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

Anonymous authors
Paper under double-blind review

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 is led by heuristic tree search and online self-play, but the game may 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 recent LLM approaches and rival or exceed the best heuristic search engines. Playing anonymously in online battles against humans, our agents surpass a 50% estimated win rate in all four generations and climb inside the top ranked players in the game's longest-horizon rulesets.

Contribution(s)

1. We build an offline RL dataset comprising nearly 1M trajectories reconstructed from years of human gameplay in the complex decision-making task of Competitive Pokémon Singles.
   Context: None
2. We demonstrate our dataset's ability to train black-box adaptive 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.

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

Anonymous authors
Paper under double-blind review

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 is led by heuristic tree search and online self-play, but the game may 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 recent LLM approaches and rival or exceed the best heuristic search engines. Playing anonymously in online battles against humans, our agents surpass a 50% estimated win rate in all four generations and climb inside the top ranked players in the game's longest-horizon rulesets.

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 a team from billions of possibilities and battle against an opponent's partially observed team. 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's complexity is a significant challenge for AI and creates an exciting research opportunity in Reinforcement Learning (RL). Previous efforts rely on heuristic search in custom simulators (Mariglia, 2024) or test-time MCTS with self-play (Wang, 2024). 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 on the online ranked ladder, thereby unlocking a naturally occurring source of offline RL data (Levine et al., 2020; Lange et al., 2012) that grows larger every day. This dataset enables a perspective on the CPS AI problem that has previously been impractical: that sequence models might be able to learn to play without explicit search by using model-free RL and long-term memory to infer their opponent's team and tendencies.

Our experiments provide a case study in the process of training, evaluating, and improving large policies (Fig. 1 Left) (Springenberg et al., 2024; Lampe et al., 2024). We create 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 Transformer (Vaswani et al., 2017) policies of up to 200M parameters trained by IL and offline RL. When deployed on the Pokémon Showdown website 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 50-75% and rise onto the global leaderboard. LLM-Agent approaches (Hu et al., 2024b) prove uncompetitive in the long horizons of early generations, and our best agent — without search — exceeds or at least closely rivals the best heuristic search engine (Mariglia, 2024) across all four generations.

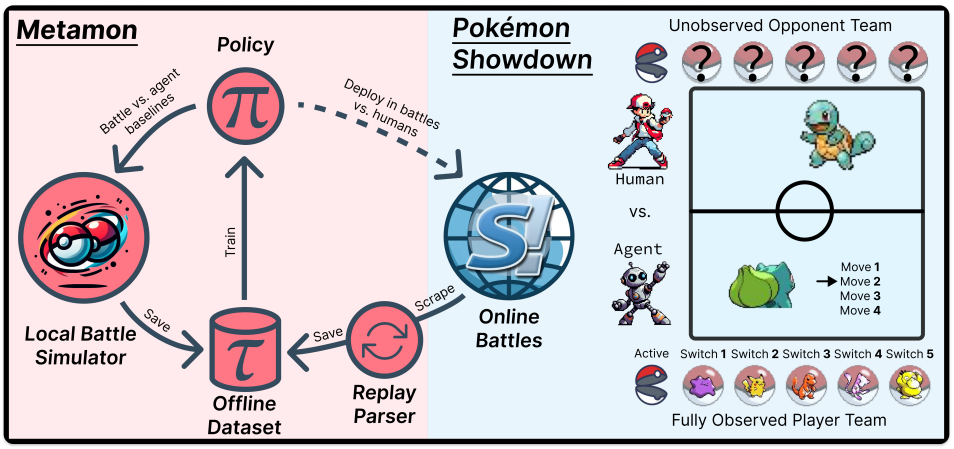


Figure 1: **Batch Training and Evaluation in CPS.** We develop a platform called `Metamon` that enables an offline RL workflow on a dataset of human gameplay from Pokémon Showdown.

2 Background: Competitive Pokémon Singles

If the reader is unfamiliar with CPS, it is difficult to overstate how complicated top-level strategy can be. The game combines opponent modeling (Nashed & Zilberstein, 2022) 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 and unintended behavior. This complexity is notable both because it hinders the sample efficiency of any learning method and because it effectively ensures independent implementations (i.e., to speed up tree search) will not be perfectly accurate. The ground-truth simulator is Pokémon Showdown (PS) — a popular 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 ban Pokémon and enforce various rules to maintain competitive balance. Each generation of each tier is essentially treated as its own game — or rather, two games played consecutively: team *design* and *control*. Players design teams before they are matched with their opponent and must consider all the threats they believe they will face. Team design converges to an equilibrium that narrows the search to perhaps thousands of meaningfully distinct teams that are considered competitively viable. However, this set shifts to counter the latest trends and has changed significantly over time.

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 composition 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 does not have space to bring both. We may try to mislead our opponent by revealing information that suggests one team design only to surprise them when they are no longer defending against our real strategy. Players make (most) decisions simultaneously. Accurately predicting the opponent's choices based on their team and previous tendencies is the key skill differentiating 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 two important player metrics on PS. **Glicko-1** is an ELO-like skill rating. The matchmaking system on PS prefers to pair players with similar ratings. **GXE** corrects for this matchmaking bias to estimate a player's odds of defeating a randomly sampled opponent.

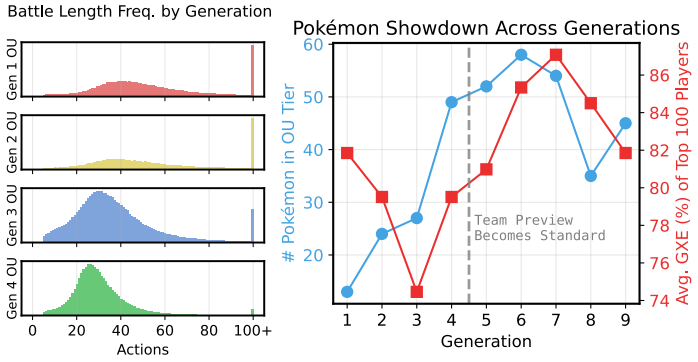


Figure 2: **Episode Length, Team Diversity, and Variance By Generation.** GXE statistics taken in February 2025. Episode length data is compiled from our replay dataset.

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 the team design question entirely 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 of CPS, making it the most popular and therefore the tier with the most data to learn from (Section 3). Each generation of OU increases the number of team combinations and gameplay mechanics (Figure 2 Right). Importantly, the size of team space becomes so unmanageable from Generation 5 onwards that PS adopts a mechanic called "team preview" that reveals the opponent's team before the start of the battle. 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 (Figure 2 Left). The early generations are an almost independent competitive community with a long history and a 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. 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. Gen4 resembles modern versions in that many Pokémon can eliminate their opponent in a single move — leading to a faster pace of play. Appendix B.1 Figure 13 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 black-box sequence-based 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 self-play MCTS (Huang & Lee, 2019) for random battles formats. CPS is primarily played and discussed on the internet, and this affords considerable gameplay knowledge to recent LLM-Agents techniques (Hu et al., 2024b; Karten et al., 2025b). Appendix A provides a survey of AI in CPS.

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[1] 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. 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 problem that may arise when trying to use 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 would require a conversion to this format that opens up a sim2real-like gap between the reconstructed (PO)MDP and the real world.

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 all available historical data to model the distribution of human-constructed teams and simplify by ignoring the fact that team construction trends on PS are non-stationary for a number of reasons. 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 B.2 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.

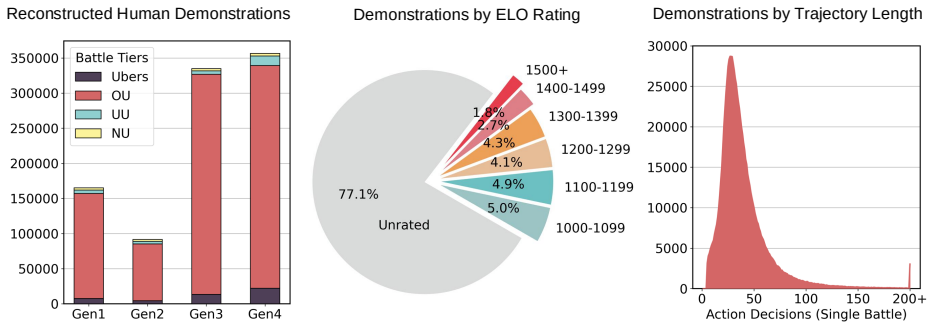


Figure 3: **Dataset Summary.** The initial version of our offline dataset includes 475k battles — summarized here by their PS format (left), ELO rating (center), and length in agent timesteps (right).

This process is not always successful, as some gameplay mechanics cannot be reconstructed from incomplete information. A long list of checks identifies trajectories that have entered ambiguous situations and (quite 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 2015 (Figure 3). Each battle yields two point-of-view trajectories for a total of about 950k sequences containing 38M timesteps. Our pipeline is now actively downloading new battles in the 15 minute window before they are deleted (regardless of whether a player chose to save them). This

[1] https://replay.pokemonshowdown.com/

has significantly increased the growth rate of the dataset for future work, but the experiments in this paper will only be using the original 950k trajectory version.

4 Search-Free Pokémon with Offline RL On Sequence Data

Pokémon 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. We can follow a meta-RL perspective (Beck et al., 2023; Ghosh et al., 2021) where we consider our opponent's choices part of the environment's unknown transition function $T(s_{t+1} \mid s_t, a_t, \pi_o, c_o, c_p)$ (Zintgraf et al., 2021a). Our goal is to find a policy that maximizes return over some distribution of these latent variables, which in our case would be the distribution of opponents currently active on PS and our own distribution of teams:

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

Context-based methods aim to maximize $\eta(\pi)$ by conditioning the policy on estimates of the unobserved variables derived from previous experience. Here, this would amount to using the entire history of a battle[2] (states, rewards[3], 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 the black-box meta-RL framework (Duan et al., 2016; Wang et al., 2016) — which has seen great success in large-scale problems (Team et al., 2023). In black-box meta-RL 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 rewards (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 exploratory actions that improve the sequence model's representation if this increases expected returns. CPS strategies like feeding our opponent misleading information about our own team also follow from this framework.

We will be using the offline dataset ($\mathcal{D}$) from Section 3 to approximate the expectations in Eq. 1. The implicit assumption is that the distribution of teams and playstyles across history is identical to that of the current game (Dorfman et al., 2020). This is false, but it may be close enough, particularly in the highly-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. 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 overestimate the odds it is facing our synthetically generated teams or opponents.

In summary, we have reduced the problem of learning to play CPS to the problem of training a sequence model with offline RL. However, this model may need to be quite large, so training is non-trivial. We can use an update that safely reduces to behavior cloning (BC) but gives room to skew the loss function towards return-maximizing behavior if we decide the offline RL risks are sufficiently small (Springenberg et al., 2024; Wu et al., 2019; Fujimoto & Gu, 2021). Ideally, BC becomes a lower bound we can improve upon. Solutions of this kind are actor-critics that train their critic to output Q-values with standard one-step backups. Actor loss functions take the general form:

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

[2] 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.

[3] 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.

Where in our setting 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 heuristic w that 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 notation and implementation details are closer to popular continuous-action variants where off-policy actor-critics are necessary even without the desire to reduce to BC (Lillicrap et al., 2015; Fujimoto et al., 2018; Chen et al., 2021). Pokémon's discrete actions let us compute $\mathbb{E}_{a\sim\pi}[Q(h,a)]$ directly (Degris et al., 2012; Christodoulou, 2019). Our experiments will study various 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., 2024) implementation used throughout our experiments. One detail worth highlighting is the parallel training of multiple γ values (Eq. 1), which is an effective trick for problems like CPS that have sparse rewards but require long horizons ($\gamma > .99$).

Figure Nickname	$w(z,a) =$	λ
"IL"	1	0
"Exp" (or just "RL")	$\exp(\beta A^\pi(z,a))$ (clipped)	0
"Binary" (or "FBC")	$A^\pi(z,a) > 0$	0
"DPG" (or "DPG+FBC", "DPG Binary")	$A^\pi(z,a) > 0$	> 0

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

Before discussing the network architecture, we need to define a state space, action space, and reward function for CPS. Our agent needs enough information to mirror human decisions, and a good point of reference is the UI of the PS website. 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 state size, memory difficulty, generalization over Pokémon's damage formula, 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 4 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**; states only include the opponent's active pokémon. We are confident in our sequence models' ability to recall previous states, and this makes it worth avoiding the sim2real exposure of tracking the opponent's team as it is 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 state 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 C provides more details.

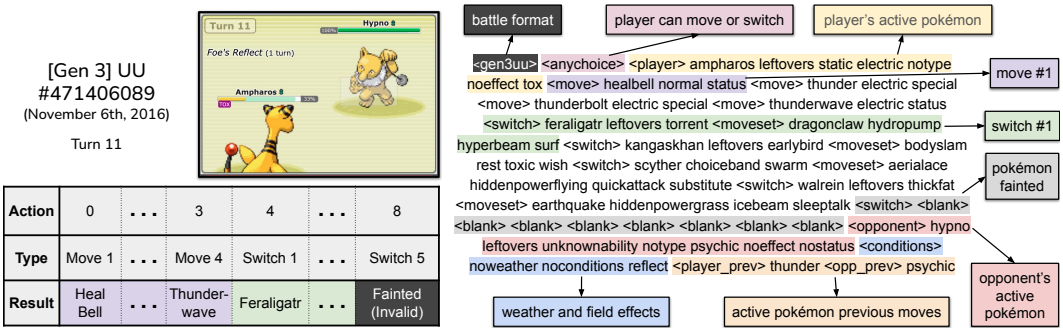


Figure 4: **Annotated Text State and Action Space.** Text order is important, but words can be tokenized into arrays with a consistent length (of 87). States also include 48 numerical features. The meaning of each action index varies by turn but is presented in the text in a consistent order.

The state, previous action, and previous reward at each timestep are processed by a Transformer encoder that uses designated summary tokens to create an embedding by attending over the multi-modal sequence (Dosovitskiy et al., 2020; Devlin et al., 2019). Text is encoded by tokenizing the Pokémon vocabulary based on our replay dataset with an "`<unknown>`" token for rare cases we

may have missed. The resulting sequence of turn representations is the input to a causal Transformer with actor and critic output heads (Figure 5).

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 2. Models are named in results according to their size and training objective (Table 1). Results will be discussed in a semi-chronological order, though some figures will spoil results from a synthetic self-play process discussed in Section 5.4. Our goal is to compete against human players, but this is expensive and creates a challenging offline evaluation problem: which methods (and checkpoints of those methods) do we deploy on PS? Our efforts to answer this question result in extensive evaluations against various opponents.

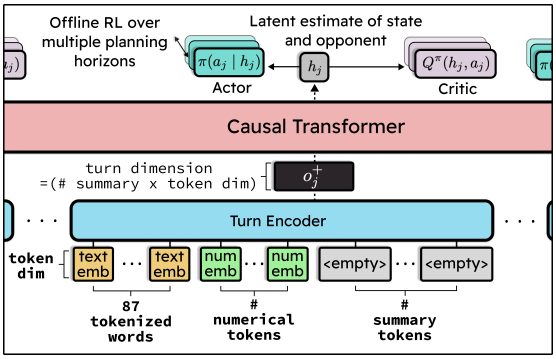


Figure 5: **Architecture**. Actions are predicted based on representations of the state, action, and reward of every turn of the current battle.

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

We create a suite of 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 B.1. The average win rate against 6 of these heuristics on the Variety set forms a "Heuristic Composite Score." The main benefit of this metric is that it represents 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 6 documents a predictable decline from OU to NeverUsed (NU) gameplay, and is the first example of a consistent theme where RL outperforms IL. We evaluate many variants of the $\mathcal{L}_{\text{actor}}$ objective (Eq. 2), but do not find significant differences between them. We tune the Turn Encoder architecture (Fig. 5) with RNN trajectory models S_θ between 500k-4M parameters trained by BC. The predictive accuracy of these models, and their performance against the heuristics, is documented in Appendix D. The best BC-

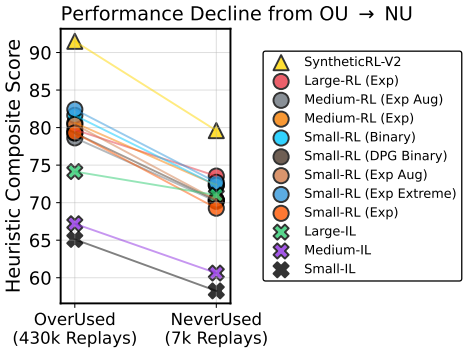


Figure 6: **OU → NU.** Heuristic opponents represent a fixed target and highlight the discrepancy between popular OU tiers with strong data coverage and unpopular tiers with far fewer replays.

RNN models lead the early Heuristic Composite rankings, and these will become the next rung on the ladder towards human-level gameplay. Clear signs of underfitting motivate the starting point of 15M for our Transformer agents.

5.2 Model-Based Evaluations

Appendix D 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 RL. Following Grigsby et al. (2024), 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 7 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 8 highlights the win rate of key models.

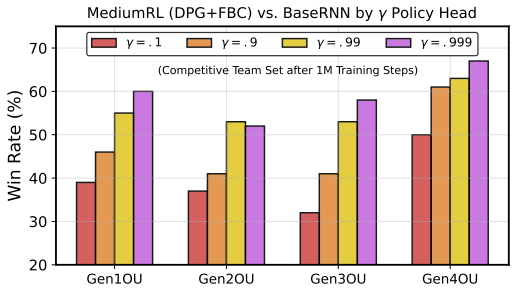


Figure 7: **Multi-γ Action Selection.**

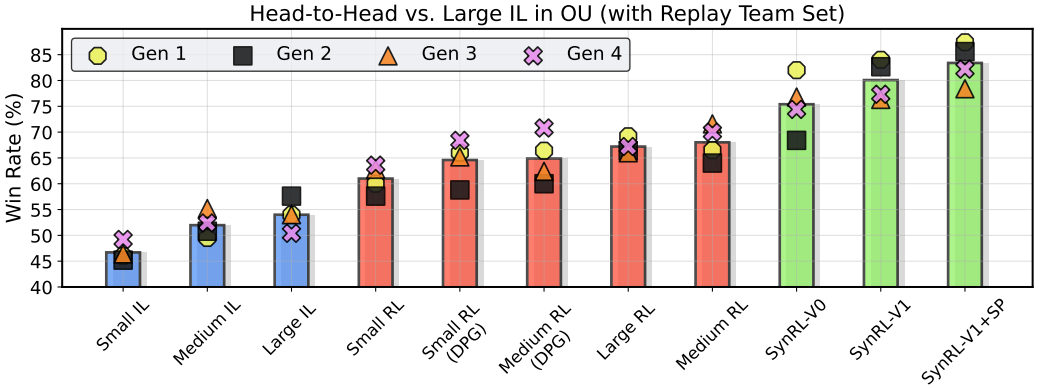


Figure 8: **Self-Evaluation Against Large-IL.** Results are determined by the best checkpoint over the last 200k training steps and models are sorted by their average win rate across generations.

5.3 Playing Humans On the Pokémon Showdown Ranked Ladder

We play against human players by queuing for ranked battles on the public PS ladders. The player pool of early generations is relatively small. We evaluate our agents over periods of several days and frequently switch between generations for sample sizes of at least 400 battles. Models' Glicko-1 and GXE stats at the end of their final battle are shown in Figure 9. We include the results of the PokeEnv (Sahovic, 2020) baseline heuristic[4] agent from Fig. 13 for additional context.

The Large-RL model rises to the level of an intermediate player, and is favored to win a battle against a randomly selected opponent in Generations 1 and 2. Qualitatively, our models display human-like gameplay; during our evaluation process, we saved sample replays on the PS website, which can be viewed by searching their assigned usernames (Table 3) on `replay.pokemonshowdown.com`. Learning from data, our agents play reasonable openings, make safe pokémon switches, and can anticipate the moves of their opponent. Figure 10 evaluates the impact of memory on the win rate of a policy competing against the full-context-length version of itself. However, they can suffer from the kind of accumulating errors we might expect from a sequence policy, and can begin to make

[4]The PokeEnv heuristic is chosen for ladder evaluations because it appears in recent work (Hu et al., 2024b; Wang, 2024). Against PokeEnv, the Small-IL and Large-RL models have win rates of $\approx 55\%$ and $\approx 75\%$, respectively.

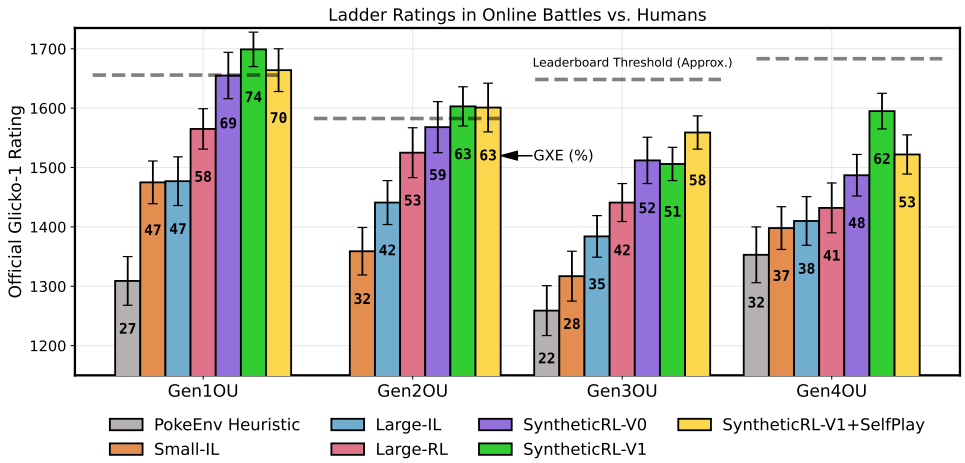


Figure 9: **Human Evaluations.** We visualize the Glicko-1 ladder rating (with its rating deviation). Bar labels represent GXE statistics. To compare across generations, we plot the heuristic baseline's performance and the average Glicko-1 of the bottom 100 players on the Top 500 global leaderboard.

nonsensical decisions in long battles — particularly when the opponent is playing with a rare team or uncommon strategy.

5.4 Synthetic Data from Self-Play

Our offline dataset yields policies capable of collecting useful human-level gameplay on the public ladder. These agents are now actively contributing 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 on the timescale of a single project this data is not making a significant difference. We can speed up this process by deploying agents on a local PS ladder, adding their trajectories to the human gameplay dataset, and then retraining from scratch with offline RL (Figure 1). 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 do this by hosting local ladders with the Variety team set and using a mix of many checkpoints of all our agents — prioritizing diversity over realism. The hope is that this data may still be valuable for model-free learning of Pokémon's stochastic transitions. The **SyntheticRL** models are Large-RL (DPG) (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. 20), BC-RNN (with win rates as high as 95% in Gen1OU and 85% in Gen3OU), Large-IL (Fig. 8), and humans (Fig. 9). **SyntheticRL-V1** takes this dataset and adds generations 2 and 4 to reach a total of 3M trajectories. Playing under the username `TheDealyTriad`, it ends its evaluation rank #46 on the Gen1OU ladder. While leaderboard rankings are quite noisy[5], its results in Figure 9 are safely human-level by any standard.

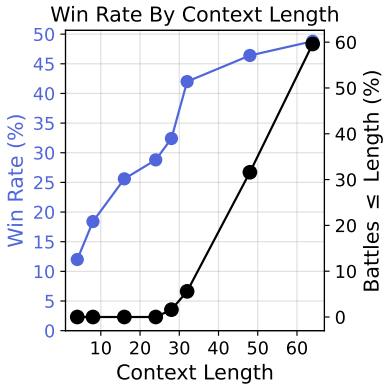


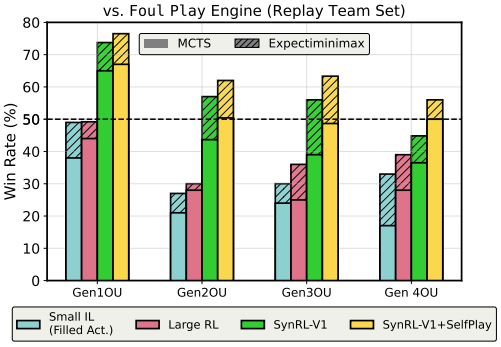
Figure 10: **Evaluating Memory.** A 200M policy battles with varying context lengths against a version of itself that can recall the entire battle.

[5]The leaderboards are sorted by PS's ELO metric. Glicko-1 considers the full history of battles under the same username and is a much better metric for our purposes. By Glicko-1, SyntheticRL-V1 would not be a top 50 player, but certainly deserves to be on the leaderboard, with a Glicko-1 of 1699 after 171 battles.

We might wonder whether any of the caution of the "synthetic" data process was necessary. We test this by letting SyntheticRL-V1 battle itself with the more realistic Replay team set until the offline dataset is 5M trajectories. Afterwards, we resume training for another 200k gradient steps. As expected, the resulting model is significantly better against itself (even when accounting for the extra training budget by continuing on the original dataset) (Appendix Table 4), but this translates to inconsistent improvement against real players in Figure 9.

5.5 LLM-Agents and Heuristic Search

`Foul Play` (Mariglia, 2024) is an advanced engine for CPS that uses a custom high-throughput 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. We challenge the engine to matches of 300 battles per generation on the Replay team set, with results shown in Figure 11a. PokéLLMon (Hu et al., 2024a) is a more general approach that takes advantage of Pokémon 's extensive web presence to build an LLM-Agent. Prompts are constructed with Pokémon type matchups and damage calculations similar to our heuristic agents, and the LLM is tasked with deciding between the available moves. Hu et al. (2024b) evaluate in a random battles tier and note that the agent struggles with long-term planning; this effect is more noticeable in the longer battle lengths of our setting. We tune the general Pokémon system prompt to be specific to the tier and experiment with changing the LLM backend from the original GPT-4 (1106-preview) to GPT-4o-mini and the o1-mini reasoning model. Results against these modifications are listed in Figure 11b.



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

PokéLLMon Matchup	Win Rate (%)
Small-IL vs. GPT-4 in Gen1OU	73
Large-RL vs. GPT-4 in Gen1OU	76
SynRL2 vs. GPT-4 in Gen1OU	92
SynRL2 vs. GPT-4o mini in Gen1OU	96
SynRL2 vs. o1-mini in Gen1OU	85
Large-RL vs. GPT-4 in Gen4OU	68
SynRL2 vs. GPT-4o mini in Gen4OU	92

(b) **PokéLLMon Matchup.** Evaluations use a custom system prompt for early-gen OU and vary the LLM backend from the original paper.

6 Conclusion

Our work presents a scalable offline RL approach for Competitive Pokémon Singles, and shows that sequence models trained on historical gameplay data can be competitive with humans in the rulesets of Generations 1-4 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. In CPS more specifically, there may be significant room for improvement by iterating on the learning update, architecture, and self-play data generation techniques to reach expert-level performance.

References

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.

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.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 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.

Foul Play, 2019. URL https://github.com/pmariglia/foul-play.

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

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Future Sight AI, 2020. URL https://www.pokemonbattlepredictor.com/FSAI/how-fsai-works.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in neural information processing systems*, 34:25502–25515, 2021.

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

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

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. Pokellmon: A human-parity agent for pokemon battles with large language models, 2024a. URL https://arxiv.org/abs/2402.01118.

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

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.

Seth Karten, Andy Luu Nguyen, and Chi Jin. Pokechamp: an expert-level minimax language agent for competitive pokemon, 2025a. URL https://openreview.net/forum?id=zi8YBcmXqA.

Seth Karten, Andy Luu Nguyen, and Chi Jin. Pokechamp: an expert-level minimax language agent for competitive pokemon, 2025b. URL https://openreview.net/forum?id=zi8YBcmXqA.

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

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.

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.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

P. Mariglia. Foul play - a competitive pokémon ai research project. https://github.com/pmariglia/foul-play, 2024. Accessed: 2025-02-27.

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

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

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.

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

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.

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.

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Technical Machine, a Pokemon AI, 2010. URL https://github.com/davidstone/technical-machine.

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.

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 Pokémon

A.1 Tree Search

Tree search agents use the Pokémon game simulator to try out various actions and evaluate the resulting game states with heuristic score functions (Technical Machine, a Pokemon AI; Harrison Ho, 2014; Foul Play). Other approaches use deep learning to model value functions (Sarantinos, 2023; Lee & Togelius, 2017). Notably, Future Sight AI trains an imitation learning policy and value function on replays scraped from Pokémon Showdown.

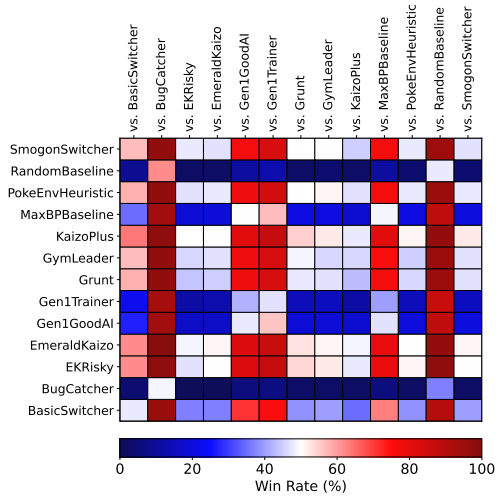


Figure 12: **Heuristic Round-Robin.** Entries denote the win rate of the row player against the column player in 500 battles under the Variety team set.

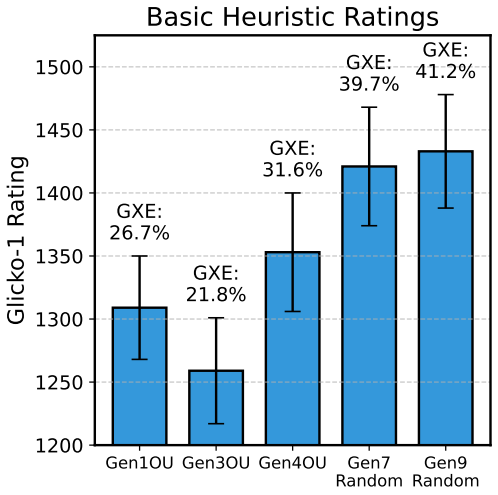


Figure 13: **PokeEnv Heuristics vs. Humans**. Early-Gen OUs are unique games that prioritize long-horizon control over memorization of damage matchups between pokémon.

A.1.1 RL Self-Play

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 Pokémon Showdown ladder.

A.1.2 Large Langauge Model Agents

The generalization and reasoning capacities of large language models (LLMs) allow them to digest and act on provided Pokémon game states — which involve many categorical variables that can be formatted as natural language. PokeLLMon (Hu et al., 2024a) conditions the LLM on a history of states, 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. They achieve a 49% win rate on the Gen8 Pokémon Showdown ladder but do not report Glicko-1 or GXE statistics that control for matchmaking bias. Karten et al. (2025a) combine an LLM with an Expectiminimax algorithm to achieve a 1500 ELO in the Gen9OU Pokémon Showdown ladder and a 76% win rate against PokeLLMon. By using an LLM to select actions and evaluate states, they create an effective minimax tree search agent. Finally, Zhang et al. (2023) use an LLM for reward design to improve sample efficiency when training a DQN policy against heuristic baselines.

B Additional Environment Details

B.1 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 type matchups, damage formula, and similar information to select actions. Figure 12 summarizes the relative performance of these heuristics. Ultimately, we find it difficult to generate meaningful diversity from this larger set, and choose to 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 Pokemon Generation 1 games. It usually chooses random moves. However, it prefers using status boosting moves on the second turn and super effective moves if it has any.
- **Grunt** is a maximally offensive player that selects the move that will deal the highest damage against the current opposing Pokemon using Pokemon's damage equation and a type chart, and selects the best matchup by type when forced to switch. By using the damage formula instead of the listed base power of each move, it creates an improved version of a common heuristic in Pokémon AI work.
- **GymLeader** improves upon Grunt by additionally taking into account factors such as health. It prioritizes using stat boosts when the current Pokemon is very healthy, and heal moves when the current Pokemon is unhealthy.
- **PokeEnv** is the `SimpleHeuristicsPlayer` baseline provided by Sahovic (2020). It chooses the highest damage move using each move's base power, accuracy, and type. It attempts to calculate favorable matchups and switches Pokemon when a switch is calculated to be optimal. It prioritizes stat boosts when the current Pokemon is healthy.
- **EmeraldKaizo** is an adaptation of the AI in Emerald Kaizo, a Pokemon 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. Actions are selected by scoring the available options against a rule set that includes handwritten conditional statements for a large portion of the moves in the game.

B.2 Replay Reconstruction

Pokémon Showdown generates a log ("replay") for every battle, capturing move selections and their effects. However, the raw replay lacks two crucial elements: (1) the complete movesets for each Pokémon on a player's team and (2) the observation states for each turn. Figure 14 shows a snippet of a raw replay downloaded from the Pokémon Showdown server. The replay can be viewed in a browser with the following link: https://replay.pokemonshowdown.com/gen4nu-776588848.

To extract complete battle information, we follow a process visualized by a simpler example by Figure 15. For each turn, we add newly revealed information to the running estimation of the team stats. By the end of the battle, some details may still be missing. We infer these using Pokémon Showdown statistics. Since Player A has full knowledge of their own team, we will reconstruct a fully observed perspective for them by filling the missing information.

After reconstruction, we will obtain: (1) the complete team composition for each player and (2) per-turn observations from one player's point of view. Figure 17 uses the above linked replay as an example — where the left side shows the initially observed team, and the right side shows the inferred full team after reconstruction. Finally, Figure 18 shows the fully reconstructed replay file, containing all necessary information for model training. The dataset extends back to the early years of Pokémon Showdown (Figure 16) and in some cases needs to account for changes in the PS log API over that time.

C Model Training Details

C.1 Reward Design

In each turn, the rewards is composed by four terms:

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 14: A real Gen4 NU replay file downloaded from PS server.

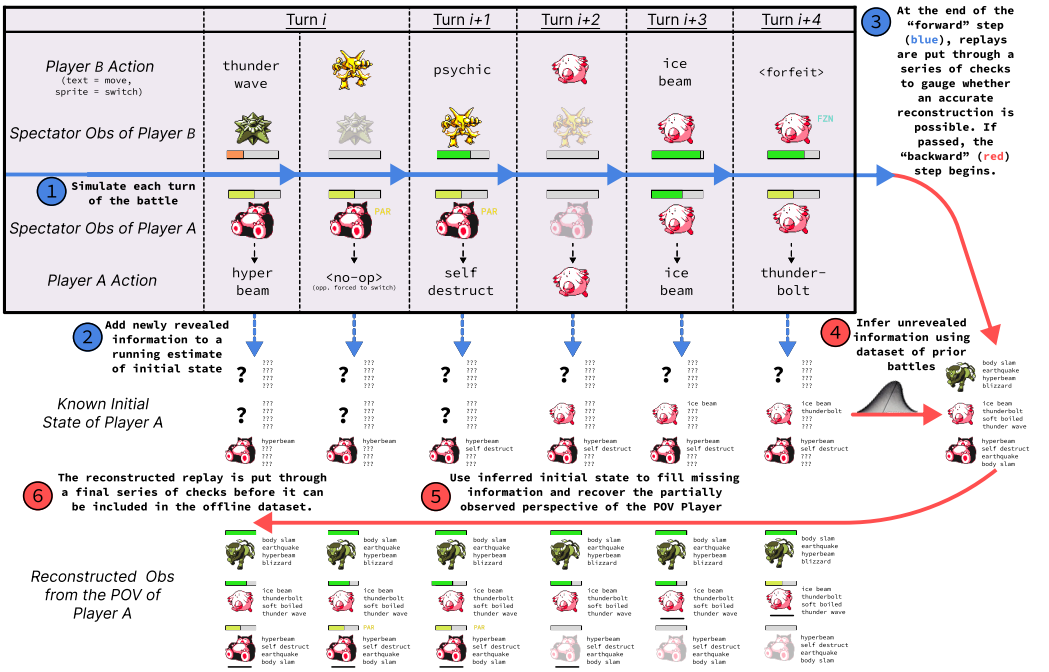


Figure 15: Simplified Replay Reconstruction for the POV of Player A in a Gen1OU example with teams of 3 Pokémon .

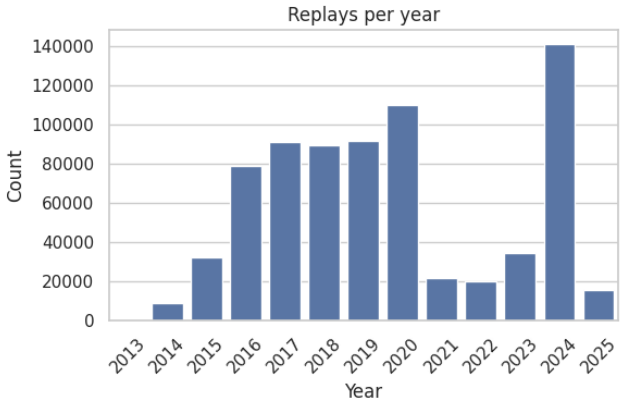


Figure 16: **Raw Replay Frequency by Battle Date.**

$$r_{\text{metamon}} = r_{\text{hp}} + 0.5 * r_{\text{stat}} + r_{\text{faint}} + 100 * r_{\text{win}}$$

HP Reward r_{hp}: Calculated by the damage dealt to the opponent's Pokémon plus the HP restored by the active Pokémon, both measured as a percentage.

Status Reward r_{stat}: Calculated by the status given to the opponent pokemon minus the status received by the active Pokémon.

Fainted Reward r_{faint}: Calculated by the number of opposing Pokémon knocked out during the turn minus the number of fainted Pokémon in the player's team.

Win Reward r_{win}: Assigned a value of 1 if the player wins the game and -1 if they lose.

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 17: A real Gen4 NU example of 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> <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> <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 18: A real Gen4 NU example of the reconstructed replay file.

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.

C.2 Training Hyperparameters

	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)	
Causal Transformer Activation		Leaky ReLU	

Table 2: **Training Hyperparameters by Model Size.** In reference to the architecture in Figure 5 and the AMAGO training configuration (Grigsby et al., 2024).

D Additional Figures

Model Name	PS Username
Small-IL	SmallSparks
Large-IL	DittoIsAllYouNeed
Large-RL	Montezuma2600
SyntheticRL-V0	Metamon1
SyntheticRL-V1	TheDeadlyTriad
SyntheticRL-V1 + Self-Play	ABitterLesson

Table 3: **Public Ladder Usernames.** Models are tied to unique usernames throughout evaluations, and we use the official PS account statistics for results in Figure 9.

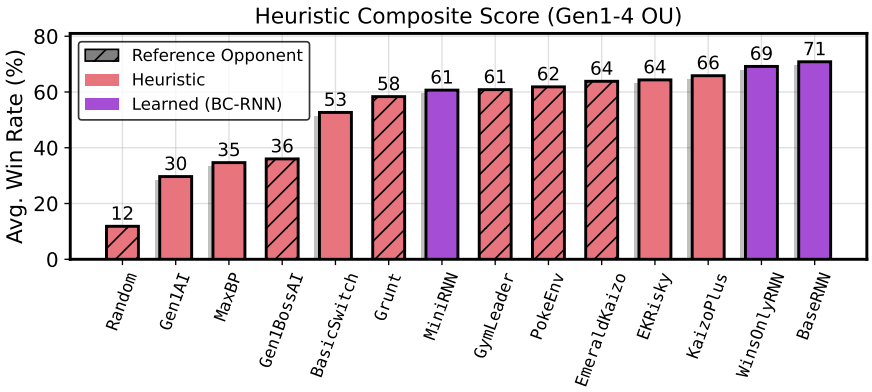


Figure 19: **Heuristic Composite Scores.** The average win rate against six of our heuristic opponents creates a data-agnostic reference point for Gens 1-4 OU, UU, NU, and Ubers.

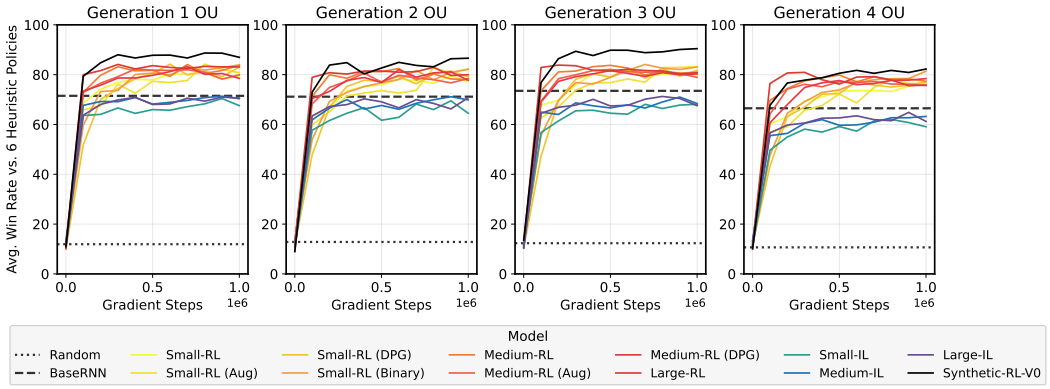


Figure 20: **Heuristic Composite Learning Curves.** Performance converges quickly but shows no sign of degrading over long training runs. BC and offline RL form two clear clusters with $\mathcal{L}_{\text{actor}}$ changes and model size having no clear impact.

	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 4: **SyntheticRL-V2 Self-Play Win Rates.** 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.

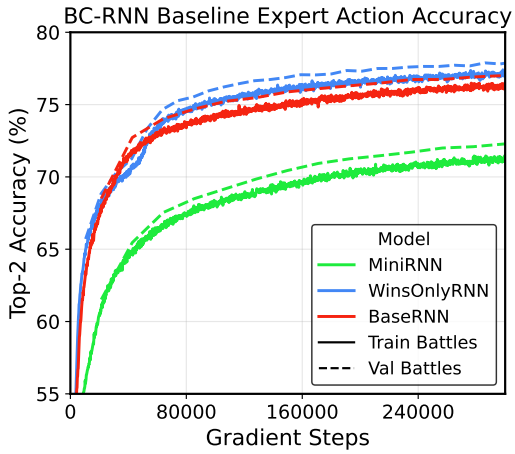


Figure 21: **BC-RNN Accuracy**. Pokémon action labels are high-entropy and we find Top-2 accuracy to be a more useful metric for tuning. "BaseRNN" is 3.5M params, "MiniRNN" ablates to 800k, and "WinsOnlyRNN" follows the filtered BC approach of only imitating decisions from the POV of the winning player (cutting its train/val sets in half).

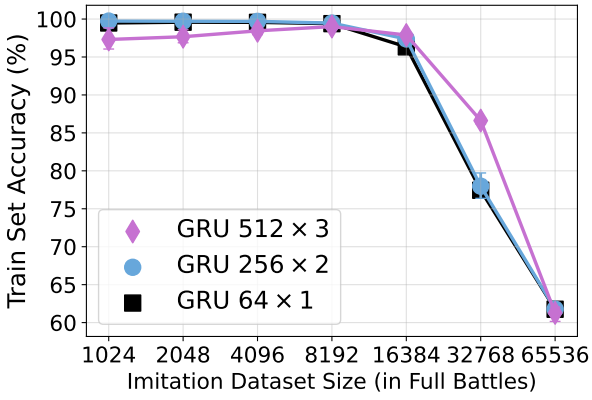


Figure 22: **Underfitting on the PS Replay Dataset.** We report the train-set accuracy of (small) recurrent BC policies on increasingly large datasets of human gameplay. Error bars denote the maximum and minimum over four random subsets. Model sizes are reported by their hidden state and number of recurrent layers

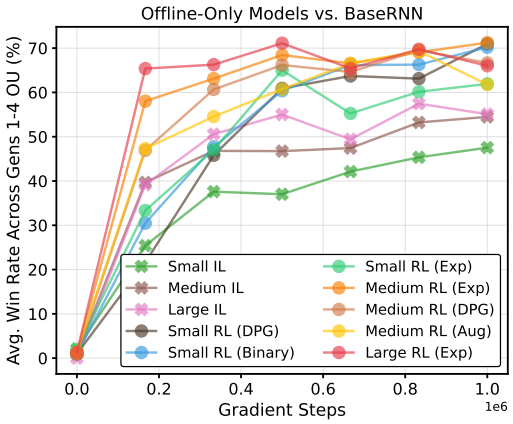


Figure 23: **Transformer IL and RL vs. RNN BC.** We evaluate the performance of Transformer policies trained on the offline replay dataset against a smaller RNN-based model designed for CPU-only inference. The RL updates do not display meaningfully distinct performance, but outperform BC at all model sizes.

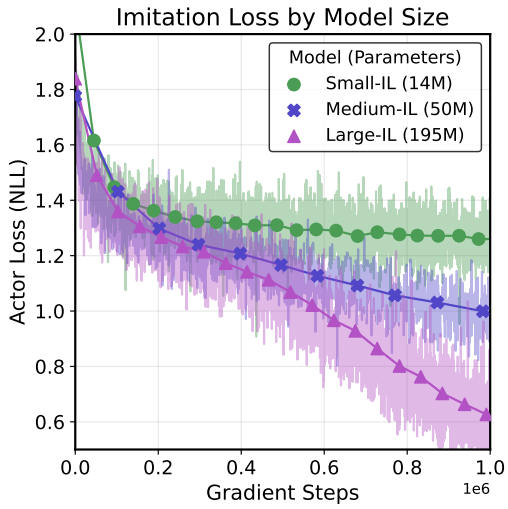


Figure 24: **Transformer IL Train Loss Curves.**

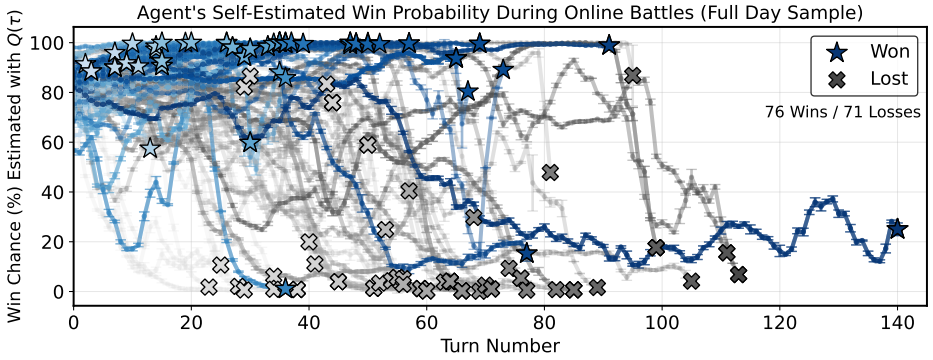


Figure 25: Q**-functions as a win estimate.** We track critic value predictions (for $\gamma = .999$) during battles across a 24-hour period of the Large-RL model's gameplay on the PS ladder. If we simplify by ignoring the reward function's small shaping terms and the discount favor, we can plot these values as a more interpretable estimate win probability. We mark these value series by their true outcome. Small error bars denote two standard deviations over the ensemble of 4 critics.