A Self-Play Policy Optimization Approach to Battling Pokémon

Dan Huang
San Francisco, CA, United States
hello@yuzeh.com

Scott Lee
Irvine, CA, United States
randomperson2727@gmail.com

***Abstract*—Pokémon is a popular role-playing video game franchise with a long-lived competitive scene that has evolved throughout the last two decades. The game exhibits several properties that come together to present a worthy challenge for AI agents to tackle. In this work, we present a low-cost self-play based reinforcement learning approach to the competitive battling aspect of the game. The proposed agent was tested and trained in a variety of environments designed to simulate possible use cases of such an AI. Experiments demonstrate that the agent is capable of performing on par with the state of the art in search-based Pokémon AI, as well as being competitive with human players on a popular matchmaking ladder. Furthermore, we investigate the transferability of trained skill–whether an agent trained in one environment performs well in a different environment.**

I. Introduction

In game-playing AI, there are several known challenges that that make both design and implementation of agents difficult. For example, some techniques, like minimax search, often require a means of simulating the impact of a decision on a game state. The simulation itself can be computationally expensive, and implementation is a non-trivial task, particularly for complex, partially observable, and stochastic games. On the other hand, techniques that rely on data must consider the validity of their data, representation of a complex state, and the cost of training. To further compound these challenges, updates to modern games will often materially change their dynamics. Whenever a new expansion upends a game's mechanics, one has to revisit their simulator or recollect data, which can be a pain point for developers and researchers.

In this work, we present a self-play based reinforcement learning (RL) approach to battling in Pokémon, a turn based adversarial game. Our approach does not require the implementation of a simulator, is low-cost to train, and performs well in a game that heavily features nondeterminism, partial observability, and a high cardinality game state with predominantly categorical features. This work discusses the algorithm, training methodology, results against a state-of-the-art search-based agent, as well as performance against human players "in the wild". We aim to show that this technique can perform well in a variety of environments, as well as investigate the ability of an agent trained on one format to perform in another.

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

II. Background

Pokémon is a popular video game franchise featuring a combat system around which a competitive scene has developed. Battles are turn-based adversarial games in which players construct teams of six Pokémon each and aim to defeat their opponent's Pokémon before they themselves are defeated. Over the course of seven major iterations, the game's ruleset and competitive metagame have changed drastically, with the game mechanics changing after each iteration. As an AI benchmark, modern Pokémon has several properties that make it challenging [1]. Game states in Pokémon are high-dimensional and the majority of its features are both categorical and partially observable. Player decisions are also processed atomically, and individual moves tend to have large impacts on game state. As a result, a game state can change drastically and unpredictably between steps. The game also features teambuilding on a scale that is several orders of magnitude more complex than many MOBAs, like Dota.

Several members of the competitive Pokémon community have developed agents to tackle this problem. The methodologies comprising the current state of the art consist primarily of search-based and heuristic-based systems, with limited forays into machine learning [2] [3] [4]. We believe the RL approach presented here is novel to the space and provides a different avenue of AI development for Pokémon battles that manages to deal well with several of the challenges that the game presents.

III. Methodology

We treat a Pokémon battle as a POMDP[1], where actions correspond to phases in the battle where the player is required to make a decision. Examples of actions include "switch to the Pokémon in slot 3" or "use move 2 on the active Pokémon." For the rest of this work, we let n denote the cardinality of the action space of the POMDP.

Inspired by self-play systems OpenAI's Dota AI [5], we represent our agent using an actor-critic neural network. Actor-critic RL methods [6] combine policy-based and value-based RL methods by predicting both policy and value for a given state, and then using the value prediction (the "critic") as

[1]While formalisms such as the *extensive-form game* more accurately describe gameplay in multi-player games, treating the game as a POMDP allows us to use methods that have been developed by the deep reinforcement learning community in recent years.

Feature	Type	Dims	Description
species	categorical	1×1023	e.g. Pikachu
item	categorical	1×368	e.g. Leftovers, Choice Band
ability	categorical	1×238	e.g. Rough Skin, Shadow Tag
moveset	categorical	4×731	e.g. Flamethrower, Surf
lastmove	categorical	1×731	The latest move used
stats	continuous	6	HP, Atk, Def, SpA, SpD, Spe
boosts	continuous	6	Temporary boosts for stats
hp	continuous	1	Current number of hitpoints
maxhp	continuous	1	Number of HP at full health
ppUsed	continuous	4	# times a move was used
active	indicator	1	1 if Pokémon is active, else 0
fainted	indicator	1	1 if Pokémon has no HP, else 0
status	indicator	28	e.g. sleep, burn, paralysis
types	indicator	18	e.g. Bug, Fire
volatiles	indicator	23	e.g. Leech Seed, Perish Song

TABLE I: An incomplete list of features used to describe a single Pokémon in a battle. These features are fed as part of the input to our Pokémon battling agent.

an estimate of expected return when updating the policy prediction (the "actor"). In our case, both actor and critic are represented by a two-headed neural network f_θ (parameterized by θ). Neural network training proceeds via self-play RL.

A. Neural network

The input to the neural network is the current state of the game, from the point of view of the player. This is a complex multi-level tree-like structure:

1) The **battle** consists of two **teams**, along with weather effects.
2) Each **team** consists of six **Pokémon**, along with side conditions (e.g. entry hazards, Reflect).
3) Each **Pokémon** has many features. Table I contains a partial list.

The network has two outputs: 1) a probability distribution $\pi \in \mathbb{R}^n$ over actions to take, and 2) an estimate of player strength in the current state $v \in \mathbb{R}$. To compute π:

1) The network outputs an intermediate vector $p \in \mathbb{R}^n$. Each of the colored cells in Figure 1 correspond to an element of p.
2) We obtain a probability distribution $\pi' \in \mathbb{R}^n$ by using the softmax function: $\pi'_i = \frac{\exp p_i}{\sum_{j=1}^{n} \exp p_j}$.
3) Because not every action is valid in every state (for example, a switch to a Pokémon is invalid if that Pokémon is already fainted), we need to make sure our agent has zero probability of taking invalid actions. To do this, we take a mask $s \in \{0,1\}^n$ as part of the input, and renormalize probabilities to obtain π: $\pi_i = \frac{s_i \pi'_i}{s^{\mathrm{T}} \pi'}$.

Two key design decisions are worth mentioning here. First, we use 128-dimensional entity embedding layers for each of the categorical variables. This enables us to capture similarities between different moves, species, items, and abilities without having to directly model their (often complicated) effects. Second, the parameters for computing p from above are shared among all n actions.

The network is described by Figure 1 and contains 1,327,618 parameters in total.

B. Training

Algorithm 1: The self-play neural network training loop for our Pokémon battling agent.

initialize θ_0 with random values
$i \leftarrow 0$
while *true* **do**
  simulate m self-play matches using f_{θ_i} as both players. Sample from π to select the action to take at each turn.
  update the neural network parameters using the $2m$ self-play matches as training data to obtain new neural network parameters θ_{i+1}.
  $i \leftarrow i + 1$
end

Training for our agent proceeds serially as described in Algorithm 1. We assign a reward of $+1$ for a win and -1 for a loss at the end of a match. To speed up learning, we construct a dense reward signal using reward shaping. Auxiliary rewards are assigned based on events that occur over the course of the match. For example, we add a reward of -0.0125 when a Pokémon on the player's side faints, and a reward of $+0.0025$ whenever the player's Pokémon makes a super effective move.

To update the neural network, we use Proximal Policy Optimization [7], which optimizes an objective function that combines expected reward, accuracy of state-value prediction, and a bonus for high entropy policies. To reduce the variance of policy gradient estimates, we use Generalized Advantage Estimation [8].

In our experiments, training proceeded for 500 iterations of the loop (so our end parameters are θ_{500}). For the number of matches played per iteration, we picked $m = 7680$ (a completely arbitrary choice). At the end of training, 3,840,000 self-play matches had been played by the neural network.

The cost of training was relatively low; the agent was trained using Google Cloud Platform over the course of 6 days and cost approximately $91 USD.

IV. Format 1: Random Battling

The agent was trained on Pokémon Showdown's `gen7randombattle` format, in which both players are given randomly[2] generated teams to use. This format is useful as it provides wide coverage on the teams the agent may encounter or be made to use.

A. Performance Against Other Agents

After training, the agent was evaluated against a set of agents of varying complexity:

- `random` - The agent selects a random move each turn.

[2]For each Pokémon, moves are randomly drawn from a pool that contains 4 to 14 moves, depending on the Pokémon. This is not uniformly random and is done to ensure a good gameplay experience.

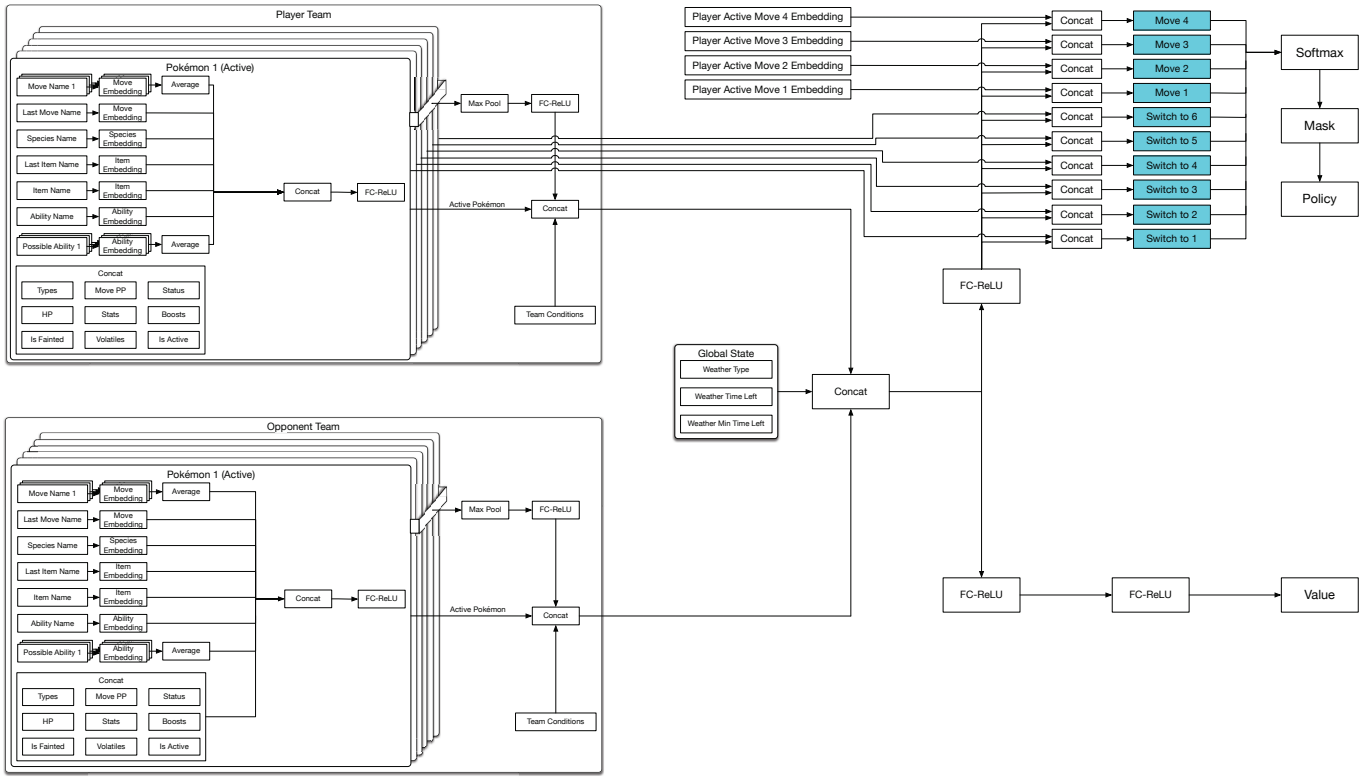

Fig. 1: The actor-critic neural network architecture describing our Pokémon battling agent. Input is the game state at a given turn. Outputs are 1) a probability distribution over actions to take (denoted "Policy") and 2) an estimate of the strength of the agent's position in the current state (denoted "Value"). Best viewed on a computer.

Opponent	Wins	Losses
`random`	995	5
`most-damage`	929	71
`most-damage-typed`	829	171
`pmariglia`	612	388

TABLE II: The performance of `RL-rb` against a set of opponents in `gen7randombattle`.

- `most-damage` - The agent selects the highest damage move each turn. This aligns with beginner level play.
- `most-damage-typed` - Similar to `most-damage`, except that the agent has knowledge of Pokémon type weaknesses and resistances.
- `pmariglia` [3] - An open-source tree-search agent with a bespoke heuristic-based state evaluator.

The random nature of team selection in this format means that matches can be unbalanced before play begins; outcomes in matches are influenced by the quality of the teams as well as the quality of the agents. To reduce the variance in our estimates of comparative skill, we simulate many matches between pairs of competitors.

We play 1000 `gen7randombattle` matches between `RL-rb` and each of the other agents. Table II shows the results. `RL-rb` outperforms the simple heuristic based ones, while `pmariglia` offers a usable baseline against which we can evaluate subsequent experiments.

B. Performance Against Human Players

We sought to determine whether `RL-rb` was competitive against humans. Every 100 training iterations, we would evaluate it by having it play 300 matches on the `gen7randombattle` ladder on the Pokémon Showdown server[3], and used its Glicko-1 [9] rating[4] at the end of matchmaking as an indicator of skill. At the end of 500 training iterations, `RL-rb` attains a 1677 Glicko-1 rating, which roughly corresponds to a 72% chance of defeating an opponent selected uniformly at random from the ladder.

V. Format 2: Established Metagame

A. Experiment Structure

Communities surrounding competitive games with strong teambuilding (or deckbuilding) elements tend to converge on a set of teams (or decks) that are commonly used among the population of players. This is often referred to as the *metagame*. Teams in a metagame tend to exhibit complexity in strategy; successful use of a complex team requires the

[3]https://play.pokemonshowdown.com/

[4]Pokémon Showdown exposes an Elo rating for competitors, but we do not use that because their Elo rating is not a true Elo system. [10] contains a discussion of rating systems for Pokémon Showdown servers.

		pmariglia Team		
		OF	PS	TR
RL-rb Team	OF	0.435	0.21	0.87
	PS	0.82	0.24	0.8
	TR	0.145	0.12	0.635

TABLE III: The performance of RL-rb against against pmariglia in 3team. Each value represents the win rate of RL-rb after 200 matches.

		pmariglia Team		
		OF	PS	TR
RL-meta Team	OF	0.73	0.555	0.99
	PS	0.975	0.895	0.97
	TR	0.81	0.67	0.96

TABLE IV: The performance of RL-meta against against pmariglia in 3team. Each value represents the win rate of RL-meta after 200 matches.

execution of multi-step action sequences. To simulate this environment and investigate RL-rb's ability to execute more complex strategies, we designed three teams of varying levels of complexity. The three teams are:

- Offense (OF) - A minimally complex team consisting of 6 offensively strong Pokémon.
- Psyspam (PS) - A more complex team that is currently very popular in the Pokémon metagame. The team revolves around Tapu Lele, a Pokémon notable for its ability to create a favorable game state by controlling terrain.
- Trick Room (TR) - A complex niche team made up of Pokémon that revolves around the use of Trick Room.

We call the resulting three-team metagame 3team.

B. Results

1) RL-rb: Table III shows the results of playing RL-rb against pmariglia in 3team. As can be seen, RL-rb performs somewhat poorly against pmariglia. Empirically inspecting a sample of replays from this experiment indicates that many of the agent's losses can be attributed either to an inability to properly utilize the team's core strategy or an unfavorable edge case to a conventional strategy.

2) RL-meta: A follow-up experiment was performed to investigate whether an agent repeatedly exposed to the three teams would perform better in 3team. We took the neural network for RL-rb, and trained for another 50 iterations. For each of the $50 \times 7680 = 384,000$ matches played, both players' teams were selected at random from our set of three: $\{\text{OF}, \text{PS}, \text{TR}\}$. We call the resulting agent RL-meta.

Table IV shows RL-meta's win rates. As can be seen, the specialized agent's win rates are significantly higher than those of the generalized agent, and outperforms pmariglia in each matchup.

C. Can RL-meta effectively play gen7randombattle?

In a head-to-head matchup of RL-rb and RL-meta in the generalized format (gen7randombattle), RL-meta only wins $77/500$ matches. The specialized agent exhibited poor performance in gen7randombattle, likely because the strategies required to use the 3team teams are effectively inapplicable for the vast majority of team configurations.

VI. Discussion and Future Work

The results generally indicate that the agent can be trained to perform well in multiple environments, with the caveat that the agent must be retrained for each. This is a minor issue, as training the agent is inexpensive and can be bootstrapped using a previously trained model as a baseline. The low cost and overhead of training will also be helpful when the game undergoes significant changes upon the release of the next generation of games.

The agent also opens up several potential avenues for future work. Further improvements to the agent may possibly achieved through different network architectures or the addition of a recurrent element like an LSTM to better model human memory during the course of a game. It would be similarly interesting to investigate more expansive simulacra of the competitive metagame, with future experiments evaluating the agent's play using "metagame" teams against human players.

Additionally, there is other work to be done with Pokémon. RL-rb has been shown to be capable of using a wide variety of teams, and may provide a helpful tool in procedurally generating teams. The difference in performance between RL-rb and RL-meta surface the implication that the two formats are different enough that they require wholly different strategies. This training and evaluation approach could be useful in investigating the design and difficulty of different play environments.

References

[1] S. Lee and J. Togelius, "Showdown ai competition," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2017, pp. 191–198.

[2] vasumv, "vasumv/pokemon_ai," Apr 2016, accessed: 2019-05-13. [Online]. Available: https://github.com/vasumv/pokemon_ai

[3] pmariglia, "pmariglia/showdown," Apr 2019, accessed: 2019-05-13. [Online]. Available: https://github.com/pmariglia/showdown

[4] D. Stone, "Technical machine," http://doublewise.net/pokemon/, accessed: 2019-05-13.

[5] OpenAI, "Openai five," https://blog.openai.com/openai-five/.

[6] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[8] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[9] M. E. Glickman, "The glicko system," *Boston University*, 1995.

[10] Antar, "Resource - everything you ever wanted to know about ratings," Aug 2013. [Online]. Available: https://www.smogon.com/forums/threads/everything-you-ever-wanted-to-know-about-ratings.3487422/