Bachelor Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Design, Implementation and Evaluation of Different Strategies for Playing Pokémon Battles

Julian Schubert
Institute for Computer Science
Chair for Computer Science VI

Prof. Dr. Frank Puppe
First Reviewer

Dr. Markus Krug
First Advisor

Submission
07. February 2022

www.uni-wuerzburg.de

Abstract

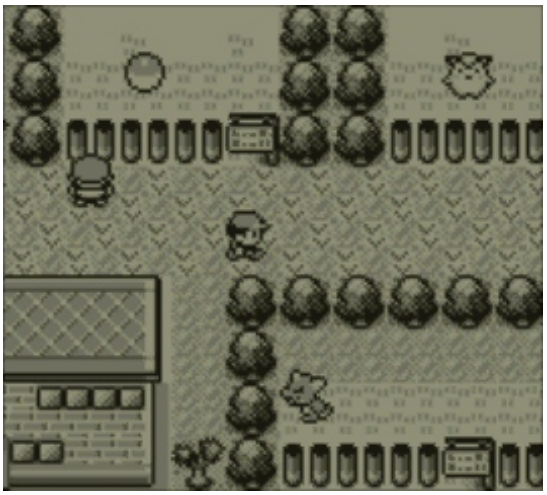
Pokémon is the highest-grossing video game franchise and revolves around collecting Pokémon with the ultimate goal of becoming "the very best, like no one ever was". This thesis provides a detailed introduction to competitive Pokémon battles and summarizes previous research on the topic. Current search-based agents lack proper long term planning as the large quantity of possible states limits the amount of turns that can be looked ahead. A combination of existing *MiniMax*- and *Rule Based*-Agents that allows for planning and execution of a match plan is introduced. After playing over 1,400 battles on the *Pokémon Showdown* ranked ladder against human players, the bot achieved a mean Elo of $1,227$ and a maximum of $1,432$. While the agent only won 273 out of 1,000 games against the best open source implementation, analysis of battles lead to the conclusion that defeats are often caused by lack of features that due to time restrictions were not fully implemented. Additionally, the initial state of over 70,000 games was rated and then played out by two baseline agents. Results showed that both a random agent and an agent that prioritizes the most damaging move have a lower win rate on unfavorable board ratings while winning more games on a positive rating.

Contents

1. Introduction

Pokémon (an abbreviation for **Pocket Monsters**) is a media franchise managed by *The Pokémon Company*, a company founded by *Nintendo*, *Game Freak* and *Creatures* (Wikipedia, 2022b). Figure 1.1 shows an image of *Pikachu*, one of the most popular Pokémon. Among multiple movies and series, a large variety of Pokémon games was released, starting with *Pokémon Red* and *Pokémon Blue*, which were published on 28. September 1998, also known as generation 1 (gen1)-games. In order to promote trading between players, *Nintendo* published two very similar games at the same time. Both releases differ in the available Pokémon and some minor changes to the story. Therefore, in order to collect all available Pokémon, players either had to buy two copies of the game or trade with a player that owns the counterpart. As of writing, there are eight major releases, also called *mainline games* with the latest being *Pokémon Sword* and *Pokémon Shield*. While the graphics evolved a lot over the years (see figures 1.2 and 1.3), the key concept of the game remained mostly unchanged. The player starts his journey in his hometown to become the *Pokémon Champion*, which is the highest known level of rank for a Pokémon trainer (Bulbapedia, 2022h). In order to achieve this goal, the player has to catch wild Pokémon and use them to create a team of up to six individual Pokémon which he then trains to unleash their full potential. This Thesis focuses exclusively on the battling aspect of the game as there are detailed lists of all locations and secrets there is to explore within the games. Pokémon battles are turn based where both players, unlike in for example chess, make their decisions at the same time. While the core battle mechanics are very simple, the game provides a lot of depth which led to the formation of a strong competitive battling scene. As catching and training Pokémon to a level where they are competitive viable is a time-intensive task, battle simulators such as the open source project *Poké-*

Figure 1.1.: Pikachu, one of the most popular Pokémon (Fandom, 2016)

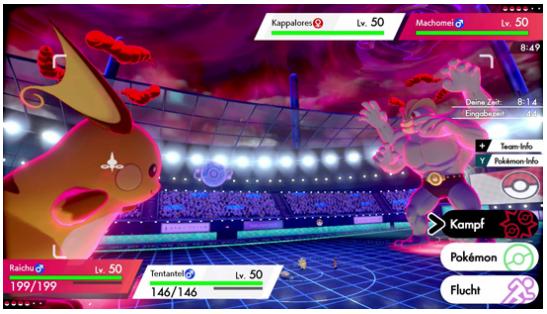


Exploring the map.

Fighting another trainer.

Figure 1.2.: Screenshots of Pokémon Red, the earliest Pokémon game. The game was released in 1996 for the Game Boy and has since been re-released multiple times. Image source: (Nintendo, 2022)

Exploring the map.
Image source: (Nintendo, 2019a)



Fighting another trainer.
Image source: (Nintendo, 2019b)

Figure 1.3.: Screenshots of Pokémon Sword, the latest major Pokémon game released for the Nintendo Switch in 2019. While the graphics have evolved, the core concept of the game, exploring and collecting Pokémon, remained the same.

mon Showdown (Smogon, 2022b) have arisen. On these fan made platforms, players have access to all available Pokémon and can compete against each other in a large variety of formats. Figure A.1 shows a battle between two players on *Pokémon Showdown*. The screenshot was taken from a generation 8 random battles (gen8randbats) between *Buckfae* and another player[1]. The screenshot was taken at the start of turn 19. Currently, the *Mewtwo* of *Buckfae* is at full health (indicated red) while the opposing *Kommo-o* has 33% of hit points (hp) left. Below the avatars of both players, the team is displayed, marked yellow for player two. While player one has three Pokémon remaining, his opponent has four members alive. The Pokéball (bottom right) indicates a to the player unknown Pokémon. Moreover, the blue box highlights the status modifications as well as boosts of the enemy Pokémon which will be covered in section 2.1.3 and 2.1.10 respectively. Below the game window, the possible choices of the player are displayed. *Mewtwo* has access to the moves *Fire Blast*, *Recover*, *Psystrike* and *Nastyplot*. In addition to *Dynamaxing* which will be coverd in section 2.2 the player also has the option to switch to either *Sirfetch'd*

[1]The other name was blurred

or *Vespiquen*. Lastly, on the right-hand side a log of the previous turns can be found. In addition to the moves a Pokémon can use, a Pokémon has one ability and can hold an item that yields advantages in battle.
Pokémon battles are simultaneous, meaning that both players move at the same time. Furthermore, each player has imperfect information about the possible team of his opponent. Currently, there are approximately 1.5×10^{39} (see section 2.4.3) possible starting positions for random battles which combined with the high dimensional feature vectors of states and the complex rules of the game result in Pokémon presenting an interesting challenge for AI to tackle.
This thesis gives an introduction to the main mechanics of the Pokémon genre while no prior knowledge of the game is required. Chapter 3 introduces and compares different approaches and provides an in depth summarization of the current state-of-the-art battling AI. Furthermore, an agent combining previously existing rule- and *MiniMax*-based implementations is developed to improve long term planning. Challenges of evaluating Pokémon AI and the results of this research are presented in 5.

2. Background

2.1. Basic rules

One of the key aspects of the Pokémon games is to battle other Pokémon. In the mainline games, you can have up to six Pokémon in your team, also known as party. There is the option to swap a Pokémon with another Pokémon, but you can not have more than six Pokémon at any point in your team. When playing the original games, you explore the world to find more Pokémon and use your team to defeat wild Pokémon and other Pokémon trainers. This thesis focuses on random battles taking place on *Pokémon Showdown*. In a random battle, both competitors get a team of six random Pokémon. At the start of the battle, you know each of your six Pokémon but only the currently active enemy Pokémon. Every turn, both players can choose to either use a move of their currently active Pokémon or switch to another Pokémon. Moves can either deal direct damage to the enemy Pokémon or yield other advantages like increasing the damage dealt by the next move. Moves will be covered in more detail in section 2.1.2. Each Pokémon has an amount of hp. The hp of a Pokémon can be lowered by attacking it with a move. If the hp of a Pokémon drops to zero, it faints and can not be used in this battle anymore. A player wins, once all enemies fainted.
Note that in the mainline games there is the possibility to heal or even revive a fainted Pokémon during battle using *Healing Items* like *Revive* or *Hyper Potion*. In competitive Play, only *Held Items* like *Leftovers* are allowed. Items will be explained in depth in section 2.1.6.

2.1.1. Types

Pokémon implements a rock-paprer-scissors (rps)-like system. Each Pokémon has either one or two of 18 types. Like *rock* beats *scissors* but looses to *paper*, *Fire*-type Pokémon have an *advantage* against *Grass*-type Pokémon but are on a *disadvantage* against *Water*-types. This is also called a type being *strong* / *weak* against another type. Figure 2.1 shows how different Pokémon types interact with each other. Unlike in rps, type modifiers will be multiplied if a Pokémon has two types. For instance, a *Fire*-type attack will deal 4 times the damage against *Parasect* as *Parasect* has the types *Grass* and *Bug* (Veekun, 2021).

2.1.2. Moves

Moves can be split up into three categories: *physical*-, *special*- and *atatus*-moves. While *physical*- and *special*-moves usually deal damage to the opponent Pokémon, *status*-moves

Figure 2.1.: Damage multiplier for all types. The damage dealt will be multiplied with two if a move is *super effective* (highlighted green), halved if the defender *resists* the attacking type (highlighted red) or set to zero if the move has no effect on the attacker (highlighted gray) (Pokémon-Database, 2022).

can for example change the weather, which plays a role in damage calculation explained in section 2.1.9, inflict status effects, raise or lower the stats of a Pokémon. In addition, a move also has exactly one of the 18 possible types. Often, *status*-moves are also referred to as *buffing* moves if they grant a benefit to the user or as *debuffing* if they have a negative effect on the opponent. While some moves will never miss, most moves have an *accuracy* of less than 100%. If a move misses, the turn is effectively skipped.

2.1.3. Pokémon

A key-concept of Pokémon battles are the *stats* of a Pokémon. The stats, among other factors which will be covered in section 2.1.9, determine how much damage a Pokémon is capable of dealing and receiving before fainting as well as how fast it can act in battle.

Explanation of stats

HP: The hp determines how much damage a Pokémon can receive before fainting.
Attack: The attack stat (atk) determines how much damage a Pokémon will deal when using a *physical*-move.
Defense: The defense stat (def) determines how well a Pokémon can resist against *physical* attacks.
Sp. Atk: The special attack stat (spa) determines how much damage a Pokémon will deal when using a *special*-move.
Sp. Def: The special defense stat (spd) determines how well a Pokémon can resist against special attacks.
Speed: The speed stat (spe) determines how fast a Pokémon can act. Usually, the Pokémon

with a higher spe will move before the slower one. The exact order of actions in battles is covered in section 2.1.8.

Determination of stats

The total stat of a Pokémon is calculated as described in equation 2.1 and equation 2.2 (Bulbapedia, 2022m).

$$HP = \left\lfloor \frac{(2 \times Base + IV + \lfloor \frac{EV}{4} \rfloor) \times Level}{100} \right\rfloor + Level + 10 \tag{2.1}$$

$$OtherStat = \left\lfloor \left(\frac{2 \times Base + IV + \lfloor \frac{EV}{4} \rfloor) \times Level}{100} + 5 \right) \times Nature \right\rfloor \tag{2.2}$$

Base: Refers to the base stat of a Pokémon. Two Pokémon of the same species will always have the same base-stats. As seen in figure 2.2, a *Charizard* will always have a base-atk of 84.

Level: As mentioned in section 2.1, the goal of the mainline games is to create a team of six Pokémon and to make that team stronger by fighting other Pokémon. If a Pokémon defeats enough other Pokémon, it grows a Level. The maximum level of a Pokémon is 100. In *Pokémon Showdown*, the level of a Pokémon is set at the start of the battle and won't increase (Mitarai, 2021).

Nature: A Pokémon has a nature. Most natures enhance the growth of one stat, while hindering the growth of another. After all other calculations are finished, the stat that the Nature enhances will be 110% of what it would be without the Nature, and the stat hindered will be 90% of its normal value (Bulbapedia, 2022m). In this thesis nature can be neglected as all Pokémon in random battles have a neutral nature, meaning no stat is enhanced or hindered (Mitarai, 2021).

IV: Refers to the individual values (iv) of a Pokémon. These cause two Pokémon of the same species to have different stats (Bulbapedia, 2022m). Pokémon in *Pokémon Showdown* will always have the best possible iv stat, 31, unless it is a disadvantage for the Pokémon, then it will be zero (Mitarai, 2021). For example, having a high spe is undesirable for a Pokémon that uses the move *Trick Room* as this move reverses the move order so that Pokémon with a lower spe stat attack first while those with a higher spe will attack last. Details regarding the order of events are explained in section 2.1.8.

EV: These are the effort values (ev) of the Pokémon. ev are what causes a trained Pokémon to have higher stats than an untrained counterpart of the same level. For every 4 ev gained, a level 100 Pokémon will have 1 extra point in the given stat. A Pokémon can earn up to 510 ev, but can not have more than 255 ev in a single stat (Bulbapedia, 2022m). Random Pokémon on *Showdown* will always have 85 ev in each stat, or 0 in the case that having a high stat being detrimental (Mitarai, 2021).

As an example, the stats for *Charizard* are denoted in figure 2.2.

2.1.4. Switching

Instead of using a move with the current Pokémon, the player also has the option to switch out the active Pokémon for another Pokémon is his party. Switching always takes place before the execution of moves. However, the player does not know whether the opponent is switching or using a move. Therefore, if the player decides to switch out a non-fainted Pokémon, the enemy gets to use his move on the new Pokémon. If a Pokémon faints, the player has to switch in a new Pokémon and then the next turns tarts. This means that the opponent gains a one turn advantage if the player decides to switch out a healthy Pokémon, but won't get to attack an additional time if the Pokémon was defeated.

Stat		Range	
		At Lv. 50	At Lv. 100
HP: 78		138 - 185	266 - 360
Attack: 84		80 - 149	155 - 293
Defense: 78		74 - 143	144 - 280
Sp. Atk: 109		102 - 177	200 - 348
Sp. Def: 85		81 - 150	157 - 295
Speed: 100		94 - 167	184 - 328
Total: 534	Other Pokémon with this total		

- Minimum stats are calculated with 0 EVs, IVs of 0, and (if applicable) a hindering nature.
- Maximum stats are calculated with 252 EVs, IVs of 31, and (if applicable) a helpful nature.
- This Pokémon's Special base stat in Generation I was **85**.

Figure 2.2.: Summary of the stats for *Charizard*. The left column shows the *base stats* of the Pokémon. To make comparison with other Pokémon easier, the second column shows these stats as bar chart. The last two columns show the possible ranges of final stats with ev and iv taken into account (Bulbapedia, 2022a).

2.1.5. Status condition

Moves may inflict so-called *status conditions* that affect a Pokémon negatively. The most important status conditions are

- **Burn:** If a Pokémon suffers from the status condition burn (brn), it will lose 1/8 of its total hp every turn. In addition to that, a burned Pokémon will only deal half as much damage when using a *physical* move.
- **Freeze:** If a Pokémon suffers from the status condition freeze (frz) it won't, with a few exceptions, be able to use moves.
- **Paralysis:** If a Pokémon suffers from the status condition paralysis (par) it won't be able to use the selected move 25% of the time and their spe is halved.
- **Poison:** If a Pokémon suffers from the status condition poison (psn) it will, with a few exceptions, take damage equal to 1/8 of its total hp at the end of every turn. A Pokémon can also be *badly poisoned*. Badly poison initially inflicts damage equal to 1/16 of the Pokémon's maximum hp, with the damage inflicted increasing by 1/16 each turn. This means that the Pokémon will take 2/16 damage on the second turn, 3/16 on the third turn (and so on).
- **Sleep:** If a Pokémon suffers from the status condition sleep (slp) it won't be able to use moves, except *Snore* and *Sleep Talk*.

At any point, a Pokémon can only suffer from one status condition at a time, this means that a burned Pokémon can not fall asleep. While a sleeping Pokémon wakes up after one to three turns and a frozen Pokémon has a 10% chance to unfreeze each turn, the other status conditions remain until they are cured. A status condition can either be cured by abilities like *Natural Cure*. This ability heals any status condition when the Pokémon is switched out. Items can also remove negative effects, for example, *Lum Berry* cures all status conditions but gets consumed in the process.

2.1.6. Items

A Pokémon can also hold an item that yields benefits in battle. There are various purposes that items can fulfill. For example, the item *Life Orb* boosts damage dealt by the holder's damaging move by 30%[1], but the holder takes damage equal to 10% of its maximum hp after it uses a damaging move[2] (Bulbapedia, 2021o). *Leftovers* restore 1/16 of the holder's maximum hp[3] at the end of each turn whereas the item *Air Balloon* makes the holder *ungrounded*, which means that the holder is immune to *Ground*-type moves as well as several related effects (Bulbapedia, 2021b). The generation of items in Pokémon Showdown is described in more detail in section 2.4.2.

Important items

In this section, a quick introduction to the most important items is given.

- **Choice Band:** When held by a Pokémon, this item boosts the atk by 50%, but only allows the use of the first move selected. This effect resets when the holder is switched out (Bulbapedia, 2021d).
- **Choice Scarf:** When held by a Pokémon, this item boosts the spe by 50%, but only allows the use of the first move selected. This effect resets when the holder is switched out (Bulbapedia, 2021e).
- **Choice Specs:** When held by a Pokémon, this item boosts the spa by 50%, but only allows the use of the first move selected. This effect resets when the holder is switched out (Bulbapedia, 2021f).
- **Heavy-Duty Boots:** The holder is unaffected by the effects of entry hazards. Entry hazards are described in section 2.3.
- **Assault Vest:** Raises the holders spd by 50% but also prevents the holder from selecting any status moves[4] (Bulbapedia, 2021c).
- **Focus Sash:** If the holder has full hp and is hit by an attack that would otherwise cause fainting, it survives with one hp (Bulbapedia, 2022f).

2.1.7. Field effects

There are multiple *field effects* that affect combat. Field effect are different from items as they are not tied to a specific Pokémon but rather always effect the currently battling Pokémon. There are field effects that only target the side of one player as well as effects that target both sides.

Terrain

Terrain is set up by the respective move with identical name and last for five turns and effects both players. All of them are beneficial to *grounded* Pokémon. A Pokémon is *not grounded* if any of the following conditions apply: The Pokémon

- has the *Flying*-type
- has the ability *Levitate*
- is holding the item *Air Balloon*

[1] This boost is approximated as $5324/4096 \approx 1.29980$
[2] Rounded down, but not less than 1
[3] Rounded down, but not less than 1
[4] Except *Me First*

- is under the effect of *Magnet Rise* or *Telekinesis*.

Grounded Pokémon are with a few exceptions those Pokémon, that are not *ungrounded*. A Pokémon will be grounded if any of the following conditions apply:

- The Pokémon is holding an *Iron Ball*
- The Pokémon is under the effect of *Ingrain*, *Smack Down* or *Thousand Arrows*.
- The field effect *Gravity* is in effect.

More information about grounding can be found at (Bulbapedia, 2021k) There are five different possible *terrain*-states.

- **None:** The default state, no other effects are applied.
- **Electric Terrain:** Grounded Pokémon can not fall asleep and the power of *Electric*-type moves is increased by 50%.
- **Grassy Terrain:** The HP of grounded Pokémon is restored by 1/16 of their maximum HP at the end of each turn. In addition, the power of *Grass*-type moves is increased by 50% and the moves *Earthquake*, *Magnitude* and *Bulldoze* halve in power.
- **Misty Terrain:** Protects all grounded Pokémon from status conditions The power of *Dragon*-type moves is halved while in effect.
- **Psychic Terrain:** Prevents grounded Pokémon from being hit by priority moves. Priority moves will be covered in section 2.1.8. The power of *Psychic*-type moves is also increased.

It is important to note that only one *terrain* can be active at a time, yet, *terrain* can coexist with other *field effects* like *weather*.

Weather

The *weather* is a set of mechanics that change the battle environment, activating abilities, modifying certain moves and potentially damaging the Pokémon in battle or affecting their stats. Only one type of weather may be present at a time, and only the most recent type of weather will take effect Bulbapedia, 2022q. List of different *weather*-conditions:

- *Clear skies*: (also: *None*) Absence of weather, this is the default state.
- *(Harsh-) Sunshine:* (also: *(Harsh-) Sunlight*) Strong sunlight shines on the battlefield.
- *(Heavy-) Rain:* Rain falls on the battlefield.
- *Sandstorm:* Stinging sand whips across the battlefield. At the end of each turn, each Pokémon, with a few exceptions, takes damage equal to 1/16 of its maximum hp unless it is a *Rock*-, *Steel*- or *Ground*-type (Bulbapedia, 2021t).
- *Hail:* Pelting hail falls on the battlefield. At the end of each turn, each Pokémon, with a few exceptions, takes damage equal to 1/16 of its maximum hp unless it is an *Ice*-type (Bulbapedia, 2021m).

There are a few additional special weather conditions which will not be covered in this thesis as they only occur in very specific scenarios.

Other

The list below contains the remaining field effects that cover the entire field not covered so far:

- *Wonder Room:* swaps the def and spd of all Pokémon, but stat changes remain on their respective stat (Bulbapedia, 2021y).
- *Magic Room:* suppresses the effect of all items held by the Pokémon on the field (Bulbapedia, 2021s).
- *Gravity:* causes the field to undergo intense gravity which multiplies the accuracy stat of all Pokémon by 5/3 (Bulbapedia, 2022g).
- *Trick Room:* reverses the move order within each priority bracket. Move priority will be covered in section 2.1.8.

Each of these effects remain on the field for five turns.

Single Side Effects

Among *Hazards* which will be covered in section 2.3, the most important single side effects are *Reflect* which halves the damage done to Pokémon on the given side by *physical* moves (Bulbapedia, 2022j) and *Light Screen* which works equivalent but for *special* moves (Bulbapedia, 2021p).

2.1.8. Order of events

In battle, switching always is executed before moves meaning that if a player switches and his opponent chooses to use a move, the move will always hit the newly brought in Pokémon. Move order is determined based on *priority*.

Priority

Priority is a characteristic of moves, such that any move with a higher priority will always be performed first. When two moves have the same priority, the users spe stat will determine which one is performed first in battle, the Pokémon with the higher spe moves first. Each move has a hidden priority value in the game data, with values ranging from $+5$ to -7 The vast majority of moves have the standard priority value of 0. A move with a positive priority is called *priority move*. Moves that have the same priority are said to be in the same priority bracket. An example for a move with priority $+2$ is *Extreme Speed*, a *Normal*-type move with a base-power of 80 (Bulbapedia, 2022d). Having a *priority*-move can be useful if the own active Pokémon has a lower spe-stat than the opponent with low hp as then the own Pokémon does not have to take an additional hit before defeating his enemy.

2.1.9. Damage calculation

The damage dealt by a move mainly depends on the *level* of the Pokémon that uses the move, its effective atk or spa stat, the opponent's effective def or spd stat and the move's effective power.

Precisely, the damage is calculated as follows (Bulbapedia, 2021i):

$$\begin{aligned}\text{Damage} = &\left(\frac{\left(\frac{2\times\text{Level}}{5}\right)\times\text{Power}\times\text{A / D}}{50}+2\right)\times Targets\times Weather \\ &\times Badge\times Critical\times random\times STAB\times Type\times Burn\times other\end{aligned} \tag{2.3}$$

The only exception for this are moves that deal direct damage. A list of these moves can be found at Bulbapedia, 2016. In the following paragraph, parameters of the formula are covered in detail. An explanation of the *Type*-modifier is omitted since it is covered in section 2.1.1.

Level

Level refers to the level of the attacking Pokémon. In *Pokémon Showdown*, the level is displayed next to the name of the Pokémon (Bulbapedia, 2021i).

A / D

A is the effective atk stat of the attacking Pokémon if the used move is a physical move, or the effective spa stat of the attacking Pokémon if the used move is a special move.
D is likewise the effective def stat of the target if the used move is a physical move, or the effective spd of the target if the used move is a special move (Bulbapedia, 2021i).

Power

Power is the effective power of the used move. The *Base Power* of a move in *Showdown* can be seen when hovering over a move in the move list.
Note: The same move will always have the same base power. For example, *Fire Punch* will always have a base power of 75 (Bulbapedia, 2022e).

Weather

The *Weather* modifier is 1.5 if a *Water*-type move is used during *Rain* or a *Fire*-type move during *Harsh Sunlight*. The modifier is 0.5 if a *Water*-type move is used during *Harsh Sunlight* or a *Fire*-type move during *Rain* (Bulbapedia, 2021i).

Critical

In the latest generation, a critical hit (crit) deals 1.5 times the damage compared to a normal hit. If the crit rate is not increased, the chance of landing a crit is 1/24 (Bulbapedia, 2022b). Increasing crit rate, as well as other stats, will be explained in chapter 2.1.10.

Random

Random is a random integer percentage between 85% and 100%. Because of this, the same move may deal different damage in the same scenario (Bulbapedia, 2021i).

STAB

STAB stands for *Same Type Attack Bonus*. It is a multiplier of 1.5 if the current move is of the same type as the attacking Pokémon. Otherwise, it is 1.0 (Bulbapedia, 2021i).

Burn

Burn is 0.5 if the attacking Pokémon is burned, and the attack is physical[5]. Otherwise, it is 1.0 (Bulbapedia, 2021i).

Other

The *other* modifier is usually 1. A list of exceptions can be found at (Bulbapedia, 2021i).

[5]This does not apply if the attacking Pokémon has the Ability *Guts* or the used move is *Facade*

2.1.10. Effective Stats

When a stat is used in a calculation in battle, a number of modifiers may be applied during the calculation. During a battle, a Pokémon's effective stats may be raised or lowered by certain moves abilities and held items. Some attacks may only have a chance of lowering stats, while certain abilities and held items may require a triggering event to activate stat modifications.
The modifiers conferred by most moves operate on a sliding scale of *stages*. When a given stat is raised or lowered, its current stage is increased or decreased by the amount dictated by the move; up to a maximum of +6 or a minimum of −6. A given stat corresponds to a given multiplier that will modify the stat when it is used in calculations. The exact multipliers for stages are described in table 2.1 When playing the mainline games, the

Table 2.1.: Stage multipliers that are applied for atk, def, spa, spd and spe. For example, a Pokémon with atk raised by two stages will have his existing atk stat multiplied by two (Bulbapedia, 2022m).

-6	-5	-4	-3	-2	1	0	1	2	3	4	5	6
2/8	2/7	2/6	2/5	2/4	2/3	2/2	3/2	4/2	5/2	6/2	7/2	8/2

player has to memorize the current stat changes of a Pokémon. In contrast to that, *Pokémon Showdown* displays the current stat multiplier of both Pokémon. Stat changes caused by boosts do not persist between switches, meaning that the stages for all stats of a Pokémon are reset to zero upon being switched in. Furthermore, a Pokémon also has an *accuracy* and *evasion* stat. Both can also be increased and lowered in stages but do not use the same modifiers. In gen8randbats the player very rarely has influence on these multipliers which therefore will not be covered in depth. Additional information can be found at (Bulbapedia, 2021a).

2.2. Dynamaxing

Dynamaxing is a temporary transformation affecting Pokémon that was introduced in generation 8 (gen8). Dynamaxing increases a Pokémon's size drastically, as well as changing the moves of the Pokémon and doubling their max and current hp, but can only be used once per battle and ends after three turns or if the Pokémon is switched out.

2.3. Hazards

An *entry hazard* is a condition that affects one side of the field. It causes any Pokémon that is sent into battle on that side of the field to be afflicted by a negative effect. Entry hazards are usually created by status moves (Bulbapedia, 2021q).

2.3.1. List of entry hazards

Currently, there are five moves that create an entry hazard:

Spikes

Spikes is a *Ground*-type entry hazard that causes the opponent to lose 1/8 of their maximum hp when they enter the field. This effect can be stacked up to three times. Two layers of spikes will deal 1/6 and three layers will deal 1/4 of the enemies maximum hp. Spikes are set by the move *Spikes* (Bulbapedia, 2022l).

Stealth Rock

The move *Stealth Rock* sets an entry hazard around the target Pokémon causing Pokémon on the target's field to receive damage upon being switched in. The amount of damage inflicted is affected by the effectiveness of the type *Rock* against the target. Unlike Spikes, this entry hazard does not stack. The damage taken from the victim's maximum is denoted in table 2.2 (Bulbapedia, 2022n).

Table 2.2.: Damage dealt to Pokémon by *Stealth Rocks* (Bulbapedia, 2022n).

Type effectiveness	**Damage (max. hp)**
0.25x	3.125%
0.5x	6.25%
1x	12.5%
2x	25%
4x	50%

Sticky Web

The entry hazard set by the *Bug*-type move *Sticky Web* lowers the opponents spe by one stage upon switching in (Bulbapedia, 2022o).

Poison spikes

Poison Spikes set by the *Poison*-type move *Toxic Spikes* cause the opponent to become poisoned. If two layers of spikes are set, the Pokémon instead becomes badly poisoned (Bulbapedia, 2021v).

Sharp steel

This entry hazard works very similar to Stealth Rock described in 2.3.1. However, Sharp steel can only be set by the *Steel*-type move *G-Max Steelsurge* which is the exclusive G-Max Move of *Gigantamax Copperajah*. The damage dealt by Sharp steel does not stack, the amount of damage dealt is based on the Type effectiveness of the *Steel*-type against the target. Exact damage modifiers can be found in table 2.3 (Bulbapedia, 2021j).

Table 2.3.: Damage dealt to Pokémon by *Sharp Steel* (Bulbapedia, 2021j).

Type effectiveness	**Damage (Max. hp)**
0.25x	3.125%
0.5x	6.25%
1x	12.5%
2x	25%
4x	50%

2.3.2. Hazard counterplay

There are some moves that can remove entry hazards. *Rapid Spin* (Bulbapedia, 2022i) removes entry hazards from the user's side of the field and *Defog* (Bulbapedia, 2022c) removes entry hazards on both sides of the field[6]. In addition, *Court Change* (Bulbapedia, 2021h) will exchange the entry hazards on each side of the field, along with other one-sided

[6] In older games *Defog* would only remove hazards on the target's side of the field. But as we only investigate the latest version, this won't be covered in detail.

field conditions. If a grounded *Poison*-type Pokémon enters the battle, it will remove *Toxic Spikes*, described in 2.3.1, from its side of the field. Pokémon holding the item *Heavy-Duty Boots* (Bulbapedia, 2021n) are unaffected by entry hazards, but grounded *Poison*-type Pokémon can still remove *Toxic Spikes* even if they hold the boots (Bulbapedia, 2021q). Additionally, there are various other exceptions to hazards which won't be covered in detail here.

2.4. Showdown random battles

Pokémon Showdown features a large variety of game modes, the most popular ones being *Sw / Sh Singles OU* and *Sw / Sh Singles Random Battle*. *Sw / Sh* refers to Pokémon *Sword* and *Shield*, the latest Pokémon Games. These two games form the eight generation of Pokémon games and are thereby also often referred as gen8-games. As *Pokémon Showdown* solely focuses on battling every player has access to every available Pokémon. This mainly plays a role in the *OU*-format. In this game mode, each player knows all enemy Pokémon but does not know their exact builds, that is, the abilities, iv-, ev-combinations and items are unknown. As the success of a player in this format heavily depends on his team as well as which other teams are currently popular, this thesis will focus on gen8randbats. In this format, each player gets a team of six random Pokémon and unlike in *OU* only knows the currently active opposing Pokémon. It is important to note that there are many possible builds for a Pokémon and therefore, information management plays a huge role. An example for this will be provided in section 4.2.
The development for battling bots is explicitly allowed and encouraged by the developers. Furthermore, there is no way to figure out whether the current opponent is a real human or an AI. This is beneficial as humans might behave differently when they know they are facing a machine potentially leading to invalid results.

2.4.1. Sets

As described in section 2.1.3, Pokémon created for random battles usually have 85 evs and 31 iv in every stat with a neutral nature, meaning a nature that does neither boost nor hinder any stat (Mitarai, 2021). There are some cases where a high stat is not beneficial, an example would be the move *Gyro Ball*. Unlike most moves, the *Base Power* of this move described in the damage calculation described in 2.1.9 is not a fixed value. It is determined as described in 2.4 (Bulbapedia, 2021l).

$$BasePower = \min\left(150, \frac{25 \times CurrentSpeed_{target}}{CurrentSpeed_{user}}\right) \tag{2.4}$$

Since the damage dealt by *Gyro Ball* gets bigger, the lower the spe of the attacker, Pokémon using this move have 0 ev and 0 iv in the spe stat.
Note: Being able to outspeed the opponent is extremely valuable, but the only two Pokémon using *Gyro Ball*, *Stakataka* and *Ferrothron*, already have a very low spe stat and are slower than almost all other Pokémon in random battles. A complete list of Pokémon with their respective spe stat can be found at (Bulbapedia, 2021r).
This knowledge can be exploited to gather additional information about the enemy. Section 4.2 describes how this is achieved.

2.4.2. Items

Items in random battles are procedurally generated by *Showdown* and depend on the Pokémon's moves, base stats and ability. As stated in Mitarai, 2021, the exact implementation is "changed frequently with the intention of optimizing set generation", yet, item assignment follows these rules:

- Pokémon with 2 or fewer attacking moves will get *Leftovers*, or *Black Sludge* if *Poison*-type.
- Pokémon with 3 attacking moves will get *Life Orb*, if the sum of their base hp, def and spd is less than 275. Otherwise, these Pokémon get *Leftovers* or *Black Sludge*.
- Pokémon with 4 matching attacks get a *Choice* item which follows these rules:
  - Pokémon with four physical attacks or four special attacks, a base spe between 60 and 108 and base atk or spa of 100 or more can get a *Choice Scarf* 2/3 of the time. If the Pokémon does not meet one of the stat qualifications or does not get the 2/3 chance, they'll get *Choice Specs* or *Choice Band* instead.
  - Pokémon with 3 special attacks and the move *U-turn* always get *Choice Specs*. *U-turn* is a physical, *Bug*-type move that switches the user out after damage is dealt (Bulbapedia, 2022p).
  - Pokémon with *Trick* (Bulbapedia, 2021w) or *Switcheroo* (Bulbapedia, 2021u), both moves that allow to switch items with the opponent, will always get a choice item. If they meet the above-mentioned speed range, they will always get a *Choice Scarf*. Otherwise, they will always get *Choice Specs* or *Choice Band*.
  - Having priority moves will always prevent a *Choice Scarf* from being generated in all situations.
- Pokémon with 4 attacks that do not qualify for choice items, will get an *Assault Vest* if their hp + def + spd ≥ 235. Otherwise, *Expert Belt*, *Leftovers* or *Life Orb* is generated.
- Pokémon that are weak to *Rock* will get *Heavy-Duty Boots* if they don't get a higher priority item, such as *Assault Vest* or a choice item. Pokémon that are four times weak to *Rock*, such as *Charizard*, will always get *Hevay-Duty Boots*. This is done as these Pokémon would otherwise lose up to 50% hp to the entry hazard *Stealth Rock* described in 2.3.1. The only exception is *Scyther*, which can get *Eviolite*.
- Pokémon in the lead slot will get *Focus Sash* if their hp + def + spd < 255, and they would otherwise get *Leftovers* or *Life Orb*.
- Pokémon that get a Speed-boosting move will be given a *Weakness Policy* if their hp + def + spd ≥ 300, and they aren't four times weak to *Ground*. This item boosts the atk and spa by two stages each if hit by a super effective move. After that, the item breaks (Bulbapedia, 2021x).

There are also some species that will always roll the same item, either because it's their signature item or because doing so supports a niche ability or set. For example, Pikachu always has *Light Ball*.

2.4.3. Amount of states

At the time of writing, 457 different Pokémon are available in gen8randbats. On average, each Pokémon has approximately nine different possible builds which leads to 1.2×10^{13} different possible teams for one player. As for one of these teams, each of the six Pokémon could be in the first slot, this number has to be multiplied by 6 to get the total amount of possible teams for one player. Equation 2.5 shows how the total amount of possible starting states.

$$\left(\left(\binom{457}{6} \cdot 9^6\right) \cdot 6\right)^2 \approx 1.5 \times 10^{39} \tag{2.5}$$

2.5. Pokémon Matchups

Due to the typing system, there is no best Pokémon that is the best option in all situations. Therefore, we have to determine how good a Pokémon is against another Pokémon in a given situation. In this case, the *situation* refers to the current state of both Pokémon like current hp and status conditions as well as field effects like weather.

2.5.1. Check and Counter

MattL, 2014 refers to a Pokémon *checking* another Pokémon if it can beat the enemy Pokémon in every scenario and can safely be switched in at any point. A *counter* is also capable of defeating the enemy Pokémon but may lose in some situations. The most notable being if switched in without the previous active Pokémon fainting as this grants the opponent an additional attack.
The key difference between check and counter is that a check is also stronger if it takes damage once more while a counter is not guaranteed to win in this situation.
Note: Every check to a Pokémon is also always a counter while counter could also be a check, but is not guaranteed to.

3. Related Work

In this section, previous research regarding the topic of battles on Pokémon Showdown is described and compared.

3.1. Baseline Agents

A good way to get a rough idea on how well an agent performs is to compare it against a baseline agent. There are two popular baseline agents, the *Random*-Player and the *MaxDamage*-Player. While the *Random*-Agent always chooses either a random move or a random switch, the *MaxDamage*-Agent always picks the move with the highest base power. If no move is available, the agent will switch to a random Pokémon. This is roughly equal to the skill level of an inexperienced beginner human.

3.2. Breadth-first search

Given a root battle object representing the current game state, breadth-first search (bfs) explores the outcomes of all possible choices, treating these resultant states as child nodes. This algorithm traverses the game tree until it finds a state in which the enemy Pokémon is fainted. As a non-adversarial algorithm, the agent assumes that the agent does not move at all. This agent won 75 out of 90 total games played against a random agent (Lee & Togelius, 2017).

3.3. MiniMax

MiniMax builds upon bfs as this algorithm deals with adversarial paradigms by assuming the opponents act in their best interest (Lee & Togelius, 2017). There are multiple possibilities on how to implement the *MiniMax*-Algorithm for Pokémon games. The main difference in implementations lies in state evaluation and the assumptions about the opponents. Additionally, the amount of features taken into consideration have a considerable impact. In the implementation proposed by Lee and Togelius, 2017, a node represents the worst case scenario that would occur as a result of the current choice. The agent also employes alpha-beta pruning, ignoring any node in which the agents Pokémon faints. One drawback of this procedure is that, a Pokémon would never use the move *Explosion*, a very powerful *Normal*-type move that also faints the user. This move can for example be used if the active member is already at very low hp. The tree itself is traversed using a

greedy strategy, which terminates when a state in which the enemy Pokémon is fainted is reached. Both the traversal order and the worst-case evaluation are performed using the evaluation function 3.1 (Lee & Togelius, 2017):

$$Eval = \frac{\text{current hp}_{\text{Own Pokémon}}}{\text{max hp}_{\text{Own Pokémon}}} - 3 \cdot \frac{\text{current hp}_{\text{Enemy Pokémon}}}{\text{max hp}_{\text{Enemy Pokémon}}} - 0.3 \cdot \text{depth} \tag{3.1}$$

A YouTube-Video released by RemptonGames, 2021 alters this evaluation function by increasing the rating of a game state when an enemy takes damage and decreasing the rating when a member of the own team takes damage. Both evaluation functions do not take hazards, status condition or boosts into account. Due to the similarity of both evaluation functions, both agents performed very similarly: The first agent described by Lee and Togelius, 2017 won 70 out of 90 total games against a random agent, resulting in a win-ratio of $\approx 86\%$. The second agent defeated the random player in 831 out of 1,000 total games, yielding a win-ratio of $\approx 83\%$. Due to the large amount of random number generation (rng) in battles, both agents are assumed to be at an equal level of play.
The current state of the art search-based algorithm was developed by *pmariglia* and is fully available at pmariglia, 2022. This approach uses the *ExpectiMiniMax*-Algorithm and takes hazards, boosts and status into consideration. In addition to "min" and "max" nodes of a traditional *MiniMax*-tree, this variant has "chance" nodes, which take the expected value of a random event occurring (Wikipedia, 2022a). Currently, the agent calculates two turns in advance, and reaches an Elo rating of $1,461$ as well as a Glicko-1 rating of $1,633$ in generation 7 (gen7) random battles on the *Showdown* ladder. In addition, *pmariglia* implemented an algorithm to estimate the item a Pokémon is holding based on the damage it dealt.

3.4. Rule based agents

The YouTube-Video released by RemptonGames, 2021 also introduces two rule-based agents. *Smart Damage* was written by the author of the video and uses Pokémon type and the spe-stat to determine the favorability of a matchup. On a bad matchup, this agent will switch to the team member with the best matchup. Otherwise, a simplified damage calculation is used to determine the move dealing the most amount of damage. *Simple Heuristics* was developed by, *Haris Sahovic*, the author of the *Poke-Env*. The implementation for this agent can be found at Sahovic, 2022b. This agent uses simple rules to decide the next move or switch and takes boosts as well as hazards into account. The results for both agents are denoted in table 3.1. The *MiniMax*-Agent in the table refers to the implementation of *Rempton Games*.

Table 3.1.: Denotes how many games the Agent in the column won against the agent in the row. The *Heuristics*-Agent developed by Sahovic, 2022a outperforms the agents created by RemptonGames, 2021.

	Random	*MaxDamage*	*SmartDamage*	*MiniMax*	*Heuristics*
Random	N / A	897 / 103	957 / 43	831 / 169	992 / 8
MaxDamage		N / A	829 / 171	834 / 166	955 / 45
SmartDamage			N / A	331 / 669	720 / 280
MiniMax				N / A	181 / 819

3.5. Other approaches

The authors of the *Showdown AI Competition* (Lee & Togelius, 2017) compared many simple AI implementations with each other. Approaches not covered so far will be summarized in this chapter.

One Turn Lookahead

One Turn Lookahead is a heuristic agent designed to encapsulate a greedy strategy that prioritizes the damage output. The agent operates by estimating the damage dealt by all usable moves, including those usable by the agent's inactive but usable Pokémon. If the highest damaging move belongs to the active Pokémon, the agent will use that attack. If the most damaging move belongs to an inactive Pokémon, the agent will switch to that Pokémon (Lee & Togelius, 2017). Depending on implementation details, this agent is very similar to *MaxDamage* or *Rule Based* agents.

Type Selector

This is a variation of the *One Turn Lookahead*-Agent that utilizes a short series of if-else statements in its decision-making. At first, if the current Pokémon knows a move that drains the opponents hp to zero, this move is selected. Otherwise, the favorability of the current matchup is evaluated. If the current type matchup is undesirable, the agent will switch to the Pokémon with an acceptable type matchup. If no such Pokémon exists, the agent will default to the most damaging move (Lee & Togelius, 2017).

Pruned Breadth-First Search

This agent is designed to demonstrate a simple way to utilize domain knowledge as a cost-cutting measure. This is achieved by making modifications to the Breadth First Search agent. First, the algorithm does not simulate any actions that involve using a damaging move with a resisted type, nor does it simulate any actions that involve switching to a Pokémon with a subpar type matchup. Additionally, rather than selfishly assuming the opponent skips their turn in each simulation, the agent assumes its opponent is a *One Turn Lookahead*-agent and simulates accordingly (Lee & Togelius, 2017).

Results

Table 3.2 displays the results of the agents described in this section.

Table 3.2.: The table denotes how many out of 90 games the agents developed by Lee and Togelius, 2017 won against a *Random*-Player.

	Random
One Turn Lookahead	77 / 90
Type Selector	67 / 90
Pruned BFS	75 / 90

3.6. Self-Play Policy Optimization Approach

Researchers from *New York University* (Lee & Togelius, 2017) were the first ones to apply *Q-Learning* to the field of Pokémon battles. Two agents using *Q-Learning* were developed: A single layer perceptron as well as a multi layer perceptron. Both agents were used to output the expected reward of all current moves and switches. Based on this, the best action was picked. Both agents were rewarded for defeating opponent Pokémon and punished for allowing one of its own Pokémon to faint. Because decisions made tend to have long term consequences, weights are updated using the last three (State, Action) pairs rather than the most recent pair only. Additionally, in order to promote exploration, the agent employs an epsilon-greedy selection policy, causing it to randomly override its decision with a probability of 0.1. The single layer perceptron was trained using the *Delta*

Rule, while the multilayer perceptron was trained using *Delta Rule* plus *Backpropagation*. The agent using a single layer perceptron won 90 out of 180 games against a random agent whereas the multi layer perceptron won 86 out of 180 games. Due to the large amount of rng in Pokémon games, more games would need to be played in order to confirm the superiority of the single layer perceptron in this particular use case. Randomness of games will be covered in more detail in section 5.1.1.

One year after the publication of Lee and Togelius, 2017, Chen and Lin, 2018 used proximal policy optimization (ppo) (Schulman et al., 2017) to train an agent. They also used embeddings to improve the representation of a Pokémon. Data available at alopez247, 2017 was used to create embeddings for each Pokémon. The dataset contains stats of the first 721 Pokémon, each row contains the name, type(s), numerical stat data (such as hp, atk, spe) and some other data such as color, height and whether the Pokémon is considered legendary in game. To create embeddings for each Pokémon, the data was turned into a graph to be used with Node2Vec (Grover & Leskovec, 2016) which creates embeddings from graph data in similar to Word2Vec (Mikolov et al., 2013). This algorithm samples random walks of some number of nodes of a given graph. Using these random walks, a skip-gram model is created which can be trained to generate embeddings. The Pokémon graph consisted of the name, type(s), numerical attributes (total stats, hp, atk, def, spa, spd, spe) as well as two special nodes, *Legendary* and *Mega*[1]. Lastly, Node2Vec was applied to the graph. Table 3.3 displays the discovered similarity using this approach. Here, a similar

Table 3.3.: Similar Pokémon within the embedding space of Chen and Lin, 2018. According to the embeddings developed, *Mewtwo* is similar to *Ho-Oh* despite both having entirely different strengths and weaknesses.

Pokémon	Most similar Pokémon
Bulbasaur	Chikorita, Turtwig, Nuzleaf, Petilil, Exeggucte, Skiploom, Jumpluff, Oddish, Budew
Caterpie	Wurmple, Weedle, Kakuna, Metapod, Paras, Ledyba, Spinarak, Venonat, Silcoon
Mewtwo	Lugia, Mesprit, Mew, Victini, Celebi, Cresselia, Volcation, Ho-Oh, Uxie

Pokémon to *Mewtwo*, a *Psychic*-Type, is *Ho-Oh*, a *Flying-Fire*-Type. However, these two have entirely different strengths and weaknesses. In addition to that, they do not share a single move in their move set: In generation 6 (gen6), *Ho-Oh* has access to the following moves: *Aura Sphere*, *Calm Mind*, *Fire Blast*, *Ice Beam*, *Psystrike* and *Recover* whereas *Lugia*'s move pool consists of *Brave Bird*, *Earthquake*, *Flame Charge*, *Roost*, *Sacred Fire*, *Substitute* and *Toxic* (Honko, 2022). This inappropriate classification is likely caused by both *Pokémon* having similar stats and both being legendary.

Features are derived from the battle state in the simulator. At a high level, battles consist of two sides. Each side consists of a team of Pokémon, and each Pokémon has some set of moves. Each of these objects (battle, side Pokémon, and move) have attributes that are used to derive a feature vector. After experimenting with multiple network architectures, the authors settled on a three-layer fully connected neural network, each with 512 neurons and a ReLU activation function.

Chen and Lin, 2018 trained the agent against a random agent as well as a *MiniMax*-Agent[2] until the reward curve stabilized which was around 100 epochs where an epoch is a single

[1] Mega is a mechanic similar to dynamaxing. Mega is not available in the latest version of the game and therefore won't be covered in detail

[2] The Authors don't provide implementation details of their *MiniMax* implementation

battle between two players. A reward of +1 is given to the agent if it wins the battle, −1 if it loses the battle and 0 for all other cases. The average epoch reward after training convergence for this approach can be found in table 3.4 The authors of Chen and Lin, 2018

Table 3.4.: Average epoch rewards after training convergence for opponent agents (Chen & Lin, 2018). The agent outperforms random and default agent but fails to defeat the *MiniMax*-Agent.

Opponent	Average epoch reward
Random	0.85
MiniMax	−0.9

describe their final agent as flawed as while the agent learned to switch Pokémon when the active Pokémon reaches low health, it almost always chooses to switch to the Pokémon in the last slot. In addition, this agent preferentially chooses the fourth move.

3.6.1. State of the art

In 2019, Huang and Lee, 2019 published a paper titled *A Self-Play Policy Optimization Approach to Battling Pokémon*. Due to its fine-grained analysis and the resulting agent performing on par with the state-of-the-art search based AI, this paper will be summarized in more depth.
Similar to OpenAI, 2018, the agent is represented using an actor-critic neural network. Actor-critic RL methods 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 an estimate of expected return when updating the policy prediction, the "actor" (Konda & Tsitsiklis, 2000). The authors represent both actor and critic using a two-headed neural network which is trained via self-play RL (Huang & Lee, 2019).

Neural Network

Input to the neural network is the current state of the game, from the point of view of the player, represented as 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 described in section 2.1.7.
3. Each *Pokémon* has many features. Table 3.5 contains a partial list[3].

The network has two outputs: a probability distribution $\pi \in \mathbb{R}^n$ over actions to take, and an estimate of player strength in the current state $v \in \mathbb{R}$. Huang and Lee, 2019 compute π as follows:

1. The network outputs an intermediate vector $p \in \mathbb{R}^n$. Each of the colored cells in figure A.2 correspond to an element of p
2. A probability distribution $\pi' \in \mathbb{R}^n$ is computed by using the softmax function: $\pi' = \frac{\exp(p_i)}{\sum_i \exp(p_i)}$
3. As not every action is valid in every state, for example, a switch to a Pokémon is invalid if that Pokémon is already fainted, the authors ensure their agent has zero probability of taking invalid actions. To do this, they take a mask $s \in \{0,1\}^n$ as part of the input, and renormalize probabilities to obtain $\pi : \pi_i = \frac{s_i \pi'_i}{s^T \pi'}$

[3]The authors state that this list is not complete, but no additional information is provided.

Table 3.5.: Features used to describe a single Pokémon battle (Huang & Lee, 2019).

Feature	Type	Dims	Description
species	categorical	$1 \times 1,023$	e.g. Pikachu
item	categorical	1×368	e.g. Leftovers, Choice Band
ability	categorical	4×238	e.g. Rough Skin, Shadow Tag
moveset	categorical	4×731	e.g. Flamethrower, Surf
lastmove	categorical	1×731	The last move used
stats	continuous	6	hp, atk, def, spa, spd, spe
boosts	continuous	6	Temporary boosts for stats
hp	continuous	1	Current number of hp
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. slp, brn, par
types	indicator	18	e.g. Bug, Fire
volatiles	indicator	23	e.g. Leech Seed, Perish Song

The authors point out the following two key design decisions: First, a 128-dimensional entity embedding layer for each of the categorical variables is used. This enables capturing similarities between different moves, species 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 resulting network is described by figure A.2 and contains 1,327,618 parameters in total (Huang & Lee, 2019).

Training the network

Training was done serially: After $m = 7,680$ games per iteration, the neural network parameters are updated using the $2m$ self-play matches as training data to obtain new neural network parameters. A reward of $+1$ for a win and -1 for a loss is assigned at the end of the match. To speed up learning, a dense reward signal using reward shaping was constructed. Auxiliary rewards are assigned based on events that occur over the course of the match. For example, a reward of -0.0125 is added when a Pokémon of the agent faints, and a reward of $+0.0025$ whenever the player's Pokémon makes a super effective move[4].
To update the neural network, the authors use ppo (Schulman et al., 2017), 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, *Generalized Advantage Estimation* (Schulman et al., 2018) is used.
After 500 iterations of the training loop, 3,840,000 self-play matches had been played by the neural network. Training was performed using Google Cloud Platform over the course of 6 days with an approximated cost of $91 USD.

Evaluation

The refined embeddings as well as the complex network architecture lead to this agent outperforming the RL-Agent described in Chen and Lin, 2018. Furthermore, the agent played 1,000 games against the open source agent of *pmariglia* mentioned in 3.3 Table

[4]While the authors do not provide an exact definition, we assume that a move is regarded as *super effective*, if the type modifier described in Damage calculation is bigger than one

Table 3.6.: Performance of the agent developed by Huang and Lee, 2019. The agent yields great performance and manages to defeat pmariglia, 2022, the best search-based AI.

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

3.6 describes the performance of the agent developed by Huang and Lee, 2019. While the authors do not provide the resulting Elo oft the agent, they state that a *Glicko-1* rating of 1,677 was reached. It is important to note that this agent played against an older version of *pmariglia* which was released in May 2019. Results of this agent described in section 3.3 and section 5 used the latest release as of February 2022.

3.7. Supervised Approach

As of writing, there is only one publication researching supervised learning for Pokémon battles, a YouTube video uploaded by TheThirdBuild, 2021. Unfortunately, the video does not cover much technical and implementation details, possibly to reach a broader audience on YouTube. The author obtained two million replays of human games from the creators of *Pokémon Showdown*. Using these replays, the input vector for a game contained the following features is created:

- **Pokémon attributes:** hp, whether the Pokémon is active, status condition and stat boosts.
- **Player's side attributes:** side conditions (like *Light Screen*), entry hazards (like *Stealth Rocks*), volatile conditions like *Leech Seed* and the last used move
- **Battlefield attributes:** weather, pseudo-weather[5] and terrain

Using these features, a neural network using TensorFlow for JavaScript was then trained to predict the win chance of a given player at a given state of the game. This network was able to predict the winner at a given state with an accuracy of 81%. In addition to that, the model was able to predict the next switch in with an accuracy of 86%, the next move with a certainty of 93% and whether the player would switch with an accuracy of 88%. Unfortunately, the author neither states the game type of the replays nor the exact architecture and evaluation method of the model.
This model was then used to pick the best move in a given scenario which allowed the agent to play games on the *Pokémon Showdown* ladder. Unlike other agents quoted in this thesis, this agent does not play gen8randbats on *Pokémon Showdown*, but rather *Gen 8 OU Singles*.
Furthermore, *Inverse Damage Calculation* is introduced. As the exact item of enemies are unknown, the author predicts the given item of a Pokémon by comparing the damage dealt of a move with the expected damage for the move modified by possible items. Despite no concrete implementation details given, we assume this approach to work similar to the implementation of pmariglia, 2022 described in section 3.3. According to the video, this agent reached a maximum Elo of $1,630$ which ranks the agent among the best 3.6% of players.

[5] The video does not give an exact definition of the word *Pseudo-Weather*

4. Approach

This thesis combines existing *MiniMax*- and *Rule Based*-Agents to allow for long term planning which is achieved by determining all matchups and storing the expected outcome. Next, these expected outcomes are used to build a traditional *MiniMax*-Tree to create and follow a long term match plan.

4.1. Communication with Pokémon Showdown

The communication with *Pokémon Showdown* is handled using Sahovic, 2022a. This library provides a lot of the core functionality needed, like accessing the current Pokémon in battle as well as switch and move options. However, it does not provide functionality for damage calculation. We use the *Pokémon Damage Calculator* (Smogon, 2022a), a node library written by the smogon-team for that. Communication between the two libraries is implemented by capturing stdout and stdin using the *subprocess* python library.

4.2. Gathering Information about the enemy Pokémon

As mentioned in section 2.4, the same Pokémon can occur in various different builds, meaning the combination of moves, abilities and items. Knowing the exact build is crucial for the decision-making process. Consider the following example:

- **Player1** has an active *Charizard* with *Heavy-Duty Boots* and 150 hp remaining on the field.
- **Player2** has just sent out a *Drapion* with 160 hp remaining.
- The *Charizard* is faster but can not kill the enemy *Drapion* in one turn as his move *Fire Blast* deals between 127 and 151 damage to the *Drapion*.
- Therefore, if **Player1** decides to attack, *Drapion* is guaranteed to survive this turn and can attack *Charizard* as well.

In this scenario, the optimal strategy for **Player1** depends heavily on the move set of the enemy *Drapion*. Possible moves for *Drapion* are:

- *Aqua Tail*: A damaging *Water*-type move
- *Earthquake*: A damaging *Ground*-type move

- *Knock Off*: A damaging *Dark*-type move
- *Poison Jab*: A damaging *Poison*-type move
- *Swords Dance*: A move to raise the own atk by two stages
- *Taunt*: A move that makes the afflicted Pokémon unable to use status moves
- *Toxic Spikes*: A move that sets an entry hazard.

Hereby it is important to note that *Drapion* only knows the move *Aqua Tail*, if it knows four total damaging moves[1]. In the given scenario, **Player1** should switch out his *Charizard* if the enemy *Drapion* knows the move *Aqua Tail* as this attack would kill *Charizard*. We can determine whether the enemy knows *Aqua Tail* based on his item:

- If *Drapion* rolls two status moves, it will have the item *Black Sludge* and therefore doesn't know *Aqua Tail*. Because *Drapion* is already damaged, we know that it has this item if it healed 1/16% of his max hp in his last turn.
- If *Drapion* rolls one status move, it will have the item *Life Orb*. If *Drapion* already attacked, we know if it has a *Life Orb* or not as this item causes it to lose 10% of his maximum hp after an attack.
- If *Drapion* has neither *Black Sludge* nor *Life Orb*, it has to have a *Choice Band* and as this item will only generate if the Pokémon knows four matching attacks, and therefore has to know the move *Aqua Tail*.

Therefore, the player should switch out his *Charizard* if the *Drapion* has neither *Black Sludge* nor *Life Orb* as then it is guaranteed to have a *Choice Band*, knows *Aqua Tail* and would therefore be able to defeat *Charizard*. In any other case, the **Player1** can safely kill *Drapion* using his active *Charizard*.

Implementation details

The first step to predicting enemy sets is to determine all possible sets and the likeliness of each individual set. In order to achieve this, a script that starts a battle between an information gathering player and a random agent was created. In the next step, the script extracts all builds of all Pokémon and stores them. Then, it forfeits, and a new battle is started. Once enough battles are fought, the script will store the builds as well as how often they appeared in text files, one file for each Pokémon.
In actual battles, if a new Pokémon enters the enemy side, we assume it to have the most likely build for its species. Once more information becomes available on items, moves and abilities, we rule-out non-matching builds and always assume the enemy to have the remaining most likely build.

4.3. Scoring the current game state

In order to not only rate the current board state, but also individual Pokémon, we implement the following scoring algorithm:

$$score(e_{i,j}) = \text{Expected Damage that Pokémon } i \text{ will deal to Pokémon } j \tag{4.1}$$

The expected damage is the damage dealt if both Pokémon behave optimal in the amount of turns that the bot looks into the future. Section 4.5 covers how the optimal moves are determined.

$$value(i) = \sum_{j \in \text{Enemy Pokémon}} score(e_{i,j}) \tag{4.2}$$

[1] This information was extracted from all possible *Drapion* builds. Details on how these builds were obtained are covered in section 4.2

Using this, we can also introduce a *value* for each of our Pokémon where a higher value implies a more important Pokémon. It is important to note that scores are determined independently of each other meaning that we do not take into account damage taken by the attacker. This does imply that this metric does not determine how good the Pokémon is if it has to battle *all* enemy Pokémon but rather against how many other Pokémon it *could* be used. This is done as the order in which a Pokémon battles multiple Pokémon plays a critical role. The reasoning behind this as well as the determination of an optimal order is explained in section 4.4. This metric also has multiple flaws because it only takes the damage dealt to the enemy into account, other important factors like damage received, healing, the availability of status moves and hazards is not considered. Using this rating, we can rank our team based on importance which is used in the switching routine described in section 4.4.

4.4. Stages of the game

We divide the game into two phases, the first one being the *Discover*-Phase whereas the second phase is called *Defeat*-Phase. Our goal and therefore our play style, is different in both phases.

Discover Phase

At the beginning of the game, we play safely until we know our opponent's entire team. In other words, we try to gather information about the enemy team while sacrificing as little hp as possible. In this stage, we act according to these rules:

1. Kill the opponent if we are guaranteed to kill him this turn. This either leads to us defeating the enemy Pokémon, or possibly new information if the enemy switches.
2. Healing our Pokémon. If we have a healing move that will heal us more than the expected damage we receive this turn, and we are not at full hp we will heal our Pokémon. Doing so will force the enemy to switch as we are otherwise gaining an advantage over him.
3. If we have a hazard setting move available, we will use this move as they will help us in the *Defeat*-Phase. Other beneficial side-conditions like *Light Screen* will be set as well.
4. Using moves that inflict status to the opponent like brn.

If none of these conditions apply, we decide whether to switch out our Pokémon or not. If our current Pokémon is a check or counter against the enemy Pokémon, we will not switch. Otherwise, we switch to a check, or counter if present. Next, we check if the current matchup is unfavorable. This applies, if the enemy is expected to survive the current matchups for two turns longer than we do, meaning that our Pokémon would not be able to defeat the enemy, even if it was allowed to attack two additional times. If this is the case, we determine the next action as follows:
We start by determining the score of each of our Pokémon as described in equation 4.1 and exclude our two most valuable Pokémon from the next steps. This is done as we assume to depend heavily on these Pokémon to defeat other enemies. Next up, we pick the Pokémon with the lowest score that fulfills the following criteria:

- The Pokémon has to survive for at least three turns against the enemy.
- At least one of the following criteria apply:
  - It is able to set a *hazard* or other beneficial field effect that is not already present.

– It can inflict a status condition on the enemy.

If no Pokémon matches these criteria, the Pokémon with the lowest score is picked instead. Score is calculated as described in section 4.3.
In any other case, the currently active Pokémon will use the best move calculated in section 4.5.

Defeat Phase

The *Defeat*-Phase starts as soon as we know all enemy Pokémon. At the beginning of this stage, we have to create a match plan for defeating the enemy. The main goal is to figure out which Pokémon to use against which enemies, especially the best order to send them into battle. Figure A.3 shows a possible end-game scenario. In this simple example, we assume both players to each have three Pokémon with full hp and no status conditions. Each circle represents two Pokémon battling each other. Therefore, the circle in the top center indicates, that our *Roserade* is currently fighting against the enemy *Charizard*. As *Charizard* is a *Fire*-type Pokémon which has a type advantage against his enemy, our *Roserade* will faint in this matchup, expressed by the underlined name. This means that *Player1* has to switch in a new Pokémon. A green circle around a node illustrates that the first player has to make a turn as his Pokémon fainted, and a red circle marks a decision for *Player2*. The first line below the names of the two opposing Pokémon displays the amount of hp the first team has left after this battle took place. As *Roserade* fainted, it has zero hp left whereas the other two Pokémon in his team are still at full health. Below that, the remaining hp of the second team is displayed. *Charizard* is *expected* to survive the battle with an *expected* amount of 141 hp remaining. Next, we have two possible options remaining, we can either send *Kyogre* or *Azumarill* to defeat the enemy *Charizard*. Taking a look at the leaves of the tree, the *value* of a leaf node is -1 if the enemy wins the battle and 1 if we win the battle. The value of a non-leaf node is the sum of the *values* of the children nodes. The *value* of two means that we expect to win the battle. As a result of this, the best choice in our example scenario is to use *Azumarill* to defeat the already damaged *Charizard*. The *Minimax*-Tree was generated using the most likely builds scraped from replays and does not use the default set of Smogon, 2022a. For example, the most likely build for *Charizard* contains the move *Roost* (Bulbapedia, 2022k) which heals the Pokémon.

4.5. Determining matchups

In order to determine whether to attack or to switch, the optimal moves for a Pokémon against another Pokémon need to be calculated. As stated before, the amount of possible combinations combined with the non-deterministic nature of the game makes it unfeasible get the optimal move combination by simulating every possible combination of actions and reactions over the span of multiple turns. Therefore, we determine the optimal moves for a Pokémon using this simplified method:
We start by generating all possible move combinations for a Pokémon with a given length[2]. Then, we simulate the outcome of the battle. If the attacking Pokémon would use these moves in the defined order given the enemy would not do anything[3]. Here, we also take boosting, items, status effects and the possibly changing field state into account. The combination resulting in the lowest amount of turns until the enemy faints is selected. It is important to note that this is not necessarily the move that the bot will use in the next turn as in the *Discover*-Phase healing, status and other beneficial effects are prioritized.

[2]The final implmenentation used a length of 3
[3]The option of not doing anything in a turn does not exist in Pokémon, if possible, the player is always forced to either select a move or switch

In order to improve performance, only matchups including the currently active Pokémon are re-evaluated each turn.

4.6. Predictions

Predicting the next enemy action plays a huge role in competitive Pokémon. In many situations, it is very likely that the opponent will switch to another Pokémon to avoid a bad matchup. For example, players often withdraw *Fire*-type Pokémon if the opponent knows a *Water*-type move as this will deal twice the amount of damage due to the type multiplier discussed in section 2.1.1. Therefore, players often choose to pick a move countering the expected Pokémon the opponent will send out or even switch themselves in anticipation of an enemy action. Our agent mimics this behavior in the following way:
The agent keeps track how often the opponent chose to withdraw his active Pokémon when it is checked and how often he decided to stay in. In order to make the actions taken by the agent less predictable, rng is used on favorable matchups to decide how to act. The chance that the agent predicts an enemy switch is zero if this is the first time that our Pokémon counters an enemy and otherwise as described in equation 4.3.

$$\text{Chance make a prediction} = \frac{\#\text{ The enemy switched when countered}}{\#\text{ Total times the enemy was countered}} \tag{4.3}$$

Currently, the agent assumes the enemy to switch in a random Pokémon that counters the active Pokémon and will then pick the move, that will yield to the best outcome for this matchup. Flaws and possible improvements to this simple routine will be discussed in section 5.2.2.

4.7. Dynamaxing

At the time of writing, *Memezboi69* is the top ranked player on the Pokémon Showdown ladder. He described the use of dynamaxing as very situation dependent as it can either be used offensive or defensive and proposed a simple rule set that can be build upon: Dynamax if the current matchup is favorable, the entire enemy team is known and the active Pokémon has more than 70% hp remaining. In addition, the agent will dynamax if the current Pokémon is the last remaining Pokémon with full hp. *Memezboi69* also laid out a more sophisticated rule set, which is described in section 5.2.2, that can be applied in future iterations of the agent.

5. Evaluation

5.1. Challenges for evaluation

Different researchers use different metrics to evaluate the performance of their agents. There are multiple factors that increase the difficulty of properly evaluating the performance.

5.1.1. Randomness of battles

As teams are generated randomly, one player often ends up with a slightly better team than his opponent. In extreme cases, one player may not even have a chance at winning the battle. While battling our agent during the evaluation process, one particular game stood out as the first Pokémon of **Player1** was capable of defeating the entire enemy team. The Pokémon of **Player1** was a *Volcarona* with the following moves:

- *Fire Blast*, a damaging *Fire*-Type move
- *Quiver Dance*, a *Bug*-Type move that boots the users spa, spd and spe by one stage each.
- *Bug Buzz*, a damaging *Bug*-Type move
- *Roost*, a move that restores half of the user's maximum hp

This Pokémon was able to defeat the entire enemy team with little to no possible counter play: The first enemy Pokémon, *Leafeon*, a *Grass*-Type Pokémon was killed in one hit using *Fire Blast* after damaging *Volcarona* using *X-Scissor*.
Next, *Glalie*, an *Ice*-Type was sent into battle. *Glalie* uses his best move, *Earth Quake* which brings *Volcarona* to 52% hp. As the enemy does not prose much thread to *Glalie*, **Player1** decided to boost using *Quiver Dance*. Now, *Volcarona* is faster than his enemy and kills it again in one hit using *Fire Blast*.
Then, *Mr. Mime (Galar)* is sent into battle. He fails to pressure *Volcarona* and therefore, **Player1** can heal his Pokémon using *Roost* and further boost using *Quiver Dance*. After defeating *Mr. Mime (Galar)*, *Volcarona* is back to 84% hp and boosts of 2.5 spa, 1.5 spd and 2.5 spe. Boosted this high, *Volcarona* can one-shot both the enemy *Volcarona*, *Pheromosa* and the dynamaxed *Scraggy* using *Fire Blast*.
To eliminate the impact of these extreme cases, evaluation of agents against other agents should be done using multiple hundreds to thousands of games against each other.

In order to quantify and better visualize the randomness of battles, over 70,000 gen8randbats were generated, and the team information was stored. Next, the board rating was calculated. Let e be a Pokémon and C the set of all enemy Pokémon that are countered by e. We compute the score of a team member e as follows:

$$\text{member-score (Pokémon: } e) = \sum_{c \in C} 1 \tag{5.1}$$

The score of a team p is defined by:

$$\text{team-score (Team: } p) = \sum_{e \in p} \text{member-score}(e) \tag{5.2}$$

Finally, the board rating of two teams p_1 p_2 is defined as the difference between their respective team score

$$\text{board rating (Team: } p_1\text{, Team: } p_2) = \text{team-score}(p_1) - \text{team-score}(p_2) \tag{5.3}$$

Therefore, a low board rating indicates that the player has fewer counters than his opponents. Finally, the generated games were played out by *Random*-Player as well as *MaxDamage*-Player. The win-rate of both agents at a given board rating is displayed in figure 5.1. Most notably, fair games, meaning games with a board rating close to zero,

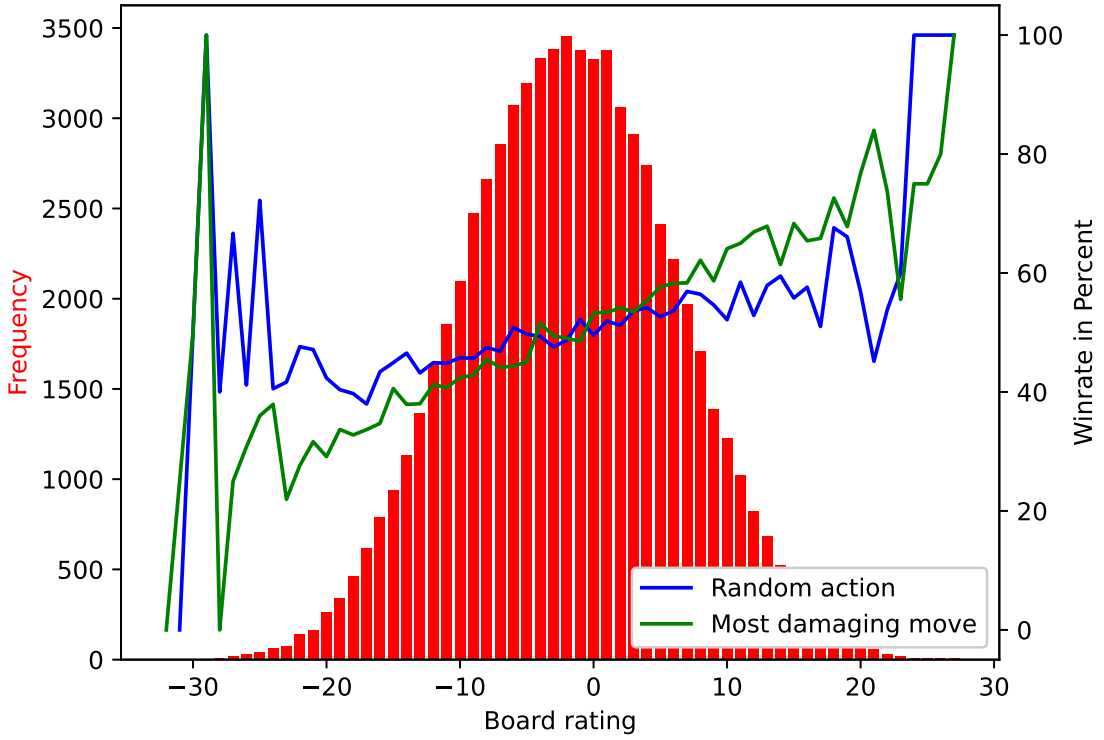


Figure 5.1.: Win rate of the baseline agents at a given board rating. Both agents move have a lower win rate on unfavorable board ratings while winning more games on a positive rating.

are gaussian-distributed with the majority of the matchups being almost fair according to our metric. The blue line indicates the win rate of a *Random*-Player when playing against another *Random*-Player at the given board rating, indicated green is the win rate

of two *MaxDamage*-Players. There are multiple important observations: Firstly, at negative board ratings, both agents have a significantly lower win rate against the opponent while positive board ratings indicate a high chance of winning. Despite collecting over 70,000 data points, board ratings worse than −20 or better than +20 will be ignored in this analysis as the large irregularities in these areas indicate an insufficient amount of data points. The steeper slope of the green line is likely explained by the rng of the actions performed by random agents having a much higher influence than the actual team composition. As a random agent rarely makes the correct choice, it fails to utilize the full potential of a stronger team.

5.2. Agents

During this thesis, two different agents were developed, *HerrDonner* and *HerrGewitter*.

5.2.1. HerrDonner

This agent was designed to establish a good baseline and to demonstrate the capabilities of a very simple rule set. The agent is capable of looking multiple turns into the future. To determine the moves to be used, the agent estimates the damage dealt for every possible move, leading to *HerrDonner* being an improved *MaxDamage*-Player as it behaves similarly, but has a more refined damage calculation algorithm. No drawback moves that heal the agent, set hazards or field conditions or inflict status conditions are not considered unless they result in the highest amount of damage dealt. Also, stat changes are not taken into account, neither for damage calculation nor for rating of matchups. This results in the bot often spamming moves like *Draco Meteor*, a special *Dragon*-type move that deal a lot of damage but also lower the users spa by two stages resulting in the move dealing less damage every time it is carried out. When the agent is forced to switch, it will switch to a check if available. If no check is available, a counter is sent into battle if one exists. Otherwise, a random Pokémon will be picked.
At the start of each turn, *HerrDonner* will evaluate the current matchup, a matchup is deemed unfavorable, if the current Pokémon is neither check nor counter to the current enemy. On a bad matchup, the bot will switch to an available check or counter. If neither is available, the bot won't switch and try to defeat the current opponent with his active team member.
Dynamaxing is implemented in a very simple and naive way: The agent will always dynamax the active Pokémon as soon as more than four enemy Pokémon are known. Lastly, if the current Pokémon is dynamaxed, the agent will not switch, even if the current matchup is not favorable. The challenges of properly using this mechanic will be explained in section 5.2.2.

5.2.2. HerrGewitter

HerrGewitter behaves like described in section 4. Here, the most notable differences between both agents are highlighted, and limitations of this agent are discussed.
Firstly, *HerrGewitter* takes more factors into consideration. Now, damage calculation is done in regard to current stat changes and status conditions. Additionally, abilities and items are taken into account. Furthermore, recoil from moves, healing both from items like *Leftovers* and moves like *Recover* are not neglected anymore.
Switching and the selection of moves is done as described in section 4.
These improvements lead to *HerrGewitter* avoiding mistakes of *HerrDonner*. For example, this agent will burn a physical attacker using *Will-O-Wisp* (Bulbapedia, 2022r) in order to reduce damage taken over the next turns. The agent will also boost and heal itself in favorable situations which stalls the game and forces the opponent to react. Another

major improvement is that the agent switches out the current Pokémon if stat changes resulted in an unfavorable matchup which is especially important as stat changes reset on a swap. However, there are still a lot of features that *HerrGewitter* is lacking.

Weather and Field effects

One such feature is a proper support for weather and field effects in the damage calculation as well as in the *MiniMax*-Algorithm. For instance, a consequence of this is the agent lacking awareness of the fact that a *Fire*-Type move deals 1.5 more damage during *Harsh Sunlight*.

Hazards

Currently, the agent will always try to set a non-present Hazard in the early game as this usually results in a long term benefit. There are however some notable exceptions to this that are not yet implemented:

- The agent will always set as many hazards as possible in the early game, even if the current matchup is unfavorable, including always setting up to two layers of spikes. A small test on human players indicated that this leads to slightly better results than only setting hazards on good matchups, but due to the very small sample size, future work is needed to determine the best strategy for setting hazards.
- The agent does not take the damage taken by hazards into account when switching Pokémon.
- The agent will always use *Toxic Spikes* even if the opponent has a *Poison*-Type Pokémon on his team that will remove this hazard upon being switched in.
- The agent will use Hazards even if the current enemy is known to have a hazard-clearing move like *Defog* (Bulbapedia, 2022c).
- The agent will not clear hazards.

Choice Items

As mentioned in section 2.1.6, Pokémon holding a *Choice*-item are locked into using always the same move until they are switched out. The agent has two major flaws in regard to these items: When the active Pokémon of the agent is holding a choice item and is already locked into a move, the agent is not aware of the fact that once the Pokémon is switched out, it will regain access to his other moves which leads to an incorrect prediction for future matchups. As described in section 4.5, the only matchups re-evaluated on a given turn are matchups that include one of the currently active Pokémon. The following example illustrates how this design decision leads to issues on Pokémon with *Choice*-items:
In the given scenario, our agent has an active *Garchomp* which is locked into using *Earth Quake*. The *Garchomp* also has access to the *Rock*-Type move *Stone Edge*. This turn *Butterfree*, a *Bug / Flying*-Type Pokémon is sent into Battle. As the *Ground*-Type move *Earth Quake* has no effect on *Butterfree*, the agent will switch out *Garchomp* for another Pokémon. In the current implementation, matchups for *Garchomp* are not re-evaluated. While this won't lead to problems in the early game, this results in an incorrect *MiniMax* calculation as for matchups involving *Garchomp* and any non-active opponent, *Garchomp* is still assumed to only have access to the move *Earth Quake*. In this scenario, the agent would fail to realize that *Garchomp* also has access to *Stone Edge* and would incorrectly assume *Garchomp* to lose all matchups against *Flying*-Type Pokémon.
While this behavior rarely effects battle, the agent failing to notice that an enemy is choice-locked has more often a negative impact on the battle: If the enemy is known to

be choice-locked into *Earth Quake* we can safely switch a *Flying*-Type into battle. This applies especcially if the enemy Pokémon is known to have the *Rock*-type super effective move *Stone Edge* as the enemy can not use this move until switched out and back in again. In this scenario, the agent wrongfully would not prefer to switch in a *Flying*-Type Pokémon due to the thread posed by *Stone Edge*. Switching in a Pokémon resisting *Earth Quake* in this scenario forces the enemy to switch to another Pokémon. This gained turn advantage can either be used to land an extra move on the next opponent, set hazards, beneficial field conditions, inflict status conditions or boost the current Pokémon.

Damage Calculator

The current implementation relies on the Pokémon Showdown Damage Calculator. As of now, this open source project does only support direct damage dealt by attacking and lacks functionalities like recoil, healing from items and moves. While most of these features were added to *HerrGewitter*, some moves are still not properly implemented. For example, the move *Counter* has a move priority of -5 and works as follows: If the last mount of damage dealt before the use of *Counter* is greater than zero and was dealt by a physical move, *Counter* will do twice as much damage to the opponent. Otherwise, the move will miss. Additionally, *Counter* has a lot of extra rules regarding other special moves in place (Bulbapedia, 2021g). Issues like these are especially obvious on *Wobbuffet* as all of his four most likely moves, *Mirror Coat*, *Encore*, *Counter* and *Encore* are very useful yet do not deal any damage and are not implemented yet which leads the agent to believe that this Pokémon is bad in every possible matchup and has no good use scenarios whatsoever.

Dynamaxing

As previously stated, dynamaxing is a very complex mechanic that requires sophisticated long term planning for proper usage. According to *Memezboi69*, the safest usage of this mechanic is to dynamax in a late stage of the game on a good matchup. He stated multiple reasons for this: Dynamaxing at a later stage of the game reduces the likeliness of an opponent being able to properly react due to the limited amount of Pokémon alive. Another flaw of *HerrGewitter* is the fact that defensive dynamaxing is not yet taken into consideration. If the opponent dynamaxes offensively, a good way to minimize damage received when no counter is available, is dynamaxing defensively and using the move *Max Guard* which protects the user from moves. If a Pokémon has access to a status move, this move will be replaced by *Max Guard* when dynamaxing. Lastly, dynamaxing can under some circumstances also be used in an earlier stage of the game if the current matchup is favorable.

Predictions

When play testing the bot, predictions were noticeable but only seemed to occur rarely. This is likely caused by the optimal move against the current Pokémon being equal to the best attack targeting the predicted switch. In order to compete with higher ranked players, more investigation on the prediction routine is required. Future work could also include switching in anticipation of the opponent withdrawing his active Pokémon.

Other special cases

This list contains more currently unhandled cases which will be addressed in future versions:

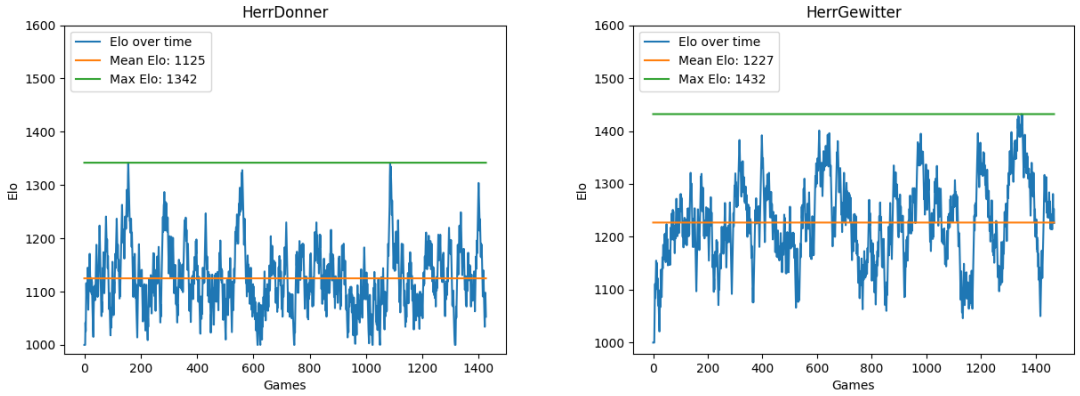
- The Pokémon *Ditto* can transform itself into the Pokémon of the current opponent.
- The Pokémon *Zoroark* can transform itself into another team member.
- The ability *Trace* changes the ability of a Pokémon to the ability of his opponent.

MiniMax

The *MiniMax*-Algorithm described in paragraph 4.4 only supports changes in health but ignores other important factors such as boosts and status conditions. Therefore, the agent will not recognize the possibility to weaken a very strong physical attacker like *Garchomp* by burning it first and then defeating it with another Pokémon. A simple way to include brn into this algorithm is to multiply the expected damage dealt by a burned Pokémon with 0.5.

5.3. Results

5.3.1. Ranked results



HerrDonner *HerrGewitter*

Figure 5.2.: Elo of *HerrDonner* and *HerrGewitter* after playing 1,400 ranked games respectively.

Both agents were evaluated at the same time over the span of multiple days by playing over 1,400 ranked games each against human opponents on *Pokémon Showdown*. Figure 5.2 shows the Elo rating of *HerrDonner* and *HerrGewitter* over time. In order to make a comparison of both agents more easy, figure 5.3 shows the smoothed Elo of both agents over time. Smoothing was achieved by dividing the Elo history in chunks of size 20 and then plotting the average Elo of each chunk.

```
step = 20
smoothed_elo = []
for i in range(0, len(elo_history), step):
    sec = elo_history[i: i + step]
    smoothed_elo += [(i, sum(sec) / len(sec))]
```

Listing 5.1: Smoothing Elo values

The first thing to note is that *HerrGewitter* has a higher mean Elo (1,227) and a higher max Elo (1,432) than *HerrDonner* who achieved an average Elo of 1,125 and peaked at an Elo of 1,342. Table 5.1 shows the performance of different agents when directly competing against each other. It is very important to note that the agent of Huang and Lee, 2019 played against an older version of pmariglia, 2022 than our bot did and is therefore not included in this table as this possibly leads to misleading interpretations. Each entry displays the results of a thousand games played between both agents. The first number indicates the amount of games the agent in the current column won. There are multiple things to point out here:
While *HerrDonner* and *HerrGewitter* achieved almost similar results when battling a

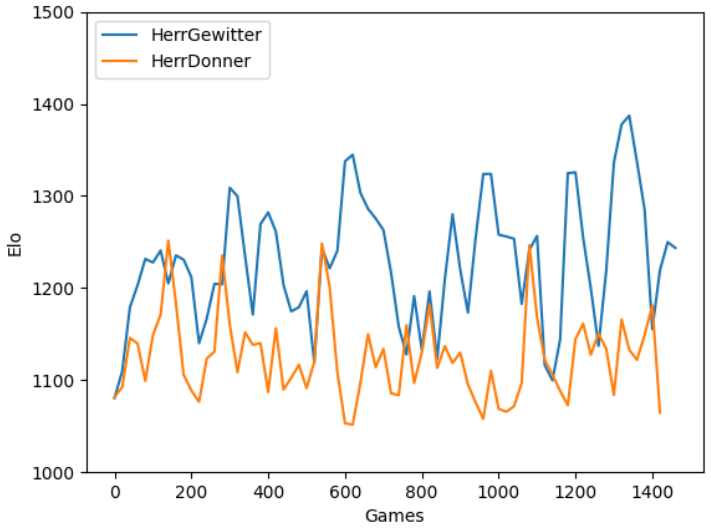


Figure 5.3.: Smoothed Elo of *HerrDonner* and *HerrGewitter* over the course of 1,400 ranked games.

Table 5.1.: Battle results of 1,000 games between the agents. The AI of pmariglia, 2022 outperforms *HerrGewitter*. *HerrDonner* manages to achieve good results against the baseline agents.

	HerrDonner	*HerrGewitter*
Random	992 / 8	993 / 7
MaxDamage	906 / 94	951 / 49
Pmariglia	-	273 / 727
HerrDonner	-	580 / 420

random agent, *HerrDonner* performed notably worse against the *MaxDamage*-Agent which always picks the move with the highest base power. Also, *HerrGewitter* only managed to win 58% against *HerrDonner* despite notably better results against human players, described in section 5.4, possibly because *HerrGewitter* fails to properly exploit weaknesses of *HerrDonner* and has less weaknesses that humans can use to gain an advantage. At the time of writing, neither *poke-env* nor replays created by *Showdown* allow the retrieval of the Glicko-1 rating. After playing for 1,400 games, *HerrDonner* ended up with a Glicko-1 rating of $1,438$ and *HerrGewitter* achieved a ranking of $1,509$.

5.4. Investigating Replays

In addition to playing against humans on the *Showdown* ranked ladder, the bot was also evaluated by playing against *Jonathan Krebs* and *Markus Krug*. While *Markus* was able to

Table 5.2.: Performance of both agents against human players. *HerrGewitter* posed a harder challenge but is unable to gain a lead against *Markus*, an experienced human player.

	HerrDonner	*HerrGewitter*
Jonathan	28 / 23	37 / 23
Markus	N / A	4 / 11

reliably beat *HerrGewitter*, the agent provided a significant challenge and required careful planning to overcome. Replays retrieved from these games confirm assumptions made about the bot in the previous sections. *HerrGewitter* is very good at picking the most damaging move, but can be exploited using mechanics not yet implemented.

5.4.1. Pro Replays

Memzboi69, the top ranked player, agreed to play three games against *HerrGewitter*. While our agent lost all three games, we were able to gain some very interesting insights from these games:

Game One

In the first game, the bot made a huge mistake that ultimately lead to defeat. The agents *Shuckle* faced a *Mr. Mime (Galar)* which had access to *Rapid Spin*, a move that removes hazards. As *Shuckle* has access to the moves *Sticky Web* and *Stealth Rock*, the agent tried to set both hazards which were immediately removed. In the extra turn, *Mr. Mime (Galar)* boosted its speed stat and slowly killed his opponent using *Freeze-Dry*. This increased spe led to an advantage that the bot was not able to make up for. A future version of the agent will need to include a better hazard routine to prevent this kind of scenarios.

Game Two

Game two was lost due to a lack of counters against the opponents *Gengar*. Prior to sending in this Pokémon, *Memzboi69* set the entry hazard *Sticky Web* which decreases the spe by one stage on switch in. After boosting spa by two stages using *Nasty Plot*, *Gengar* was able to defeat the four remaining Pokémon of the agent. Further investigation revealed another flaw in the logic of the agent. When switching in the *Discover Phase* with no check or counter available, the agent ranks all possible options based on how much damage they are capable of dealing as described in section 4.3. If no Pokémon exists that is capable of surviving for more than two turns, the Pokémon with the lowest score will be sent into battle. In the given scenario, the agent sent out *Klefki* as his last Pokémon. This Pokémon always has access to either *Thunderwave*, a move that applies par or *Toxic* that poisons the opponent. While the chances of winning would still be very low when sending out this Pokémon, bringing in *Klefki* would have increased the odds. Additionally, the agent did not take the entry hazard into account which lead to the wrong assumption that *Zamazenta* is faster than *Gengar* and could therefore kill the opponent. This resulted in *Zamazenta* getting killed in one shot.

Game Three

The last battle was the closest and ended with *Memezboi69* having only two Pokémon remaining. The agent even managed to set up a burned *Obstagoon*, boosting atk to stage two and def to stage three. Unfortunately, this Pokémon was then killed by *Buzzwole* with the super effective move *Close Combat*. Even after a lot of investigation, it is hard to determine how this match could have been won by the agent, one possibility might have been switching in a *check* to *Buzzwole* and dynamaxing next turn.

6. Conclusion

While simple rules like picking the most damaging move are sufficient to defeat *Random-Player* most of the time, more complex routines are required in order to rival human players. Existing search-based agents lack proper long term planning as the large amount of possible states limits the amount of turns that can be calculated in advance. This thesis introduces a combination of existing *Rule Based*- and *MiniMax*-Agents in order to create and execute a match plan. Proposed agents were able to defeat baseline opponents reliably and are challenging to defeat, even for more experienced humans.
Huang and Lee, 2019 developed a DQN-Agent that makes use of embeddings which is able to defeat the search-based agent of pmariglia, 2022. Evaluation of agents for *Pokémon Showdown* needs to be done with regard to the large amount of rng that results in one player often having an objectively stronger team than his opponent. *HerrGewitter* failing to rival *pmariglia* is likely caused by its inferior item prediction algorithm and handling of edge cases.

Further possible improvements to *HerrGewitter* include item prediction and the refinement of the *MiniMax*-Algorithm by including status and boosts. Lastly, a refined dynamaxing-routine and improved predictions of enemy actions are worth investigating.

Pointed out by Huang and Lee, 2019, the DQN-Agent might be improved by introducing "different network architectures or the addition of a recurrent element like an LSTM to better model human memory during the course of a game". Experiments combining supervised learning and reinforcement learning similar to the *StarCraft II* AI AlphaStar, 2019 developed by *DeepMind* may improve the agent even further.

The advantage of a rule based approach is the capability to quickly adapt rules to other game modes, as no re-training is required. Therefore, our approach can be quickly applied in newer games which are released by Nintendo on a regular basis.

List of Figures

List of Tables

Listings

Acronyms

hp hit points

atk attack stat

def defense stat

spa special attack stat

spd special defense stat

spe speed stat

crit critical hit

iv individual values

ev effort values

brn burn

frz freeze

par paralysis

psn poison

slp sleep

rng random number generation

bfs breadth-first search

ppo proximal policy optimization

rps rock-paprer-scissors

gen1 generation 1

gen6 generation 6

gen7 generation 7

gen8 generation 8

gen8randbats generation 8 random battles

Bibliography

alopez247. (2017). Pokémon for data mining and machine learning. Retrieved January 24, 2022, from https://www.kaggle.com/alopez247/pokemon

AlphaStar. (2019). Alphastar: Mastering the real-time strategy game starcraft ii. Retrieved February 3, 2022, from https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Bulbapedia. (2016). Category:moves that use stats from different categories. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Category:Moves_that_use_stats_from_different_categories

Bulbapedia. (2021a). Accuracy. Retrieved February 5, 2022, from https://bulbapedia.bulbagarden.net/wiki/Accuracy

Bulbapedia. (2021b). Air balloon. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Air_Balloon

Bulbapedia. (2021c). Assault vest. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Assault_Vest

Bulbapedia. (2021d). Choice band. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Choice_Band

Bulbapedia. (2021e). Choice scarf. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Choice_Scarf

Bulbapedia. (2021f). Choice specs. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Choice_Specs

Bulbapedia. (2021g). Counter (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Counter_(move)

Bulbapedia. (2021h). Court change (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Court_Change_(move)

Bulbapedia. (2021i). Damage. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Damage

Bulbapedia. (2021j). G-max steelsurge (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/G-Max_Steelsurge_(move)

Bulbapedia. (2021k). Grounded. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Grounded

Bulbapedia. (2021l). Gyro ball (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Gyro_Ball_(move)

Bulbapedia. (2021m). Hail (weather condition). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Hail_(weather_condition)

Bulbapedia. (2021n). Heavy-duty boots. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Heavy-Duty_Boots

Bulbapedia. (2021o). Life orb. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Life_Orb

Bulbapedia. (2021p). Light screen (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Light_Screen_(move)

Bulbapedia. (2021q). List of moves that cause entry hazards. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/List_of_moves_that_cause_entry_hazards

Bulbapedia. (2021r). List of pokémon by base stats (generation vii). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_base_stats_(Generation_VII)

Bulbapedia. (2021s). Magic room (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Magic_Room_(move)

Bulbapedia. (2021t). Sandstorm (weather condition). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Sandstorm_(weather_condition)

Bulbapedia. (2021u). Switcheroo (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Switcheroo_(move)

Bulbapedia. (2021v). Toxic spikes (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Toxic_Spikes_(move)

Bulbapedia. (2021w). Trick (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Trick_(move)

Bulbapedia. (2021x). Weakness policy. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Weakness_Policy

Bulbapedia. (2021y). Wonder room (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Wonder_Room_(move)

Bulbapedia. (2022a). Charizard (pokémon). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Charizard_(Pok%C3%A9mon)

Bulbapedia. (2022b). Critical hit. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Critical_hit

Bulbapedia. (2022c). Defog (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Defog_(move)

Bulbapedia. (2022d). Extreme speed (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Extreme_Speed_(move)

Bulbapedia. (2022e). Fire punch (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Fire_Punch_(move)

Bulbapedia. (2022f). Focus sash. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Focus_Sash

Bulbapedia. (2022g). Gravity (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Gravity_(move)

Bulbapedia. (2022h). Pokémon champion. Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_Champion

Bulbapedia. (2022i). Rapid spin (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Rapid_Spin_(move)

Bulbapedia. (2022j). Reflect (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Reflect_(move)

Bulbapedia. (2022k). Roost (move). Retrieved February 5, 2022, from https://bulbapedia.bulbagarden.net/wiki/Roost_(move)

Bulbapedia. (2022l). Spikes (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Spikes_(move)

Bulbapedia. (2022m). Stat. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Stat

Bulbapedia. (2022n). Stealth rock (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Stealth_Rock_(move)

Bulbapedia. (2022o). Sticky web (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Sticky_Web_(move)

Bulbapedia. (2022p). U-turn (move). Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/U-turn_(move)

Bulbapedia. (2022q). Weather. Retrieved February 2, 2022, from https://bulbapedia.bulbagarden.net/wiki/Weather

Bulbapedia. (2022r). Will-o-wisp (move). Retrieved February 3, 2022, from https://bulbapedia.bulbagarden.net/wiki/Will-O-Wisp_(move)

Chen, K., & Lin, E. (2018). Gotta train 'em all: Learning to play pokémon showdown with reinforcement learning. Retrieved February 3, 2022, from https://cs230.stanford.edu/projects_fall_2018/reports/12447633.pdf

Fandom. (2016). Ashs pikachu. Retrieved January 27, 2022, from https://pokemon.fandom.com/de/wiki/Ashs_Pikachu?action=history

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks.

Honko. (2022). Pokémon damage calculator (generation vi). Retrieved January 24, 2022, from https://calc.pokemonshowdown.com/randoms.html?gen=6&mode=randoms

Huang, D., & Lee, S. (2019). A self-play policy optimization approach to battling pokémon. *2019 IEEE Conference on Games (CoG)*, 1–4. https://doi.org/10.1109/CIG.2019.8848014

Konda, V., & Tsitsiklis, J. (2000). Actor-critic algorithms. In S. Solla, T. Leen, & K. Müller (Eds.), *Advances in neural information processing systems* (p. 7). MIT Press. https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Lee, S., & Togelius, J. (2017). Showdown ai competition. *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 191–198. https://doi.org/10.1109/CIG.2017.8080435

MattL. (2014). What are checks and counters? Retrieved February 3, 2022, from https://www.smogon.com/smog/issue32/checks-and-counters

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality.

Mitarai, R. (2021). Random battles competitive guide. Retrieved February 2, 2022, from https://www.smogon.com/articles/randbats-guide-gen8

Nintendo. (2019a). Das kampf-stadion. Retrieved February 3, 2022, from https://swordshield.pokemon.com/de-de/gameplay/pokemon-battle-stadium/

Nintendo. (2019b). Pokémon schwert. Retrieved February 3, 2022, from https://www.nintendo.de/Spiele/Nintendo-Switch/Pokemon-Schwert-1522111.html

Nintendo. (2022). Pokémon rote edition. Retrieved February 3, 2022, from https://www.nintendo.de/Spiele/Game-Boy/Pokemon-Rote-Edition-266109.html

OpenAI. (2018). Openai five. Retrieved February 2, 2022, from https://blog.openai.com/openai-five/

pmariglia. (2022). Github/pmariglia-showdown. Retrieved February 2, 2022, from https://github.com/pmariglia/showdown

Pokémon-Database. (2022). Pokémon types & type chart. Retrieved February 2, 2022, from https://pokemondb.net/type

RemptonGames. (2021). Programming ai for pokemon showdown + bot battle royale! Retrieved February 3, 2022, from https://www.youtube.com/watch?v=rhvj7CmTRkg

Sahovic, H. (2022a). Github/poke-env. Retrieved February 2, 2022, from https://github.com/hsahovic/poke-env

Sahovic, H. (2022b). Poke-env: Baselines.py. Retrieved February 2, 2022, from https://github.com/hsahovic/poke-env/blob/master/src/poke_env/player/baselines.py

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2018). High-dimensional continuous control using generalized advantage estimation.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms.

Smogon. (2022a). Github/damage-calc. Retrieved February 2, 2022, from https://github.com/smogon/damage-calc

Smogon. (2022b). Github/pokemon-showdown. Retrieved February 3, 2022, from https://github.com/smogon/pokemon-showdown

TheThirdBuild. (2021). How an a.i. is becoming the world's best pokemon player. Retrieved February 3, 2022, from https://www.youtube.com/watch?v=rhvj7CmTRkg

Veekun. (2021). Parasect. Retrieved February 2, 2022, from https://veekun.com/dex/pokemon/parasect

Wikipedia. (2022a). Expectiminimax — Wikipedia, the free encyclopedia [[Online; accessed 24-January-2022]]. http://en.wikipedia.org/w/index.php?title=Expectiminimax&oldid=952355961

Wikipedia. (2022b). Pokémon. Retrieved January 27, 2022, from https://en.wikipedia.org/wiki/Pok%C3%A9mon

Appendix

A. Figures

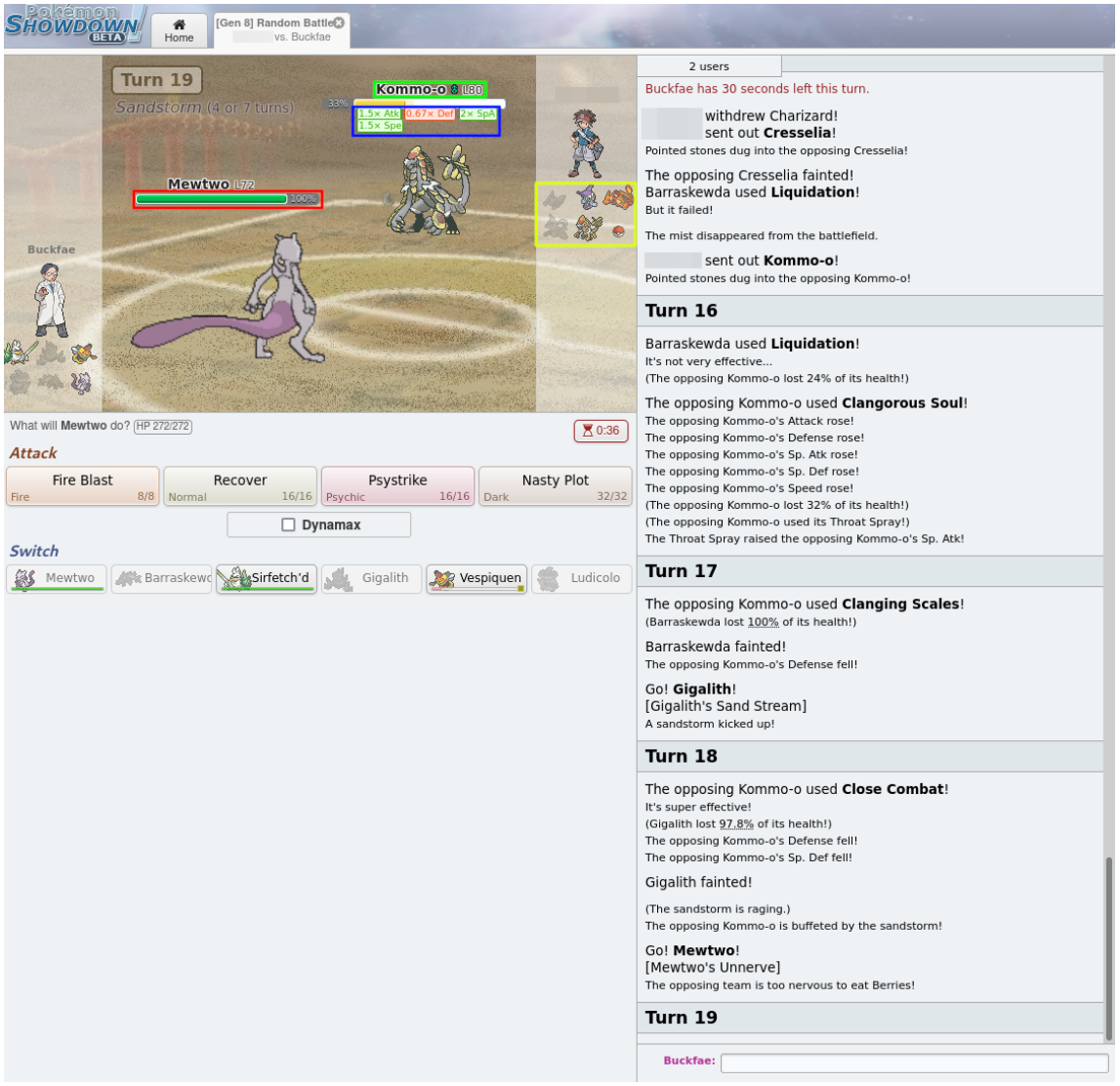


Figure A.1.: Battle between two players on *Pokémon Showdown*. The *Mewtwo* of the player *Buckfae* is at full health while the opposing *Kommo-o* has 33% of his health remaining. Below the game window, all possible actions for the player is displayed. A log of previous actions is displayed on the right-hand side.

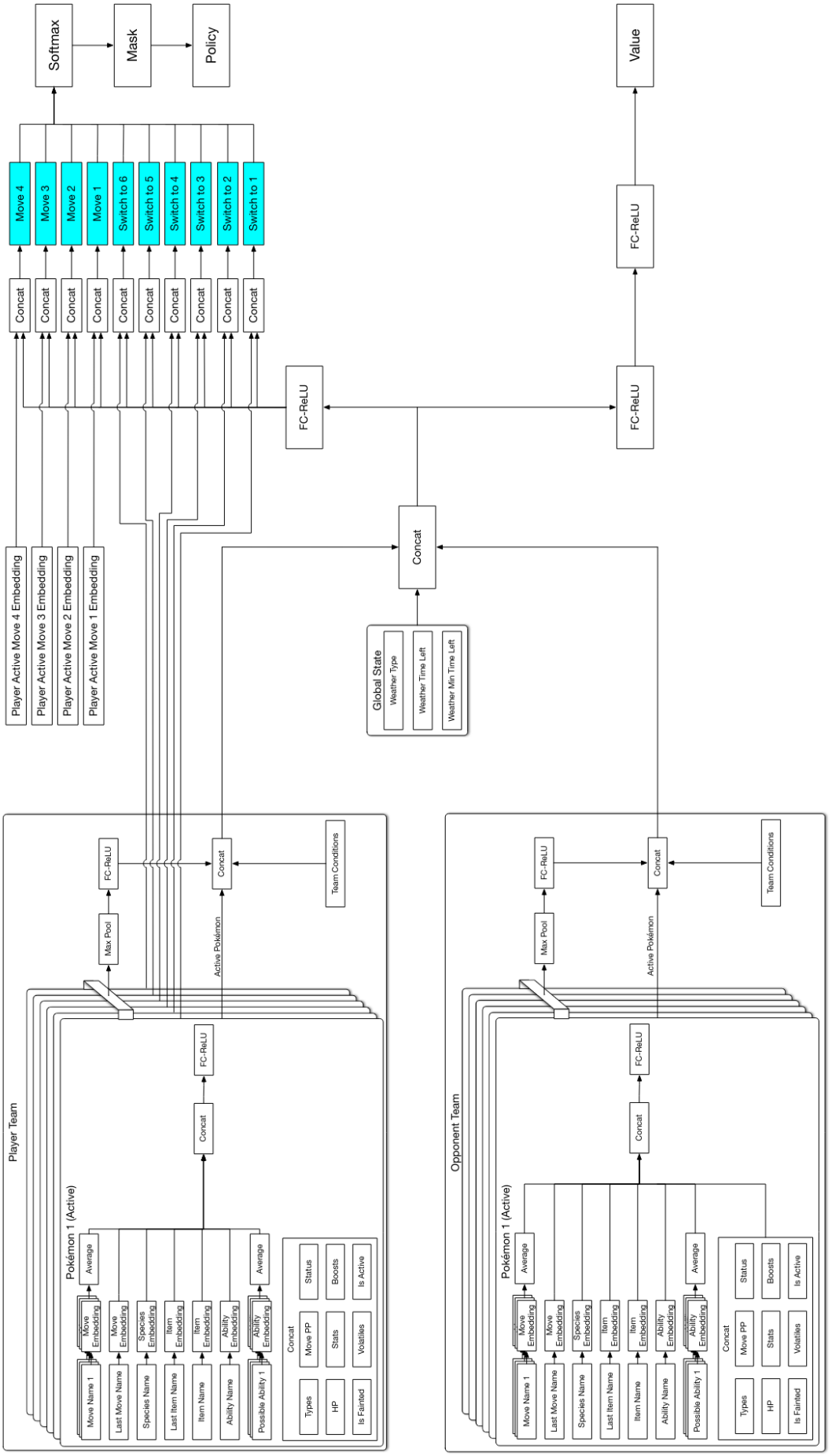


Figure A.2.: The actor-critic neural network used by the authors of Huang and Lee, 2019. Embeddings are used to properly represent different game objects. The network has two outputs: a probability distribution over actions to take, and an estimate of player strength in the current state.

Figure A.3.: Possible end game scenario. In the first line of a node are the two active Pokémon, the Pokémon that is expected to faint first is underlined. The lines below shows the remaining hp of all Pokémon for both teams. The last line shows the Value of the given state. A positive value indicates that the player is expected to win. If the node is outlined green, the first player has to bring in a new team member as his active Pokémon fainted, a red node indicates that the opponent has to send a new Pokémon into battle.

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich keiner anderer als der in den beigefügten Verzeichnissen angegebenen Hilfsmittel bedient habe. Alle Textstellen, die wörtlich oder sinngemäß aus Veröffentlichungen Dritter entnommen wurden, sind als solche kenntlich gemacht. Alle Quellen, die dem World Wide Web entnommen oder in einer digitalen Form verwendet wurden, sind der Arbeit beigefügt.

Weitere Personen waren an der geistigen Leistung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Ghostwriters oder einer Ghostwriting-Agentur in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit ist vollständig. Mir ist bewusst, dass nachträgliche Ergänzungen ausgeschlossen sind.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung zur Versicherung der selbstständigen Leistungserbringung rechtliche Folgen haben kann.

Würzburg, 07. February 2022

.....

(Julian Schubert)