning Conference*, 2025.

Sihao Hu, Tiansheng Huang, and Ling Liu. PokéLLMon: A human-parity agent for Pokémon battles with large language models. *arXiv preprint arXiv:2402.01118*, 2024.

Seth Karten, Andy Luu Nguyen, and Chi Jin. PokéChamp: an expert-level minimax language agent. In *International Conference on Machine Learning*, 2025.

Scott Lee and Julian Togelius. Showdown AI competition. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 191–198, 2017.

Viliam Lisý and Michael Bowling. Equilibrium approximation quality of current no-limit poker bots. In *AAAI-17 Workshop on Computer Poker and Imperfect Information Games*, 2017.

Viliam Lisý, Vojtěch Kovařík, Marc Lanctot, and Branislav Bošanský. Convergence of Monte Carlo tree search in simultaneous move games. In *Advances in Neural Information Processing Systems 26*, 2013.

Guangcong Luo and contributors. Pokémon Showdown. https://pokemonshowdown.com, 2011. Open-source Pokémon battle simulator.

Patrick Mariglia and contributors. Foul Play. https://github.com/pmariglia/foul-play, 2019. Pokémon Showdown battle bot; accessed August 2026.

Meta Fundamental AI Research Diplomacy Team (FAIR), Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

Pokemon Champions Battle Data. Aggregated ranked-battle usage statistics. https://championsbattledata.com, 2026. Per-species moves, items, natures, and stat spreads; accessed August 2026.

Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G. Zacharias Holland, Elnaz Davoodi, Alden Christianson, and Michael Bowling. Student of Games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Mohammad Shafiei, Nathan Sturtevant, and Jonathan Schaeffer. Comparing UCT versus CFR in simultaneous games. In *IJCAI Workshop on General Game Playing (GIGA)*, 2009.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Smogon Forums. Ladder bots and usage-based tiering. https://www.smogon.com/forums/threads/ladder-bots-and-usage-based-tiering.3774656/, 2025. Policy Review thread, December 2025, marked Implemented April 2026; accessed September 2026.

David Stone. Technical Machine. http://doublewise.net/pokemon/, 2010. Pokémon battle AI; source at https://github.com/davidstone/technical-machine.

Mandy J. W. Tak, Marc Lanctot, and Mark H. M. Winands. Monte Carlo tree search variants for simultaneous move games. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 2014.

VGCPastes. VGCPastes Repository. https://docs.google.com/spreadsheets/d/1axlwmzPA49rYkqXh7zHvAtSP-TKbM0ijGYBPRflLSWw, 2026. Community-maintained repository of competitive VGC teams; accessed August 2026.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20*, 2007.

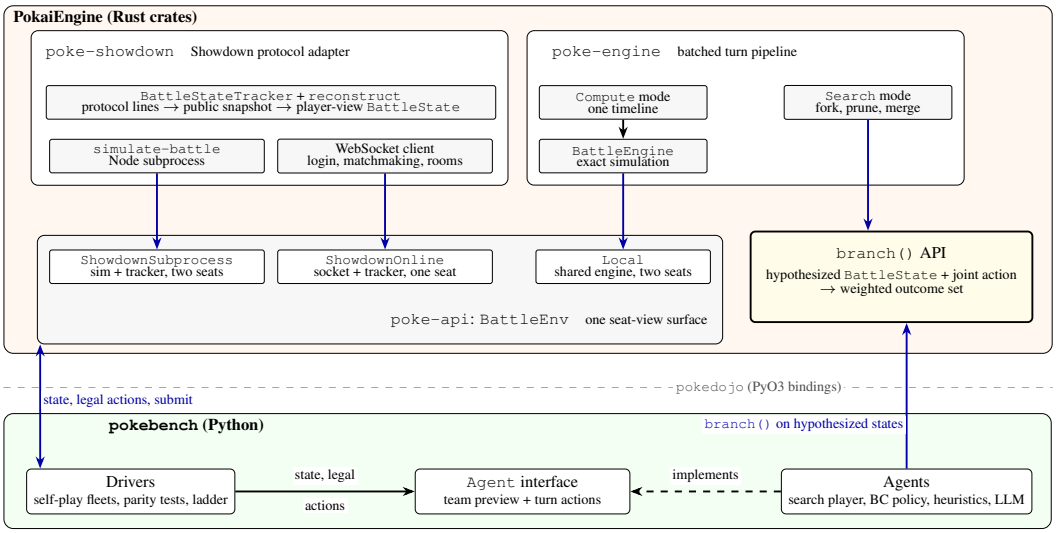

Figure 3: System architecture. The engine runs one rules code in `Compute` or `Search` mode, the latter exposed as the `branch()` API returning weighted outcome sets; the Showdown adapter reconstructs the same player-view `BattleState` the engine emits from protocol lines; and `BattleEnv` is one seat-view surface over all three backends. Python agents implement one `Agent` interface and run unchanged on any backend; the search player additionally calls `branch()` on hypothesized states.

A Engine Details

A.1 System Architecture and Backends

Figure 3 details the stack behind the shared battle abstraction of Section 3: local engine play, an offline Showdown process, and the live Showdown ladder as interchangeable backends under one seat-view surface.

A.2 Optimization Details

Representing outcomes compactly. A Pokémon's HP is carried as a distribution over the damage rolls it has absorbed, and an opponent Pokémon's hidden stat spread can be carried as an in-state distribution under the simplifying assumption that spreads of different Pokémon are statistically independent. In reality teammates are trained with one another in mind. We have not measured the strategic cost of ignoring that correlation, and accept it for the computation the compaction saves. When a later event does depend on position within an in-state distribution, such as an HP threshold that triggers an item or ability, or a speed comparison against an uncertain stat, the engine splits the state there, partitioning the distribution into the branches that behave differently.

Sharing work across branches. Shared damage modifiers are computed once per batch, while per-state work (status-dependent pre-move checks, damage against each state's stages and spreads) iterates over members. Batches are keyed on the attributes move resolution can share and that rarely diverge across a turn's forks (which Pokémon are alive, items, abilities, the resolved action sequence, the field state), while statuses, stat stages, and spread hypotheses stay per-member.

A.3 Parity Measurement

Behavioral parity is measured three ways, and only the figure we quote is out of sample. *Random-opening probe.* The 500 random openings of Table 2 were drawn once, and each was replayed 256 times by Showdown and checked against the exhaustive enumeration before any mismatch was examined. This first pass mismatched 6 of 500 turns, traced to four engine bugs, and its rate is the $\sim$99% we report. After the fixes the same probes pass 499 of 500, which we treat as a regression result rather than a parity estimate, since these probes found the bugs. *Scenario suite.* More than 600

Component	Follows	Departure for VGC
Belief states, CFR solve, CV net at leaves	ReBeL, SoG	simultaneous-move matrix nodes; exactly enumerated chance nodes (Section 4.2)
Incremental subgame growth	SoG (GT-CFR)	walk over a matrix game; admissions priced by compute cost, budget in cost units (Section 4.3)
Value targets	SoG (solver values + TD(1) rows)	per-row λ-mix, grounded on the realized world's slot only (Section 4.6)
Extra supervision	SoG (re-solved leaf queries)	interior nodes of the same solve, reach-sampled and bootstrap-weighted
Off-line coverage	SoG (recursive queries)	hypothetical augmentation, grounded by play

Table 4: Where PokaiTrainer follows its lineage and where it departs.

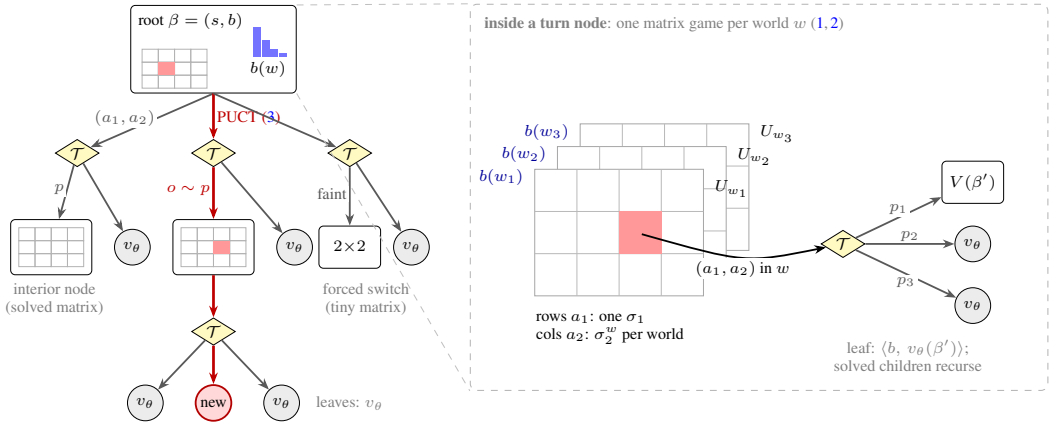

Figure 4: Anatomy of one budgeted solve. Left: the subgame rooted at the public belief state $\beta = (s, b)$; every decision node holds a matrix game, chance nodes are the engine's exactly enumerated outcome batches $\mathcal{T}$, leaves are priced by v_θ, and the red path is one PUCT expansion walk (Equation (3)) admitting a new continuation. Right, zooming into a turn node: one payoff matrix U_w per world, weighted by the belief b and solved jointly by CFR (Equations (1) and (2)); each cell resolves through one `Search`-mode engine pass of $\mathcal{T}$, and its successors either recurse or read the leaf value $\langle b, v_\theta \rangle$.

constructed positions, each isolating one mechanic, are played turn by turn on both simulators with their battle states asserted equal. They run in continuous integration, and all pass. *Human replays.* 527 recorded ladder battles from three regulations (199 each of Regulations M-A and M-B, 129 of Regulation I) are replayed turn by turn, with the engine driven by the recorded actions and its public state compared against the log after every turn. The public state matches on all 3,452 turns. Of Showdown's per-turn events, 97% are reproduced in order, and about 80% of turns reproduce the full event sequence. Both suites were used to find and fix bugs, so they bound regressions rather than estimate parity on unseen play.

B Method Details

B.1 Relation to ReBeL and Student of Games

Table 4 summarizes, component by component, where the method of Section 4 follows ReBeL and SoG and where the domain forced a departure.

B.2 Decision Types of the Bayesian Matrix Game

Table 5 catalogs the four decision types of Section 4.2 as instances of the Bayesian matrix game, with the menus, the hidden information, and the resulting solve sizes of each; Figure 4 sketches the anatomy of one budgeted solve. In the turn shortlist, a third of the opponent cap k_2 is reserved for switch-containing joints and a third of the remainder for joints that spend the gimmick.

Decision	Chooses	$\mathcal{A}_1$ / $\mathcal{A}_2$	Hidden	Solve size
Team preview	both	bring-4 × lead pair	none ($\lvert\mathcal{W}\rvert$=1)	90×90
Turn	both	move-or-switch per slot	$w \sim b$	$\lvert\mathcal{A}_1\rvert \times k_2 \times \lvert\mathcal{W}\rvert$ root; $k_1 \times k_2 \times \lvert\mathcal{W}\rvert$ interior
Forced switch (after faints)	both	replacement	$w \sim b$	$\leq 2 \times 2 \times \lvert\mathcal{W}\rvert$
Mid-turn switch (pivot)	one seat	replacement (one slot)	w, opponent's uncommitted actions $\mathcal{H}$	$\leq 2 \times \lvert\mathcal{H}\rvert \times \lvert\mathcal{W}\rvert$

Table 5: The decision types of a VGC battle as instances of the Bayesian matrix game (Section 4.2), with solve sizes as rows × columns × worlds. k_i is seat i's menu after shortlisting by the policy prior (root solves keep our full menu; values in Section 5.1); replacement menus are bounded by the two-Pokémon bench; $|\mathcal{W}| \leq 15$ before the leads are revealed and ≤ 6 after.

B.3 Observation Encoding

- **Tokens.** A position is fourteen tokens: a summary token (a learned vector, which both heads read), our six Pokémon in roster order, the opponent's six in positional order (actives in field-slot order, then the revealed bench, then the sheet's unrevealed members), and one field token.
- **Pokémon token.** Embedded categorical ids (species, ability, item, the four moves, the two types, and the tera type) plus a 542-dimensional scalar block:
  - battle state: HP fraction, stat stages, PP, status, active flag and field slot, and 30 volatile conditions with their elapsed counters;
  - stats: base stats, and calculated stats on our own side only (masked for the opponent, whose spreads are hidden);
  - descriptors: one per move slot with the move's queued flag, and the ability and the item as currently held (a consumed or swapped item encodes as such);
  - commitments: charging, recharging, semi-invulnerable, Protect streak, choice lock, and last move;
  - revealed-ness: appeared, fraction of moves revealed, and brought / not brought / unknown;
  - the two belief summaries of Section 4.5: the phase-and-queue block (queue status, departing, switching-in slot, queued target and gimmick) and 27 stat-belief quantiles (mean, p10, p50, and p90 of each of the six stats under the spread mixture q_j, plus p10, p50, and p90 of the current HP fraction), which reduce to a point mass on our own side.
- **Field token.** A 90-dimensional block: weather, terrain, rooms, gravity, both sides' side conditions with elapsed and remaining turns, the turn fraction, gimmick-used flags, the team-preview flag, and a one-hot of the decision phase (turn, forced switch, mid-turn switch).
- **Belief as input.** Each opponent token carries its marginal $P(\text{brought} \mid b)$, and the field token carries the full distribution over the $\binom{6}{4} = 15$ bring subsets of the open sheet in a canonical order, together with its effective sample size. Nothing in the input asserts which world is true, since unrevealed opponent Pokémon are encoded as sheet members of unknown status, so one forward pass prices every world at once.
- **Belief as output.** The value head has fifteen units, one per bring subset in the same order: a world reads the unit of its bring, worlds that share a bring share a unit, and the leaf value $\langle b, v_\theta \rangle$ of Equation (1) is the belief-weighted sum of the units.
- **Trunk.** A post-norm transformer with a learned positional embedding over the fourteen tokens; both heads read the summary token, the policy head scoring the joint two-slot menu.

B.4 Interior-Row Sampling and Weighting

Up to K interior rows per decision (Section 4.6) are drawn $\propto$ reach under $\bar{\sigma}$ by Horvitz–Thompson systematic sampling: heavy nodes enter deterministically and the tail is sampled with inclusion weights, so $\mathbb{E}[\tilde{r}]$ reproduces the full capture's reach weighting. Each sampled row is weighted by

$$\eta_{\text{int}} = \tilde{r}\left(m + (1-m)\,\lambda_{\text{int}}\right), \tag{5}$$

where $\tilde{r}$ is the row's reach weight normalized within its solve and m the fraction of its backup that terminated in-tree rather than at the leaf network, so the most network-priced backups are discounted.

C HIDDEN STAT BELIEF MACHINERY

This appendix expands Section 4.4. Pokémon Champions allocates stat points on a 0–32 per-stat scale with a 66-point budget (in place of the 0–252 effort values of the main series), so the investment grid per stat is small and the nature–spread pair is the whole hidden build under open sheets. Spreads are assumed independent across Pokémon, so a world's spread belief is the product $\prod_j q_j$ of its per-Pokémon mixtures.

Prior fallback. A species' candidate menu comes first from the team-paste corpus (Section 4.4): the top three spread rows observed with the species under its revealed nature (the unconditional marginal when the corpus never pairs the two). Species the corpus lacks fall back to the per-species usage rows $\{(\xi_r, f_r)\}$ of Pokemon Champions Battle Data (2026), scraped per season and format. Usage reports nature and spread as independent marginals; since the nature is public under open sheets and the two are strongly dependent, each row is reweighted by nature compatibility $\kappa(\nu_j, \xi_r) \in \{1, 0.35, 0.01\}$: investment in the nature's lowered stat all but rules a candidate out, skipping its raised stat demotes it, and nothing is zeroed. Raw rows are near-duplicates (the top rows of one species often differ by a point or two of filler), so they are greedily clustered: rows in order of $f_r\kappa$ join the first cluster whose representative is within L_1 distance 16, pooling their mass, and the top three clusters become the candidates, spending slots on genuinely different builds (bulky versus fast) rather than variants of one. In both the corpus and fallback paths, a fixed $\tau = 0.15$ of the mass goes to base-stat archetype spreads, $q_j^0 = (1-\tau)\, q_j^{\text{head}} + \tau\, q_j^{\text{tail}}$, deduplicated against the head candidates. Under closed sheets the whole build is hidden and the menu comes from a Bayesian team-paste corpus model instead.

Outcome likelihood. After turn t we have observed the joint action a and the public outcome $o_t = \phi(s_t)$, where ϕ projects an engine state to its public part, refined by the observed move-order signature. Writing $w[\xi_j]$ for world w with Pokémon j's mixture collapsed to the single candidate ξ_j, the likelihood of that candidate is the mass the engine's own transition puts on what was seen,

$$L_j(\xi_j) = \sum_{(s', w', p) \in \mathcal{T}(s_{t-1},\, w[\xi_j],\, a)} p\, \mathbf{1}[\phi(s') = o_t], \quad q_j(\xi) \leftarrow \frac{q_j(\xi)\, \max\{L_j(\xi),\, \epsilon \max_{\xi'} L_j(\xi')\}}{\sum_{\xi''}(\cdot)}, \tag{6}$$

with $\epsilon = 0.05$, and the world weight takes $P(w) \propto P(w) \sum_\xi q_j(\xi) L_j(\xi)$, the same numbers by linearity of $\mathcal{T}$ in the mixture. The update discriminates only *within* the menu, so its failure modes are menu failures: a candidate that cannot reproduce one outcome is demoted by the floor ϵ rather than eliminated, and a menu none of whose candidates reproduces the observed move order is handled by the contradiction ledger below.

Per-turn update. Three updates run each turn in order. *Reveal conditioning* zeroes worlds inconsistent with new hard evidence (a fifth revealed species not in a world's bring, a move or ability outside its config); item mismatches are priced softly because items change hands mid-battle. *Action conditioning* is the ReBeL update $P(w) \propto P(w)\, \bar{\sigma}_2^w(a_2)$ against our own last solve, floored so a surprising action dampens a world rather than killing it. *Outcome conditioning* is the engine step: the completed turn's joint action is re-branched per world, and the matched mass of the branch on the observed public outcome (HP quantized to 16 buckets, refined by the observed move-order signature when it is reproducible) is $P_w(o \mid a)$. The same re-branch is run per spread candidate by materializing the world with one Pokémon's mixture restricted to that candidate; because the branch is linear in mixture weights, the per-candidate and per-world likelihoods are literally the same numbers, not two approximations. Hidden opponent slots whose action was not observed are handled as a completion mixture weighted by the pre-turn $\bar{\sigma}$, capped in group count since each group is one branch. Likelihoods are floored relative to the best candidate (a single observation moves a posterior at most $20\times$), worlds are evaluated heaviest-first up to a cap, and the spread posteriors live on the tracker across turns and across the games of a best-of-three set, where builds are locked. Worlds are rebuilt from the current state when every world is contradicted or the effective sample size collapses while reveals are incomplete.

Contradiction ledger. Within-support evidence is absorbed by the posterior, but an observation that no candidate reproduces conditions nothing and was previously discarded: the ordering channel fell back to the public-only channel for everyone, because mixed-channel scoring would move

Team	Elo	Pokémon
mb486	1676	*Charizard*, Garchomp, Venusaur, Incineroar, Toxapex, Annihilape
mb355	1579	*Charizard*, *Floette-Eternal*, Kingambit, Whimsicott, Basculegion, Garchomp
mb520	1531	*Tyranitar*, *Staraptor*, Excadrill, Gholdengo, Sinistcha, Milotic
mb300	1523	*Raichu*, Basculegion, Garchomp, Kingambit, Whimsicott, Ninetales-Alola
mb443	1520	*Charizard*, *Aerodactyl*, Garchomp, Sylveon, Farigiraf, Kingambit
mb210	1516	*Froslass*, *Scovillain*, Lycanroc-Dusk, Kingambit, Basculegion, Sneasler
mb142	1508	*Metagross*, *Dragonite*, Ninetales-Alola, Garchomp, Sneasler, Kingambit
mb340	1505	*Blastoise*, *Delphox*, Sneasler, Kingambit, Incineroar, Sinistcha
mb366	1484	*Charizard*, Garchomp, Gholdengo, Whimsicott, Incineroar, Basculegion
mb521	1482	*Gengar*, *Swampert*, Archaludon, Politoed, Vivillon, Incineroar
mb524	1473	*Charizard*, Grimmsnarl, Basculegion, Pelipper, Venusaur, Archaludon
mb65	1462	*Staraptor*, Grimmsnarl, Garchomp, Gholdengo, Sinistcha, Incineroar
mb166	1452	*Swampert*, *Floette-Eternal*, Pelipper, Grimmsnarl, Sinistcha, Archaludon
mb244	1447	*Froslass*, *Blaziken*, Kingambit, Sinistcha, Basculegion, Sneasler
mb333	1426	*Pyroar*, *Blastoise*, Torkoal, Farigiraf, Sylveon, Venusaur
mb235	1418	*Mawile*, Kangaskhan, Farigiraf, Torkoal, Staraptor, Vivillon

Table 6: The eval16 evaluation pool, sorted by strength. Each team is a public paste from the VGCPastes repository (VGCPastes, 2026) (linked; author credit on the paste page), identified by its corpus index. Elo: Bradley–Terry fit over 2,940 internal head-to-head games among the 16 (mean 1500). Italics mark Pokémon holding a Mega Stone.

belief toward refuted candidates. The ledger catches exactly this signature, a public outcome that reproduces under some candidate while the move order reproduces under none, which distinguishes a speed-shaped contradiction from a degraded log. It then probes rather than inverts: candidate spreads are synthesized along the species' speed-investment axis, substituted one at a time into a value-copy of the world, and scored through the same branch-and-match pipeline, in a coarse-to-fine sweep, so Trick Room, Tailwind, paralysis, stat stages and spread-dependent abilities are correct by construction. The feasible hull is stored as a unary interval fact (opponent speed versus one of our own, whose speed is known) or a pair fact (opponent versus opponent). A second producer reads the end-of-turn residual phase, where Showdown applies chip and healing per bracket in speed order, giving a uniform-context speed observation on most turns with two or more holders in a bracket. Facts are kept as supersets of the truth (ties inclusive, rounding outward) so they compose by intersection and never need revisiting; dominated facts are dropped on insert and tightenings are recomputed by closure. A binding fact marks the tracker dirty, and the next rebuild filters the candidate menu through the closed constraints and synthesizes a minimal-perturbation candidate per binding constraint, using the same recipe helpers the probes use so a probed candidate and a synthesized one can never disagree. The ledger persists across the games of a set. Known gaps: an order fact between two opponent Pokémon that are both off-menu is not resolved by marginal sweeps, and bulk and knock-out-timing axes are not built. The ledger was off in every training run, where opponent spreads are drawn from the same corpus mixture the menu is built from, and on in all three ladder runs of Section 5.4, where hand-tuned human spreads outside that mixture are exactly the case it targets.

D Team Strength and Pool Effects

This appendix rates specific team builds within the 2026 Regulation M-B metagame. It is written mostly for readers who play VGC; the findings will mean little without some familiarity with the format.

Table 6 lists the eval16 pool of Section 5.1: sixteen public team pastes from the VGCPastes repository (VGCPastes, 2026), hand-picked to span the meta's primary archetypes.

Eval16 was chosen for archetype diversity, not strength, and it turns out to be a mid-tier pool. Rating every build in the corpus with a Bradley–Terry fit over 18,000 self-play games (648 canonical builds from 797 pastes, median 48 games each), the 16 eval teams scatter from rank 3 to rank 459 of 648 (median 217), with four in the bottom half. The orderings are also strongly pool-dependent: the corpus-wide ratings agree with the eval16 round-robin hierarchy only weakly (Spearman $+0.38$), and neither correlates with per-team ladder residuals at current sample sizes. Being good against these

16 is substantially a matchup fact, not a strength fact. Both consequences matter for the main text: internal win rates are compressed by mirror-adjacent matchups, and the ladder runs pilot mid-tier public builds against opponents free to bring the meta's best. A ladder run on a corpus-top pool is reported at the end of this appendix.

To rate the strong end under conditions matching live play, the 32 highest-rated builds also played a full round-robin (496 unordered pairs $\times$ 3 games $=$ 1,488 games) at a deepened deployment shape of Section 5.3 (B=128, widened 200-iteration child solves). Table 7 shows the standings; three readings stand out.

The strong end is the human meta's strong end. The top of Table 7 is dominated by tournament-proven builds popularized by established competitors — eval16's own `mb486`, tenth here, is the team of Shiliang Tang, a top VGC player and team builder — and the winner is clear: the Mega Staraptor sand team `mb1` swept 16 of its 31 opponents. The one entrant near the top a player would likely not have picked is `mb695`, a sand build around Mega Sceptile, rated fourth corpus-wide and holding 60.2% against the elite pool.

Human-meta strength is a different axis. `mb702`, a Mega Gengar Yawn-trap team with an un-Mega'd Swampert, won a recent major tournament, so we forced it into the round-robin over its bottom-half corpus rating — and it finished dead last. Its gameplan monetizes surprise against human expectations, value the self-play-trained search neither finds nor falls for. The corpus fit and the elite pool also disagree at the top in both directions (the cRk column): the corpus top two — the `mb188` rain build and the dual-Mega Staraptor–Delphox "StarFox" team `mb348` — both post losing records against the elite pool, while the winning sand team entered only 11th of the 32 by corpus rating.

Durable teams convert search. The corpus fit rates teams at self-play budgets $B \in \{32, 64\}$; the round-robin plays a far deeper shape, and the rank moves between the two are systematic. Durable builds rose — `mb200`, Pelipper–Archaludon rain behind Grimmsnarl screens, from 22nd within the 32 to 8th, its screenless sibling `mb198` from 20th to 5th — while the Mega Froslass plus Mega Scovillain hyper offense `mb727` fell from 8th to 21st. These moves confound search depth with the opponent pool, but a split of the corpus fit by budget, which holds the pool fixed, points the same way: the species gaining the most rating from B=32 to B=64 (Excadrill, Staraptor, Raichu) are three of the winning sand team's six. Our reading is that bulkier structures give a deep solve more turns in which its edge compounds, while all-in offense has played out before the extra depth matters.

A corpus-top ladder run. We ran the ladder a third time at the scaled shape of Section 5.4, drawing from the top eight of Table 7 (cut to the top six after 91 sets, dropping `mb194` and `mb423`), for 142 sets against a field averaging $\sim$1310 Elo. The stronger pool bought nothing in aggregate: 80–62 sets (56.3%) and 172–155 games (52.6%), below both eval16 runs, with the round-robin winner `mb1` at 15–8 and the two teams cut at 5–8 and 5–7. The run is also the paper's clearest picture of single-account noise: its first 81 sets went 52–29 (64.2%) and reached 1415 Elo, and the remaining 61 went 28–33, including a 4–12 stretch that an audit of the build, the solve counters, and the opponents' ratings could attribute to nothing but variance. Self-play rates a team within its pool; against the human field the rating is at best a weak prior.

Within-set adaptation. A best-of-three lets the same opponent face the agent up to three times, so exploitation within a set would show as a win rate that falls from game to game. Over the three ladder runs the agent went 156–136 (53.4%) in game one, 141–138 (50.5%) in game two, and 68–42 (61.8%) in game three. Game one includes 13 sets in which the opponent walked out, and game three is played only after a split, but no decline appears. Nor does game three rest on spread inferences carried over from earlier games. It went 39–30 when the opponent's roster had been fully seen in games one and two, and 29–12 when the opponent brought a Pokémon not yet seen, so the agent does at least as well when the opponent changes plan.

Private challenge matches. After the ladder runs the agent was hosted as an opt-in practice partner. Any Showdown user who messaged the account could challenge it to a best-of-one or best-of-three, with the agent piloting either a house team or a team the user supplied as a paste. In this mode the agent never entered a ladder queue. After each game the agent posted a recap into the battle room, listing for every solved decision the root value of the solve, its own solved action mix, and the

#	Team	Win%	cRk	Pokémon
1	mb1	76.3	11	*Tyranitar*, *Staraptor*, Excadrill, Basculegion, Sinistcha, Raichu
2	mb194	69.9	6	*Froslass*, Incineroar, Gholdengo, Basculegion, Kingambit, Arcanine-Hisui
3	mb537	68.8	19	*Charizard*, *Mawile*, Whimsicott, Basculegion, Farigiraf, Garchomp
4	mb345	65.6	13	*Charizard*, *Garchomp*, Venusaur, Farigiraf, Incineroar, Sylveon
5	mb198	64.5	20	*Swampert*, *Floette-Eternal*, Whimsicott, Pelipper, Archaludon, Basculegion
6	mb423	63.4	7	*Raichu*, *Tyranitar*, Meowscarada, Basculegion, Kleavor, Talonflame
7	mb350	62.4	15	*Raichu*, *Floette-Eternal*, Whimsicott, Basculegion, Garchomp, Gholdengo
8	mb200	62.4	22	*Swampert*, *Metagross*, Grimmsnarl, Pelipper, Archaludon, Sinistcha
9	mb417	62.4	5	*Charizard*, *Aerodactyl*, Kingambit, Farigiraf, Garchomp, Sylveon
10	mb486	61.8	3	*Charizard*, Garchomp, Venusaur, Incineroar, Toxapex, Annihilape
11	mb534	61.3	17	*Charizard*, Whimsicott, Garchomp, Incineroar, Basculegion, Gholdengo
12	mb695	60.2	4	*Sceptile*, *Tyranitar*, Rotom-Wash, Corviknight, Arcanine-Hisui, Houndstone
13	mb675	58.6	9	*Kangaskhan*, *Charizard*, Incineroar, Hatterene, Farigiraf, Torkoal
14	mb128	58.1	14	*Kangaskhan*, Vileplume, Farigiraf, Torkoal, Basculegion, Kommo-o
15	mb326	57.0	12	*Charizard*, Farigiraf, Venusaur, Garchomp, Incineroar, Sylveon
16	mb413	57.0	10	*Delphox*, *Blastoise*, Incineroar, Maushold-Four, Sinistcha, Sneasler
17	mb383	51.6	30	*Pyroar*, *Floette-Eternal*, Vileplume, Torkoal, Incineroar, Farigiraf
18	mb354	49.5	23	*Delphox*, *Blastoise*, Incineroar, Sinistcha, Maushold, Sneasler
19	mb188	45.2	1	*Metagross*, *Swampert*, Sinistcha, Pelipper, Archaludon, Hydreigon
20	mb348	43.0	2	*Staraptor*, *Delphox*, Gholdengo, Grimmsnarl, Basculegion, Sinistcha
21	mb727	40.9	8	*Froslass*, *Scovillain*, Kingambit, Basculegion, Sneasler, Lycanroc-Dusk
22	mb450	39.2	16	*Charizard*, *Floette-Eternal*, Whimsicott, Basculegion, Garchomp, Kingambit
23	mb459	39.2	21	*Gardevoir*, Maushold-Four, Kingambit, Garchomp, Talonflame, Basculegion
24	mb273	36.6	25	*Charizard*, *Floette-Eternal*, Sneasler, Sinistcha, Incineroar, Kingambit
25	mb576	36.6	29	*Dragonite*, *Scovillain*, Gholdengo, Basculegion, Sylveon, Garchomp
26	mb580	36.6	26	*Raichu*, *Aerodactyl*, Sylveon, Kingambit, Basculegion, Sneasler
27	mb341	33.9	24	*Froslass*, Glaceon, Sneasler, Kingambit, Garchomp, Talonflame
28	mb187	31.2	28	*Floette-Eternal*, Whimsicott, Incineroar, Farigiraf, Garchomp, Basculegion
29	mb603	29.0	27	*Staraptor*, *Delphox*, Garchomp, Whimsicott, Glimmora, Kingambit
30	mb255	26.9	31	*Charizard*, Bellibolt, Sableye, Garchomp, Basculegion, Sylveon
31	mb171	26.3	18	*Eelektross*, Glimmora, Whimsicott, Basculegion, Garchomp, Farigiraf
32	mb702	24.7	32	*Gengar*, Swampert, Incineroar, Ninetales-Alola, Sneasler, Dragonite

Table 7: Full standings of the 32-team elite round-robin at the deepened deployment shape. Win% is over 93 games per team; cRk is the team's rank within the 32 by the corpus-wide fit. Links and italics as in Table 6.

opponent mix it solved against, so a challenger could review why it played what it played. Between 28 August and 20 September 2026, 13 users completed 83 matches (48 best-of-three and 35 best-of-one, at deployment budgets of 32, 64, and 256). The agent won 57–26 by set and 98–56 by game (63.6%). In 52 matches it piloted one of 12 distinct user-supplied teams, seven of which match no self-play corpus team on all six species, and won 39–13 by set and 80–43 by game (65.0%). With house teams it went 18–13. The sample is small and self-selected, and the users' ratings are unknown. Two users account for 45 matches. Against the heavier user (72 games, almost all with teams they supplied), the agent's win rate fell from 27 of 36 games in the first half to 21 of 36 in the second, while against the other (34 games) it rose from 10 of 17 to 12 of 17. The data therefore neither shows nor rules out adaptation by a persistent opponent.

E Additional Experimental Results

E.1 Search and Training Configuration

Defaults of the final run, beyond those stated in Section 5.1: 300 CFR iterations at the root and 100 at interior nodes, 160 for the 90×90 preview; root own menu unrestricted; prune floor 2% with at most 8 branches kept per joint action; belief-width exponent $\rho = 1$ and switch-continuation cost $c_{\text{sw}} = 0.01$; interior rows capped at 96 per decision and weighted by walk visit share, with $\lambda_{\text{int}} = 0.15$ rising to 0.25 from round six; outcome-grounding mix $\lambda = 0.5$; replay window shrinking from 5 to 2 rounds over the ramp. The compute ramp: rounds 1–5 at B=8 and 1k games per round, 6–8 at B=16/2k, 9–20 at B=16/3k, and 21–32 at B=32/3k.

E.2 The Reference Agent and Encoder Lineage

The D1SW reference agent. D1SW is a previous-tier agent trained for 15 rounds with a "depth 1" solve that exhaustively recurses forced-switch and mid-turn switch nodes but not actual turn nodes. It is a hero run in the unflattering sense: it uses an older encoder with multiple bugs, an exhaustive search shape that costs ∼30 s per decision, and a value head that passes only 26 of 170 direct-read tactical checks (Appendix E.6). It nevertheless beat every deeper, wider, and later agent we built for ten days of campaigns — including its own search shape re-run on later encoder tiers — so intermediate campaigns were reported against it round-matched, and its final checkpoint D1SW15 remains the fixed foreign anchor of Figure 2. Round matching is not compute matching: D1SW rounds were 1–2k games at 30 s per decision, where budgeted solves cost 10–23 s.

Encoder tiers. The final run's encoder (*v7*) is the earlier tier (*v5*) plus the two belief summaries of Section 4.5: v5 lacked the turn phase and queued commitments, so mid-turn leaves were priced as pre-turn states, and it left hidden stats to the belief alone, with no spread quantiles in the input. Everything else in the agent (solver, targets, training) is indifferent to the tier. The two summaries arrived in separate tiers, and only the second was measured in isolation: with every search and training knob held fixed, moving the recipe from the phase-only tier to v7 was worth +8–12pp against round-matched D1SW over its first five rounds (33–37% → 42–46%, 512-game cells), and the deeper 6-layer trunk added ≈6pp more by round five (49.9 vs. 43.8%, one cell, ≈ 2σ). The phase block's own contribution is confounded with search changes in the runs that introduced it, which still lost to round-matched D1SW. Every cell here is a loss: at the round-matched budget of 8 the lineage trailed D1SW on every tier, and first reached parity only when the budget was raised to 32 (Section 5.2). The mid-game pair probes that motivated the spread quantiles are in Appendix E.6.

E.3 The Policy Recipe at Matched Training Compute

The π-ONLY agents of Section 5.2 are the policy heads of our own networks. To compare against the VGC-Bench recipe (Angliss et al., 2026) on our own format, we also trained a policy by behavior cloning on the same public replays followed by PPO self-play, over the same Regulation M-B team corpus (361 teams at the time) and with a policy architecture carrying a 32-decision temporal context. The PPO run made 2,000 iterations of 128 self-play games (256k games) on one RTX 4090 with 32 rollout workers, and its final policy reaches 39.5% against GREEDY (101–155). It beats the policy head of our round-one network 58–42, as a quarter-million PPO games should beat a head fit to one round of 1k searched games. Under search the order reverses. The same round-one network beats the

PPO policy 73.0% (187–69) at the native round-32 search shape and 74.2% (190–66) at its own B=8 shape, and the behavior-cloned parent, which has seen no self-play at all, beats it 70.3% (180–76). The same parent under the same search loses 50–206 (19.5%, Wilson [15.1, 24.8]) to the round-32 network. Search alone carries the recipe past PPO, and 32 rounds of self-play add thirty points on top. Against GREEDY those three search cells are 54.3, 53.1, and 50.8%. Round one is the first and cheapest of the run's 32 rounds (Appendix E.1), so the PPO side is not the compute-starved one, and training compute does not separate the recipes. Decision-time search does, before any self-play round has been paid for. All cells are eval16, single seat, argmax, 256 games (±6pp), and the comparison is to the recipe rather than to the released agents, whose team pool and regulation differ from ours.

E.4 TRAINING ABLATIONS

A caveat governs this subsection. At our compute scale a campaign arm is two to seven rounds and a cell is 256–512 games, so the resolving power is ±4–6pp, and almost every individual lever we tried lands inside that band. Our working rule has therefore been to ablate in order to make sure a change is not worse, and then to keep the variant that makes more sense on its own terms (cheaper, simpler, better-covered gradients, better-calibrated), rather than to claim a win for it. The arms below are reported in that spirit; Table 8 collects the head-to-heads, cells against D1SW are round-matched (Appendix E.2), and win rates are over all games in a cell, so the rare game with no winner counts as a loss.

Expansion and budget. The budget binds on every walk: at $B \in \{8, 16, 32\}$ the spent cost reaches the budget on 100% of walks and the early-stopping criterion never fires, while median walk depth grows 2→3→4. Marginal units are cheap (per-walk wall 15.9→23.6→31.7 s for $B = 8 \to 16 \to 32$ on the local box), and budget spent at inference converts directly (Section 5.3; already visible mid-campaign, where a round-five network gained +4.3pp at 2× its training budget). Spent at data generation instead, doubling the budget did not move a plateaued lineage in two rounds (34.8/35.7% vs. 35.5/33.6%), foreshadowing the final run's budget-insensitive back half. Replacing the best-first expansion heap by the PUCT walk held parity against the queue-mode sibling (48.8% at round three, 52.0% at round five) on roughly half the training rows per round and −8% wall per decision, and was kept for that reason (the same arm cut hypothetical forks from three to one per game, so the two changes are confounded; the row saving is the walk's, which admits about a third as many forced-switch plans per decision, while the fork count was a wash on its own, Table 8); weighting interior rows by walk visit share instead of telescoped reach was a wash (50.0/52.3/52.3% at rounds 1/3/5) and was kept because it un-kills the gradient on the 56% of interior rows that had weight below 10^{-3}. Entry-point continuations (recursing the replacement and preview roots) won 52.0% at round five and 57.0% at round seven against their sibling; deploying the flags on the sibling's network alone gives 50.8%, so the edge is in the training data.

Interior rows. On a round-one dataset (146k rows: 26k roots, 120k interior at a cap of six per decision) the interior share of the value loss is linear in λ_{int}, from 10% at λ_{int}=0 to 43% at 1; retraining across $\lambda_{\text{int}} \in \{0, 0.05, 0.15, 0.5, 1\}$ is free up to 0.15 and costs +0.003–0.005 MSE and 0.07 of corr(v, z) at 1, and by round six play is flat in λ_{int} (49.6/50.6/48.6% against round-matched D1SW at 0.15/0.5/1.0, 512-game cells). We use 0.15 rising to 0.25. The one thing interior rows clearly did was qualitative: theirs was the first lineage whose self-play rounds did not erase the behavior-cloned parent's tactical discrimination (Appendix E.6). At the production cap of 96 rows the sampler captures 80–84% of interior reach mass at $B \in \{8, 16\}$; already at a cap of 24, interior rows are 94% of a round's rows while carrying 23% of its value weight.

Menus and solver iterations. Menu width never pays at matched cost in training-time cells either: on a B=32 round-five network, widening the opponent root cap from 16 to 32 is 47.3% head-to-head against the native deploy, cutting our own root menu to 24 is 44.5%, and uncapping both root menus is flat (53.1 vs. 51.2% against D1SW15) at 1.84× the cost; a shortlist probe on 59 self-play roots puts the knee at own 24 / opponent 16. Root CFR iterations are a dead lever (300 → 2000 moves the root value by 0.003, and 100 → 400 interior iterations flips no argmax — though at the wider deployment shapes of Table 3, where the same iterations spread over larger matrices, child iterations become the biggest lever), and CFR beats the fictitious-play solver of the ReBeL paper at equal wall clock (field win rate over eight rounds 49.5 vs. 46.0%, 13.5k and 11.5k games against the same field). Both

Arm	Change	Result	Verdict
no grounding	λ=0, λ_{int}=1	50.0% (256/512) h2h, same deploy	wash at 2 rounds; calibration drifts
decoupled + zero-sum	θ' re-priced targets, moment loss, λ=0	42.6% (109/256) vs sibling r3	loses; dropped
hypothetical forks, 3/game	augmentation only	48.8% (125/256) compute-matched	wash per unit compute
hypothetical forks, 1/game + PUCT		52.0% (133/256) on half the rows	kept
full-z on hypo rows	λ=1 on augmented rows	40.6% (104/256) vs control	loses
entry continuations	replacement + preview roots	52.0% r5, 57.0% r7	kept
visit-weighted interior	$\tilde{r}$ from walk visits	50.0/52.3/52.3%	kept (gradient coverage)
D1SW shape on v7	phased depth 1, switch-all	43.0% (110/256) r5	loses
outcome-only head	re-init v_θ, train on z only	49.2% (126/256) vs r20	parity; see Appendix E.6
decoupled-UCT solver	solver swap at deploy, 10× iterations, +21% wall	51.7% (2,048 games) h2h; 85.5% vs GREEDY	parity vs fixed field; exploitable (Appendix E.5)

Table 8: Target and expansion ablations on the v7 deep line; each row is a head-to-head against its round-matched sibling unless noted. Most levers are within a 256-game cell's ±6pp of parity: the binding constraint is the value network, not the target construction.

solvers converge to an equilibrium, so this compares solvers. Appendix E.5 tests a non-equilibrium baseline. The one deploy-side menu lever that was unambiguous is shortlist *ranking*: opponent menus ranked by the belief-weighted policy prior instead of by support count are worth +16pp on the same network (57.8 vs. 41.8%, 512 games), and fixed the Protect-omission failure mode of the earliest ladder deployments, whose attack-biased shortlists had omitted Protect on 48% of the turns it was legal.

Targets. Removing outcome grounding costs nothing at two rounds but lets calibration drift (+0.03–0.05 on matched states, in-sample corr(v, z) 0.64 vs. 0.78), so we kept it; re-pricing leaves with a frozen θ' plus a zero-sum moment loss at λ=0 loses outright; grounding augmented rows fully by their suffix outcome loses; hypothetical forks are a wash per unit compute and were kept at one per game for coverage. Table 8 is the honest summary: at 2–5 rounds and 256 games, almost every lever lands within noise of parity.

E.5 Equilibrium Quality and Exploitability

Within the solved subgame. We instrumented every CFR call in 96 self-play games of the final agent on the eval16 pool (2,172 root solves, 138k child solves), recorded the per-world root payoff matrices after each solve, and computed the NashConv of the profile the agent plays from, in payoff units on $[-1, 1]$. An informed opponent's best response gains only 0.0004 on average against the mix (median 0.0002, p90 0.0011), with team preview at 0.0011 and replacement games exact. Cold Linear CFR on the same matrices reaches 0.041 at 10 iterations, 0.0019 at 100, and 0.0006 at the deployed 300. The solver is converged. This measurement does not bound what the subgame leaves out (menus, pruning, the leaf network). The same matrices price the play temperature. The best response gains 0.117 (p90 0.31) against the argmax the evaluation cells submit, 0.032 against the ladder's T=0.5 draw, and 0.0004 against the full mix.

The decoupled-UCT arm. The non-equilibrium baseline of Section 5.2 is decoupled UCT (Shafiei et al., 2009; Tak et al., 2014), the selection rule of Foul Play. Each seat runs UCB1 over its own actions and submits the most-visited one. We swap it in for every matrix solve of the round-32 agent at its native shape, keeping the network, engine, menus, beliefs, and expansion walk, and give it ten times the iterations (c=1 on the $[-1, 1]$ payoff scale). This costs 21% more search wall time per game (148 s against 122 s). It plays the CFR agent at parity, 51.7% ([49.5, 53.8]) over 2,048 games, and beats GREEDY 85.5% against CFR's 78.5%. On 400 captured root matrices its mix has NashConv 0.24, against 0.0009 for CFR at a tenth of the iterations, and the action it submits is exploitable by

Agent's solver	Agent's play	Best-response win% [95% CI]		Gain / decision
CFR	argmax (eval cells)	77.7	[72.2, 82.4]	0.097
CFR	T=0.5 (ladder)	52.3	[46.2, 58.4]	0.027
CFR	T=1 (solved mix)	49.6	[43.5, 55.7]	0.0003
decoupled UCT	argmax	74.6	[68.9, 79.6]	0.193
decoupled UCT	T=0.5	72.7	[66.9, 77.8]	0.110
decoupled UCT	T=1	83.6	[78.6, 87.6]	0.058

Table 9: An informed local best response, seated second, against the round-32 agent under each solver and play mode (eval16, 256 games per cell). Gain is the best response's in-model advantage per decision over the agent's expectation, in payoff units on $[-1, 1]$. The two solvers are at parity against fixed opponents (51.7%, 2,048 games) and 34pp apart here.

0.23, against 0.13 for CFR's argmax. Ten times the iterations halved its NashConv (from 0.43 at 300), so more would not close the gap. At these budgets its visit distribution is flat, with the most-visited row holding about a fifth of the root visits, so it is a weakly converged baseline rather than a greedy one.

An informed local best response. A local best response (Lisý & Bowling, 2017) bounds the exploitability of an agent whose best response is intractable. At every decision the opponent is handed the agent's strategy and replies with the action that is best against it under a fixed local evaluation. Ours is informed by the agent's own solved subgame and the true world. It sits in the second seat and reads the distribution the agent actually draws from (a point mass under argmax, $\bar{\sigma}^{1/T}$ otherwise), never the realized draw. It best-responds inside the agent's per-world payoff matrices, in the belief world that matches its own roster, and plays as the agent itself at team preview, at replacements, and on turns the agent decided without a solve. Its win rate over 256 eval16 games lower-bounds the agent's exploitability (Table 9), and 50% means unexploited. The CFR mix is unexploited, with an in-model gain of 0.0003 per decision, and the per-decision gains under argmax and T=0.5 (0.097 and 0.027) reproduce the within-subgame prices above in live play. The decoupled-UCT agent plays fixed opponents at parity yet is exploited under every play mode, most of all under its own mix, because sampling from a flat non-equilibrium visit distribution plays many rows that a best response punishes.

E.6 Value Network Quality

Instrument. The tactics suite grades value networks against search-certified optima on hand-built Champions positions: 17 crafted positions (speed control, ability interactions, critical-hit lines, Perish Song, weather, gimmick timing, Protect reads) and 129 generated discrimination pairs, 248 positions in which one fact is flipped between a certified win and a certified loss (speed by raw stat, Trick Room, Tailwind, stages, Choice Scarf; bulk by matchup, screens, stages; burn, paralysis; weather damage and speed). Values are certified by an exact solve whose leaves are re-filled with $0, \pm 1$; fill-independence is the certificate. Generated pairs are graded from the network's direct read at the root, since a depth-one solve rescues any network; we report *root separation*, the mean read on certified wins minus certified losses, against an exact gap of about 2.0, and *gap passes*, pairs the network separates by at least 0.7, over the 123 pairs whose certified gap has a direction.

The discrimination wall. Every checkpoint through mid-August separates wins from losses by only $+0.05$ to $+0.14$ (one B=32 arm reaches $+0.20$) and passes at most 3 of 123 gaps: the network gets the move right under search (crafted argmax 11–14/14, conversion walks 8–10/10) but does not see why. The behavior-cloned parents are often the best direct readers, and earlier lineages' self-play rounds erased that knowledge (one parent passes 63/170 root checks, its self-play child 30/170); the interior-row lineage was the first whose rounds did not. Only deep into the final run does the wall move: rounds 17–20 reach separations of $+0.16$ to $+0.23$ with 2–3 gap passes, and the B=32 back half pushes certified-loss hard passes to a campaign-high 25/120 by round 32.

Optimism. The final run prices certified losses positive in every round but the second (mean loss read up to $+0.75$): win-side root checks climb to 114/141 while loss-side checks stay at or below 13/120 through round 20, with the bias plateauing over rounds 11–15 ($+0.71$ to $+0.75$) and falling

to $+0.10$ by round 32 before re-inflating in the saturated extension rounds. The v5 family had the opposite sign. The optimism is not a suite artifact: on $\sim$36k of its own self-play roots a round-two network reads $+0.12$ above the realized outcome on average, $+0.28$ on turns 1–2 falling to $+0.01$ past turn 7, the same phase profile the ladder solves show live (Section 5.4). Re-initializing a late checkpoint's value head and training it on realized outcomes only (1.5M grounded rows, $\approx 3\%$ of the pool) recovers a head with the same play strength (49.2% head-to-head, 61.3% vs. D1SW15), better outcome correlation (0.72 vs. 0.69), and separation $+0.21$, yet still prices certified losses at $+0.22$: the blindness is a coverage problem of self-play, not a property of the solver targets or the BC initialization.

Where the error goes. On 33 late-endgame roots with certified value bands, budgeted solves (B=16) land inside the band for every network tested, including the worst-calibrated one (mean band distance 0.13, 79% of direct reads outside): search absorbs leaf error where the horizon is short — the mechanism behind the live calibration-by-phase profile of Section 5.4. The depth-one shape leaks, and there the solve error tracks the network's oracle error (Pearson 0.40–0.86). In mid-game pair probes the pre-v7 networks under-encode speed facts relative to D1SW's (network-to-solve response ratio for Tailwind 0.15–0.27 vs. 0.53) and no network encodes screens, which a 16-unit solve proves worth $+0.11$ to $+0.19$; this probe is what motivated encoding raw spread quantiles in v7.