Human-Level Competitive Pokémon via Scalable Offline Reinforcement Learning with Transformers

Jake Grigsby, Yuqi Xie[†], Justin Sasek[†], Steven Zheng[†], Yuke Zhu

Keywords: Pokémon , Offline RL, Imitation Learning

Summary

Competitive Pokémon Singles (CPS) is a popular strategy game where players learn to exploit their opponent based on imperfect information in battles that can last more than one hundred stochastic turns. AI research in CPS has been led by heuristic tree search and online self-play, but the game may also create a platform to study adaptive policies trained offline on large datasets. We develop a pipeline to reconstruct the first-person perspective of an agent from logs saved from the third-person perspective of a spectator, thereby unlocking a dataset of real human battles spanning more than a decade that grows larger every day. This dataset enables a black-box approach where we train large sequence models to adapt to their opponent based solely on their input trajectory while selecting moves without explicit search of any kind. We study a progression from imitation learning to offline RL and offline fine-tuning on self-play data in the hardcore competitive setting of Pokémon's four oldest (and most partially observed) game generations. The resulting agents outperform a recent LLM Agent approach and a strong heuristic search engine. While playing anonymously in online battles against humans, our best agents climb to rankings inside the top 10% of active players. All agent checkpoints, training details, datasets, and baselines are available at `metamon.tech`.

Contribution(s)

1. We build and release an offline RL dataset comprising 3.5M trajectories reconstructed from years of human gameplay in the complex decision-making task of Competitive Pokémon.
   Context: PokéChamp Karten et al. (2025) concurrently released a dataset of Pokémon battles. The datasets differ in that:
   - Ours covers all available data (2014-Present) for a smaller list of popular game modes. This provides more demonstrations per mode and explores the challenges of learning from strategies that evolve over time.
   - Ours is distributed in a flexible RL format that allows for customization of observations, actions, and rewards outside of LLM prompts.
   - Ours reconstructs the agent's partially observed perspective from spectator data with more accuracy thanks to a custom state-tracking and prediction pipeline designed for this purpose. Further discussion is provided in Appendix D and in our open-source release.
2. We demonstrate our dataset's ability to produce sequence policies that play Competitive Pokémon at a human level.
   Context: Prior work has used online self-play and heuristic search to build successful Pokémon agents in other rulesets.

Human-Level Competitive Pokémon via Scalable Offline Reinforcement Learning with Transformers

Jake Grigsby, Yuqi Xie[†], Justin Sasek[†], Steven Zheng[†], Yuke Zhu
{grigsby,yukez}@cs.utexas.edu

The University of Texas at Austin

[†] Equal contribution. Order determined by a surprise Pokémon tournament.

Abstract

Competitive Pokémon Singles (CPS) is a popular strategy game where players learn to exploit their opponent based on imperfect information in battles that can last more than one hundred stochastic turns. AI research in CPS has been led by heuristic tree search and online self-play, but the game may also create a platform to study adaptive policies trained offline on large datasets. We develop a pipeline to reconstruct the first-person perspective of an agent from logs saved from the third-person perspective of a spectator, thereby unlocking a dataset of real human battles spanning more than a decade that grows larger every day. This dataset enables a black-box approach where we train large sequence models to adapt to their opponent based solely on their input trajectory while selecting moves without explicit search of any kind. We study a progression from imitation learning to offline RL and offline fine-tuning on self-play data in the hardcore competitive setting of Pokémon's four oldest (and most partially observed) game generations. The resulting agents outperform a recent LLM Agent approach and a strong heuristic search engine. While playing anonymously in online battles against humans, our best agents climb to rankings inside the top 10% of active players. All agent checkpoints, training details, datasets, and baselines are available at metamon.tech.

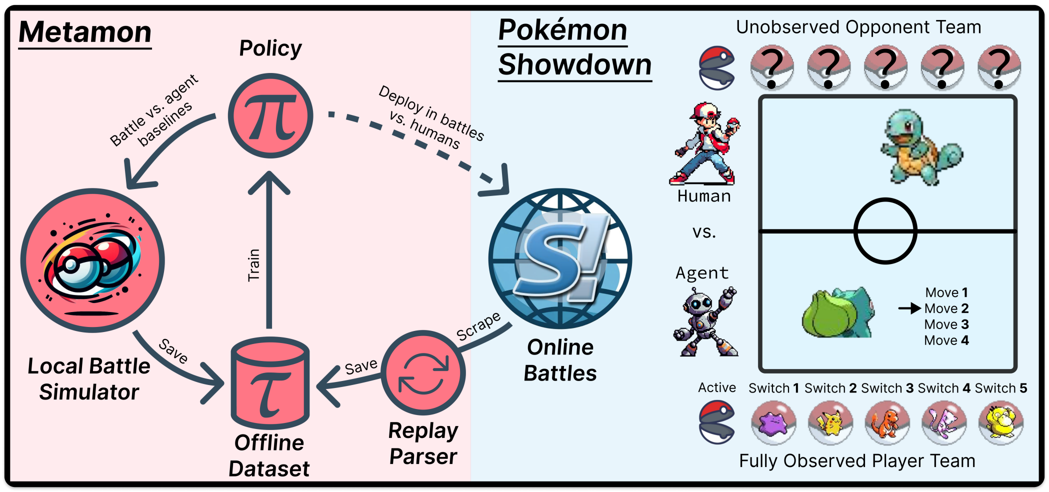


Figure 1: **Batch Training and Evaluation in CPS.** We develop a platform called Metamon that enables an offline RL workflow on a dataset of human gameplay from Pokémon Showdown.

1 Introduction

Competitive Pokémon (Singles) (**CPS**) is a two-player strategy game that combines the long planning horizons of chess with the imperfect information, opponent modeling, and stochasticity of poker — and then adds so many named entities and niche gameplay mechanics that it takes an encyclopedia to document them all. In CPS, players construct teams from billions of possibilities and battle against an opponent. On each turn of the battle, players can choose to use a move from the Pokémon already on the field or switch to another member of their team (Figure 1 Right). Moves can deal damage to the opponent, eventually causing it to faint, until the last player with active Pokémon wins. CPS AI is an exciting Reinforcement Learning (RL) problem because it requires reasoning under uncertainty in an incredibly large state space. The best Pokémon AI relies on heuristic search in custom simulators (Mariglia, 2019) or test-time Monte Carlo tree search with self-play (Wang, 2024). Notably, Competitive Pokémon is played on a website that saves turn-by-turn records of battles dating back over a decade. We develop a pipeline to convert these logs to the partially observed point-of-view of an agent playing against humans in official ranked battles, thereby unlocking a naturally occurring source of offline RL data (Lange et al., 2012) that grows larger every day. Our "reconstruction" process is specific to CPS and will create further CPS-specific problems that RL will need to overcome. At a high level, though, it is an example of a challenge that may arise when using existing data to kickstart a data flywheel. There are applications of RL (healthcare, finance) where lots of data *surrounding* the problem exists (patient records, time series) but is not formatted as trajectory data from the point-of-view of an agent, and any conversion to this format would open up a "sim-to-real" gap between the reconstructed (PO)MDP and the real world.

Our dataset enables a general perspective on the CPS AI problem that has previously been impractical: that sequence models might be able to learn to play without explicit search or heuristics by using model-free RL and long-term memory to infer their opponent's team and tendencies. Our experiments take this perspective to its extreme and create a case study in the process of training and evaluating large policies (Fig. 1 Left). We develop a suite of heuristic and imitation learning (IL) opponents for offline evaluation with procedurally generated Pokémon teams. With these opponents as a benchmark, we evaluate Transformers (Vaswani et al., 2017) of up to 200M parameters trained by IL and offline RL. When deployed in ranked battles against human players in the highly competitive realm of CPS's first four generations — where battles are longest and reveal the least information about the opponent's team — our largest RL policy is officially estimated to have a 41-58% chance to defeat a randomly sampled opponent (depending on the generation). Rather than waiting for more data to accumulate in our dataset, we explore the idea that our models would benefit from training on intentionally unrealistic self-play data that does not attempt to recreate the unknown distribution of teams and opponents in online battles. The resulting agents improve to win rates of 64-80% — rising into the top 10% of active usernames and onto the global leaderboards. A recent LLM Agent (Hu et al., 2024) proves uncompetitive in the long horizons of the early generations, and our best agents match or surpass the strongest heuristic search engine.

2 Background: Competitive Pokémon Singles

If the reader is unfamiliar with Competitive Pokémon , it is difficult to overstate how complicated top-level strategy can be. The game combines opponent modeling with stochastic transitions, complex dynamics, long-horizon planning, and a large initial state space. Pokémon is highly stochastic, and gameplay revolves around nuanced mechanics with endless edge cases. CPS is played on Pokémon Showdown (**PS**) — a website with thousands of daily players. PS simulates the combat mechanics of each major commercial game release (or "generation"). Some fundamentals transfer, but competitive play relies on details specific to each generation. PS divides generations into "tiers" that enforce various rules to maintain competitive balance. Each tier of each generation is its own game — or rather, two games played consecutively: team *design* and *control*. Players design teams before they are matched against an opponent and make trade-offs to counter threats they believe they may face. Team design converges to an equilibrium that helps narrow the search to perhaps many thousands of meaningfully distinct teams that are considered competitively viable.

In addition to navigating Pokémon's randomness, team control (battling) focuses on decision-making under imperfect information. Details of the opponent's Pokémon are only revealed when they directly impact the battle. We can gain an advantage by inferring our opponent's team based on what they have already revealed. For example, we might know that Pokémon A is often used alongside Pokémon B and that Pokémon A commonly brings move x or y but rarely brings both. We may try to mislead our opponent by revealing information that suggests one team design only to surprise them later in the battle. Players make (most) decisions simultaneously. Accurately predicting the opponent's choices based on their team and previous tendencies is the key skill that differentiates high-level players. For example, a move may win the battle but only be safe to select if we believe our opponent will switch their Pokémon on this turn. In short, Pokémon players are constantly updating a prior over the opponent's team and strategy to improve their decision-making.

There are three player metrics on PS. **ELO** is a standard rating system, but PS's version is intentionally noisy, and ELO is not comparable across game modes. **Glicko-1** is an ELO-like rating that considers the full history of a player's battles and is a much better estimate of true skill for our purposes. The matchmaking system on PS prefers to pair players with similar ELO ratings. **GXE** corrects for this matchmaking bias to estimate a player's odds of defeating a randomly sampled opponent. Pokémon has the kind of inherent variance that would be familiar to Heads-Up No-Limit Texas Hold'em players: minimizing risk is considered a key skill, but some losses are inevitable. The very best players have a GXE between 74-90$\%$ (Figure 2 Right).

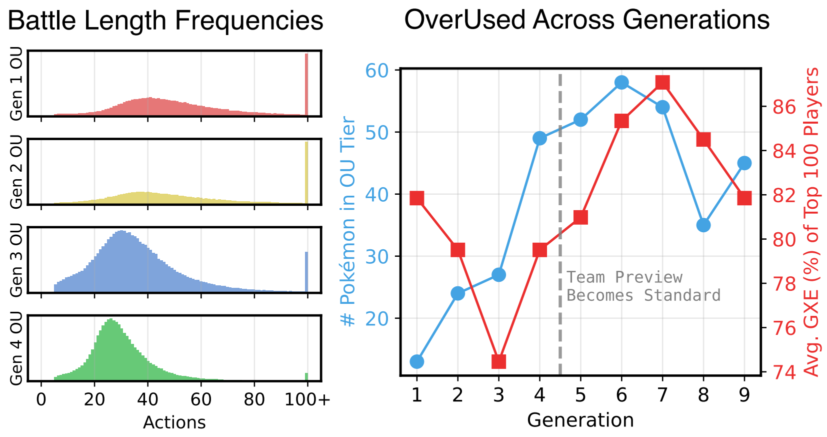


Figure 2: **Episode Length, Team Diversity, and Variance by Gen.** Battle lengths are based on our replay dataset and binned with a max length of 100. GXE statistics are captured in February 2025.

AI research in PS faces the question of which generation and tiers to study. The standard choice is the most recent generation's "random battles" tier. Random battles remove team design by providing each player with a procedurally generated team. This ruleset has a more casual player base, and we will focus on formats where players design teams tailored to their playstyle. **Our agents will learn to play four different tiers, but evaluations will focus on "OverUsed" (OU)**. OU is the definitive competitive format, making it the most popular and, therefore, the tier with the most data to learn from (Section 3). Broadly speaking, each generation of OU increases the number of team combinations and gameplay mechanics (Figure 2 Right). Importantly, the size of the team space creates so much variance from Generation 5 onwards that PS adopts a mechanic called "team preview" that reveals the opponent's team before the start of the battle. We are particularly interested in the partial observability of CPS. For this reason, we focus on the first four generations.

Early Generation OverUsed. In addition to their signature lack of team preview, the early generations of CPS are defined by their unique gameplay mechanics and outlier battle lengths (Fig. 2 Left). Gen1 and Gen2 are infamously stochastic, and reduced offensive power shifts focus away from team composition and towards battle strategy over long exchanges. Gen3 is notable for its enduring popularity and competitive balance — with a narrow margin between median and top-level players by GXE (Fig. 2 Right). Gen4 resembles modern versions in that many Pokémon can eliminate their opponent in a single move; the fast pace of play leads to high-stakes decisions over short planning depths. The early generations are an almost independent competitive community with a long history and a relatively small but self-selective player base. The people we will be playing against have intentionally sought out the competitive format of a $15+$ year-old game because it is their interest and expertise. There are few casual players here; many of the "low rated" usernames we will face are experienced players logged into alternate accounts for various reasons. Appendix C finds that a heuristic using basic Pokémon principles and lookup tables is far less effective against human players in early-generation OU than modern random battles.

While our use of model-free long-context RL and focus on Early Gen OU are novel, there is existing work on AI for CPS. The best Pokémon bots focus on heuristic tree search with custom high-throughput simulators. Some work has experimented with network-based state evaluation and Monte Carlo tree search (MCTS) (Browne et al., 2012) for random battles formats (Huang & Lee, 2019). Pokémon is primarily played and discussed on the internet, and this affords considerable gameplay knowledge to recent LLM-Agent techniques (Hu et al., 2024; Karten et al., 2025). Key baselines will be discussed as we play against them in Section 5. Appendix A provides a survey of AI in CPS, while Appendix B discusses related work in offline RL and gameplaying.

3 Building an Offline RL Dataset of Real Human Battles

PS creates a log ("replay") of every battle that expires after a brief period unless saved. Players save replays for later study, to share a fun outcome with friends, or as a way to record official tournament results. PS has been the home of Competitive Pokémon for over a decade — time enough to accumulate millions of replays. The PS replay dataset is an exciting source of naturally occurring data. However, there is a critical problem: CPS decisions are made from the partially observed point-of-view of one of the two competing players, but PS replays record the perspective of a third-party spectator who has access to information about neither team. We unlock the PS replay dataset by converting spectator views to each player's perspective separately.

Replay reconstruction involves four high-level steps. First, we **simulate** the current state of the battle from a spectator perspective according to the PS API. Throughout this process, we use incoming information to estimate the initial configuration of both unobserved teams. At the end of the battle, we **infer** any information that was never revealed. To do this, we need a way to model the distribution of competitive teams in each generation and tier. Fortunately, the PS community tracks Pokémon usage statistics to measure trends and evaluate rule changes. We use available usage data and the revealed teams of similar replays to model the distribution of human-constructed teams. Next, we **backfill** inferred team rosters for a chosen point-of-view player to replicate the information they would have observed when their decisions were made. Finally, we **convert** the reconstructed trajectory to a format identical to the online simulator. Appendix D walks through a simplified example and uses a real replay to visualize the raw input, inferred team, and trajectory output according to the observation space, action space, and reward function discussed in the next section.

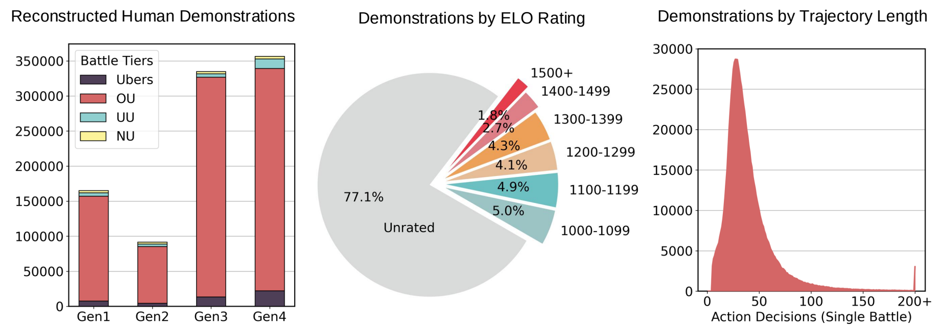


Figure 3: **Dataset Summary.** The initial version of our offline dataset includes 475k battles — summarized here by their PS format (left), ELO rating (center), and length in agent timesteps (right).

This process is not always successful, as some gameplay mechanics cannot be reconstructed from incomplete information. A list of checks identifies trajectories that have entered ambiguous situations and conservatively discards them. All told, we are able to download and reconstruct more than 475k human demonstrations (with shaped rewards) from historical Gen 1-4 battles dating back to 2014 (Figure 3). Each battle yields two point-of-view trajectories for a total of about 950k sequences containing 38M timesteps. Player names and chats are anonymized, and trajectories are stored in a flexible format that lets researchers customize observations, actions, and rewards. Our pipeline is actively downloading new battles and has recently expanded to include Gen 9 OU, bringing the total to 3.5M trajectories. However, the experiments in this paper use the original 950k-trajectory dataset, with a cutoff of September 2024.

4 Search-Free Pokémon with Offline RL On Sequence Data

Players discuss and teach the game based on the idea that their decision-making policy π is conditioned on their current estimate of their opponent's policy (π_o) and team *c*omposition (c_o). Let c_p be our own team composition. This paper will take a Bayesian RL (Ross et al., 2007; Ghavamzadeh et al., 2015) or meta-RL (Beck et al., 2023) perspective where we consider our opponent's choices part of the environment's unknown transition function $T(s_{t+1} \mid s_t, a_t, \pi_o)$ (Zintgraf et al., 2021a). Our goal is to find a policy that maximizes return over some distribution of latent environment variables, which in our case would be the opponents active on PS and our distribution of teams:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi_o,\, c_o \sim p(\pi_o, c_o), c_p \sim p(c_p)} \left[\mathbb{E}_{\tau \sim p(\tau \mid \pi, \pi_o, c_o, c_p)} \left[\sum_{t=0}^{T} \gamma^t \, R(s_t, a_t) \right] \right] \tag{1}$$

Context-based methods condition the policy on estimates of the unobserved variables derived from previous experience. Here, this would amount to using the entire history of a battle[1] (observations, rewards[2], and the actions of both players) to estimate (c_o, π_o). If we want to avoid explicitly predicting c_o or π_o (Humplik et al., 2019) (which is difficult to formulate) or modeling the complicated dynamics of Pokémon (Zintgraf et al., 2021b), we can follow a simple black-box framework (Duan et al., 2016; Wang et al., 2016) where a sequence model S_θ takes all prior experience under the current latent variables (the entire battle up until the current timestep, $\tau_{0:t}$) as input and outputs a representation h_t for the policy network π_ϕ. The system is trained end-to-end to maximize Eq. (1) as in standard deep RL. Because a better estimate of the opponent will increase win rate, the sequence model will implicitly learn that behavior. The policy navigates an exploration-exploitation trade-off at test time, where it may take actions that reveal new information if this increases expected returns.

We will be using the offline dataset ($\mathcal{D}$) from Section 3 to approximate the expectations in Eq. (1), which assumes that the distribution of teams and playstyles across history is identical to that of the current game (Dorfman et al., 2020; Li et al., 2024). This is false, but it may be close enough, particularly in the optimized world of Early Gen OU. If we want to expand our dataset (i.e., by self-play), we need to try to select teams and opponents that match the true distribution. Alternatively, we can collect data that is *unambiguously* out-of-distribution (OOD). For example, we can place a rare Pokémon in the lead-off position so that when the policy begins a real battle and sees a more standard choice, it has no reason to believe it is facing our synthetically generated teams or opponents.

Pokémon has a complex state space, and our policy may need to be large and non-trivial to train with offline RL. To stabilize, we can frame the problem from a behavior cloning (BC) perspective: predicting the actions of a human player requires reasoning about the strategy of the player we are imitating and their understanding of the opponent. Accurate predictions will require long context inputs. RL is a tool to sort through the noise of a large dataset that includes the decisions of all levels of players in both competitive and casual settings. We arrive at the same setup but prefer an update that safely reduces to BC while allowing room to skew the loss function towards return-maximizing behavior if we decide the offline RL risks are sufficiently small (Springenberg et al.,

[1]A natural extension of the context-based framework here would include previous battles between the same players alongside their current battle. This may allow for adaptation in a tournament best-of-three match format.

[2]Because our Pokémon reward function never changes, it would be considered part of the state space and happens to be important for inferring the *outcome* of the previous turn in our setup.

2024; Wu et al., 2019; Fujimoto & Gu, 2021). Ideally, BC becomes a lower bound upon which we can improve. Solutions of this kind are actor-critics that train their critic to output Q-values with standard one-step temporal difference backups. Actor loss functions take the general form:

$$\mathcal{L}_{\text{Actor}} = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\frac{1}{T} \sum_{t=0}^{T} \left(-w(h_t, a_t) \log \pi(a_t \mid h_t) - \lambda \mathbb{E}_{a \sim \pi(\cdot \mid h_t)} \left[Q\left(h_t, a\right) \right] \right) \right] \tag{2}$$

Where h_t is the output of the sequence model $S_\theta(\tau_{0:t})$ that replaces the state. The first term is a BC objective that re-weights decisions according to a function w and constrains learning to actions taken in the offline dataset (Wang et al., 2020; Nair et al., 2020). The second term is the standard online off-policy actor update that risks overestimating the value of OOD actions when used offline (Kumar et al., 2019). Our experiments will study configurations of Equation (2) summarized by Table 1. For further discussion of RL engineering details, we refer the reader to the AMAGO (Grigsby et al., 2024a) implementation used throughout our experiments.

Model Name	$w(h, a) =$	λ
"IL"	1	0
"Exp" (or just "RL")	$\exp(\beta A^{\pi}(h, a))$ (clipped)	0
"Binary"	$A^{\pi}(h, a) > 0$	0
"Binary+MaxQ"	$A^{\pi}(h, a) > 0$	> 0

Table 1: $\mathcal{L}_{\text{actor}}$ **Configurations (Eq. (2)).** Advantages are estimated by the critic: $A^{\pi}(h, a) = Q(h, a) - \mathbb{E}_{a' \sim \pi}[Q(h, a')]$.

Next, we need to define an observation space, action space, and reward function for CPS. Our agent needs enough information to mirror human decisions, and the user interface of the PS website is an obvious point of reference. However, our models have memory, and we do not need to provide all of this information at every timestep. We have a trade-off between dimensionality, memory difficulty, generalization over Pokémon's complex dynamics, and exposure to sim2real errors between replay reconstruction and deployment. We settle on a compromise of 87 words of text and 48 numerical features. The text component is semi-readable, and Figure 5 provides an example from a replay in our dataset. **The most important detail is that we are relying entirely on memory to infer the opponent's team**; observations only include the opponent's active Pokémon. The memory demands of our CPS observations are more comparable to those in the commercial video games than the PS web interface. We are confident in our sequence models' ability to recall previous timesteps, and this makes it worth avoiding distribution shift over features of the opponent's full team as it is slowly revealed. There are nine discrete actions, where the first four indices correspond to the moves of the active Pokémon, and the remaining five switch to another team member. The observation conveys the precise meaning of these actions in a predictable order. The reward function is dominated by binary win/loss but includes light shaping for damage dealt and health recovered. Appendix E provides more details.

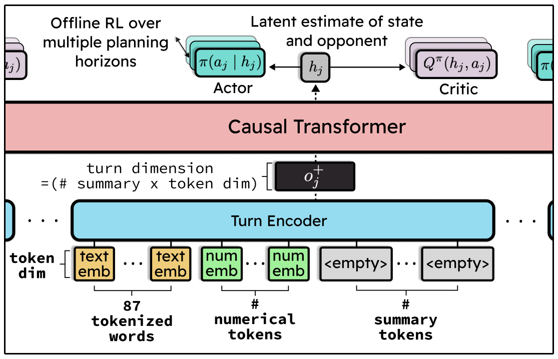
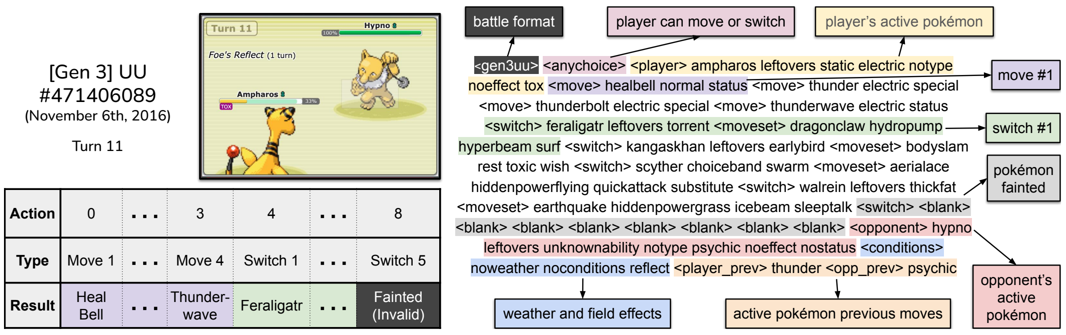


Figure 4: **Model Overview.** Actions are predicted based on representations of the observation, action, and reward of each turn in the current battle.

The observation, previous action, and previous reward at each timestep are processed by a Transformer encoder that uses designated summary tokens to attend over the multi-modal sequence (Devlin et al., 2019). Text is encoded by tokenizing the Pokémon vocabulary based on our dataset with an `<unknown>` token for rare cases we may have missed[3]. The resulting sequence of turn representations is the input to a causal Transformer with actor and critic output heads (Figure 4).

[3]We experiment with an augmentation scheme that sets tokens to `<unknown>` to force recovery from previous timesteps. Models above 100M parameters use this strategy by default, while its use in smaller models is indicated by "Aug." We do not find evidence that this strategy impacts performance.



Action	0	. . .	3	4	. . .	8
Type	Move 1	. . .	Move 4	Switch 1	. . .	Switch 5
Result	Heal Bell	. . .	Thunder-wave	Feraligatr	. . .	Fainted (Invalid)

<gen3uu> <anychoice> <player> ampharos leftovers static electric notype noeffect tox <move> healbell normal status <move> thunder electric special <move> thunderbolt electric special <move> thunderwave electric status <switch> feraligatr leftovers torrent <moveset> dragonclaw hydropump hyperbeam surf <switch> kangaskhan leftovers earlybird <moveset> bodyslam rest toxic wish <switch> scyther choiceband swarm <moveset> aerialace hiddenpowerflying quickattack substitute <switch> walrein leftovers thickfat <moveset> earthquake hiddenpowergrass icebeam sleeptalk <switch> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <opponent> hypno leftovers unknownability notype psychic noeffect nostatus <conditions> noweather noconditions reflect <player_prev> thunder <opp_prev> psychic

Figure 5: **Observation and Action Space.** Text order is important, but words can be tokenized into arrays with a consistent length (of 87). Observations also include 48 numerical features. The meaning of each action index varies by turn but is presented in the text in a consistent order.

5 Experiments

We will begin evaluating a progression of increasingly RL-heavy training objectives across model architectures with "Small" (15M), "Medium" (50M), and "Large" (200M) parameter counts summarized by Table 3. Models are named in results according to their size and training objective (Table 1). Table 4 provides a complete list of model configurations. Results will be discussed in semi-chronological order, though some figures will spoil win rates of models trained on "synthetic" self-play datasets described in Section 5.3. Our goal is to compete against human players, but this is expensive and creates a challenging evaluation problem: Which model checkpoints do we deploy on PS? Our efforts to answer this question result in extensive evaluations against various opponents.

Training uses the offline dataset to assign our players' teams, but we need to "prompt" our agents with a set of teams during evaluations. We use three sets: 1) The **Variety Set** procedurally generates 1k intentionally diverse teams per gen/tier and will be used to evaluate OOD gameplay and to generate unambiguous self-play data as mentioned in Section 4. 2) The **Replay Set** approximates the choices of top players based on their replays and infers unrevealed details as done in Section 3. 3) The **Competitive Set** comprises 10-20 complete "sample" teams per gen/tier scraped from forum discussions; these are generally designed for beginners by experts. Win rates are measured over large samples of hundreds or thousands of battles unless otherwise noted. Evaluations use `poke-env` (Sahovic, 2020) to interact with a locally hosted PS server and the public website.

5.1 Heuristic Evaluations

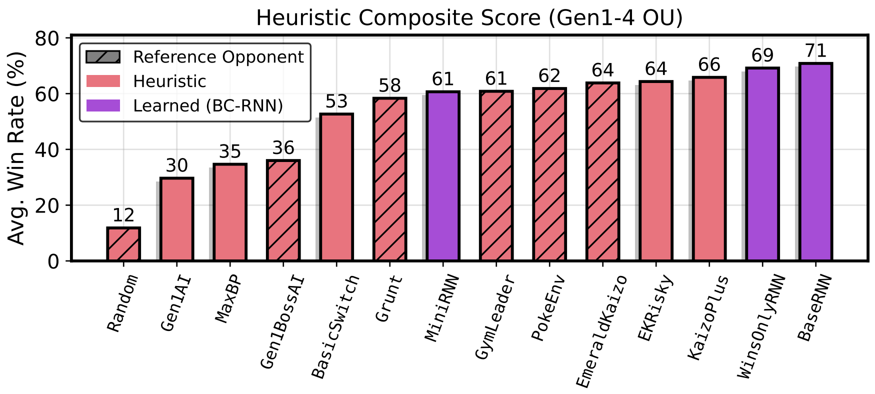


Figure 6: **Heuristic Composite Scores.** The average win rate against six of our heuristics measures core game knowledge and creates a relatively fixed point of reference across different game modes.

We create a suite of a dozen heuristic opponents that evaluate core game knowledge. Strategies are based on fundamental Pokémon concepts and re-implementations of policies from official versions of Pokémon, fan-made ROM hacks with inflated difficulty, and popular CPS AI baselines. Full descriptions of these policies and their relative performance are provided in Appendix C. The average win rate against 6 of these heuristics on the Variety Set forms a "Heuristic Composite Score" (Figure 6). We tune the Turn Encoder architecture (Fig. 4) with RNN trajectory models S_θ between 500k-4M parameters trained by BC. Appendix F.1 documents the predictive accuracy of these models and provides further details. The best BC-RNN models lead the early Heuristic Composite rankings, and these will become the next rung on the ladder toward human-level gameplay. Clear signs of underfitting motivate the starting point of 15M for our Transformer agents. While we will go on to saturate this benchmark in OU, heuristics represent a fixed target unaffected by the discrepancies in data availability between OU and the other three tiers our agents are trained to play (Fig. 3). Figure 7 documents a predictable decline from OU to NeverUsed (NU) gameplay. We evaluate many variants of the $\mathcal{L}_{\text{actor}}$ objective (Eq. 2) but do not find significant differences between them.

5.2 Model-Based Evaluations

Appendix F.1 evaluates our larger Transformer models against our best RNN baseline. RL updates significantly outperform the pure-BC Transformers, but there is little difference between the many RL variants considered. The expected relationship between model size and performance is clearer for BC than it is for RL. Following Grigsby et al. (2024a), we are optimizing actor and critic network outputs for a set of γs in parallel. At test time, we are able to select the action corresponding to any of these horizons. Figure 8 verifies that our agents are using long-term value estimates to improve their win rate. All other evaluations follow the policy for $\gamma = .999$. With RL comfortably outplaying our smaller IL baselines on the more limited Competitive Team Set, we shift to playing against Large-IL on the Replay Set. Figure 9 highlights the win rate of key models in OU.

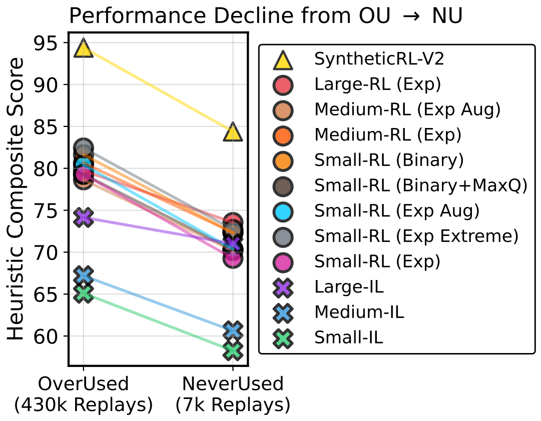


Figure 7: **OU → NU.** Heuristics highlight a gap between OU tiers and those with fewer replays. OU scores are directly comparable against Fig. 6.

5.3 Synthetic Data from Self-Play

Section 5.5 will find that our offline dataset yields policies capable of human-level gameplay on the public ladder. Our agents contribute to each day's batch of new replays and grow the dataset alongside human players. In principle, we could wait to retrain new policies on a larger dataset, but this data is not making a significant difference on the timescale of a single project. We can speed up the process by deploying agents on a local PS ladder, adding their trajectories to the human gameplay dataset, and retraining or fine-tuning (Figure 1 Left). However, we need to be wary of a shift between the frequency of teams and opponents implied by the new offline dataset and the true distribution on PS. One approach would be to try and generate data that is clearly different from the original set so that when conditioned on a real battle, our model's implicit estimate of $p(\pi_o, c_o \mid \tau_{0:i})$ should be unchanged at small i. We let a mix of checkpoints from all our agents compete on a locally hosted PS ladder, playing with teams from the Variety Team Set. By prioritizing diversity over realism, we hope this data will cover replay reconstruction failures and improve model-free learning of Pokémon's stochastic transitions without biasing estimates of human teams and strategies.

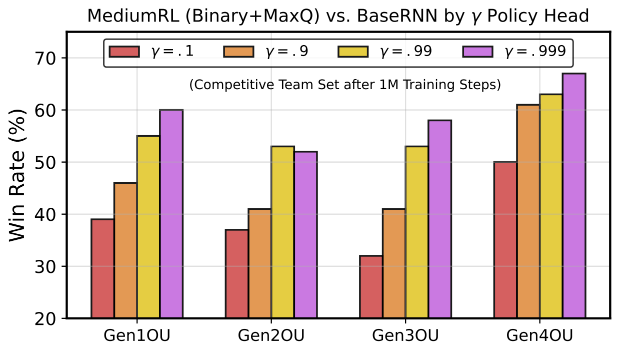


Figure 8: **Multi-γ Policies.** Models train over multiple value horizons, but long-term planning increases win rate.

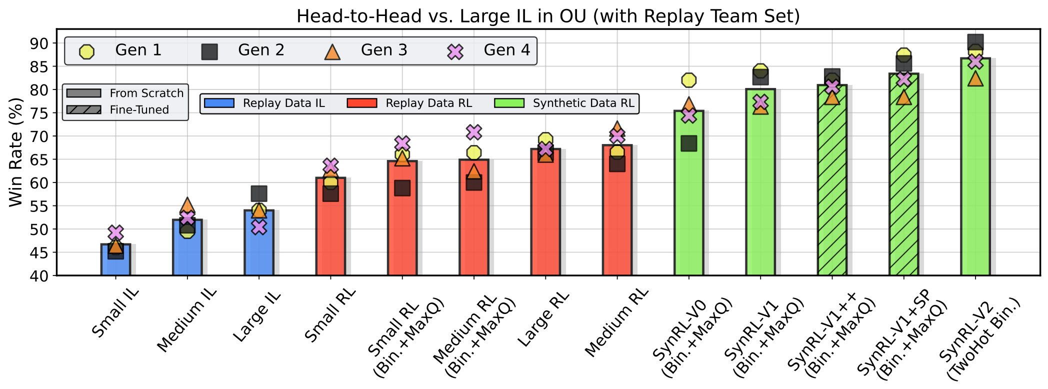
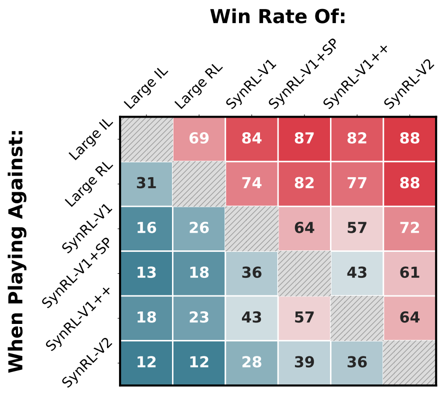


Figure 9: **Self-Evaluation Against Large IL.** Results are determined by the best checkpoint over the last 200k training steps with a sample size of 500 battles per generation.

The SyntheticRL (SynRL) models are Large Binary+MaxQ (Eq. (2)) policies trained from scratch. **SyntheticRL-V0** trains on "synthetic" variety data for generations 1 and 3 only, for a total dataset size of 2M trajectories. It is a promising improvement over our previous policies against heuristics (Fig. 28), BC-RNN (with win rates as high as 95% in Gen1OU and 85% in Gen3OU), and Large-IL (Fig. 9). **SynRL-V1** takes this dataset and adds generations 2 and 4 to reach a total of 3M trajectories (retraining a 200M policy from scratch) for a consistent improvement across generations.

We might wonder whether the caution of the "synthetic" data process was necessary. We test this by letting SynRL-V1 battle recent checkpoints of itself with the more realistic Replay Set until the offline dataset is 5M trajectories. Afterward, we resume training for another 200k gradient steps to create **SynRL-V1+SelfPlay (SP)**. As expected, the resulting model is significantly better against itself (Figure 10), and a key baseline in Section 5.4, but this will translate to inconsistent improvement against real players in Section 5.5. Battle replays make it clear that the model believes it is playing SynRL-V1. We backtrack and expand the dataset to 5M with unrealistic teams and IL opponents and fine-tune SynRL-V1 again to create **SynRL-V1++**. Finally, we train a new 200M model from scratch on the SynRL-V1++ dataset with 50k new human replays. Instead of Binary+MaxQ, we use a simple binary weighted BC update with value prediction converted to two-hot classification (Schrittwieser et al., 2020; Hafner et al., 2023; Farebrother et al., 2024) as implemented in this setting by Grigsby et al. (2024b). This trick is often motivated by hyperparameter insensitivity and invariance to return magnitudes in multi-task RL. In our case, improved critic accuracy leads to an entirely new level of pessimism in the binary BC filter (Figure 25) — a potential improvement considering our dataset is now primarily composed of decisions made at beginner or intermediate human levels (Sec. 5.5). The resulting **SynRL-V2** model is our best by every metric (against heuristics, other models, and key external baselines yet to be discussed).

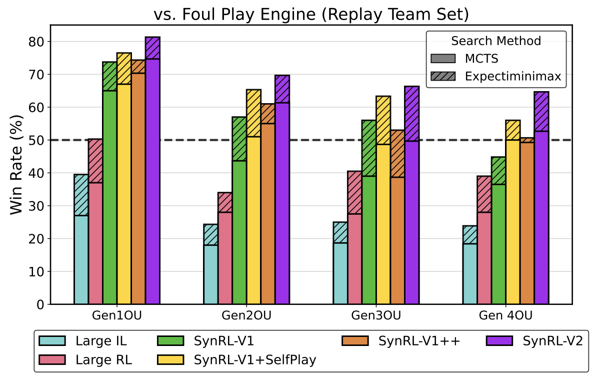


	Large IL	Large RL	SynRL-V1	SynRL-V1+SP	SynRL-V1++	SynRL-V2
Large IL	▨	69	84	87	82	88
Large RL	31	▨	74	82	77	88
SynRL-V1	16	26	▨	64	57	72
SynRL-V1+SP	13	18	36	▨	43	61
SynRL-V1++	18	23	43	57	▨	64
SynRL-V2	12	12	28	39	36	▨

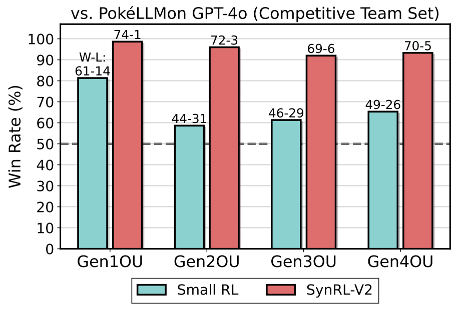
Figure 10: **Gen1OU Self-Play.** Sample of 500 battles on the Replay Set.

5.4 LLM Agents and Heuristic Search

Foul Play (Mariglia, 2019) is an advanced engine for CPS that uses a custom simulator to search over Pokémon's game tree. With extensive domain knowledge, it implements much of the behavior we would hope our policies can learn from data. For example, it infers its opponent's team during battles using PS usage statistics, much like we do during dataset construction. A January 2025 update to Foul Play introduced support for the early generations. We challenge the engine to matches of 300 battles per generation on the Replay Team Set, with results shown in Figure 11a. We manage to play the best version of the bot to a draw in Gens 3 and 4 (where the effective search depth would be lowest), and outperform it in the long horizons of Gens 1 and 2. PokéLLMon (Hu et al., 2024) is a more general approach that takes advantage of Pokémon's extensive web presence to build an LLM-Agent. Prompts are constructed with domain knowledge such as Pokémon type matchups and move descriptions, and the LLM is tasked with deciding between the available moves. Hu et al. (2024) evaluate in a random battles tier and note that the agent struggles with long-term planning; this effect is much more noticeable in the longer battle lengths of Gen1-4 (Figure 11b).



(a) **Foul Play Evaluation.** Using both available search algorithms and `poke-engine` v0.31.0. Sample of 300 battles.



(b) **PokéLLMon**. GPT-4o backend with custom prompts for Gen1-4. Sample of 75 battles.

5.5 Playing Humans On the Pokémon Showdown Ranked Ladder

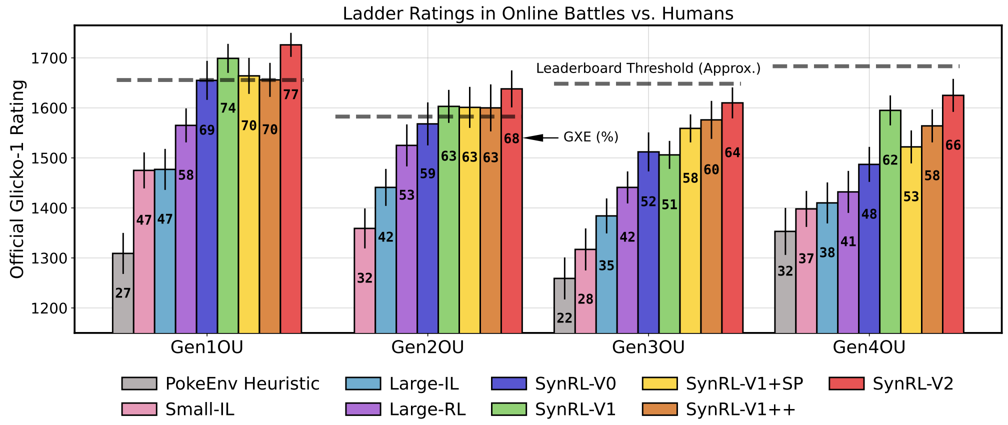


Figure 12: **Human Evaluations.** We visualize the Glicko-1 ladder rating (with its rating deviation). Bar labels represent GXE statistics. To compare across generations, we plot a heuristic baseline's performance and the average Glicko-1 of the bottom 100 players on the Top 500 global leaderboard.

We compete against human players by queuing for ranked battles on the public PS ladders. We evaluate our agents over periods of 4-8 days — frequently switching between generations to sample a wider variety of opponents and achieve large sample sizes of at least 400 battles. Evaluations run from late December 2024 through late March 2025. Models' Glicko-1 and GXE stats at the end of their final battle are shown in Figure 12. We include the results of a heuristic agent for additional context. Figure 14 converts ladder statistics to a percentile among active usernames. Percentiles are more interpretable without a CPS background, but the distribution of player stats necessary to compute them is *not* public information; PS only displays the ratings of the top 500 active usernames. However, we are able to create a reasonable estimate because our dataset is reconstructing (most) battles played over the latter half of the evaluation period. We recover the ratings of all the unique usernames that are active enough to have a Glicko-1 deviation $\leq \pm 100$. This metric is still not ideal because players frequently use multiple usernames. Top players have clear competitive reasons to make new accounts, but we are unable to account for this. The evaluations of SynRL-V1++ and SynRL-V2 in Gen1OU are impacted by a weeks-long tournament that requires participating (top) players to make new accounts and leads to massive rating deflation in our high skill bracket[4].

The Large-RL model rises to the level of an intermediate player and is favored to win against a randomly selected opponent in Gens 1 and 2. High-variety self-play data leads to dramatic improvements over the course of our work. Our SynRL-V2 model is a reasonably advanced player estimated to be inside the top decile across generations. Although ELO ratings are noisy, the SynRL-V1 and SynRL-V2 models reach peak global rankings of #46 and #31 in Gen1OU, respectively, and SynRL-V2 makes two appearances inside the top 300 in Gen3OU. All RL models sit inside the top 500 in Gen2OU. To the best of our knowledge, this is the first time an AI has achieved *any* of SynRL-V2's ladder ratings in *any* of the Early-Gen OU tiers — and it achieves this without a dynamics model or falling back on Pokémon heuristics while learning to play 16 rulesets at the same time (Appendix A). Qualitatively, our models display human-like gameplay. During our evaluation process, we saved sample replays on the PS website that can be viewed by searching models' usernames (Table 6) at this link. Policies learn to play reasonable openings, make safe Pokémon switches, and anticipate the moves of their opponent. However, our agents occasionally suffer from the accumulating errors we might expect from a sequence policy and can begin to make nonsensical decisions in long battles — particularly when the opponent is playing with a rare team or uncommon strategy. Figure 13 evaluates the impact of memory on the win rate of a policy competing against the full-context-length version of itself.

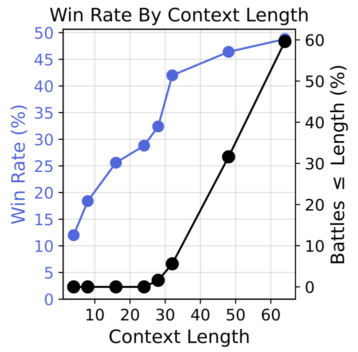


Figure 13: **Memory.** SynRL-V1 battles a version of itself that can recall the entire battle.

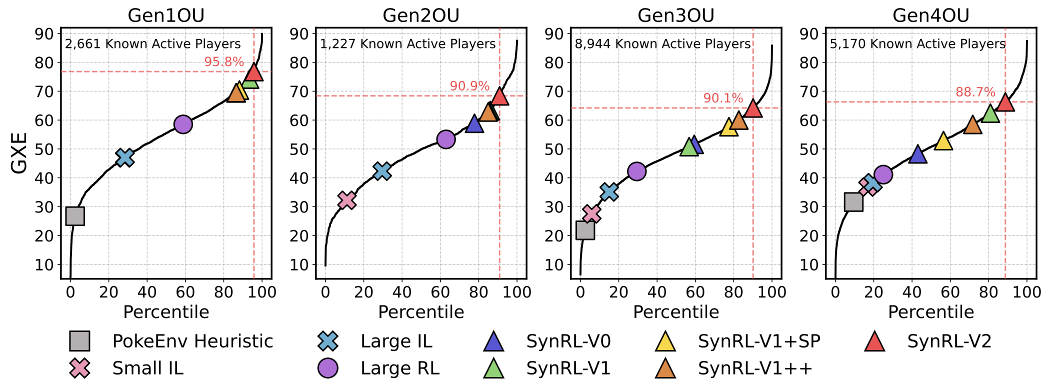


Figure 14: **Ladder Percentiles.** Replays downloaded between Feb-Mar 2025 identify 14022 active Gen1-4 usernames. Using Gen1 as an example, 5095 of these usernames played Gen1OU, while 2661 were active enough to have a valid GXE statistic when results were finalized.

6 Conclusion

Our work enables a scalable offline RL approach to Competitive Pokémon Singles and demonstrates that sequence models trained on historical gameplay can be competitive with humans in the challenging setting of Early-Generation OverUsed. Our PS trajectory dataset will continue to grow over time and may be of broader interest in offline RL as a way to evaluate new research on a complex task. We hope our dataset and baseline models will inspire research interest in Competitive Pokémon. Alternative training details and large-scale self-play techniques may create a path to super-human performance. Our code, pretrained models, and datasets are available on GitHub at: `UT-Austin-RPL/metamon`.

[4]SynRL-V2 plays 613 human battles and settles at a Gen1OU GXE of 79.9% (Glicko-1 1761 $\pm$ 35) after more than 100 battles. However, its rating declines over its next 100 games because we stop avoiding a competition where top players are playing with fresh (low-rated) usernames. Figures 12 and 14 conservatively report the final metrics.

Acknowledgments

We would like to give special thanks to Felix You and Emil Velasquez — undergraduates at UT Austin and key early contributors to what would go on to be an unusually long research effort. Thanks also to the `poke-env` and Pokémon Showdown projects, as well as Pokémon communities like Bulbagarden and Smogon. This project would not have been possible without fan-made resources. Our research was supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project), a Sony Research Award, and JP Morgan.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In International conference on machine learning, pp. 104–114. PMLR, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. arXiv preprint arXiv:2301.08028, 2023.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680, 2019.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818, 2023.

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, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. Advances in neural information processing systems, 33:17057–17069, 2020.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1):1–43, 2012. DOI: 10.1109/TCIAIG.2012.2186810.

Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. Artificial Intelligence, 134(1):57–83, 2002. ISSN 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00129-1. URL https://www.sciencedirect.com/science/article/pii/S0004370201001291.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. arXiv preprint arXiv:2101.05982, 2021.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pp. 4171–4186, 2019.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta learning of exploration. arXiv preprint arXiv:2008.02598, 2020.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779, 2016.

FAIR FAIR Diplomacy Team, 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.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. arXiv preprint arXiv:2403.03950, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. Advances in neural information processing systems, 34:20132–20145, 2021.

Quentin Gallouédec, Edward Beeching, Clément Romac, and Emmanuel Dellandréa. Jack of all trades, master of some, a multi-purpose transformer agent. arXiv preprint arXiv:2402.09844, 2024.

Matthias Gerstgrasser, Rakshit Trivedi, and David C. Parkes. Crowdplay: Crowdsourcing human demonstrations for offline learning. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=qyTBxTztIpQ.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. Foundations and Trends® in Machine Learning, 8(5-6):359–483, 2015.

Jake Grigsby, Linxi Fan, and Yuke Zhu. AMAGO: Scalable in-context reinforcement learning for adaptive agents. In The Twelfth International Conference on Learning Representations, 2024a. URL https://openreview.net/forum?id=M6XWoEdmwf.

Jake Grigsby, Justin Sasek, Samyak Parajuli, Ikechukwu D Adebi, Amy Zhang, and Yuke Zhu. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. Advances in Neural Information Processing Systems, 37:87473–87508, 2024b.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. Advances in Neural Information Processing Systems, 33:7248–7259, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. arXiv preprint arXiv:2301.04104, 2023.

Varun Ramesh Harrison Ho, 2014. URL https://varunramesh.net/content/documents/cs221-final-report.pdf.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In International conference on machine learning, pp. 805–813. PMLR, 2015.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. Multi-task deep reinforcement learning with popart. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pp. 3796–3803, 2019.

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.

Dan Huang and Scott Lee. A self-play policy optimization approach to battling pokémon. In 2019 IEEE Conference on Games (CoG), pp. 1–4, 2019. DOI: 10.1109/CIG.2019.8848014.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. arXiv preprint arXiv:1905.06424, 2019.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. arXiv preprint arXiv:2210.03094, 2022.

Yuheng Jing, Kai Li, Bingyun Liu, Yifan Zang, Haobo Fu, QIANG FU, Junliang Xing, and Jian Cheng. Towards offline opponent modeling with in-context learning. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=2SwHngthig.

Akshay Kalose, Kris Kaya, and Alvin Kim. Optimal battle strategy in pokemon using reinforcement learning. Web: https://web. stanford. edu/class/aa228/reports/2018/final151. pdf, 2018.

Seth Karten, Andy Luu Nguyen, and Chi Jin. Pok\'echamp: an expert-level minimax language agent. arXiv preprint arXiv:2503.04094, 2025.

KGS. Kgs go game archives, 2025. URL https://www.gokgs.com/archives.jsp. Accessed: 2025-03-21.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. IEEE transactions on intelligent transportation systems, 23(6):4909–4926, 2021.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. In The Eleventh International Conference on Learning Representations, 2022.

Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, et al. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 7772–7779. IEEE, 2024.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In Reinforcement learning: State-of-the-art, pp. 45–73. Springer, 2012.

Romain Laroche and Rémi Tachet des Combes. Multi-batch reinforcement learning. Proceedings of the 4th Reinforcement Learning and Decision Making (RLDM), 2019.

Dongsu Lee, Chanin Eom, and Minhae Kwon. Ad4rl: Autonomous driving benchmarks for offline reinforcement learning with value-based dataset. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 8239–8245. IEEE, 2024.

Scott Lee and Julian Togelius. Showdown ai competition. In 2017 IEEE Conference on Computational Intelligence and Games (CIG), pp. 191–198, 2017. DOI: 10.1109/CIG.2017.8080435.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

Lanqing Li, Hai Zhang, Xinyu Zhang, Shatong Zhu, Yang Yu, Junqiao Zhao, and Pheng-Ann Heng. Towards an information theoretic framework of context-based offline meta-reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), Advances in Neural Information Processing Systems, volume 37, pp. 75642–75667. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/8a30aba6514b56d02976f49797f6338a-Paper-Conference.pdf.

Lichess. Lichess game database, 2025. URL https://database.lichess.org/. Accessed: 2025-03-21.

P. Mariglia. Foul play - a competitive pokémon ai research project. https://github.com/pmariglia/foul-play, 2019. Accessed: 2025-02-27.

Michaël Mathieu, Sherjil Ozair, Srivatsan Srinivasan, Caglar Gulcehre, Shangtong Zhang, Ray Jiang, Tom Le Paine, Richard Powell, Konrad Żołna, Julian Schrittwieser, et al. Alphastar unplugged: Large-scale offline reinforcement learning. arXiv preprint arXiv:2308.03526, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, 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.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. arXiv preprint arXiv:2006.09359, 2020.

Amna Najib, Stefan Depeweg, and Phillip Swazinna. Iterative batch reinforcement learning via safe diversified model-based policy search. In CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World, 2024,.

Samer Nashed and Shlomo Zilberstein. A survey of opponent modeling in adversarial domains. Journal of Artificial Intelligence Research, 73:277–327, 2022.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 6892–6903. IEEE, 2024.

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.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Transactions on Neural Networks and Learning Systems, 2023.

Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, et al. Scaling instructable agents across many simulated worlds. arXiv preprint arXiv:2404.10179, 2024.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. arXiv preprint arXiv:2205.06175, 2022.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), Advances in Neural Information Processing Systems, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/3b3dbaf68507998acd6a5a5254ab2d76-Paper.pdf.

Max Rudolph, Nathan Lichtle, Sobhan Mohammadpour, Alexandre Bayen, J Zico Kolter, Amy Zhang, Gabriele Farina, Eugene Vinitsky, and Samuel Sokota. Reevaluating policy gradient methods for imperfect-information games. arXiv preprint arXiv:2502.08938, 2025.

H. Sahovic. poke-env: A python interface for training reinforcement learning agents in pokémon battles. https://github.com/hsahovic/poke-env, 2020. Accessed: 2025-02-27.

Yuta Saito, Shunsuke Aihara, Megumi Matsutani, and Yusuke Narita. Open bandit dataset and pipeline: Towards realistic and reproducible off-policy evaluation. arXiv preprint arXiv:2008.07146, 2020.

Nicholas R. Sarantinos. Teamwork under extreme uncertainty: Ai for pokemon ranks 33rd in the world, 2023. URL https://arxiv.org/abs/2212.13338.

Martin Schmid. Search in imperfect information games. arXiv preprint arXiv:2111.05884, 2021.

Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G Zacharias Holland, et al. 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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

Sam Shleifer, Jason Weston, and Myle Ott. Normformer: Improved transformer pretraining with extra normalization. arXiv preprint arXiv:2110.09456, 2021.

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.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Bechtle, Steven Kapturowski, Roland Hafner, et al. Offline actor-critic reinforcement learning scales to large models. arXiv preprint arXiv:2402.05546, 2024.

David Stone, 2010. URL https://github.com/davidstone/technical-machine.

Gerald Tesauro. Temporal difference learning and td-gammon. Commun. ACM, 38(3):58–68, March 1995. ISSN 0001-0782. DOI: 10.1145/203330.203343. URL https://doi.org/10.1145/203330.203343.

Dhruva Tirumala, Thomas Lampe, Jose Enrique Chen, Tuomas Haarnoja, Sandy Huang, Guy Lever, Ben Moran, Tim Hertweck, Leonard Hasenclever, Martin Riedmiller, et al. Replay across experiments: A natural extension of off-policy rl. arXiv preprint arXiv:2311.15951, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, 575(7782):350–354, 2019.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. arXiv preprint arXiv:1611.05763, 2016.

Jett Wang. Winning at pokémon random battles using reinforcement learning. Master of engineering thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2024. Submitted to the Department of Electrical Engineering and Computer Science.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. Advances in Neural Information Processing Systems, 33:7768–7778, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361, 2019.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M Susskind. Stabilizing transformer training by preventing attention entropy collapse. In International Conference on Machine Learning, pp. 40770–40803. PMLR, 2023.

Alex Zhang, Ananya Parashar, and Dwaipayan Saha. A simple framework for intrinsic reward-shaping for rl using llm feedback. 2023. URL https://alexzhang13.github.io/assets/pdfs/Reward_Shaping_LLM.pdf.

Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1712–1714, 2021a.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. Journal of Machine Learning Research, 22(289):1–39, 2021b.

A AI in Competitive Pokémon

A.1 Online Tree Search

Many CPS AI approaches rely on model-based online tree search with heuristic value approximations — much like the methods that led to early successes in games like chess and Go. Harrison Ho (2014) use shallow search and mostly ignore imperfect information to reach 55% GXE in Gen6RandomBattles. The best heuristic Pokémon engines use PS team composition statistics to estimate private information at the current root node and reduce CPS to perfect-information depth-limited search (Stone, 2010). Sarantinos (2023) adds more complex heuristic value functions, search pruning, and private information inference to peak at rank #33 in Gen7RandomBattles. Sarantinos (2023) play a comparable number of human battles as each of our main models (600+). However, they are evaluating a single policy in a single ruleset — enabling a large effective sample size that clearly demonstrates the extreme variance of PS's ELO and world ranking metrics. Glicko-1 and GXE are not reported but are far better metrics, and we encourage their use in future comparisons. Based on results in old forum posts, years of continued development, and our knowledge of method details and feature coverage relative to competitors, Foul Play (Mariglia, 2019) is the strongest open-source engine today.

A.2 RL and Self-Play

Kalose et al. (2018) evaluate small-scale Q-learning in a simplified version of CPS against random and minimax heuristic agents with limited success. Prior works use an online self-play process by collecting on-policy data against their own policy. Huang & Lee (2019) train PPO (Schulman et al., 2017) self-play agents without tree search. They achieve a 1677 Glicko-1 and 72% GXE on the Gen7RandomBattle Pokémon Showdown ladder. Wang (2024) augments PPO with MCTS at test-time and achieve a 1756 Glicko-1 and 79.5% GXE on the Gen4RandomBattle ladder.

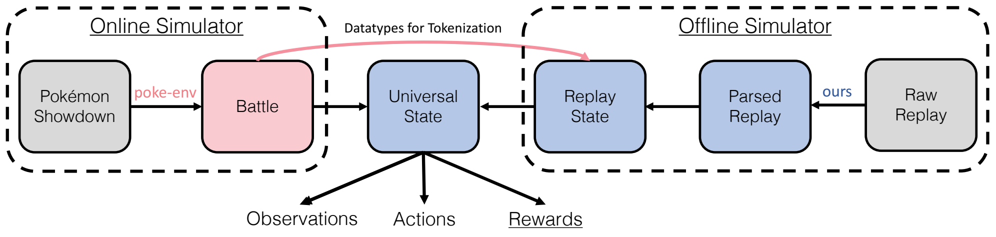


Figure 15: **Creating an Offline `poke-env` Dataset.** Our offline replay reconstruction pipeline interprets PS replays in a custom implementation designed to parse historical replays, improve team inference beyond the PS viewer, and diagnose failures. The resulting trajectory is then converted to a representation that can also be recovered from the online `poke-env` interface.

Lee & Togelius (2017) propose CPS and Pokémon Showdown as an important benchmark for AI research. `poke-env` (Sahovic, 2020) has made the PS domain much more accessible and has become the default for recent work, including ours and those in Appendix A.3. Our `Metamon` release aims to be the final bridge connecting academic RL research and PS; we use a custom version of `poke-env` geared towards the early generations and add 1) a suite of additional baseline opponents, 2) standardized team sets, 3) a template for BC experiments, and 4) direct compatibility with large-scale RL training (Grigsby et al., 2024a). Fifth, and most importantly, we create the PS replay dataset with a complex reconstruction process (Section 3). From the perspective of the user, our dataset appears to provide offline trajectories of human gameplay recorded via `poke-env`. At test time, the online `poke-env` interface is used to play against other agents and humans on the public ladder. However, this compatibility is an illusion enabled by closing a sim2sim gap between our own replay parser and `poke-env` (Figure 15). More discussion in Appendix D.

A.3 Large Langauge Model Agents

Pokémon's web presence lets large language models (LLMs) act on Pokémon game states that involve many categorical variables that can be formatted as natural language. PokéLLMon (Hu et al., 2024) conditions the LLM on a history of observations, actions, and turn results to select the next action. They also use retrieval-augmented generation from a Pokémon knowledge database to inform the LLM's decisions. PokéLLMon achieves a 49% win rate on the Gen8RandomBattles Pokémon Showdown ladder but does not report Glicko-1 or GXE statistics that control for matchmaking bias. Note that the expected raw win rate on the Pokémon Showdown ladder in modern generations (where large player pools allow for even matchmaking) is $\approx 50\%$ unless well below human-level. Karten et al. (2025) extend the LLM prompting setup to model the opponent's decisions and enable depth-limited search with heuristic value functions. Prompts include information like move damage calculations that let future outcomes inform action selection. Pokéchamp's planning allows for a 76% win rate against PokéLLMon in Gen8RandomBattles and Gen9OU. Pokéchamp is concurrent work, and non-trivial modifications are needed to convert its model-based search to the gameplay mechanics of the early generations. Finally, Zhang et al. (2023) use an LLM for reward design to improve sample efficiency of DQN (Mnih et al., 2015) against heuristics.

B Broader Related Work

Imitation Learning and Offline RL. Many large-scale agents in complex domains are trained by imitation learning. These methods prioritize training scalable sequence models on large datasets and avoid RL obstacles (Reed et al., 2022; Gallouédec et al., 2024; Jiang et al., 2022; Raad et al., 2024; Brohan et al., 2023). Offline RL (Prudencio et al., 2023; Levine et al., 2020) learns policies that outperform their demonstrations and has found success at scale (Kumar et al., 2022; Springenberg et al., 2024). In practice, offline RL can be used in a way-off-policy or multi-batch setting (Laroche & des Combes, 2019; Najib et al., 2024,) where models are iteratively retrained or fine-tuned as data accumulates or better training techniques are found (Lampe et al., 2024; Tirumala et al., 2023); the ability to learn stable policies from large mixed-quality datasets unlocks a flexible engineering workflow. Many solutions prevent the learned policy from deviating too far from the offline dataset (Wang et al., 2020; Nair et al., 2020; Fujimoto & Gu, 2021). These approaches create a spectrum between unconstrained RL and behavior cloning and let a single objective replace the two-stage process of BC pre-training $\rightarrow$ RL fine-tuning.

Offline RL targets real-world use cases where (1) data collection is expensive or (2) deployment mandates some minimum performance standard well above random exploration. Playing Pokémon against humans leads to both problems: battles are slow (and there are limits to how many games we can play in parallel), and finding competent strategies across the full range of Pokémon teams and game modes is a daunting exploration challenge. In simulated RL domains, it is common to mimic the process of learning from existing data by first training online RL agents and then saving their rollouts for offline research (Reed et al., 2022; Fu et al., 2020; Gulcehre et al., 2020; Agarwal et al., 2020). If online RL cannot solve the task, it may be possible to crowdsource demonstration datasets (O'Neill et al., 2024; Gerstgrasser et al., 2022). It would be more realistic (and more convenient) if offline datasets already existed and grew naturally without requiring researchers to collect data. Our Pokémon dataset falls in this category — as do other games played on the internet like Chess (Lichess, 2025), Go (KGS, 2025; Silver et al., 2016), Diplomacy (FAIR Diplomacy Team et al., 2022), and Starcraft II (Vinyals et al., 2019; Mathieu et al., 2023). Other examples include autonomous driving (Kiran et al., 2021; Lee et al., 2024) and e-commerce (Saito et al., 2020).

Gameplaying. Games have always been key benchmarks for AI and RL research (Campbell et al., 2002; Tesauro, 1995). High-profile successes include AlphaZero in chess and Go (Silver et al., 2018), AlphaStar in StarCraft II (Vinyals et al., 2019), OpenAI Five in DOTA 2 (Berner et al., 2019), and DeepNash in Stratego (Perolat et al., 2022). Applications of model-based search to imperfect information games (IIGs) like poker (Moravčík et al., 2017; Brown & Sandholm, 2018; Brown et al., 2020) create methods at the intersection of RL and game theory. We refer interested readers to Schmid et al. (2023) for a detailed overview. Policy learning in CPS (Section 4) could also be viewed from the perspective of IIG formalisms like Factored Observation Stochastic Games (Schmid, 2021). Model-free RL against hybrid populations of opponent agents is a viable alternative despite lacking theoretical guarantees to converge to optimal (equilibrium) policies (Vinyals et al., 2019; Rudolph et al., 2025; Heinrich et al., 2015). Finally, long-context sequence models have been used to model the decisions of opponents (Nashed & Zilberstein, 2022) in multi-agent settings (Jing et al., 2024).

C Heuristic Opponents

In an attempt to evaluate a variety of Pokémon fundamentals, we develop an array of heuristic opponents. These policies are unable to cheat by accessing unrevealed information about their opponent's team but are otherwise free to use ground-truth knowledge of Pokémon's mechanics to select actions. Figure 16 summarizes the relative performance of these heuristics. Ultimately, we find it difficult to generate meaningful diversity from this larger set and focus on six heuristics:

- **RandomBaseline** selects a legal move (or switch) uniformly at random and measures the most basic level of learning early in training runs.
- **Gen1BossAI** emulates the decision-making of opponents in the original Pokémon Generation 1 games. It usually chooses random moves. However, it prefers using stat-boosting moves on the second turn and "super effective" moves when available.
- **Grunt** is a maximally offensive player that selects the move that will deal the greatest damage against the current opposing Pokémon using Pokémon's damage equation and a type chart and selects the best matchup by type when forced to switch. Its strategy amounts to greedy one-ply search and is an improvement over a common "MaxBasePower" agent in related work.
- **GymLeader** improves upon Grunt by additionally taking into account factors such as health. It prioritizes using stat boosts when the current Pokémon is very healthy, and heal moves when unhealthy.
- **PokeEnv** is the `SimpleHeuristicsPlayer` baseline provided by Sahovic (2020).
- **EmeraldKaizo** is an adaptation of the AI in a Pokémon Emerald ROM hack intended to be as difficult as possible. The game's online popularity has led to a community effort to document its decision-making in extensive detail. We use this documentation to re-implement the policy. It selects actions by scoring the available options against a rule set that includes handwritten conditional statements for a large portion of the moves in the game.

Figure 17 evaluates the PokeEnv Heuristic against humans on the ladder. We choose PokeEnv for this task because it appears in external work, but its strengths and weaknesses are similar to several other heuristics in our set. We use the same Competitive Team Set as our main model evaluations but evaluate over a smaller sample of battles per ruleset. The relationship between battle format and heuristic performance in Fig. 17 is predictable given knowledge of the PS metagames. Players correctly accuse the heuristic of being a bot in the online chat, and we decide we have made our point and stop evaluations. Notably, these accusations are not rooted in the super-human reaction time of the policy, but in its lack of move diversity and multi-turn strategy while playing at the (low) level people have come to expect from hobbyist bot projects and the Pokémon video games. Our learning-based agents do not suffer from these problems, and we will return to this discussion in Appendix F. We would expect the heuristic to perform worst (and play least like a human) in Gen2OU, but are not comfortable evaluating this. Glicko-1 ratings can be slow to converge when this far below the mean, and it is possible that Fig. 17 is an overestimate. However, our low rating skews matchmaking in our favor (we are matched against the lowest ELO players) — making this a rare case where raw win-loss records can be informative as an upper bound on win rate (Table 2).

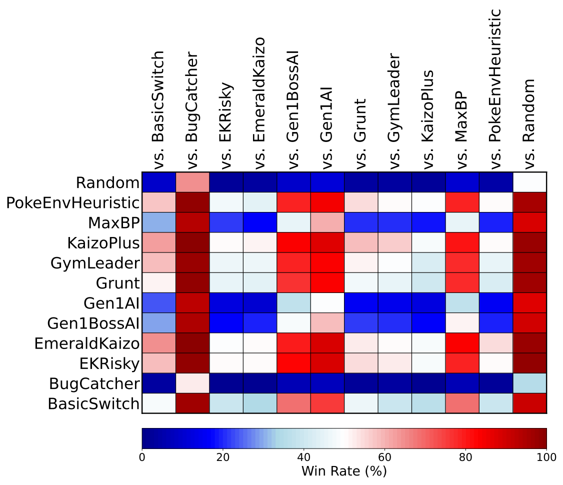


Figure 16: **Heuristic Round-Robin.** Entries denote the win rate of the row player against the column player in 2k Variety Set battles across Gen1-4OU.

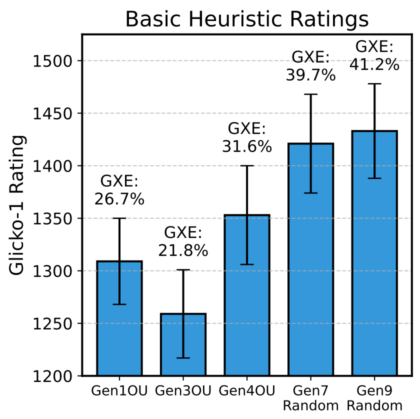


Figure 17: **PokeEnv Heuristic vs. Humans**. Early-Gen OU tiers are unique games that prioritize long-horizon control over memorization of damage matchups between pokémon.

Battle Format	Wins	Losses
Gen1OU	16	59
Gen3OU	16	54
Gen4OU	21	36
Gen7RandomBattle	24	32
Gen9RandomBattle	28	32

Table 2: **PokeEnv Heuristic Win-Loss Records on the PS Ladder.**

D Replay Reconstruction

As mentioned in Appendix A.2, we build a custom replay reconstruction pipeline designed to interpret years-old records of human gameplay and identify teams from a spectator point-of-view (POV). The resulting trajectories train offline policies that can be deployed online via `poke-env` (Sahovic, 2020).

We follow a process visualized by a simplified example in Figure 18 to extract complete battle information. On each turn, we add newly revealed information to a running estimate of the initial team configuration. By the end of the battle, some details may still be missing and are inferred with Pokémon Showdown statistics. We then backfill the inferred team through the trajectory, accounting for any changes to the roster that occur during the battle. Since Player A should have full knowledge of their own team, but limited knowledge of Player B's, we save a trajectory from Player A's perspective by using the inferred version of Player A's private state and the original spectator POV of Player B's state.

During reconstruction, we will obtain: (1) the complete team composition for each player and (2) per-turn observations from one player's POV. We can use a real battle as an example. You can view the relay at this link. Figure 21 gives a sample of the raw PS log for this replay, while Figure 22 shows a chosen POV player's observed and inferred team. Finally, Figure 23 shows the fully reconstructed replay containing all necessary information for model training.

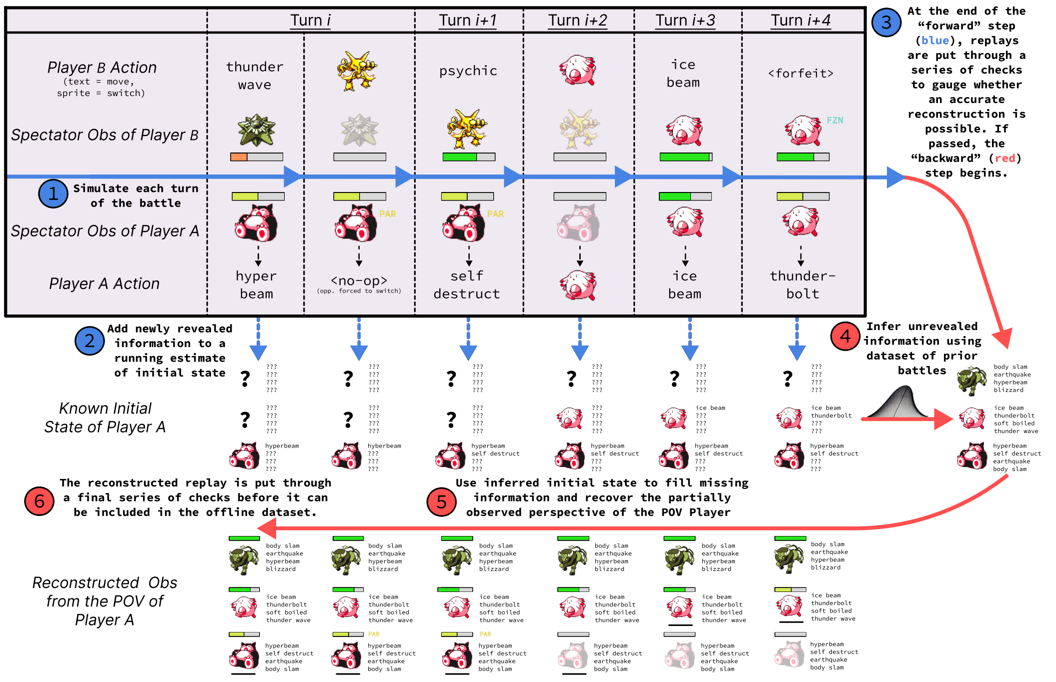

Turn *i*
Turn *i+1*
Turn *i+2*
Turn *i+3*
Turn *i+4*
Player B Action
(text = move, sprite = switch)
Spectator Obs of Player B
thunder wave
psychic
ice beam
<forfeit>
FZN
1 Simulate each turn of the battle
Spectator Obs of Player A
PAR
Player A Action
hyper beam
<no-op> (opp. forced to switch)
self destruct
ice beam
thunder-bolt
3 At the end of the "forward" step (blue), replays are put through a series of checks to gauge whether an accurate reconstruction is possible. If passed, the "backward" (red) step begins.
2 Add newly revealed information to a running estimate of initial state
Known Initial State of Player A
4 Infer unrevealed information using dataset of prior battles
body slam
earthquake
hyperbeam
blizzard
ice beam
thunderbolt
soft boiled
thunder wave
hyperbeam
self destruct
earthquake
body slam
6 The reconstructed replay is put through a final series of checks before it can be included in the offline dataset.
5 Use inferred initial state to fill missing information and recover the partially observed perspective of the POV Player
Reconstructed Obs from the POV of Player A

Figure 18: **Simplified Replay Reconstruction.** We walk through the reconstruction of the perspective of Player A in a Gen1OU example with teams of 3 Pokémon.

D.1 Reconstruction Failures

A challenge in the replay reconstruction process is that inaccurate team inference can create inaccurate records of human decision-making: An expert player may have may have only picked the action in the replay because they did *not* have access to the moves or Pokémon our dataset says they did. This is a fundamental problem created by the spectator POV, but it could be improved by team inference strategies that are more sophisticated than sampling from historical statistics.

Offline RL always confronts a distribution shift problem created by sampling a finite dataset from a large state/action space (Levine et al., 2020). Replay reconstruction can fail, and these failures add an additional challenge in that some specific state/actions will never appear in a dataset of any size (Figure 19). Some of these failures are caused by unimplemented game mechanics that rarely occur but could be improved. Others are caused by fundamentally ambiguous situations from the spectator perspective — even the PS browser replay viewer gets these wrong or warns that values may be inaccurate. A long list of checks throughout the reconstruction process attempts to find and discard trajectories in these states. These situations are rare and discarding them may be needlessly cautious.

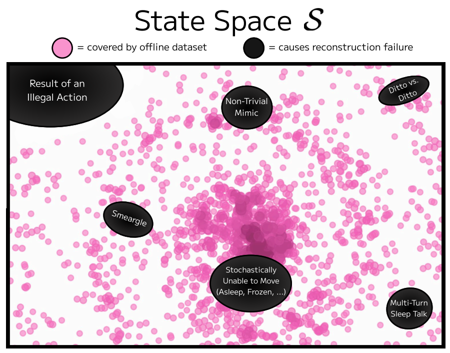

State Space $\mathcal{S}$
= covered by offline dataset
= causes reconstruction failure
Result of an Illegal Action
Non-Trivial Mimic
Ditto vs. Ditto
Smeargle
Stochastically Unable to Move (Asleep, Frozen, ...)
Multi-Turn Sleep Talk

Figure 19: **Replay Parser Failure States.** An informal visualization of how the replay reconstruction process creates holes in our dataset on top of the more standard distribution shift inherent to offline RL.

There are two gaps in the replay dataset that we cannot ignore. Our solutions impact our findings and are worth discussing in detail:

Illegal Actions. Pokémon always has *up to* 9 discrete actions, but some of these actions become invalid as the battle progresses. Humans are not given the option to select invalid actions, so they never appear in the dataset. Offline RL should be able to handle this problem. Our policies are clearly told which actions are invalid, and we let their mistakes become indicators of accumulating OOD behavior[5]. We send a random valid action to PS if an invalid action is selected. Invalid action masking was added to our open-source release long after the experiments in this paper and predictably made little difference when enforced only at test time — though it may improve value estimation during training.

(Stochastically) Unrevealed Moves. There are situations where the player's action choice has no impact on the battle and is not revealed to spectators. These occur too often to discard the trajectory, so we either need to mask or fill the action label. BC-RNN baselines (Appendix F.1) *mask* unrevealed action labels. When training offline RL along a full trajectory sequence in parallel, it is risky to back up the Q-values of timesteps where the actor or critic is not trained. Therefore, the RL models *fill* action labels and the main Transformer BC models follow suit to create a direct comparison between variants of the actor objective (Eq. (2)). We initially fill missing actions with a small BC-RNN model trained on a much earlier version of the dataset. The precise accuracy of these moves may not seem important because they have no impact on the battle. However, there are stochastic gameplay mechanics (mainly sleep and paralysis) where they *could have* impacted the battle. We eventually suspect we can improve by filling missing actions with the more accurate (Figure 27) BaseRNN model. We retrain 15M IL and RL Transformer policies on this revised ("Filled Action") version of the dataset. Offline RL should have already been able to avoid sub-optimal action choices in the situations they are relevant. Indeed, we find no evidence that the new action labels impact the RL policies. However, the Small IL model is significantly improved — now ranking *between* Large IL and the RL eval scores against heuristics (Figure 20 Left), and BC-RNN (Fig. 20 Right). Though not included in the figures, Small IL with Filled Actions also ranks between Large IL and all RL scores against Large IL (Figure 9) and the Foul Play engine (Figure 11a).

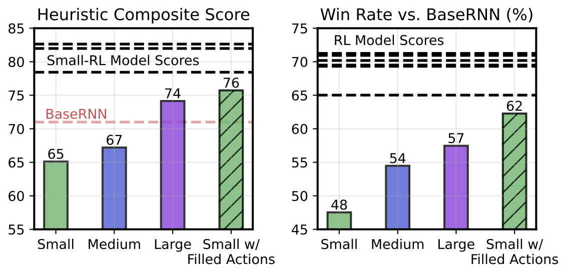
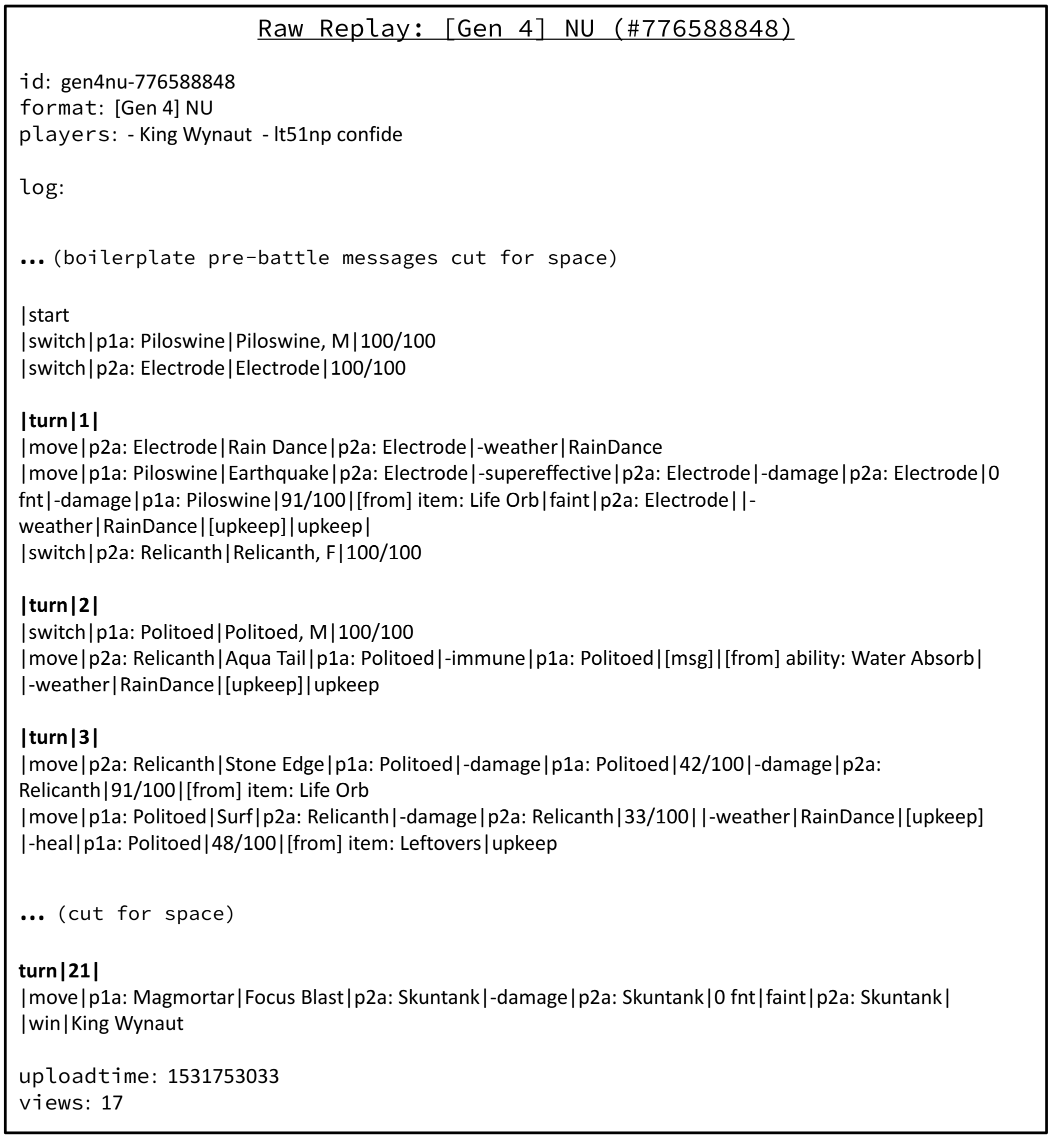


Figure 20: **Impact of Improved Missing Action Labels on a 15M Transformer IL Policy.**

We conclude that while the comparisons between IL and RL remain a fair evaluation of the same architecture trained on the same dataset, the original dataset was challenging in a way that was unintentionally similar to contrived benchmarks that dilute high-quality demonstrations with poor decisions (Fu et al., 2020). Our final batch of RL models (SynRL-V1+SP, SynRL-V1++, and SynRL-V2) use improved labels in their human battle trajectories out of caution. After the release of this paper, we added missing action masking directly into the RL training pipeline with similar results.

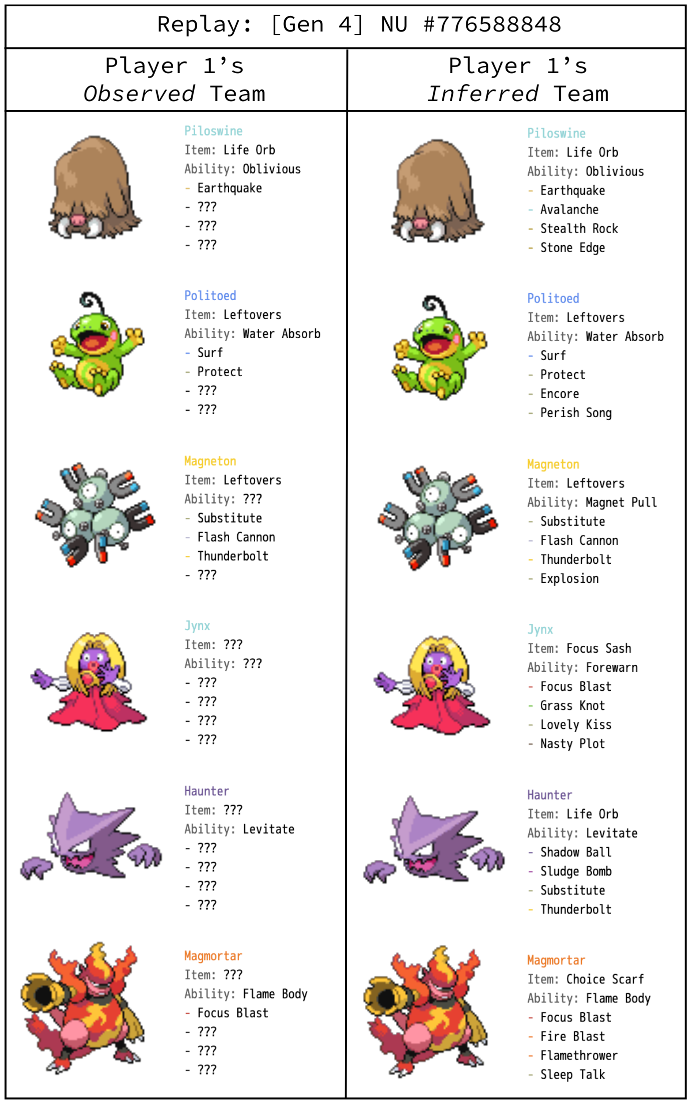
[5]For reference, all RL policies average valid action rates of 97-99% against heuristics and 95-98% against humans. Nearly all of these invalid actions occur in succession once the policy is already in a lost position or runs into a limitation of the observation space discussed in Appendix E.1.



Raw Replay: [Gen 4] NU (#776588848)

```
id: gen4nu-776588848
format: [Gen 4] NU
players: - King Wynaut  - lt51np confide

log:


... (boilerplate pre-battle messages cut for space)

|start
|switch|p1a: Piloswine|Piloswine, M|100/100
|switch|p2a: Electrode|Electrode|100/100

|turn|1|
|move|p2a: Electrode|Rain Dance|p2a: Electrode|-weather|RainDance
|move|p1a: Piloswine|Earthquake|p2a: Electrode|-supereffective|p2a: Electrode|-damage|p2a: Electrode|0
fnt|-damage|p1a: Piloswine|91/100|[from] item: Life Orb|faint|p2a: Electrode||-
weather|RainDance|[upkeep]|upkeep|
|switch|p2a: Relicanth|Relicanth, F|100/100

|turn|2|
|switch|p1a: Politoed|Politoed, M|100/100
|move|p2a: Relicanth|Aqua Tail|p1a: Politoed|-immune|p1a: Politoed|[msg]|[from] ability: Water Absorb|
|-weather|RainDance|[upkeep]|upkeep

|turn|3|
|move|p2a: Relicanth|Stone Edge|p1a: Politoed|-damage|p1a: Politoed|42/100|-damage|p2a:
Relicanth|91/100|[from] item: Life Orb
|move|p1a: Politoed|Surf|p2a: Relicanth|-damage|p2a: Relicanth|33/100||-weather|RainDance|[upkeep]
|-heal|p1a: Politoed|48/100|[from] item: Leftovers|upkeep


... (cut for space)

turn|21|
|move|p1a: Magmortar|Focus Blast|p2a: Skuntank|-damage|p2a: Skuntank|0 fnt|faint|p2a: Skuntank|
|win|King Wynaut

uploadtime: 1531753033
views: 17
```

Figure 21: An example Gen4 NeverUsed (NU) replay file downloaded from PS server.



Replay: [Gen 4] NU #776588848

Player 1's *Observed* Team	Player 1's *Inferred* Team
Piloswine Item: Life Orb Ability: Oblivious - Earthquake - ??? - ??? - ???	Piloswine Item: Life Orb Ability: Oblivious - Earthquake - Avalanche - Stealth Rock - Stone Edge
Politoed Item: Leftovers Ability: Water Absorb - Surf - Protect - ??? - ???	Politoed Item: Leftovers Ability: Water Absorb - Surf - Protect - Encore - Perish Song
Magneton Item: Leftovers Ability: ??? - Substitute - Flash Cannon - Thunderbolt - ???	Magneton Item: Leftovers Ability: Magnet Pull - Substitute - Flash Cannon - Thunderbolt - Explosion
Jynx Item: ??? Ability: ??? - ??? - ??? - ??? - ???	Jynx Item: Focus Sash Ability: Forewarn - Focus Blast - Grass Knot - Lovely Kiss - Nasty Plot
Haunter Item: ??? Ability: Levitate - ??? - ??? - ??? - ???	Haunter Item: Life Orb Ability: Levitate - Shadow Ball - Sludge Bomb - Substitute - Thunderbolt
Magmortar Item: ??? Ability: Flame Body - Focus Blast - ??? - ??? - ???	Magmortar Item: Choice Scarf Ability: Flame Body - Focus Blast - Fire Blast - Flamethrower - Sleep Talk

Figure 22: Continuing the Gen4 NU example by listing the observed team and the inferred team after replay reconstruction.

Reconstructed Replay: [Gen 4] NU (#776588848)
King Wynaut vs. lt51np confide (from POV of King Wynaut)
Played July 16th, 2018

Text Obs #0: <gen4nu> <anychoice> <player> piloswine lifeorb oblivious ground ice noeffect nostatus <move> avalanche ice physical <move> earthquake ground physical <move> stealthrock rock status <move> stoneedge rock physical <switch> haunter lifeorb levitate <moveset> shadowball sludgebomb substitute thunderbolt <switch> jynx focussash forewarn <moveset> focusblast grassknot lovelykiss nastyplot <switch> magmortar choicescarf flamebody <moveset> fireblast flamethrower focusblast sleeptalk <switch> magneton leftovers magnetpull <moveset> explosion flashcannon substitute thunderbolt <switch> politoed leftovers waterabsorb <moveset> encore perishsong protect surf <opponent> electrode unknownitem unknownability electric notype noeffect nostatus <conditions> noweather noconditions noconditions <player_prev> nomove <opp_prev> nomove

(observations also include an array of numerical features)

Action #0: **1** (→ 2nd <move> → earthquake)
Reward #1: **0.91**

Text Obs #1: <gen4nu> <anychoice> <player> piloswine lifeorb oblivious ground ice noeffect nostatus <move> avalanche ice physical <move> earthquake ground physical <move> stealthrock rock status <move> stoneedge rock physical <switch> haunter lifeorb levitate <moveset> shadowball sludgebomb substitute thunderbolt <switch> jynx focussash forewarn <moveset> focusblast grassknot lovelykiss nastyplot <switch> magmortar choicescarf flamebody <moveset> fireblast flamethrower focusblast sleeptalk <switch> magneton leftovers magnetpull <moveset> explosion flashcannon substitute thunderbolt <switch> politoed leftovers waterabsorb <moveset> encore perishsong protect surf <opponent> relicanth unknownitem unknownability rock water noeffect nostatus <conditions> raindance noconditions noconditions <player_prev> earthquake <opp_prev> nomove

Action #1: **8** (→ 5th <switch> → politoed)
Reward #2: **0.00**

Text Obs #2: <gen4nu> <anychoice> <player> politoed leftovers waterabsorb notype water noeffect nostatus <move> encore normal status <move> perishsong normal status <move> protect normal status <move> surf water special <switch> haunter lifeorb levitate <moveset> shadowball sludgebomb substitute thunderbolt <switch> jynx focussash forewarn <moveset> focusblast grassknot lovelykiss nastyplot <switch> magmortar choicescarf flamebody <moveset> fireblast flamethrower focusblast sleeptalk <switch> magneton leftovers magnetpull <moveset> explosion flashcannon substitute thunderbolt <switch> piloswine lifeorb oblivious <moveset> avalanche earthquake stealthrock stoneedge <opponent> relicanth unknownitem unknownability rock water noeffect nostatus <conditions> raindance noconditions noconditions <player_prev> nomove <opp_prev> aquatail

Action #2: **3** (4th <move> → surf)
Reward #3: **0.15**

... (cut for space)

Text Obs #25: <gen4nu> <anychoice> <player> magmortar choicescarf flamebody fire notype noeffect nostatus <move> fireblast fire special <move> flamethrower fire special <move> focusblast fighting special <move> sleeptalk normal status <switch> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <switch> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <switch> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <switch> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <switch> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <blank> <opponent> skuntank unknownitem unknownability dark poison noeffect nostatus <conditions> noweather noconditions noconditions <player_prev> focusblast <opp_prev> crunch

Action #25: **2** (3rd <move> → focusblast)
Reward #26: **101.91**

Figure 23: Concluding the Gen4 NU example with an abridged version of the reconstructed replay.

E Training Details

E.1 Reward Function

Rewards are a combination of three shaping terms and a binary win/loss indicator (r_{win}):

$$R(s_t, a_t) = r_{\text{hp}} + \frac{1}{2} r_{\text{stat}} + r_{\text{faint}} + 100 r_{\text{win}}$$

We describe the shaping terms from the perspective of the agent's player:

- **Health Reward r_{hp}:** Encourages dealing more damage than the opponent and/or recovering more health than the opponent. Computed by net health points gained/lost by our active Pokémon versus those gained/lost by the opponent's active Pokémon (with all health values scaled $0 - 1$).
- **Status Reward r_{stat}:** Encourages dealing status conditions while avoiding taking status conditions ourselves. Status conditions are a key indicator of mid-game progress. Computed by the net gain in the binary presence of a status condition of the two Pokémon on the field.
- **Faint Reward r_{faint}:** Encourages knocking out the opponent's Pokémon while preserving our own. Computed by the number of Pokémon we made unavailable to the player on this turn minus the number we lost.

The reward function is designed to give some shaping to help the offline filter w (Equation (2)) learn to assign unique weights over short horizons but be dominated by the binary win/loss outcome we ultimately care about. We do find some qualitative evidence of models exploiting the shaped terms. For example, our agents tend to cling to life in clearly lost positions by using recovery moves.

E.2 Observation Space

Observations include a language description (depicted by Figure 5) and 48 numerical features. Numerical features include the base power and accuracy of moves and the health/stats/boosts of Pokémon. We defer a full account to the open-source release. Implementation details add the previous action and reward as policy inputs. Rewards may help resolve some ambiguity over the outcome of the previous turn (e.g., did the move hit and deal damage?). The player's previous action is a one-hot vector that is mostly redundant to information in the text observation but helps provide a history of action choices that were not revealed to the opponent.

Our observation space relies on long-term memory to track the true state of the battle. Section 4 notes that we only include the visible attributes of the opponent's active Pokémon, which reduces dimensionality and distribution shift over the opponent's team. We can infer the public state of the opponent's team from memory over the active Pokémon on previous turns and their move choices. Text tokens include the most recent move of both Pokémon on the field. The long-term memory of our models is quite effective in general. As one example, PS enforces a rule called "Sleep Clause" where attempting to put a second opponent Pokémon to sleep does nothing and wastes a turn. Our policies are remarkably good at following this rule even though their only way to track it is to recall that they put a Pokémon to sleep and that it has not reappeared and woken up.

There is a limit to the number of times a move can be used by a Pokémon in a battle. These "PowerPoint" (PP) limits break long stalemates in CPS, but PP counts are unreliable and full of edge cases in replays. While PP counts are tracked during reconstruction to help discard replays, we ultimately exclude them from the observation space. We decided to protect against sim2sim gaps because we assumed our agents would have to be unrealistically skilled to survive long enough for PP limits to be relevant. Our final policies are actually strong enough that PP stall losses are their most noticeable flaw and the leading cause of invalid action selections (Appendix D.1). PP counts can be inferred from memory over the move history. However, this is challenging in practice, especially when lacking opponent policies that force PP stalls during self-play. SynRL-V2 does demonstrate some ability to play around PP limits.

The observation space can be improved to address specific gameplay mechanics and will be version controlled for future comparisons. However, environments as complex as CPS will always have nuanced partial observability and benefit from the flexibility of sequence model policies.

E.3 Action Space

Agents play with 9 discrete actions. The first four indices correspond to the active Pokémon's moves, and the remaining indices switch to the other Pokémon on the player's team. The correspondence between action index and move/switch choices is indicated by both the text and numerical observation — which arrange their features in a consistent alphabetical order. As discussed in Appendix D.1, actions become invalid over the course of a battle. Invalid actions are also noted in the observation. If the agent selects an invalid action, it is replaced by a random valid action within the environment's transition dynamics.

E.4 Models and Hyperparameters

Table 3 details the default training configuration for Small (15M), Medium (50M), and Large (200M) model sizes. Table 4 lists changes for all the models and ablations mentioned in the paper and released in our open-source code.

	Small	**Medium**	**Large**
Learning Rate		1e-4	
Linear LR Warmup Steps		1000	
Target Critic τ		0.004	
TD Loss Coeff		10	
Grad Clip		1.5	
L2 Coeff		1e-4	
Batch Size	32	40	48
Actor Activation		Leaky ReLU	
Actor Layers		2	
Actor Hidden Dimension	300	400	512
Agent Popart (Hessel et al., 2019)		True	
Critic Ensemble Size (Chen et al., 2021)		4	
Critic Layers		2	
Critic Activation		Leaky ReLU	
Critic Hidden Dimension	300	400	512
Turn Encoder Token Dim	100	100	160
Turn Encoder Layers	3	3	5
Turn Encoder Summary Tokens	4	6	11
Turn Encoder Attention Heads	5	5	8
Turn Encoder Numerical Tokens		6	
Causal Transformer Layers	3	6	9
Causal Transformer Attention Heads	8	8	20
Causal Transformer FF Dim.	2048	3072	5120
Causal Transformer Model Dim.	512	768	1280
NormFormer (Shleifer et al., 2021)		True	
σReparam (Zhai et al., 2023)		True	
Causal Transformer Normalization		LayerNorm (Ba et al., 2016)	
Max Context Length	200	200	128
Causal Transformer Activation		Leaky ReLU	

Table 3: **Base Training Hyperparameters by Model Size.** In reference to the architecture in Figure 4 and the AMAGO training configuration (Grigsby et al., 2024a).

<u>Model Name</u>	<u>Dataset</u>	Architecture (Table 3)	Loss (Equation (2))	<u>Notes</u>
Small IL	RPS 950k	Small	Behavior Cloning	
Small IL (Filled Actions)	RPS 950k	Small	Behavior Cloning	A late ablation that replaces missing actions with improved estimates (Appendix D.1).
Small RL	RPS 950k	Small	Exponential w	Exp w default to $\beta = .5$ and clip weight values in $[1e\text{-}5, 50]$.
Small RL (Binary)	RPS 950k	Small	Binary w	
Small RL (Exp Extreme)	RPS 950k	Small	Exponential w	$\beta = 1$, clip $[-1e\text{-}5, 100]$. (Testing sensitivity to w hyperparameters).
Small RL (Aug)	RPS 950k	Small	Exponential w	Randomly sets tokens to `<unknown>` timestep-wise.
Small RL (Filled Actions)	RPS 950k	Small	Binary w	Ablation that replaces missing actions with improved estimates (Appendix D.1).
Small RL (Binary+MaxQ)	RPS 950k	Small	Binary w, $\lambda = 1$	Testing Q overestimation on the replay dataset before scaling up.
Medium IL	RPS 950k	Medium	Behavior Cloning	
Medium RL	RPS 950k	Medium	Exponential w	
Medium RL (Aug)	RPS 950k	Medium	Exponential w	
Medium RL (Binary+MaxQ)	RPS 950k	Medium	Binary w, $\lambda = 1$	
Large IL	RPS 950k	Large	Behavior Cloning	All Large architecture models use the "Aug" dropout scheme by default.
Large RL	RPS 950k	Large	Exponential w	
Large RL (Binary+MaxQ)	RPS 950k	Large	Binary w, $\lambda = 1$	
SyntheticRL-V0	RPS 950k + 1M Gen1&3 Variety Set model vs. model trajectories with an ad-hoc mixture of all policies above. 2M total trajectories.	Large	Binary w, $\lambda = 10$	
SyntheticRL-V1	SyntheticRL-V0 + 1M Gen2 & Gen4 Variety Set model vs. model battles. 3M total trajectories.	Large	Binary w, $\lambda = 10$	
SyntheticRL-V1+SelfPlay	SyntheticRL-V1 + 2M Gen1-4 SynRL-V1 self-play trajectories. Models from here to the end of the table use improved action labels in their RPS dataset (Appendix D.1). 5M total trajectories.	Large	Binary w, $\lambda = 10$	
SyntheticRL-V1++	SyntheticRL-V1 + 2M additional Variety Set model vs. model trajectories (5M total).	Large	Binary w, $\lambda = 10$	
SyntheticRL-V2	SyntheticRL-V1++ adding 100k trajectories from 50k human battles in Jan-Mar 2025 (RPS 1.05M) — including many of our own public battles. 5M total trajectories.	Large	Binary w	Value predictions are converted to two-hot classification following (Grigsby et al., 2024b) with 96 output bins spaced *evenly* between $[-110, 110]$.

Table 4: **Model Variations**. Datasets, architectures, and hyperparameter changes (from the base set in Table 3) for the 20 main Transformer models trained throughout the paper. "RPS 950k" refers to the original replay reconstruction dataset (Appendix D). "Exponential" weight functions (w) are implemented following AWAC (Nair et al., 2020). "Binary" weight functions are implemented following CRR (Wang et al., 2020). In both cases, advantage estimates approximate $V(s)$ as the mean over the critic ensemble. "Synthetic" models increase batch size from $48 \rightarrow 96$ sequences.

We train all models on a single $8\times$ NVIDIA A5000 GPU machine for at least 1M gradient steps. We default to the checkpoint at 1M, which is well after performance has converged according to our evaluations. In the open-source code and weights, an "epoch" is an arbitrary interval of 25k gradient steps, and we save checkpoints every 2 epochs. Therefore, results default to checkpoint 40 unless otherwise noted. SyntheticRL-V1+SelfPlay fine-tunes from epoch $40 \rightarrow 48$ and defaults to 48, while SyntheticRL-V2 is an exception in that we can confirm it is still improving at 1M, and so we use the last available checkpoint (of 48). These exceptions are noted by Table 6, and Appendix F.3 contains more discussion.

Figure 24 shows the relationship between model size and action prediction accuracy for behavior cloning models on the replay dataset. Figure 25 highlights the difference between scalar regression and two-hot classification for value prediction.

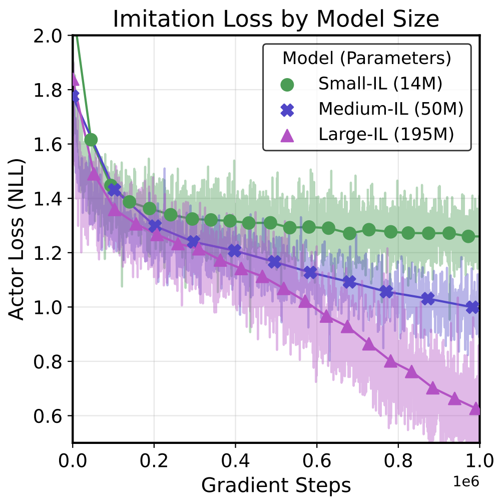


Figure 24: **Transformer IL Train Loss Curves.** Training loss on the Pokémon human replay dataset has a predictable relationship with model size when using a standard BC objective.

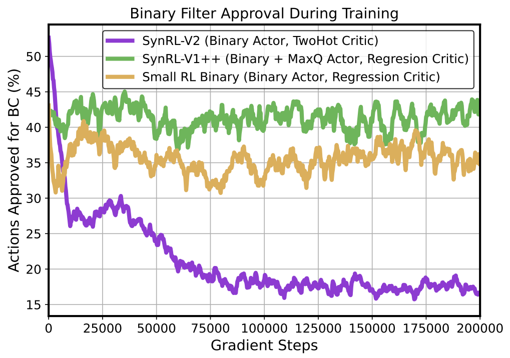


Figure 25: **Critic Filter Pessimism.** We track the percentage of the offline dataset assigned as weight $w(h, a) > 0$ (Eq. (2)) throughout training. The accuracy of the two-hot classification filter has a significant impact on the pessimism of the BC process. Curves are noisy because they track the average value of a single GPU mini-batch (of 12 battles).

F Experimental Details and Additional Figures

This section contains figures and experimental details that support Section 5 in the main text.

F.1 Early Imitation Learning Models

In the beginning of our effort, it is not apparent that the Pokémon replay dataset requires architecture sizes beyond the scale of common RL problems. We begin by building a small-scale behavior cloning pipeline (that is still available in the `Metamon` code release). Figure 26 identifies clear underfitting on the reconstructed battle replay dataset. Our early development leads to the Turn Encoder Transformer architecture (Figure 4) with a GRU-based (Cho et al., 2014) trajectory model (rather than the Transformer in Fig. 4) to create a "BaseRNN" opponent. BaseRNN leads the early Heuristic Composite Score rankings (Figure 6) and later serves as a fast (CPU-only) opponent and as a way to fill missing action labels (Appendix D.1). Figure 27 documents BaseRNN's predictive accuracy alongside two ablations. "WinsOnlyRNN" follows a common offline RL ablation by testing whether performance can be improved by manually discarding low-return trajectories from the POV of the losing player.

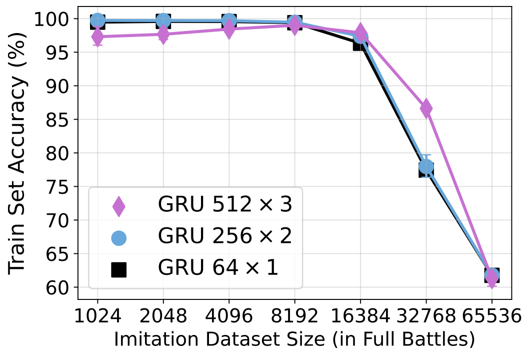


Figure 26: **Underfitting on PS Replays.** We report the train-set accuracy of (small) recurrent BC policies on increasingly large datasets of human gameplay. Error bars denote the maximum and minimum over four random subsets. Model sizes are reported by their hidden state and number of recurrent layers.

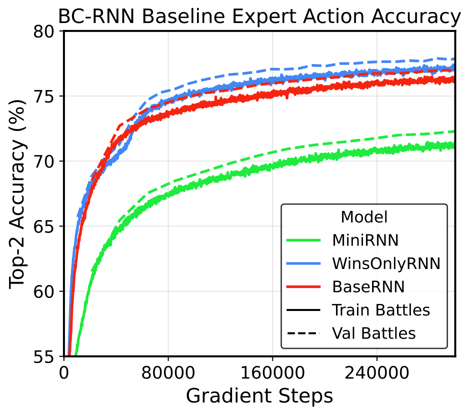


Figure 27: **BC-RNN Accuracy**. Action labels are high-entropy and we find Top-2 accuracy to be a more useful metric for tuning. "BaseRNN" is 3.5M params, "MiniRNN" ablates to 800k, and "WinsOnlyRNN" follows the filtered BC approach of only imitating decisions from the POV of the winning player (cutting its train/val sets in half).

F.2 Heuristic Evaluations

Figure 28 records the Heuristic Composite Score (Section 5.1) of various models (Table 4) throughout training. Much of our early effort goes into creating strong but inexpensive heuristics to monitor training progress, but performance converges in less than 250k training steps. Model-based opponent evaluations run fast enough to generate learning curves after the fact and shed more light on the relationship between training budget and performance (Appendix F.3).

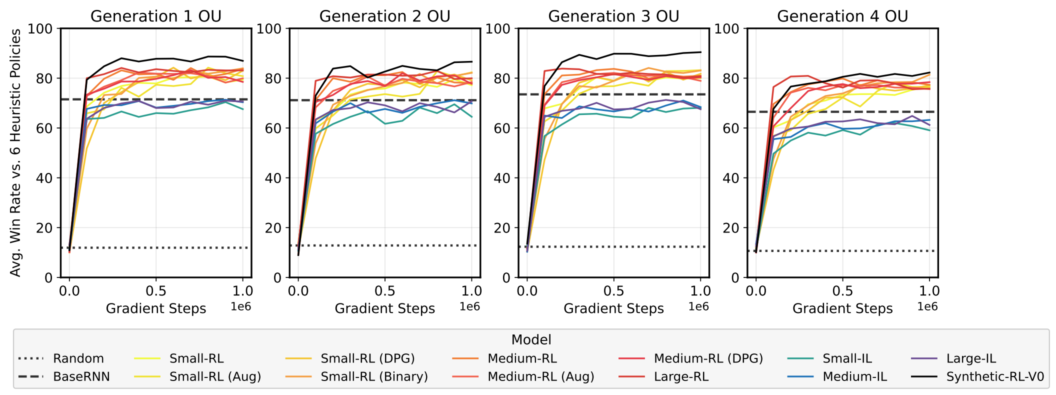


Figure 28: **Heuristic Composite Learning Curves.** Performance converges quickly but shows no sign of degrading over long training runs. BC and offline RL form two clear clusters with $\mathcal{L}_{\text{actor}}$ changes and model size having no clear impact.

F.3 Model-Based Evaluations

Figure 29 evaluates a variety of models against the BaseRNN behavior cloning model. Like the heuristic learning curve in Figure 28, performance converges well before the end of training against this opponent. Figure 30 highlights the continued improvement of our final model ("SyntheticRL-

V2") against a previous version that had climbed into the global top 50 in Gen1OU. SyntheticRL-V2 may not have converged after 1.2M steps, but training was cut short due to time constraints. Table 5 evaluates the impact of narrow self-play data with realistic teams and controls for the additional training budget of fine-tuning on this dataset versus continuing training on the original dataset.

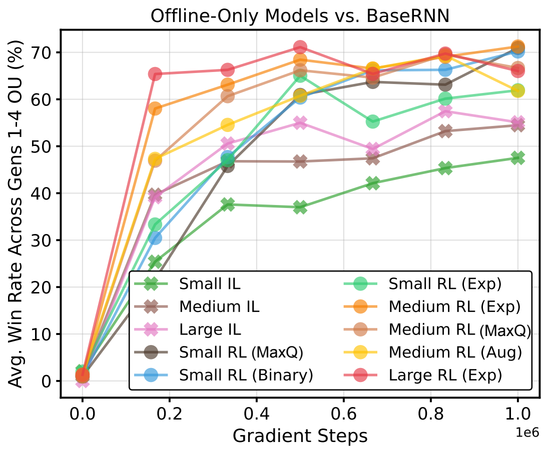


Figure 29: **Transformer IL and RL vs. RNN BC.** We evaluate the performance of Transformer policies trained on the offline replay dataset against a smaller RNN-based model designed for CPU-only inference. The RL updates do not display meaningfully distinct performance but outperform BC at all model sizes.

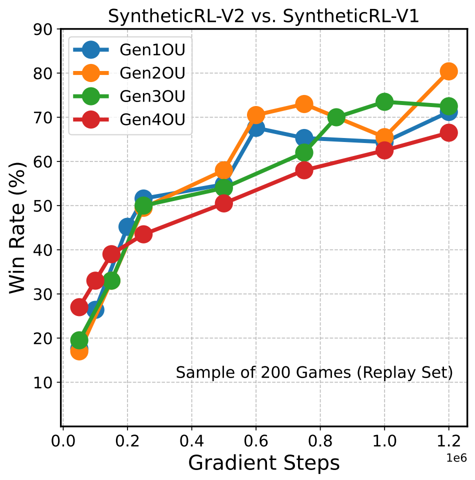


Figure 30: **Improvement of Advanced Policies.** We record the improvement of our best model ("SyntheticRL-V2") against a previous version that reached a top 50 ranking in Gen1OU.

	Gen1OU	Gen2OU	Gen3OU	Gen4OU
SyntheticRL-V1+SelfPlay @ 1.2M Steps	63.6%	59.6%	61.4%	59%
Synthetic-V1 @ 1.2M Steps	50%	53.8%	48.4%	48.2%

Table 5: **Win Rates vs. SyntheticRL-V1.** We evaluate a checkpoint fine-tuned on a dataset of self-play battles against the original version (at 1M training steps). We control for the additional training steps with a second version that maintains its original dataset. Sample size of 500 games.

F.4 Human Evaluations

Our models play under identical conditions to humans. We assign each model its own username (Table 6). Usernames are visible to the opponent, so humans can adapt to the model over repeat matchups (just as they might exploit any other player). We use the PS statistics for each username in Figures 12 and 14. Note that ratings like ELO and Glicko-1 confidence intervals decay every 24 hours, so the PS statistics at the time of reading will no longer match our figures. Table 7 records each model's overall win/loss for completeness — though we note again that such records have little meaning because PS matches stronger models against stronger players.

The PS ladder has increment time controls (similar to chess) that go into effect if requested by either player. We always request the timer in order to keep evaluations moving if our opponent disconnects from the game for an extended time. Note that most players also request time constraints, as Early Gen battles can be 20+ minutes long even when enabled. Time limits can be a key constraint for CPS AI methods involving search or LLMs (Karten et al., 2025). However, our agents select an action at the inference speed of $\leq$ 200M parameter Transformer, and this makes time constraints a non-issue. In fact, our pace of play is suspiciously fast. However, the opponent must be playing very quickly for this to be noticeable because decisions are made simultaneously, and the battle moves at the pace of the slower player. As we reach high ELO, we begin to run into the few players who can defeat our models while playing quickly. We eventually implement a random delay to hide the inference speed (while still playing faster than the opponent on most turns). Superhuman speed aside, all our policies play in an undeniably human-like style. We saved hundreds of battle replays to the PS website, which you can browse via the links in Table 6 or by searching `https://replay.pokemonshowdown.com/`. These replays are a (mostly) unbiased sample of all matches played in public (spectator-viewable) battles while the lead author was monitoring the ladder evaluations.

Model Name	**PS Username**	**Checkpoint**
Small-IL	SmallSparks	40
Large-IL	DittoIsAllYouNeed	40
Large-RL	Montezuma2600	40
SyntheticRL-V0	Metamon1	40
SyntheticRL-V1	TheDeadlyTriad	40
SyntheticRL-V1 + Self-Play	ABitterLesson	48
SyntheticRL-V1++	QPrime	40
SyntheticRL-V2	MetamonII	48

Table 6: **Public Ladder Usernames.** Models are tied to unique usernames throughout evaluations. Links lead to a replay page for each model. Miscellaneous test battles are also played under the usernames “NotableWalrus” and “PsyduckIsUbers”, which are not always the same model and do not appear in results, but may be present in replays or videos featured in our release materials.

We believe that the ability to generate human-like gameplay at fast inference speeds with arbitrarily prompted teams can be a fun and useful practice tool for human players. However, the models can be frustrating to play against because their reward function encourages delaying losses, and they do not forfeit. Figure 31 uses the Large RL model to show that Q-value predictions are calibrated enough to identify lost positions and implement auto-forfeits.

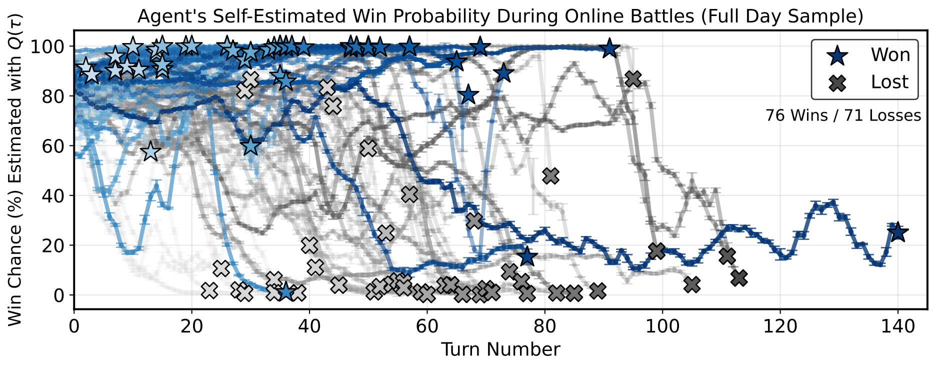


Figure 31: Q**-functions as a win estimate.** We track critic value predictions (for $\gamma = .999$) during battles across a 24-hour period of the Large-RL model’s gameplay on the PS ladder. If we simplify by ignoring the reward function’s small shaping terms and the discount factor, we can plot these values as a more interpretable estimate of win probability. We mark these value series by their true outcome. Small error bars denote two standard deviations over the ensemble of 4 critics.

Model	Username	Gen1 OU	Gen2 OU	Gen3 OU	Gen4 OU
PokeEnv Heuristic	WinningIsOptional	16 - 59	N/A	16 - 54	21 - 36
Small IL	SmallSparks	54 - 66	20 - 63	25 - 50	41 - 59
Large RL	Montezuma2600	72 - 60	49 - 56	68 - 67	32 - 42
SynRL-V0	Metamon1	57 - 42	42 - 35	61 - 52	49 - 51
SynRL-V1	TheDeadlyTriad	107 - 64	76 - 52	82 - 74	83 - 61
SynRL-V1+SP	ABitterLesson	65 - 38	51 - 30	80 - 71	64 - 56
SynRL-V1++	QPrime	72 - 52	37 - 24	48 - 32	68 - 53
SynRL-V2	MetamonII	148 - 95	75 - 40	76 - 53	68 - 48

Table 7: **PS Usernames and Win - Loss Records.**