Applying Q-Learning to Optimize Pokémon Trainer Decision Making

John Greaney-Cheng
Stanford University
johngc@stanford.edu

Shyam Sai Bethina
Stanford University
sbethina@stanford.edu

Aaron Sequeira
Stanford University
aarons01@stanford.edu

***Abstract*—Pokémon is a turn-based game that requires strategic decision-making in factors such as type effectiveness, Same-Type Attack Bonus (STAB), power versus accuracy, and switching choices. In this project, our goal was to train an agent to learn these core battle skills as a step toward optimal play. We used a model-free reinforcement learning approach based on Q-learning with a light forced-exploration phase and trained the agent separately on fixed battle scenarios to enable localized learning. Evaluation against a random agent showed that the trained agent reliably learned the targeted strategic behaviors and achieved substantially high win rates, with a win rate between $75\% - 99\%$ in fixed scenarios and 67% in randomly sampled 2v2 battles. Furthermore, the agent showed respectable performance in robust randomized situations, indicating the model's effectiveness to apply greedy strategies and explore others across thousands of situations.**

Introduction

Pokémon is a popular game worldwide, where two players, or "trainers," engage in a turn-based battle game. They deploy their creatures (Pokémon) against each other, select moves, switch teammates if available, and try to outmaneuver each other under uncertainty. Although the earliest games featured only 151 Pokémon, modern generations include over 1000 species, hundreds of moves, and many interacting mechanics, creating an enormous strategic space. Each turn involves predicting the opponent's action, managing type match-ups, and accounting for randomness of damage rolls and critical hits. This richness makes Pokémon an ideal testbed for sequential decision making. In this project, we model simplified Gen-3 Pokémon battles as a Markov Decision Process (MDP) and use reinforcement learning to train an automated trainer agent. Rather than learning among all possible Pokémon, we restrict the agent to a small set of carefully designed battle scenarios that isolate key tactical skills such as exploiting type advantages, avoiding ineffective moves, and deciding when switching is beneficial. Our goal is to study how these fundamental battle strategies can emerge purely from experience.

Motivation

Competitive Pokémon battling has grown into a large, global ecosystem supported by online simulators, community-driven strategy analysis, and officially sanctioned national and international tournaments. The depth of strategy required, ranging from team construction to in-battle predictions, makes it an especially compelling domain for studying learned decision making. Players invest significant effort into mastering type matchups, coverage, tempo, and long-term planning under uncertainty. More broadly, Pokémon shares key properties with many modern strategy-oriented video games: large state spaces, long decision horizons, stochastic outcomes, and tightly coupled game mechanics. Yet, compared to classic board games like Chess or Go, where near-optimal AI play has been extensively explored, these modern strategy games have received far less systematic attention in reinforcement learning research. This gap motivates the question of how well standard RL methods can adapt to environments that are less rigid, more uncertain, and structurally richer than traditional board games. By training an agent to learn core Pokémon battle concepts such as type effectiveness, Same-Type Attack Bonus (STAB), power–accuracy trade-offs, immunity awareness, and switching behavior, we investigate whether intelligent tactical decision-making can emerge in a stochastic, adversarial environment. Our simplified Gen-3 battle setting provides a controlled but realistic testbed for studying these challenges.

Fig. 1. Example 1v1 battle: Player 1 uses Treeko against Player 2 using Lotad. The interface shows current HP for all Pokémon, four moves per active Pokémon, and the type of the pre-selected move. Not pictured is a second part of the interface where you can choose to switch to a healthy teammate.

Problem Definition

The goal of our project is to train a reinforcement learning agent that can make strong tactical decisions in simplified Pokémon battle scenarios. Battles follow the standard turn-based format: at each turn, both players have one active Pokémon on the field and choose an action simultaneously, either selecting a damaging move or switching to another healthy teammate.

To keep the problem tractable, we restrict learning to a small, fixed set of Generation 3 Pokémon and hand-designed battle scenarios. Each scenario defines its own environment with fixed teams and move sets, and we train a separate agent per scenario. This mirrors the class example paper's MDP formulation while focusing on specific battle situations instead of the full game.

Each Pokémon in our simulator is defined by its six base stats (HP, Attack, Defense, Special Attack, Special Defense, Speed), typing, and a set of up to four damaging moves. To isolate strategic decision making, we remove status moves, items, abilities, weather, and most secondary effects, and restrict the simulator to damage-dealing moves with standard Gen-3 damage and accuracy randomness.

Within this framework, a battle strategy is a sequence of moves and switches selected over the course of the fight. A strong strategy must reason about:

- Type effectiveness and immunities
- Same-Type Attack Bonus (STAB)
- Power trade-offs between moves
- When switching is advantageous

A battle ends when all Pokémon on one side faint. The agent receives a positive terminal reward for winning and a negative reward for losing, with additional intermediate rewards based on relative team health during the battle.

Our objective is to train an agent that can consistently win in fixed battle scenarios by learning these foundational Pokémon tactics. This provides a controlled stepping stone toward more complex strategic decision-making in the full Pokémon battle ecosystem.

Literature Review

Efforts to develop optimal Pokémon battle agents have been explored both at Stanford University and in the broader research community. Within Stanford, students have constructed AI agents designed to play competitively on Pokémon Showdown, an online battle simulator. Khosla, Lin, and Qi [1] developed an expectimax-based agent whose evaluation function was trained on 20,000 battle replays using a TD-Lambda approach, achieving an ELO rating of 1344. Similarly, Ho and Ramesh [2] implemented a depth-2 minimax agent with branch-and-bound pruning, move ordering, and a speed–type effectiveness evaluation function, reaching an ELO of 1270.

Research outside Stanford has also produced several approaches to Pokémon battle AI. Panumate and Iida [3] evaluated four agents with distinct game refinement strategies (Random AI, Attack AI, Smart-Attack AI, and Smart-Defense AI), showing that different strategies are optimal against opponents of different skill levels. Lee and Togelius [4] compared Pruned BFS, Minimax, and One-Turn Look-Ahead agents and found that all three outperformed simpler baselines in head-to-head battles.

While prior work focuses primarily on search-based decision making (expectimax, minimax, BFS) with handcrafted evaluation functions, our approach instead emphasizes reinforcement learning from direct interaction. Rather than learning on the full Pokémon Showdown ladder with hundreds of Pokémon and hidden information, we restrict the environment to a set of fixed, interpretable battle scenarios. This transforms the problem into a set of small, fully observable MDPs where model-free Q-learning is feasible. We follow the classic formulation introduced by Watkins and Dayan and popularized by Sutton and Barto, in which Q-learning is an off-policy, model-free temporal-difference method that bootstraps from its own value estimates [5].

Challenges

Building an intelligent agent for Pokémon battles introduces several challenges, even under our simplified setting. A primary difficulty is the size of the underlying state space. Although we restrict ourselves to the third generation, which contains 135 Pokémon and 103 moves, the number of possible team configurations, type matchups, and move outcomes remains large. Even without status moves or secondary effects, the agent must navigate a strategic landscape shaped by type effectiveness, STAB, base power differences, and switching decisions.

Another challenge lies in accurately modeling the battle environment. Even in our simplified setting without long-term status conditions, abilities, items, or most secondary effects, many details must still be captured correctly: type interactions, accuracy and damage rolls, turn order, and switching rules. Ensuring that this logic is both faithful to the game and computationally efficient required careful design.

Training the agent also poses significant complexity. Running reinforcement learning over the full battle space would be infeasible due to the number of possible states. To overcome this, we train the agent on individual, fixed battle scenarios. Once a scenario is specified, and the corresponding player's Pokémon, movesets, and the initial matchup are defined, the resulting state space becomes restricted enough to enable a form of local search. Within each scenario, the agent must still learn nontrivial skills such as selecting moves based on typing, choosing between physical and special attacks, evaluating when switching is beneficial, and balancing long-term survival against immediate damage.

Implementation

We developed a custom Pokémon battle simulator to provide a controlled environment for our Q-learning agent. The simulator retains core Gen-3-style mechanics such as type interactions, accuracy checks, and damage rolls, but removes long-term status conditions, abilities, items, and most secondary effects. This lets us focus on strategic move and switching decisions rather than modeling every mechanic in the full games.

The simulator is written in Python and uses data from PokéAPI [6] to load species, stats, types, and moves from Generation 3. It exposes a battle environment interface that returns structured observations and accepts discrete action indices for use by the Q-learning agent.

A. Data Structure

We represent the game state using two main data structures: a Pokémon object and a Move object.

Pokémon: Each Pokémon object has the following fields:

- **ID** and species information
- **Base stats**: HP, Attack, Defense, Special Attack, Special Defense, Speed
- **Level**
- **Type(s)** (one or two types)
- **Current HP**, how healthy they are
- **Moves**: up to four move IDs

Move: Each move object has the following fields:

- **ID** and type
- **Power** and damage class (physical or special)
- **Accuracy**
- **Priority** (overriding speed when nonzero)
- **Target** (self, opponent, etc.)

B. Damage Calculation

We use the standard Pokémon damage formula:

$$\text{Damage} = \left(\frac{\left(\frac{2\times\text{Level}}{5} + 2\right) \times \text{Power} \times \frac{A}{D}}{50} + 2 \right) \times \text{Modifier}$$

$$\text{Modifier} = \text{TypeModifier} \times \text{STAB},$$

where A and D are the relevant attacking and defending stats (Attack / Defense for physical moves, Special Attack / Special Defense for special moves), TypeModifier encodes type effectiveness and immunities, and STAB (Same-Type Attack Bonus) is applied when the move's type matches the user's type.

C. Q-learning Agent

We use model-free Q-learning to learn an action-value function $Q(s, a)$ over battle states. Episodes terminate when one side has no remaining Pokémon or when a turn cap is reached. The reward signal combines dense health-based shaping with a terminal win/loss bonus:

$$r_t = \frac{1}{100} \left(\sum_{p \in \text{team}_\text{Q}} \text{HP}(p) - \sum_{p \in \text{team}_\text{opp}} \text{HP}(p) \right)$$

at non-terminal steps, and $r_T = +1$ (win), $r_T = -1$ (loss), or $r_T = 0$ (simultaneous faint) at the end of the battle. This preserves the primary objective of winning while providing informative feedback during the fight.

1) State and Action Spaces: The environment produces an observation containing:

- For each active Pokémon: species, current/max HP, types, and six combat stats.
- For each team: the name and current/max HP of every party member.

Actions are encoded as integers. Indices 0–3 correspond to using one of up to four moves of the active Pokémon. Indices $4-8$ encode switches: the index maps to an offset into the list of non-fainted bench Pokémon that are not currently active. The legal action set $\mathcal{A}(s)$ depends on the current party and whether switching is enabled for that scenario.

2) Behavior Policy: During training, the agent follows a policy of constant ε-greedy behavior over $\mathcal{A}(s)$. With probability ε it selects a random legal action; otherwise, it chooses an action maximizing $Q(s, a)$. When multiple actions share the maximum value (up to a small tolerance), we break ties using the simulator's damage model, preferring the move with the highest immediate damage and treating switches as less desirable in these ties. This encourages high-value attacks early in learning when Q-values are still close to zero.

3) Update Rule and Training: We use the standard one-step, off-policy Q-learning update with learning rate α and discount factor γ. Unless otherwise noted, we use $\varepsilon = 0.50$, $\alpha = 0.40$, and $\gamma = 0.97$. Scenario 0 uses $\varepsilon = 0$ to check that greedy play and environment dynamics behave as expected. We choose γ close to 1 so that terminal wins and losses strongly influence value estimates over battles lasting tens of turns. The learning rate $\alpha = 0.40$ allows the Q-values to adapt over a few thousand episodes without oscillating excessively under noisy rewards. Lastly, we keep ε fixed at 0.50 so that the agent continues to explore both moves and switches and avoids prematurely locking into a single pattern of play.

Each scenario uses its own number of training episodes (typically between 2,500 and 6,000) and switching configuration. We select episode counts so that, within reasonable computation limits, the win rate in that scenario stabilizes above roughly 90%.

Algorithm 1 Q-learning with ε-greedy exploration

```
Initialize Q(s, a) ← 0 for all (s, a)
for episode = 1 to N do
    Reset environment, observe o; set s ← Serialize(o)
    while not terminal and t < max_turns do
        Choose a using ε-greedy over A(s) w.r.t. Q
        Execute a; observe r, next observation o', and terminal flag
        s' ← Serialize(o')
        y ← r + γ · 1_{¬terminal} max_{a'∈A(s')} Q(s', a')
        Q(s, a) ← Q(s, a) + α(y − Q(s, a))
        s ← s'
return Q
```

4) Opponent Policy: The opponent follows a simple random policy. At each turn it selects a damaging move uniformly at random from the moves available to its active Pokémon. If switching is enabled and there is at least one non-fainted bench Pokémon, then with probability 0.10 it instead chooses a random switch action. When a Pokémon faints, the environment forces that side to switch to one of the remaining healthy Pokémon on its team.

5) Evaluation: After training, we set $\varepsilon = 0$ and evaluate the learned policy in greedy mode. For each fixed scenario, we run 1,000 evaluation episodes and record the Q agent's win rate and the average number of turns per battle. For the randomized

Scenario 7, we repeat this process across many independently sampled match-ups and report both the win rate for a single sampled scenario and the cumulative win rate aggregated over 100 random draws.

TEST METHODOLOGY

We evaluate the Q-learning agent on a suite of scenarios S0–S7, each targeting a specific tactical skill: pruning useless actions, choosing strong attacks, using switching, and exploiting defensive weaknesses. In all experiments the Q agent (Player 1) plays against the random baseline (Player 2) defined in Section *Implementation*. For each fixed scenario we train as described in Section *Q-learning Agent* and then evaluate the greedy policy ($\varepsilon = 0$) over 1,000 episodes computing an average win rate percentage. We made deterministic Pokemon selections for S0-S6 based on how well they fit the designed situation, but several other Pokemon with similar attributes could be used. For the randomized scenario S7, we randomly created 100 different random matches and ran 300 evaluation episodes per random draw to compute an average win rate across all selections.

Scenario 0: Sanity Check (Mirror Zangoose 1v1)

Scenario 0 is a symmetric 1v1 Zangoose mirror. Both sides use a normal-type Pokémon, Zangoose with two damaging moves: `dig` (Ground-type) and `double-edge` (Normal-type). In our damage model, `double-edge` is strictly better in the mirror because it has higher base power and receives a same-type attack bonus. We train with $\varepsilon = 0$ to ensure purely greedy play. A correct learned policy should always choose `double-edge`, win most games, and finish battles in one or two turns. This demonstrates that the agent will prefer higher damaging moves assuming all moves are equally accurate.

Scenario 1: Ignoring Useless Normal Moves (Zangoose vs Dusclops)

Scenario 1 is a 1v1 between Zangoose and the Ghost-type Dusclops. Zangoose has four moves: `shadow-ball` (Ghost-type) and three Normal-type moves (`double-edge`, `scratch`, `tackle`). Dusclops is immune to Normal-type attacks, so only `shadow-ball` can deal damage. A random agent wastes many turns on ineffective moves; a learned agent should discover that `shadow-ball` is the only useful option and converge to using it almost exclusively.

Scenario 2: 2v2 mirror with Switching (Dugtrio + Gyarados)

Scenario 2 is a 2v2 battle with mirrored teams of Dugtrio and Gyarados. Dugtrio is a Ground-type attacker with `earthquake`, `rock-slide`, `mud-slap`, and `scratch`, while Gyarados is a Water/Flying attacker with `hydro-pump`, `waterfall`, `tackle`, and `whirlpool`. Both players may switch. Gyarados's Water-type moves are very strong into Dugtrio, while Dugtrio's Ground-type moves are weak into Gyarados but `rock-slide` can still threaten it. A strong policy should use high-damage moves in good matchups and switch to Gyarados when facing an opposing Dugtrio, rather than switching arbitrarily. Figure 2 illustrates the 2v2 structure used in our multi-Pokémon scenarios, with one active Pokémon per side and a bench of switch targets. As this scenario starts with Dugtrio as active with Gyarados in the back, the ideal action for the agent is to switch the first turn. Each trial thus tests whether the agent can recognize when switching is the optimal move.

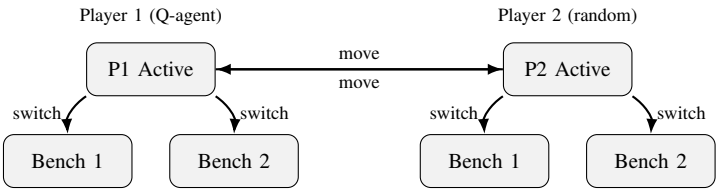

Fig. 2. Schematic of a 2v2 battle: each player has one active Pokémon and a bench of healthy teammates. On each turn, the agent may choose a damaging move or a switch action (when switching is enabled in the scenario).

Scenario 3: 2v2 mirror without Switching (Dugtrio + Gyarados)

Scenario 3 uses the same teams as Scenario 2 but disables switching for both players. This is to compare them to demonstrate the effectiveness of switching, meaning Scenario 2 should have a higher win rate than Scenario 3. Each side sends out Dugtrio first and then Gyarados after Dugtrio faints. This removes team management and isolates move choice. A learned agent should favor strong, type-appropriate moves such as `earthquake` in Dugtrio–Dugtrio matchups and Water-type attacks like `hydro-pump` when Gyarados faces Dugtrio, while mostly avoiding low-impact options like `scratch` or `mud-slap`.

Scenario 4: Power vs Reliability (Magneton vs Gyarados)

Scenario 4 is a 1v1 between Magneton and Gyarados. Magneton's key Electric-type moves are `zap-cannon` (very high power, very low accuracy) and `thunderbolt` (slightly lower power, perfect accuracy), plus weaker Normal-type moves. Both Electric moves are extremely strong against Gyarados's Water/Flying typing, so the main trade-off is raw power versus expected damage per turn. A random agent treats `zap-cannon` and `thunderbolt` similarly; a learned agent should prefer the more reliable `thunderbolt` and thereby win more quickly and consistently.

Scenario 5: STAB vs Coverage (Zangoose vs Lairon)

Scenario 5 is a 1v1 between Zangoose and Lairon. Zangoose has Normal-type moves `double-edge` and `tackle`, the Ground-type coverage move `dig`, and `shadow-ball`. Lairon is Steel/Rock-type, which heavily resists Normal attacks but is very weak to Ground. As a result, `double-edge` receives STAB but still does poor damage, while `dig` does much more. The correct behavior is to ignore the STAB bonus and treat `dig` as the primary attack whenever Lairon is active. We expect the learned agent to converge toward using `dig` almost all the time, unlike the random baseline. This is to test

that the agent won't fall into local maximums i.e. choosing highest power STAB move but will take all damage modifiers into account.

Scenario 6: Targeting the Weaker Defense (Swalot vs Kecleon)

Scenario 6 examines physical versus special preference in a controlled 1v1 setting. The agent uses the Pokémon Swalot with two main damaging moves: `facade` (a physical Normal-type move) and `shock-wave` (a special Electric-type move), plus two weaker Normal moves (`tackle`, `pound`). The random opponent is Kecleon, which is more vulnerable on the physical side. As Swalot has equal physical and special attack stats, there's no modifiers on either main move, and both main moves have the same power, the agent should choose a move based on the opponent's relative physical and special defense. A random agent treats these moves similarly, while a learned agent should infer from experience that `facade` removes more HP against Kecleon and therefore bias toward it over `shock-wave`. This scenario evaluates whether the agent takes the opponent's relative physical and special defense into account when choosing a move.

Scenario 7: Randomized 2v2 with Switching (Gen 3)

Scenario 7 is our strongest robustness test; it generalizes the 2v2 setting to randomly generated teams. For each run we sample two Gen 3 species for each side and use the database to assign up to four strong legal moves per Pokémon (highest-power moves with reasonable accuracy). Both players may switch. In the single-draw version we train and evaluate on one such matchup; in the multiple samping version we repeat this process across many draws and aggregate win rates. Here we expect the Q agent to exploit obvious type advantages and high-power moves across a variety of randomly constructed battles, achieving win rates above 50% on average despite the lack of hand-crafted structure.

Results

After learning our Q agent on each scenario, we observed the following win rates against an opponent with random decision making:

TABLE I
COMPILED TEST RESULTS ON Q-AGENT VS RANDOM

Test Scenario	Win Rate (%)	Avg. Turns	Notes
S0	75	1.0	$\varepsilon = 0$ for greedy
S1	93	3.19	
S2	95	4.91	switching = True
S3	89	5.97	switching = False
S4	98	1.0	
S5	99	1.44	
S6	84	4.14	
S7 (single)	92	5.22	random 2v2, switching = True
S7 (multiple)	67	4.98	Eval=300, reps=100

Figure 3 visualizes these win rates, highlighting that the Q-agent is strongest in the structured 1v1 and 2v2 scenarios and remains above 50% even under randomized matchups.

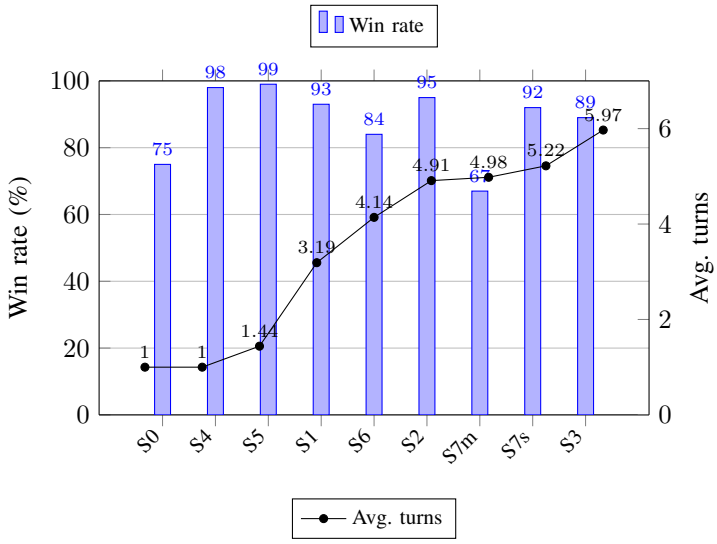

Fig. 3. Win rate and average battle length per scenario, sorted by increasing average turns. S7s = Scenario 7 (single fixed draw), S7m = Scenario 7 (multiple random draws).

In addition to aggregate win rates, we inspected individual battle replays to verify that the simulator and policies behave as intended. See the Appendix for detailed samples of given replays for scenarios S0, S2, S5, and S7.

During training we periodically evaluated intermediate policies and observed that win rates increased rapidly in early episodes and then plateaued well before the final training horizon in each scenario. This informed our choice of episode counts in Section *Q-learning Agent*, where we stop once performance appears stable. In this report we focus on win rate and average number of turns as our primary metrics; a more detailed analysis of learning curves and hyperparameter sensitivity is left for future work.

Results Discussion

Table I shows that the Q-learning agent consistently outperforms the random baseline. Across the fixed scenarios (S0–S6), win rates range from 75% to 99%, and games resolve in a small number of turns, indicating that the agent is learning high-value action sequences rather than winning by chance.

In the simpler 1v1 settings, the learned behavior aligns with the intended tactical goals. In S0, the agent reliably prefers `double-edge` over the weaker `dig`, yielding a 75% win rate where losses occur when the random agent moves first. In S1, it reaches a 93% win rate by effectively ignoring Normal-type moves that have no effect on Dusclops and focusing on `shadow-ball`. In S4 and S5, the agent almost always selects `thunderbolt` against Gyarados and `dig` against Lairon, achieving win rates of 98% and 99%, which shows that it internalizes type effectiveness, immunities, and coverage rather than simply maximizing raw power or STAB.

The 2v2 scenarios demonstrate that the agent can also handle team structure and switching. In S2, with switching enabled, the agent attains a 95% win rate and uses switches

to preserve favorable matchups (for example, bringing in Gyarados against Dugtrio) instead of switching arbitrarily. In S3, where switching is disabled, it still wins 89% of games by using strong moves such as `earthquake`, `rock-slide`, and Water-type attacks more consistently than the random baseline. The agent is also more capable of winning the match more efficiently when switching is enabled, as evidenced by 4.91 average turns in S2 vs 5.97 in S3–the highest number of average turns among all simulations. As S2 has a higher win rate than S3, we can see that the trained agent can recognize when switching is a useful option.

The more nuanced and randomized settings are harder, but the learned policy remains strong. In S6 (physical vs. special), the agent reaches an 84% win rate, suggesting that it often learns to target the opponent's weaker defensive stat. In S7, it wins 92% of games in a single sampled 2v2 matchup and 67% across 100 random draws. This drop is expected given variability in team strength and type matchups, but the win rate remains well above 50%, indicating that the heuristics learned in structured scenarios transfer to a broad range of randomly generated battles.

Future Work

Our current study is limited to small, fully observable scenarios with a tabular Q-function and a simple ε-greedy behavior policy. A natural extension is to scale the environment and state representation, for example by allowing larger teams, more diverse moves, or reintroducing additional stochastic mechanics such as accuracy and critical hits. In those settings, function approximation or state abstraction would likely be necessary to generalize across related matchups rather than storing separate values for each serialized state.

On the decision-making side, an important direction is to explore richer exploration and planning strategies. Possible extensions include decaying exploration schedules, softmax, and combining learning with search by layering Monte Carlo Tree Search (MCTS) on top of learned Q-values for short-horizon lookahead. Future work should also compare against stronger baselines, such as simple heuristic or planning-based agents, and run tuning experiments to find optimal hyperparameters (ε, α, γ) to better understand when Q-learning provides the best advantage in richer Pokémon battle settings.

Conclusion

In this project we built a custom Gen-3–style Pokémon battle simulator and used it as a controlled testbed for Q-learning. By training separate agents on a suite of fixed scenarios, we showed that a simple model-free method can acquire core tactical behaviors such as exploiting type advantages, avoiding immune moves, preferring reliable high-accuracy attacks, and using switching to reach favorable matchups. Across scenarios S0–S6 the learned policies achieve high win rates and short games against a random baseline, and they retain an advantage even in a randomized 2v2 setting.

These results suggest that standard Q-learning, combined with a carefully designed environment and reward signal, is sufficient to induce nontrivial battle strategies in a complex but structured domain. At the same time, the drop in performance under fully randomized matchups highlights the limits of our approach and motivates future work on function approximation, richer exploration policies, and search-based enhancements such as MCTS. More broadly, our framework illustrates how decision-making under uncertainty in games like Pokémon can serve as an accessible yet meaningful setting for studying reinforcement learning algorithms.

Group Member Contribution

All members contributed to the proposal, progress report, and final writeup.

John: Led project idea, scope, problem framing, and test methodology; contributed to the Abstract, Introduction, Motivation, Problem Definition, Challenges, and Literature Review.
Sai: Implemented the Pokémon simulator and contributed to the Introduction, Results Interpretation, Future Work, and Conclusion.
Aaron: Implemented the Q-learning trainer and evaluation pipeline; contributed to the Implementation and Results sections.

Acknowledgement

We used ChatGPT to help implement the Pokémon simulator, connect it to our Q-learning agent, and refine parts of the written report.

We thank Mykel Kochenderfer and the CS 238 course staff for their support throughout the quarter and for providing the opportunity to work on this final project.

References

[1] K. Khosla, L. Li, and C. Qi, "Artificial Intelligence for Pokémon Showdown," Stanford University, Stanford, CA, USA, 2018.
[2] H. Ho and V. Ramesh, "Percymon: A Pokémon Showdown Artificial Intelligence," Stanford University, Stanford, CA, USA, 2014.
[3] C. Panumate and H. Iida, "Developing Pokémon AI for Finding Comfortable Settings," Aug. 2016.
[4] S. Lee and J. Togelius, "Showdown AI competition," in *Proc. 2017 IEEE Conference on Computational Intelligence and Games (CIG)*, New York, NY, USA, 2017, pp. 191–198.
[5] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.
[6] GitHub. PokeAPI/api-data. https://github.com/PokeAPI/api-data/. 7 Dec. 2018.

APPENDIX: SAMPLE BATTLE LOGS

In this appendix we include raw battle transcripts for selected scenarios. Player 1 is the learned Q-learning agent and Player 2 is the random agent. Note that the loser of a match is the last player to have Pokémon faint.

```
S0 (mirror Zangoose 1v1)
Episode 2 | Turns: 1
  t01:
    Player 1's zangoose used double-edge and dealt 150 damage to zangoose (HP: 0/123).
    Player 2's zangoose fainted!

S2 (Dugtrio + Gyarados 2v2 with switching)
Episode 1 | Turns: 8
  t01:
    Player 1 switches to gyarados.
    Player 2's dugtrio used rock-slide and dealt 80 damage to gyarados (HP: 65/145). It's super effective!
  t02:
    Player 2 switches to gyarados.
    Player 1's gyarados used waterfall and dealt 40 damage to gyarados (HP: 105/145). It's not very effective...
  t03:
    Player 2's gyarados used whirlpool, but it missed!
    Player 1's gyarados used tackle and dealt 27 damage to gyarados (HP: 78/145).
  t04:
    Player 2's gyarados used hydro-pump, but it missed!
    Player 1's gyarados used waterfall and dealt 37 damage to gyarados (HP: 41/145). It's not very effective...
  t05:
    Player 2's gyarados used hydro-pump and dealt 21 damage to gyarados (HP: 44/145). It's not very effective...
    Player 1's gyarados used waterfall and dealt 38 damage to gyarados (HP: 3/145). It's not very effective...
  t06:
    Player 1's gyarados used waterfall and dealt 37 damage to gyarados (HP: 0/145). It's not very effective...
    Player 2's gyarados fainted!
    Player 2 sends out dugtrio!
    Player 2's dugtrio used tackle and dealt 21 damage to gyarados (HP: 23/145).
  t07:
    Player 2's dugtrio used rock-slide and dealt 167 damage to gyarados (HP: 0/145). A critical hit! It's super effective!
    Player 1's gyarados fainted!
    Player 1 sends out dugtrio!
    Player 1's dugtrio used whirlpool and dealt 24 damage to dugtrio (HP: 61/85). It's super effective!
  t08:
    Player 1's dugtrio used earthquake and dealt 127 damage to dugtrio (HP: 0/85).
    Player 2's dugtrio fainted!

S5 (Zangoose vs Lairon, STAB vs coverage)
Episode 2 | Turns: 2
  t01:
    Player 1's zangoose used double-edge and dealt 15 damage to lairon (HP: 95/110). It's not very effective...
    Player 2's lairon used rock-slide and dealt 71 damage to zangoose (HP: 52/123).
  t02:
    Player 1's zangoose used dig and dealt 109 damage to lairon (HP: 0/110). It's super effective!
    Player 2's lairon fainted!

S7 (example multi-turn battle)
Episode 3 | Turns: 6
  t01:
    Player 2's grumpig used double-edge and dealt 27 damage to tropius (HP: 122/149).
    Player 1's tropius used double-edge and dealt 49 damage to grumpig (HP: 81/130).
  t02:
    Player 2 switches to shroomish.
    Player 1's tropius used solar-beam and dealt 46 damage to shroomish (HP: 64/110). It's not very effective...
  t03:
    Player 1's tropius used hyper-beam and dealt 71 damage to shroomish (HP: 0/110).
    Player 2's shroomish fainted!
    Player 2 sends out grumpig!
    Player 2's grumpig used sludge-bomb and dealt 77 damage to tropius (HP: 45/149). It's super effective!
  t04:
    Player 2's grumpig used mega-kick, but it missed!
    Player 1's tropius used solar-beam and dealt 46 damage to grumpig (HP: 35/130).
  t05:
    Player 2's grumpig used hyper-beam and dealt 119 damage to tropius (HP: 0/149). A critical hit!
    Player 1's tropius fainted!
    Player 1 sends out kirlia!
    Player 1's kirlia used hyper-beam, but it missed!
  t06:
    Player 2's grumpig used hyper-beam and dealt 96 damage to kirlia (HP: 0/88).
    Player 1's kirlia fainted!
```