Applying Competitive Policy Optimization to Pokémon Battling

SHATAYU KULKARNI and KAMYAR AZIZZADENESHELI, Purdue University, USA

Pokémon is a popular video game involving a turn-based strategy mechanic called "battling". Battling involves building a team beforehand and employing complex tactics in-game, making it an interesting two-part problem to be solved. We seek to apply Competitive Policy Optimization (CoPO) to Pokémon battling and test its performance against humans on the online ladder Pokémon Showdown. At this time, a simplified problem involving an agent playing against itself with a fixed team was solved using an algorithm derived from Competitive Policy Optimization. The agent demonstrated knowledge of advanced strategies and successfully defeated two heuristics-based agents at about a 90% rate.

CCS Concepts: • **Computing methodologies** → **Probabilistic reasoning**; **Planning under uncertainty**; **Multi-agent planning**.

Additional Key Words and Phrases: reinforcement learning, multi-agent reinforcement learning, self-play, deep learning, neural networks

ACM Reference Format:
Shatayu Kulkarni and Kamyar Azizzadenesheli. 2021. Applying Competitive Policy Optimization to Pokémon Battling. 1, 1 (March 2021), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Multi-agent reinforcement learning is an emerging and highly popular way to discover optimal strategies to multiplayer games and similar dilemmas. It was first used by DeepMind to solve the single-player game Atari [3]. Ever since then, it has been used for more complex games, such as chess [7] and DoTA 2 [1]. Many find it attractive due to the intuitive nature of formulating a problem - with states, rewards, and actions - and the lack of a need to create a dataset. One only simply has to define the information required above, which is often self-evident in a problem, and then let the algorithm run until it converges. Oftentimes, multi-agent reinforcement learning algorithms can converge upon unexpected, although highly effective, strategies that lead to new insight on how to play the game being studied.

Competitive policy optimization (CoPO) is a novel policy gradient approach that exploits the game-theoretic nature of competitive games to derive policy updates [6]. It uses a bilinear approximation of the game objective to capture interactions between players. This concept begets two algorithms, Competitive Policy Gradient and Trust Region Competitive Policy Optimization. We will use Competitive Policy Gradient due to its increased stability relative to Trust Region Competitive Policy Optimization. A critical advantage of the algorithms derived from CoPO is that it allows an agent to reason about its opponent's potential actions and responses to its actions. In Pokémon, this means that it can potentially learn "gamesmanship" where it can make a move in response to what it predicts its opponent to do. In this paper, we will use Competitive Policy Gradient to derive an optimal strategy for Pokémon.

Pokémon is a popular multiplayer game played online. The fundamental format is two players with teams of six creatures (called Pokémon) battle in a turn-based setting. Creatures have health, moves, abilities, and various stats associated with them. Pokémon face off one at a time, with the currently active Pokémon being replaced upon losing all of its health (referred to in-game as being "knocked out"). Each turn, players simultaneously select one of four moves their Pokémon has or they switch to one of their other Pokémon. A series of in-game rules decides whose action goes first. If a Pokémon is knocked out before it gets to use its move, then the person whose Pokémon was knocked out does not execute any action until after the turn ends, when they will be given the option to send out another Pokémon.

2 RELATED WORK

There have been a few instances of using self-play methods in Pokémon Showdown. The Generalized Infinitesimal Gradient Ascent algorithm applied with the Win or Learn Fast principle (GIGA-WoLF) was able to converge to optimal behavior in a fixed, disadvantageous situation [8]. In another example of self-play, an actor-critic algorithm [2] was used to attain a Glicko-1 rating of 1677 against humans on the Pokémon Showdown random ladder, which roughly corresponds to having a 72% chance of defeating a opponent chosen uniformly at random from the Pokémon Showdown ladder [2]. The random ladder is the environment where the team is randomly chosen before battle, thus removing the burden of team-building from the player.

There have also been efforts to create a rule-based bot for Pokémon battles. The most notable is named pmariglia [4], which was able to reach a Glicko-1 rating of 1625 on the OU ladder, which corresponds to a high level of skill - but not extraordinary - on the Pokémon Showdown ladder against real-life humans. It was also able to get to a Glicko-1 rating of 1633 in the Randoms ladder, which means it has a solid general understanding of how to play with a plethora of teams against opponents using many different teams (albeit without familiarity with those teams).

This work, unlike the self-play works listed, aims to compete in the standard "OverUsed" (OU) ladder on Pokémon Showdown, which requires the player to design their own team prior to battle. For the purposes of this paper, this was done by studying a fixed team. Using a fixed team allows the agent to learn intricate strategies incorporated into the specific team as opposed to generic tactics such as exploiting type matchups. The agent will have the opportunity to experiment with matters such as which Pokémon to lead with and which matchups it prefers. This work is also not deterministic, like pmariglia, which means it has the opportunity

Authors' address: Shatayu Kulkarni, shatayu@purdue.edu; Kamyar Azizzadenesheli, kamyar@purdue.edu, Purdue University, 305 N University Street, West Lafayette, IN, USA, 47906.

XXXX-XXXX/2021/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

to learn new strategies not envisioned by a human. The tactics the agent employs will be solely discovered through its own exploration of its environment's state space.

3 MOTIVATION

Pokémon battling is a complex game with billions of different states. While there has been work on creating battling algorithms, applying Competitive Policy Optimization creates the potential for different, novel strategies. Moreover, its application to the standard battling format gives it the potential to develop more specialized solutions involving knowledge of its specific team as opposed to a general sense of "fundamentals" as proposed by other algorithms. Lastly, demonstrating the competence of this algorithm in turn means it can be applied to other multifaceted domains such as sports, robotics, and other strategy games.

4 OVERVIEW OF MULTI-AGENT REINFORCEMENT LEARNING

Multi-agent reinforcement learning is a problem involving two or more agents making decisions sequentially. It is modeled using a competitive Markov Decision Process where exactly two agents play a zero-sum game. The Markov Decision Process is modeled using a tuple $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{A}_1, \mathcal{A}_2, \mathcal{R}_1, \mathcal{R}_2, \mathcal{T}, \mathcal{P}, \gamma)$. The agents aim to maximize the reward functions $\mathcal{R}_1$ and $\mathcal{R}_2$. Since this is a zero sum game, $\mathcal{R}_1 = -\mathcal{R}_2$. They take actions $a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2$ when in states $s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2$. The state the agents transition to upon performing an action in a state depends on the transition probability $T(s'|s, a_1, a_2) \in \mathcal{T}$. In the next state, the agents select actions and the process continues. This continues until the model converges, which is judged by either letting the model train for a fixed number of epochs or by analytically judging the reward earned until the programmer is satisfied.

Multi-agent reinforcement learning is used to model Pokémon as opposed to single-agent reinforcement learning because the intrinsic nature of the game, with two opponents competing against each other, means that multi-agent reinforcement learning algorithms can be used to derive an optimal strategy for the game. Another advantage of using multi-agent reinforcement is that this agent does not "overfit" to counter any particular strategy or take undue influence from existing play. Being free of data derived from humans, such as logs of battles conducted by highly skilled humans, means that the agent is free to explore the state space and learn novel strategies that a human observer can learn from.

5 OVERVIEW OF COMPETITIVE POLICY OPTIMIZATION

Players are modeled using stochastic policies $\pi(a_k^1|s_k;\theta^1)$ and $\pi(a_k^2|s_k;\theta^2)$. Agents then generate trajectories with distribution is denoted by:

$$f(\tau;\theta^1,\theta^2) = p(s_0)\Pi_{k=0}^{|\tau|-1}\pi(a_k^1|s_k;\theta^1)\pi(a_k^2|s_k;\theta^2)T(s_{k+1}|s_k,a_k^1,a_k^2)$$

From here, the value of a state s_k can be defined as

$$V(s_k;\theta^1,\theta^2) = E_{t\ f(\cdot;\theta^1,\theta^2)}[\Sigma_{j=k}^{|\tau|-1}\gamma^{j-k}r(s_j,a_j^1,a_j^2)|s_k]$$

This function $V(s)$ represents the expected value of how much reward an agent can earn from a state s_k following policy $\pi(\theta^1)$ and $\pi(\theta^2)$.

Now that $V(s)$ is defined, the game objective can be thought of in terms of maximizing or minimizing $V(s_0)$, the expected value from the starting state. Game objective can be defined as

$$\eta(\theta^1,\theta^2) = \int_{s_0} p(s_0)V(s_0;\theta^1,\theta^2)ds_0$$

which is the expected value of all state distributions. As a consequence, the objectives of each agent can be interpreted with the below two equations respectively:

$$\theta^{1*} \in argmax_{\theta^1\in\Theta^1}\eta(\theta^1,\theta^2)$$

$$\theta^{2*} \in argmax_{\theta^2\in\Theta^2}\eta(\theta^1,\theta^2)$$

Competitive Policy Optimization exploits the game-theoretic and competitive nature of Competitive Markov Decision Processes and deploys a bilinear approximation of the game objectives for each agent:

$$argmax_{\Delta\theta^1:\Delta\theta^1+\theta^1\in\Theta^1}\Delta\theta^{1T}D_{\theta^1}\eta + \Delta\theta^{1T}D_{\theta^1\theta^2}\eta\Delta\theta^2 - \frac{1}{2\alpha}||\Delta\theta^1||^2$$

$$argmin_{\Delta\theta^2:\Delta\theta^2+\theta^2\in\Theta^2}\Delta\theta^{2T}D_{\theta^2}\eta + \Delta\theta^{1T}D_{\theta^1\theta^2}\eta\Delta\theta^2 - \frac{1}{2\alpha}||\Delta\theta^2||^2$$

The bilinear term $D_{\theta^1\theta^2}\eta$ is the smallest order term that can capture interdependence between agents. It still retains the property of being linear with respect to each player. The update rule is given by solving the two systems of linear equations in $\Delta\theta^1$ and $\Delta\theta^2$ respectively and is given by:

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha\begin{bmatrix}I & -\alpha D_{\theta^1\theta^2}\eta\\ \alpha D_{\theta^2\theta^1}\eta & I\end{bmatrix}^{-1}\begin{bmatrix}D_{\theta^1}\eta\\ -D_{\theta^2}\eta\end{bmatrix}$$

The advantage of this system is that it allows the agent to reason about its interactions with its opponent as well as its interactions with the environment. This allows it to develop significantly more complex strategies since it can identify the best action for a given state while factoring in what its opponent may do to it.

For more detail, see the original paper [6].

6 OVERVIEW OF COMPETITIVE POLICY GRADIENT

This is the algorithm derived from the principles of Competitive Policy Optimization. Given a CoMDP, players $i, j \in 1, 2, i \neq j$, and the policy parameters θ^i, θ^j:

$$D_{\theta^i}\eta = \int_\tau \Sigma_{k=0}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta^1,\theta^2)D_{\theta^i}(\log\pi(a_k^i|s_k;\theta^i))Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)d\tau$$

$$D_{\theta^i\theta^j}\eta = (1) : \int_\tau \Sigma_{k=0}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta^1,\theta^2)D_{\theta^i}(\log\pi(a_k^i|s_k;\theta^i)))$$

$$D_{\theta^j}(\log\pi(a_k^i|s_k;\theta^i))^T Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)d\tau$$

$$+(2): \int_{\tau} \Sigma_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k}; \theta^1, \theta^2) D_{\theta^i}(\log \Pi_{l=0}^{k-1} \pi(a_l^i | s_l; \theta^i)))$$

$$D_{\theta^j}(\log \pi(a_k^i | s_k; \theta^i))^T Q(s_k, a_k^1, a_k^2; \theta^1, \theta^2)$$

$$+(3): \int_{\tau} \Sigma_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k}; \theta^1, \theta^2) D_{\theta^i}(\log \Pi_{l=0}^{k-1} \pi(a_l^j | s_l; \theta^j)))$$

$$D_{\theta^j}(\log \pi(a_k^i | s_k; \theta^i))^T Q(s_k, a_k^1, a_k^2; \theta^1, \theta^2) d\tau$$

Proofs are available in the original paper [6]. The gradient term is a generalization of policy gradient [9] to competitive multiplayer. In the bilinear term, term (1) corresponds to the immediate interaction between players. Term (2) corresponds to player i's behavior up to time step k with consideration of the reaction of player j's at time step k and the environment. Term (3) corresponds to player j's behavior up to time step k with consideration of the reaction of player i's at time step k and the environment.

This algorithm operates in epochs. It starts by collecting trajectories based on its existing policy $\pi(\theta^1)$ and $\pi(\theta^2)$. From there, it estimates Q, D_{θ^i}, and $D_{\theta^i \theta^j}$. It then updates its parameters θ^i and θ^j using the rules described above and then repeats. For full details, see the original paper.

7 GAME SETUP

The agent was given a preset team of three Pokémon, each with their own set of 4 moves. These moves included a mix of damage-dealing moves, strategic moves that put the active Pokémon in advantageous positions, and moves that dealt less damage but also had other strategic benefits. The full team is detailed in Appendix A.

The action space includes up to 6 distinct actions - using any of the Pokémon's 4 available moves or switching to any of the other 2 Pokémon available. If a Pokémon faints, or if a move runs out of uses, then that action is no longer available to the agent. The state space - the space the agent was made aware of - was the current turn, whether its active Pokémon had fainted or not, the number of healthy (non-fainted) Pokémon it had, the number of moves it had, and if its active Pokémon was fainted or not. The reward function was sparse; merely a 1 if the agent won the battle, a -1 if the agent lost, and 0 for all other moves.

The agent's performance was measured against two deterministic agents; one which selected moves randomly (the "random" agent) and one which selected its most damaging move (the "max-damage" agent).

The agent played itself a total of 50,000 times to converge to a previously unknown optimal strategy. This was done on a personal laptop over the course of approximately 36 hours. In each interval of training, the agent would train for 50 episodes, each with a batch size of 50. After each interval of training, the agent's performance was benchmarked against the two deterministic agents described before by battling each of them 100 times. There were 20 such intervals of training. The agents played with the same team that the agent trained against. The agent, however, did not update its policy during these test battles, meaning the agent did not train or influence its learning in any way based on the events or outcomes of the test battles against deterministic agents.

8 RESULTS

Fig. 1. Percentage of the time that the CoPG agent won against the max-damage agent over each of its 20 training iterations. After 20 iterations, it converged to a 91% win rate against the max-damage agent.

Fig. 2. Percentage of the time that the CoPG agent won against the max-damage agent over each of its 20 training iterations. After 20 iterations, it converged to a 91% win rate against the max-damage agent.

After 20 training rounds, the CoPG agent ended up with a 91% winrate against the random agent and an 89% winrate against the max-damage agent. Based on the graphs, one can observe that the agent's winrates against the random and max-damage agents follow a relatively volatile, but steadily increasing trend.

Upon manual inspection, despite the relatively limited amount of information available to the CoPG agent, it learned several sophisticated tactics. For example, it learned how to use setup moves like Swords Dance and Substitute that in early-game situations where it was not as likely to die. It also learned to use high-damage moves that had a good chance of dealing significant damage regardless of which of the opponent's Pokémon were out.

However, the agent was not perfect. It did not always immediately use "useful" moves. For example, it would sometimes use moves that did not effect the opponent, as a result throwing away the turn. It also did not always switch into advantageous matchups or take opportunities to use setup moves that it could have. A smarter opponent would have capitalized on these openings and defeated the agent with these opponents. It is worth noting that the agent did not have very much information to work with; it could not recognize the Pokémon in front of it, which in turn limited it from finding strategies to specifically counter the opponent.

The agent also left several strategies on the board. One underutilized move was Knock Off, which has significant value in terms of removing the opponent's items as well as doing decent damage. The agent never used it, which meant it either found the move to be relatively low-value or did not explore it sufficiently. It also did not seem to favor specific lineups; it led with all three Pokémon at relatively frequent rates and instead used different strategies based on the game situation.

We also find that the strategies that the agent could develop were limited by a simple state space. It could only exhibit knowledge of early-game and late-game strategies. It had to deduce which Pokémon was active from leading with a Pokémon of its choice and switching its Pokémon around and observing the states resulted from that move, which in turn means it was likely not able to create strategies predicated on its active Pokémon and certainly not its opponent.

In summary, the agent did learn various smart, optimal strategies. However, it only employed them some of the time. The rest of the time, it defaulted to using damaging moves, which worked out typically but not as efficiently as it could've been.

9 CONCLUSION

In conclusion, we found that using Competitive Policy Gradient on this specific instance of a Pokémon battle did lead to the agent learning several sophisticated strategies and achieving highly successful winrates against the two deterministic agents it was tested on. However, it did not use them all the time due to the limited information it had to work with. This bears significant promise for expanding the information available to the agent and the complexity of the game.

The implication, therefore, is that Competitive Policy Gradient is a viable algorithm for discovering strategies in zero-sum competitive games of a complex nature. It was able to make moves exploiting its opponent's actions and setting itself up for success later down the game. Its success against deterministic opponents means that it can learn an opponent's strategy and come up with its own to counter with, which means that it is a viable tool when studying how to defeat an opponent with a specific known "style".

10 FUTURE PLANS

Future work will be done to nuance the agent to understand more states, play more complex games, and learn how to build a team.

The immediate next step is to make the agent more aware of its state. At this time, the agent is only aware of extremely minimal information. Implementing awareness of the opponent's healthy Pokémon, what information has been revealed over the course of the game, and its own state will enable the agent to make more sophisticated decisions and employ more nuanced strategies over the course of the game. A key inclusion in the future will be the teams of the agent itself and its opponent. Inclusion of this agent will allow the agent to directly reason about lineups and come up with strategies accordingly. This would unlock many of the strategies enabled by team composition and move selection, such as using Knock off to get rid of items.

From there, another action of note is to train the agent to play against more kinds of opponents. Currently, it trains exclusively with one team to fight against one team. Letting the agent learn how to play with and against many different teams will make it more versatile and intelligent, especially if it can learn how to use strategies learned from one team with others. This will enable the agent to also measure its performance against real-life humans, which is a key milestone in using this algorithm to tackle complex problems in many domains.

Lastly, the agent will be set up to play online against real humans on the Pokémon Showdown online ladder. It will compete against other humans and its Elo rating will be monitored as a measure of its abilities. This Elo rating will be compared against other deterministic algorithms and known artificial-intelligence based algorithms to benchmark this algorithm's performance in this game.

Another long-term goal is to learn the team-building stage of Pokémon battling. Team-building will present additional challenges in terms of the state space and action space, but the process to generate an optimal strategy with Competitive Policy Optimization will remain the same.

11 ACKNOWLEDGEMENTS

The author would like to thank Manish Prajapat for his help in understanding the Competitive Policy Optimization algorithm and with his help in troubleshooting the code used to run the experiments for this project.

The author would also like to thank Discord user HSahovic, creator of the `poke-env` environment, for providing various instances of boilerplate code on interacting with the poke-env environment to help development of this project.

REFERENCES

[1] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR* abs/1912.06680 (2019). arXiv:1912.06680 http://arxiv.org/abs/1912.06680

[2] Dan Huang and Scott Lee. 2019. A Self-Play Policy Optimization Approach to Battling Pokémon. In *2019 IEEE Conference on Games (CoG)*. 1–4. https://doi.org/10.1109/CIG.2019.8848014

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[4] Pmariglia. [n. d.]. PMARIGLIA/Showdown: A pokemon showdown battle bot written in python. https://github.com/pmariglia/showdown

[5] pokeaimmd. [n. d.]. peak 3 in ou by doublevee. https://pokepast.es/78d5c6bf90f769de

[6] Manish Prajapat, Kamyar Azizzadenesheli, Alexander Liniger, Yisong Yue, and Anima Anandkumar. 2020. Competitive Policy Optimization. *CoRR* abs/2006.10611 (2020). arXiv:2006.10611 https://arxiv.org/abs/2006.10611

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *CoRR* abs/1712.01815 (2017). arXiv:1712.01815 http://arxiv.org/abs/1712.01815

[8] David Simões, Simão Reis, Nuno Lau, and Luís Paulo Reis. 2020. Competitive Deep Reinforcement Learning over a Pokémon Battling Simulator. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 40–45. https://doi.org/10.1109/ICARSC49921.2020.9096092

[9] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press. https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

12 APPENDIX A: THE POKÉMON TEAM

The team is listed in the standard Pokémon Showdown format. These 3 Pokémon were taken from a team which peaked at 3 overall on the Pokémon Showdown OU ladder [5].

These moves represent a variety of moves which allow for the agent to express a breadth of strategy. For example, Substitute's inclusion allowed the agent to potentially try to learn how to set up a Substitute sweep on a safe turn. Similarly, Roost and Dragon Dance on Dragonite allowed the agent to use Dragonite as a primarily offensive Pokémon and a defensive one depending on the game. The inclusion of items on each Pokémon gives Knock Off, a move which removes items from Pokémon, additional value.

The inclusion of Tapu Fini also introduces a new nuance, the Terrain. Misty Terrain halves the power of Dragon-type moves, which reduce the effectiveness of Dragonite's moves even if Tapu Fini itself is not active. It also prevents Ice Punch and Ice Bream from freezing the opponent, which is a relatively low-probability event which would have a significant impact on game outcomes in the event where it occurs. Tapu Fini forces the agent to come up with novel strategies to eliminate Tapu Fini from battle before using Dragonite or learn how to play with a Dragonite at significantly reduced effectiveness while Tapu Fini's effects last.

The items serve to nuance the Pokémon's usages while also serving to increase the value of Knock Off, one of Tapu Fini's moves. Leftovers on Mamoswine make it bulkier, while the Expert Belt on Tapu Fini makes its moves deal more damage. Using Knock Off on Tapu Fini or Mamoswine, therefore, makes these Pokémon less powerful and bulky, respectively.

Together, these moves, items, and team composition give the agent the opportunity to develop many deep and nuanced strategies, which in turn serve to help evaluate the effectiveness of the algorithm at understanding the game and how to develop complex tactics.

Tapu Fini @ Expert Belt
Ability: Misty Surge
EVs: 4 Def / 252 SpA / 252 Spe
Modest Nature
- Hydro Pump
- Ice Beam
- Moonblast
- Knock Off

Mamoswine @ Leftovers
Ability: Thick Fat
EVs: 252 Atk / 4 SpD / 252 Spe
Adamant Nature
- Earthquake
- Icicle Crash
- Ice Shard
- Substitute

Dragonite @ Heavy-Duty Boots
Ability: Multiscale
EVs: 248 HP / 52 Atk / 56 Def / 152 Spe
Adamant Nature

- Dragon Dance
- Ice Punch
- Earthquake
- Roost

13 APPENDIX B: THE COMPETITIVE POLICY GRADIENT ALGORITHM

Algorithm 1 Competitive Policy Gradient

$\theta^1 \rightarrow$ random values
$\theta^2 \rightarrow$ random values
for epoch 1, 2, 3... until termination **do**
  Collect samples under $\pi(.|.;\theta^1), \pi(.|.;\theta^2)$
  Estimate Q, then $D_{theta^i}\eta, D_{\theta^i\theta^j}\eta$
  Update θ^1, θ^2
end for