A Performant Game Engine and Monte-Carlo Tree Search Agent for Pokemon Battles

Derek Dong, James Glenn (advisor)

January 28, 2025

1 Introduction

Pokemon is one of, if not the, most popular and highest-grossing media franchises in the world [pkm]. Its foundation lies in role-playing games centered around a complex "turn-based" battle system where players pit the team of Pokemon they've caught against opposing teams. Players can battle against each other or against rule-based opponents built into the games. The largest Pokemon tournament each year has a prize pool of over $2,000,000, and YouTube videos detailing playthroughs of the base games or their "ROM hacks" can each garner millions of views.

Developing agents to play video games is also of particular interest in the ML community. Agents have been successfully trained to excel and even surpass human performance in games ranging from reasonably complex combinatorial games like chess to higher-branching dynamic, stochastic, and/or multi-agent games like Dota, Starcraft, and Quake [RP20]. Games have controlled, well-defined environments and, simultaneously, sufficient complexity where techniques can be demonstrated or refined and then transferred to more practical applications of high-agency AI, like robotics or large language models.

The combination of the two – developing an AI skilled at Pokemon battling – is therefore of interest to both Pokemon's vast playerbase and academic researchers.

1.1 Pokemon Battles

The basic process of a Pokemon battle consists of two players ("trainers") with "teams" of up to 6 Pokemon each. Each Pokemon has its own stats, moves, and HP values. At any point in the match, each player has one Pokemon active, and they simultaneously can choose in a turn to attack with one of its moves or switch to one of their 5 other Pokemon, for a branching factor of about 9. Once a Pokemon's HP reaches 0, they "faint," and when a trainer's Pokemon all faint, they lose.

The computational difficulty of battling comes from the complex interactions and massive state space. There are over 1,000 distinct Pokemon, not including form changes. Each Pokemon has up to two of 18 types, has one of over 200 Abilities, might be holding one of over 300 Held Items (given to it before the battle by its trainer), and has access to up to 4 of over 700 moves (taught to it before the battle by its trainer). There are further restrictions on which Ability or moves a type of Pokemon can have. Stat changes, multi-hit moves, move priority, and Statuses (e.g. Sleep or Paralysis) expand the state space, as do environmental conditions like "weather," "terrain," or "hazards." An agent must understand all of these conditions/effects and how they interact – both with the rest of their team and the opposing team's – to win the game.

A player can battle against a rule-based opponent hard-coded into the game or against another player. So there are three types of participants: a human player, an artificial agent (the development of which is the goal of the field), and the rule-based opponent. We distinguish the latter two as "AI" and "opponent" to avoid confusion.

1.2 Related Work

Pokemon battles, in their original form, are in a video game and meant to be played by a human. This, combined with the fact that the games have strict anti-piracy measures, means that they are inseparable from the GUI and animations. This is rarely a problem for human players, but a severe performance and interface bottleneck. An AI can only interface with the base games by simulating clicks and parsing/processing entire still frames to understand the game state. Most techniques also require significant data, and therefore the speed for a large number of playthroughs. Thus, battle simulators are a crucial tool for research in the field.

One popular existing framework to simulate Pokemon battles is Pokemon Showdown [sho, LT17]. The implementation allows players to easily create their own teams (instead of catching/training them from scratch in the base games) and battle each other in their browsers. It also provides a protocol for programs to send input, circumventing the GUI/animation requirements. However, it's built for generality and security. Playing against an opponent that chooses moves that are strong but impossible for their Pokemon to have (recall that such pairs maybe restricted, especially in certain formats) wouldn't be fun to play against, and so Pokemon Showdown performs certain validations on each input. Showdown also supports features like a timer (to prevent players from refusing to choose a move) and move changes (so long as the other player hasn't chosen their move yet) for UX that present overhead in both the underlying algorithm and the design choices of the system itself, being implemented in TypeScript for dynamism.

Additionally, interesting work has been done in training agents themselves. The two most common approaches are MCTS [Lin19] and reinforcement learning [HL19, SRLR20], and the most impressive work (from 2024) uses a neural network with MCTS similarly to AlphaGo to achieve a peak of top-8 rank in the Pokemon Showdown ladder for its event [Wan24]. These works tend to overcome performance issues by aggressively parallelizing training/simulation, but a higher-performance engine should allow more work to be concentrated on developing the approach itself.

The primary contribution of this project is to provide a performant environment/engine for further development of artificial agents, and in particular deep/reinforcement learning agents, for Pokemon battling. We essentially convert the Pokemon Showdown simulator from TypeScript to C++, removing legality checks and simplifying the battle stream interface to reduce performance overhead. We also implement a basic Monte-Carlo Tree Search algorithm to showcase the engine, and benchmark its performance against the opponent in a few pre-determined matchups.

We specifically implement the mechanics and opponent from Pokemon Run and Bun, a ROM hack of the Pokemon Emerald game. We chose this game because of its popularity, relative modernity of mechanics, and most importantly, clearly documented opponent behavior. However, a goal of the project is to produce a code-base that can be adapted to other games' mechanics with minimal pain.

2 Timeline and Deliverables

2.1 Implementing and Testing the Engine

At each step, test cases will be written and checked (with performance improvements kept track of) by comparing to the Pokemon Showdown simulator's results.

- **Run Pokemon Showdown simulator** and set up timing benchmarks (by 2/3)
- **Implement basic mechanics and testing interface:** stats, damage, flinch, accuracy, types, speed, PP (by 2/7)
- **Implement priority mechanics** (by 2/14)

- **Implement timer mechanics:** weather, Reflect/Light Screen, Trick Room, Perish Song, Hyper Beam, etc. (by 2/21)
- **Implement ability mechanics:** special cases include Disguise, Trace, Mold Breaker, etc. (by 3/7)
- **Implement item mechanics** (by 3/21)
- **Implement form mechanics:** Transform, Mega Evolution, Stance Change, etc. (by 3/28)
- **Implement special cases:** Mimic, Mirror Move, Metronome, Illusion, U-Turn type moves, etc. (by 4/4)

2.2 Training and Benchmarking an MCTS Agent

- Write opponent logic (by 4/11)
- Write test scenarios (by 4/15)
- Evaluate performance of agent against opponent on test cases (by 4/18)

2.3 Final Products

- **Draft**
- **Poster:** create final poster (by 4/25)
- **Write/revise** final paper (by 4/30)
- **Document** (or at least finalize documentation) for code-base (by 5/1)

References

[HL19] Dan Huang and Scott Lee. A self-play policy optimization approach to battling pokémon. In *2019 IEEE Conference on Games (CoG)*, pages 1–4, 2019.

[Lin19] Norstrom Linus. Comparison of artificial intelligence algorithms for pokémon battles. Master's thesis, Chalmers University of Technology, 2019.

[LT17] Scott Lee and Julian Togelius. Showdown ai competition. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 191–198, 2017.

[pkm] Pokemon. `https://www.pokemon.com/us`. Accessed: 2025-01-28.

[RP20] Sebastian Risi and Mike Preuss. From chess and atari to starcraft and beyond: How game ai is driving the world of ai. *KI - Künstliche Intelligenz*, 34, 02 2020.

[sho] Pokemon showdown. `https://pokemonshowdown.com/`. Accessed: 2025-01-28.

[SRLR20] David Simões, Simão Reis, Nuno Lau, and Luís Paulo Reis. Competitive deep reinforcement learning over a pokémon battling simulator. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 40–45, 2020.

[Wan24] Jett Wang. Winning at pokémon random battles using reinforcement learning. Master's thesis, MIT, 2024.