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

Jonathan Krebs
First Advisor

Submission
XX. Month 20YY

www.uni-wuerzburg.de

Abstract

...

Zusammenfassung

Contents

1. Introduction

1.1. What is Pokémon

(Players move at the same time, unlike in chess.) (Game states in Pokémon are high-dimensional and the majority of its features are both categorical and partially observable) **ToDo** **ToDo**
(Game is turn based) **ToDo**

2. Background

2.1. Basic rules

(Explain Dynamaxing!) (Humans on Showdown do NOT know that they are playing against a bot. Bots are allowed on showdown.) **ToDo** **ToDo**

2.2. Battling

One of the key aspects of the Pokémon game 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't 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 trainer. This thesis focuses on random battles taking place on Pokémon Showdown. In a random battle, both you and your opponent 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 their active Pokémon 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 2.2.2. Each Pokémon has an amount of hit points (hp). The hp of a Pokémon can be dropped by attacking it with a Move. If the hp of a Pokémon drops to zero, it faints and can't be used in this battle anymore. A player wins, once all enemies fainted.

Note: 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 2.2.6.

2.2.1. Types

Pokémon implements a *Rock-Paper-Scissors*-like system. Each Pokémon has either one or two of 18 types. For example, a *Fire*-type Pokémon is weak against *Water*-type Pokémon whereas a *Water*-type Pokémon is weak against *Grass*-type Pokémon. Lastly, a *Grass*-type Pokémon is weak against *Fire*-type Pokémon. The figure 2.1 shows how different Pokémon types interact with each other. It is important to note, that the type modifiers

DEFENSE → ATTACK ↴	NOR	FIR	WAT	ELE	GRA	ICE	FIG	POI	GRO	FLY	PSY	BUG	ROC	GHO	DRA	DAR	STE	FAI
NORMAL													½	0			½	
FIRE		½	½		2	2						2	½		½		2	
WATER		2	½		½				2				2		½			
ELECTRIC			2	½	½				0	2					½			
GRASS		½	2		½			½	2	½		½	2		½		½	
ICE		½	½		2	½			2	2					2		½	
FIGHTING	2					2		½		½	½	½	2	0		2	2	½
POISON					2			½	½				½	½			0	2
GROUND		2		2	½			2		0		½	2				2	
FLYING				½	2		2					2	½				½	
PSYCHIC							2	2			½					0	½	
BUG		½			2		½	½		½	2			½		2	½	½
ROCK		2				2	½		½	2		2					½	
GHOST	0										2			2		½		
DRAGON															2		½	0
DARK							½				2			2		½		½
STEEL		½	½	½		2							2				½	2
FAIRY		½					2	½							2	2	½	

Figure 2.1.: Pokémon type chart [1]

will be multiplied if a Pokémon has two types. For example, a *Fire*-type attack will deal 4 times the damage against *Parasect* as *Parasect* has the types *Grass* and *Bug* [2].

2.2.2. Moves

Moves can be split up into three categories: *Physical*-, *Special*- and *Status*-Moves. While *Physical*- and *Special*-Moves usually deal damage to the opponent Pokémon, *Status*-Moves can for example change the weather, which plays a role in damage calculation explained in section 2.2.9, inflict status effects, raise or lower the stats of a Pokémon. In addition, a move also has exactly one of the 18 possible types.

2.2.3. Pokémon

A key-concept of Pokémon battles are the *stats* of a Pokémon. The most important stats are explained below.

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 2.2.8. **(Cover evasion / accuracy, context to showdown)**

ToDo

Determination of stats

The total stat of a Pokémon is calculated as described in equation 2.1 and equation 2.2 [3].

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

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

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 2.2, 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. If the level of a Pokémon increases, so will its stats. For each level gained (ignoring Nature), stats will increase by 1/50 the base stat value, and 1/100 the combined individual values (iv) and effort values (ev) values [3]. In Pokémon Showdown, the level of a Pokémon is set at the start of the battle and won't increase [4].

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 100% of what it would be without the Nature, and the stat hindered will be 90% of its normal value [3]. Nature can be neglected in this thesis as all Pokémon in random battles have a neutral nature, meaning no stat is enhanced or hindered [4].

IV: Refers to the iv of a Pokémon. These cause two Pokémon of the same species to have different Stats [3]. 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 [4].

EV: These are the 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't have more than 255 ev in a single stat [3]. Random Pokémon on Showdown will always have 85 ev in each stat, or 0 in the case that having a high stat being detrimental [4].

Figure 2.2 displays information about possible stat-combinations of *Charizard*.

2.2.4. 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.

2.2.5. Status condition

A Pokémon can have a status condition, this affects the Pokémon negatively. Status conditions are inflicted by moves. The most important status conditions are

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.: Possible stats of *Charizard* [5]

- **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 Speed 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.
- **Sleep:** If a Pokémon suffers from the status condition sleep (slp) it won't be able to use moves, except *Snore* and *Sleep Talk*. In the mainline games, sleep lasts randomly between one and three turns. However, in Pokémon Showdown a Pokémon will *always* be asleep for exactly two turns.

At any point, a Pokémon can only suffer from one status condition at a time, this means that a burned Pokémon can't fall asleep.

2.2.6. 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] [6]. *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[7]. The generation of items in Pokémon Showdown is described in more detail in section 2.4.2.

[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

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 [8].
- **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 [9].
- **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 [10].
- **Leftovers:** Restores 1/16 of the holder's maximum hp at the end of each turn [11].
- **Life Orb:** Boosts the damage dealt by the holder's damaging moves by 30%, but the holder takes damage equal to 10% of its maximum hp after it uses a damaging move [6].
- **Heavy-Duty Boots:** The holder is unaffected by the effects of entry hazards. Entry hazards are described in 2.3.
- **Assault Vest:** Raises the holders spd by 50%, but also prevents the holder from selecting any status move[4] [12].
- **Focus Sash:** If the holder has full hp and is hit by an attack that would otherwise cause fainting, it survives with 1 hp [13].

2.2.7. Field effects

There are multiple *field effects* that affect combat.

Terrain

Terrain is set up by the respective move with identical name and last for five turns. 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
- The Pokémon has the Ability *Levitate*
- The Pokémon is holding the item *Air Balloon*
- The Pokémon 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 [14] There are five different possible *terrain*-states.

[4]Except *Me First*

- **None:** The default state, no other effects are applied.
- **Electric Terrain:** Grounded Pokémon can't 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 (including confusion) **(Does confusion exist in Showdown?)** 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 2.2.8. The power of *Psychic*-type moves is also increased

ToDo

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

ToDo **(Explain weather)**

2.2.8. Order of events

ToDo ToDo ToDo **(Switching has the highest Priority) (Usually the faster Pokémon acts first) (Some moves have a special priority.)**

2.2.9. Damage calculation

The damage dealt by a move mainly depends on the *level* of the Pokémon that uses the move, its effective Attack or Special Attack stat, the opponent's effective Defense or Special Defense stat and the move's effective power.

Precisely, the damage is calculated as follows[15]:

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

The only exception for this are moves that deal direct damage. A list of these moves can be found at [16].

Level

Level refers to the level of the attacking Pokémon[15]. In Pokémon Showdown, the level is displayed next to the name of the Pokémon. **(Mainline games leveling)** ToDo

A / D

A is the effective Attack stat of the attacking Pokémon if the used move is a physical move,
ToDo **(Reference to physical moves)**
or the effective Special Attack stat of the attacking Pokémon if the used move is a special move. **(Reference to special moves)** ToDo
D is likewise the effective Defense stat of the target if the used move is a physical move, or the effective Special Defense of the target if the used move is a special move[15].

There are four moves that use stats from different categories, more Information can be found at [17].

Power

Power is the effective power of the used move. **(When is the power not equal to the base power)** ToDo 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[18].

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* [15]. **(Reference to weather section)** ToDo

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 [19]. Increasing crit rate, as well as other stats, will be explained in chapter 2.2.10.
Note: In earlier games, crits worked different, see [19] for more details.

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 [15].

STAB

STAB stands for *Same Type Attack Bonus*. It is a multiplier of 1.5 if the used move is of the same type as the attacking Pokémon. Otherwise, it is 1.0 [15].

Type

This is the in section 2.2.1 described type modifier [15].

Burn

Burn is 0.5 if the attacking Pokémon is burned, and the used move is a physical move[5]. Otherwise, it is 1.0 [15].

Other

The *other* modifier is usually 1. A list of exceptions can be found at [15].

2.2.10. Effective Stats

Boosting

(Boosting critical rate) ToDo

2.3. Hazards

An *entry hazard* is a condition that affects a side of the field that causes any Pokémon that is sent into battle on that side of the field to be afflicted by a negative effect. Entry hazards are created by moves, usually status moves [20].
(This paragraph is copied word by word from Bulbapedia) ToDo

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

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.

ToDo **(Removal and Immunity of Spikes)** Spikes are created by the move *Spikes*[21].

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 **??**[22]. *Note:* Stealth Rocks can also be created by the move *G-Max Stonesurge*.

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

Table 2.1.: Damage dealt to Pokémon by Stealth Rocks[22]

This damage-dealing Water-type G-Max move is exclusive to Gigantamax Drednaw [23].

ToDo **(Does this move exist in Showdown)**

Sticky Web

The entry hazard set by the *Bug*-type move *Sticky Web* lowers the opponents speed stat by one stage upon switching in [24].

ToDo **(Pokémon that are not affected by this)**

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 [25].

ToDo **(Pokémon not affected)**

ToDo **(Explain (badly) poisoning)** **(Pokemon that can remove spikes)**

ToDo

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 Copperhead. 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 **??** [26]. **(Unaffected Pokémon)** ToDo

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

Table 2.2.: Damage dealt to Pokémon by Sharp Steel[26]

2.3.2. Hazard counterplay

There are some moves that can remove entry hazards. *Rapid Spin* [27] removes entry hazards from the user's side of the field and *Defog*[28] removes entry hazards on both sides of the field[6]. In addition, *Court Change*[29] will exchange the entry hazards on each side of the field, along with other one-sided field conditions. **(What other one-sided field conditions are there?)** ToDo If a grounded[7] *Poison*-type Pokémon enters the battle, it will remove Toxic Spikes, described in 2.3.1, from its side of the field. Lastly, Pokémon holding the item *Heavy-Duty Boots*[30] are unaffected by entry hazards, but grounded *Poison*-type Pokémon can still remove Toxic Spikes even if they hold the boots[20]. There are various exceptions and special cases to hazards. **(Special cases of hazards)** ToDo

2.4. Showdown random battles

(Write introduction to this) (This has to include that the same species has different movesets) ToDo ToDo

2.4.1. Sets

As described in section 2.2.3, 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 [4]. 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 2.2.9 is not a fixed value. It is determined as described in 2.4 [31].

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

As 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 [32].
This knowledge can be exploited to gather additional information about the enemy, section 4.2 describes how this is achieved.

[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.

[7] The term *grounded* is used to describe a Pokémon that can't be affected by damaging *Ground*-type moves and several other associated effects[14].

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 [4], 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 doesn't meet one of the stat qualifications or doesn't 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 [33].
  - Pokémon with *Trick* [34] or *Switcheroo* [35], both moves that allow to switch items with the opponent, they 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. **(Either explain priority moves or explain them here)**

ToDo

- Pokémon with 4 attacks that don't qualify for choice items, will get an *Assault Vest* if their hp + def + spd $\geq$ 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 loose 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 $\geq$ 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 [36].

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*

2.5. 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

There are multiple definitions of *check* and *counter* **(Cite multiple definitions)**. In this thesis, we refer 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. **ToDo**

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

3.1. Baseline Agents

A good way to get a rough idea on how well an agent performs can be to compare it against a baseline agent. There are two very 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. Breath-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[37].

3.3. Minimax

Minimax builds upon bfs as this algorithm deals with adversarial paradigms by assuming the opponents act in their best interest. There are multiple possibilities on how to implement the Minimax algorithm for Pokémon games. The main difference in different implementations lies in state evaluation and the assumptions about the opponents. Additionally, the amount of features taken into consideration have a huge impact. In the implementation proposed by [37], a node represents the worst case scenario that would occur as a result of the current choice. The agent also uses alpha-beta pruning, ignoring any node in which the agents Pokémon faints. One drawback of this procedure is that like this, 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 wors-case evaluation are performed using the evaluation function 3.1 [37]:

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

A YouTube-Video released by *Rempton Games* [38] 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 [37] 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 1000 total games, yielding a win-ratio of $\approx 83\%$. Due to the large amount of random number generation (rng) in battles, we assume both agents 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 [39]. 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 [40]. Currently, the agent calculates two turns in advance, and reaches an Elo-Rating of 1461 in Generation VII random battles on the Showdown ladder. In addition, *pmariglia* implemented an algorithm to determine estimate the item a Pokémon is holding based on the damage it dealt.

3.4. Rule based agents

The YouTube-Video released by *Rempton Games* [38] 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 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* library [41]. The implementation for this agent can be found at [42]. This agent uses a few simple rules to determine 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 reffers to the implementation of *Rempton Games*.

	Random	*Max damage (dmg)*	*Smart dmg*	*Minimax*	*Heuristics*
Random	N / A	897 / 103	957 / 43	831 / 169	992 / 8
Max dmg		N / A	829 / 171	834 / 166	955 / 45
Smart dmg			N / A	331 / 669	720 / 280
Minimax				N / A	181 / 819

Table 3.1.: Denotes how many games the Agent in the column won against the agent in the row [38]

3.5. Other approaches

The authors of the *Showdown AI Competition*[37] 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 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 [37]. 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 [37].

Pruned Breadth-First Search

This agent is designed to demonstrate a simple way to utilize domain knowledge as a cost-cutting measure. This algorithm does so 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 [37].

Results

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

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

Table 3.2.: Results of other approaches against a random agent [37]

3.6. Self-Play Policy Optimization Approach

Researchers from *New York University* [37] 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 opponents 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 **??**

One year after the publication of [37], the authors of [43] improved this design. They used proximal policy optimization (ppo) [44] to train an agent. They also used embeddings to better represent a Pokémon. Data available at [45] 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 [46] which creates embeddings from graph data in similar to Word2Vec [47]. 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 discovered similarity using this approach. Here,

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

Table 3.3.: Similar Pokémon within the embedding space of [43]

a similar 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 don't share a single move in their move set: In generation 6, *Ho-Oh* has access to these 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* [48]. This inappropriate classification is likely caused to both *Pokémon* having similar stats and both being legendary. **(The paper states that they are confident in their embeddings. This critic is my own work. How do I note this properly?)**

ToDo

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) has 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.

The authors of [43] trained the agent against a random agent, a default agent that always tries to pick a non-switching move as well as a *Minimax*[2] agent until the reward curve stabilized which was around 100 epochs where an epoch is a single 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 [43] describe their final agent as

Opponent	Average epoch reward
Random	0.85
Default	0.85
Minmax	−0.9

Table 3.4.: Average epoch rewards after training convergence for opponent agents [43]

[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

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, the authors of [49] published a paper titled *A Self-Play Policy Optimization Approach to Battling Pokémon*. This paper will be summarized in more depth as it is very detailed, and the agents proposed are performing on par with the state-of-the-art search based Pokémon AI.
Similar to OpenAI's Dota AI [50], the agent is represented using an actor-critic neural network. Actor-critic RL methods [51] 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". The authors represent both actor and critic using a two-headed neural network which is trained via self-play RL [49].

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 2.2.7
3. Each *Pokémon* has many features. Table 3.5 contains a partial list[3]

Feature	Type	Dims	Description
species	categorical	1×1023	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

Table 3.5.: Features used to describe a single Pokémon battle [49]

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 $v \in \mathbb{R}$. The probability distribution π is computed as follows: **(Move this figure to the appendix)** **ToDo**

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

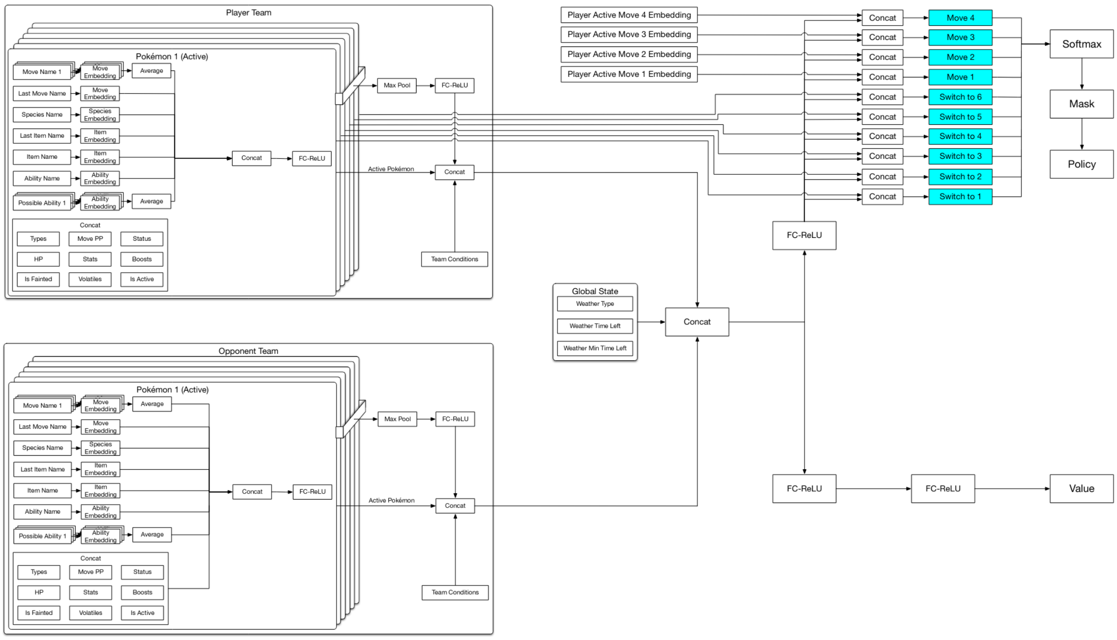

Figure 3.1.: The actor-critic neural network used by the authors in [49]

1. The network outputs an intermediate vector $p \in \mathbb{R}^n$. Each of the colored cells in figure 3.1 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 $\pi : \pi_i = \frac{s_i \pi'_i}{s^T \pi'}$

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 3.1 and contains 1,327,618 parameters in total [49].

Training the network

Training was done serially: After $m = 7680$ 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 are 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.

To update the neural network, the authors use *Proximal Policy Optimization* [44], 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* [52] 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 [43]. Futhermore, the authors played 1.000 games against the open source agent of *pmariglia* described in 3.3 Table 3.6 describes the

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

Table 3.6.: Performance of the agent developed by [49]

performance of the agent developed by [49] **(Glicko-1 Rating: 1677. Explain this rating and add Glicko-1 of HerrGewitter)** **ToDo**

3.7. Supervised Approach

As of writing, there is only one publication researching supervised learning for Pokémon battles, a YouTube video uploaded to the channel "*The Third Build*" with the title "*How an A.I. is Becoming the World's Best Pokemon Player*"[53]. 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[4] 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 of 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 the 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 described in this thesis, this agent does not play *random* battles on Pokémon Showdown, but rather *Gen 8 OU Singles*. In this format, the teams of both players are not random but rather chosen by the player. In addition, each player knows the species of all enemies at the beginning of the game but not their exact build.

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

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

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 implimentation of *pmariglia* described in 3.3. According to the video, this agent reached a maximum Elo of 1630 which ranks the agent among the best 3.6% of players.

4. Approach

(Predictions) (Matchups only recalculated on new information.) **ToDo** **ToDo**

4.1. Communication with Pokémon Showdown

The communication with Pokémon Showdown is handled using the python library *Poke-Env* [41]. 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* [54], 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 2.4, the same Pokémon can occur in various different builds, meaning the combination of moves, abilities and items. Knowing the exact enemies build is very important for decision-making. Consider the following example:

- **Player1** has an active *Charizard* with *Heavy-Duty Boots* and 150hp remaining on the field.
- **Player2** has just sent out a *Drapion* with 160hp remaining.
- The *Charizard* is faster but can't 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 play 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. 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*.

Implementation details

The first step to predicting enemy sets is to determine all possible sets as well has how likely each individual set is. In order to achieve this, I wrote a script that to start a battle between an information gathering player and a random agent. 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 played, 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 this 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. 4.1

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

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 explicitly mean 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 huge role. The reasoning behind this as well as the determination of
ToDo an optimal order is explained in **(Ref to MinMax)**. This metric also has multiple flaws

as 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 taken into account.
Similarly, we can determine the *thread level* as shown in equation 4.3

$$thread\_level(j) = \sum_{i \in \text{Own Pokémon}} score(e_{j,i}) \tag{4.3}$$

Combining the *value* and *thread level*:

$$board\_rating = \sum_{i \in \text{Own Pokémon}} val(i) - \sum_{j \in \text{Enemy Pokémon}} thread\_level(j) \tag{4.4}$$

(Maybe use can_defeat with 0 / 1 instead here) Therefore, a positive rating indicates that the board is favorable for the player, a negative rating indicates that the board is unfavorable for the player and a value close to zero indicates that no player currently has and advantage over the other. **ToDo**

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 opponents entire team. Therefore, we try to gather information about the enemy team while sacrificing as little hp as possible. In this stage, we act following 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 won't switch. Otherwise, we switch to a *check*, or *counter* if present. Next, we check if the current matchup is very 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, 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 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

- The Pokémon has to fulfill at least one of these criteria:
  - 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 not Pokémon matches these criteria, the Pokémon with the lowest score is picked instead. In any other case, the currently active Pokémon will use the best move calculated in 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 4.1 shows a possible end-game scenario. In this simple example, we

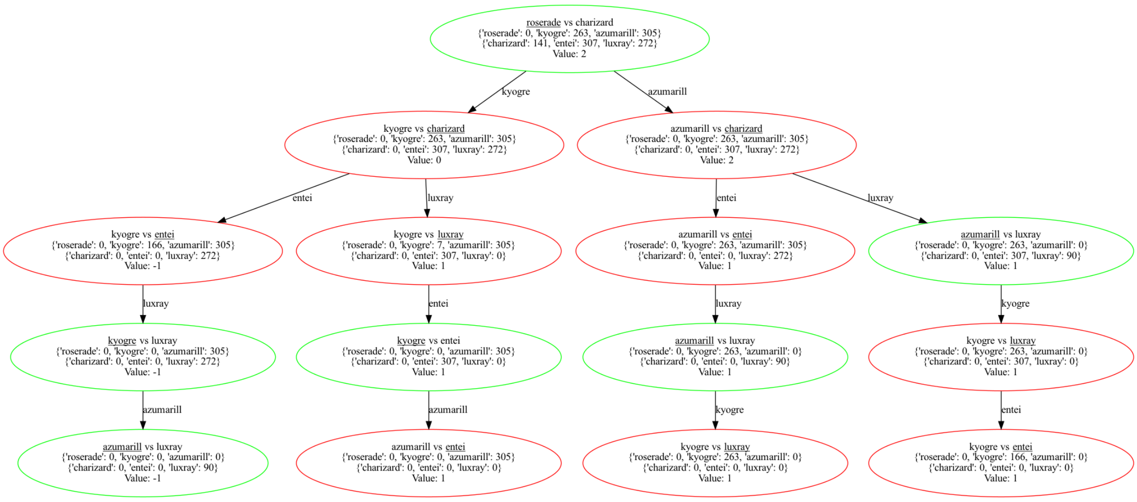

Figure 4.1.: Possible end game scenario

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, indicated by the underlined name. This means, that now Player 1 has to switch in a new Pokémon. A green circle around a node indicates that the first player has to make a turn as his Pokémon fainted, and a red circle marks a decision for Player 2. 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*.

ToDo **(Explain why MinMax Tree is correct. Pokemon don't have expected items / moves!)**

4.5. Determining matchups

In order to determine whether to attack or to switch, we need to determine how the optimal moves for a Pokémon against another Pokémon are 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. 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 [1] Here, we also take boosting, items, status effects and the possibly changing field state into account **(Reference to further work with things that do not work like recognizing focus items)**. In the early game. Then, 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. **ToDo**

[1]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.

5. Evaluation

(Lee-Paper: MinMax requires simulation) **ToDo**

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 random, one player often ends up with a slightly better team than his opponent. In very 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 Player one was capable of defeating the entire enemy team.
The Pokémon of Player on 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 doesn't pressure *Glalie* much, 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. As he fails to pressure *Volcarona* as well, 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 very extreme cases, evaluation of agents against other agents should be done using multiple hundred, better thousands of games against each other.
In order to quantify and better visualize the randomness of battles, we started by generating **(HOW MANY GAMES IN TOTAL FOR GRAPH)** random battles and collected the team information. After both teams were stored, two *MaxDamage*-agents finished the game to determine a winner. Next, we determined the matchups for each battle as described in 4.5. Finally, the board rating was calculated as follows:

ToDo

```
# Amount of counters player 1 has against player 2
counter_p1_count = sum([sum([
  1 if m.is_counter(m.pokemon_1.species, m.pokemon_2.species)
  else 0 for m in matchups if m.pokemon_1.species == p.species])
  for p in self.team_1])

# Amount of counters player 2 has against player 1
counter_p2_count = sum([sum([
  1 if m.is_counter(m.pokemon_2.species, m.pokemon_1.species)
  else 0 for m in matchups if m.pokemon_2.species == p.species])
  for p in self.team_2])

board_rating = counter_p1_count - counter_p2_count
```

Listing 5.1: Calculate Board rating

Finally, figure 5.1 was created. The figure contains how often what board-rating occurs (red) as well as the win-rate for the given rating (green). Not only does this figure show a gaussian distribution of how fair games are **(Schlechte Formulierung!)**, but also that our method of determining matchups works as intended. This is due to the fact that games in which we assume the player to be in a bad position also have a win-rate and games with a favorable board rating show the highest win ratio.

ToDo

5.1.2. Evaluation against human opponents

As described in **(Link Showdown chapter)**, Pokémon Showdown allows researchers to use bots on the ranked ladder.

ToDo

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. In order to determine what moves to be used, the Agent generates every possible move combination with the specified amount of turns into the future and calculates the expected outcome while assuming that the enemy does not move at all, similar to the bfs-based algorithm described in paragraph **??**. 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 determination of matchups. This results in the bot often spamming moves like *Draco Meteor*, a *special Dragon*-type move that deals a lot of damage but also lowers the users spa by two stages resulting in the move dealing less damage every

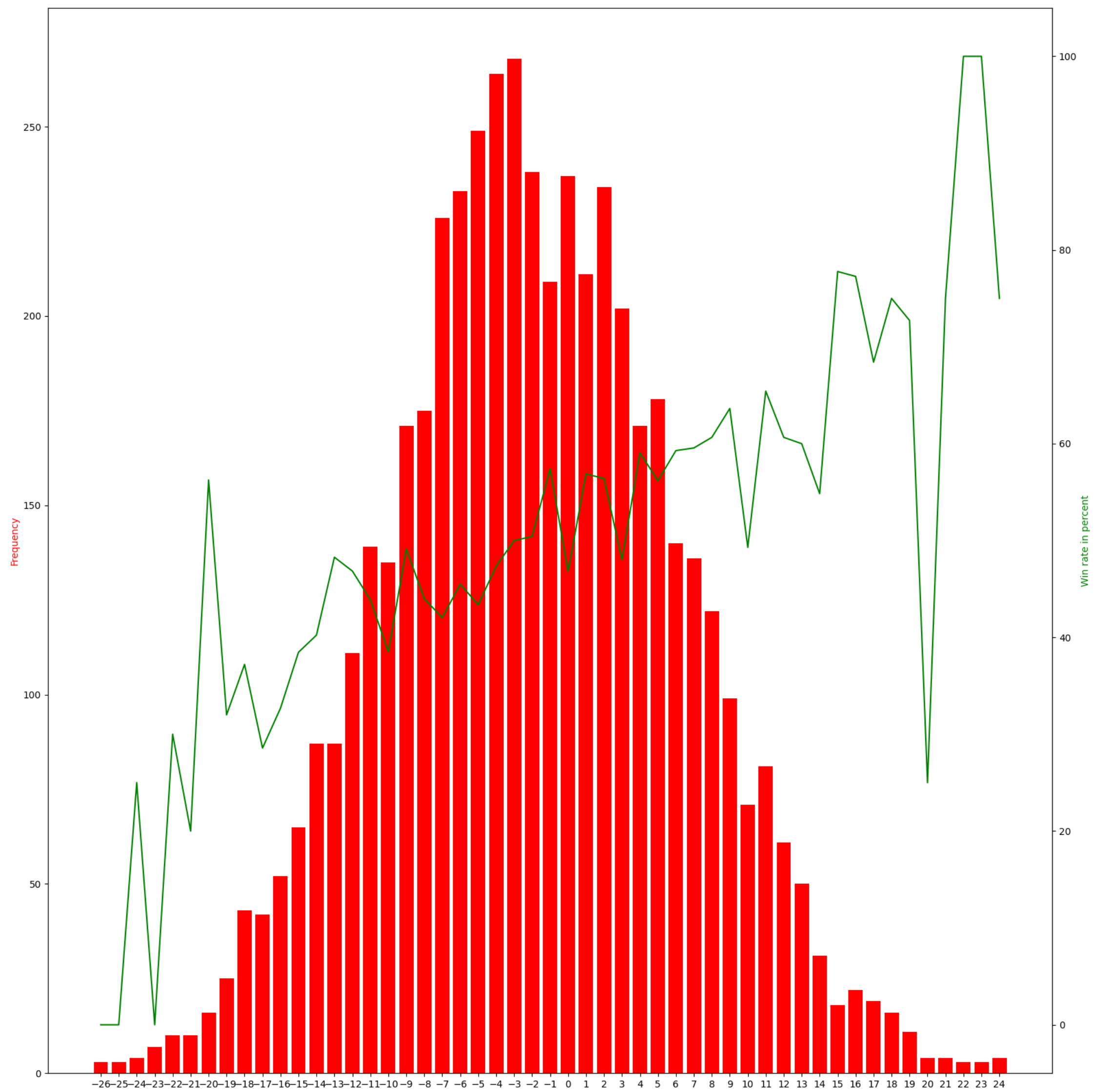

Figure 5.1.: Win rate for a given board rating

time it is used. 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 check if the current matchup is not favorable, 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.

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, more things are taken into consideration when calculating damage, current stat changes are taken into consideration as well as status conditions. In addition to that, abilities and items are considered for damage calculation. Furthermore, recoil from moves, healing both from items like *Leftovers* [11] and moves like *Recover* aren't neglected anymore.
Switching and the selection of moves is done as described in 4.
These improvements lead to *HerrGewitter* avoiding mistakes of *HerrDonner*. For example, this agent will burn a physical attacker using for example *Will-O-Wisp* [**?**] 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 swap.
There are still a lot of features that *HerrGewitter* is lacking.

Weather and Field effects

The first thing to improve in future versions is to add proper support for weather and field effects in the damage calculator as well as in the *MinMax*-Algorithm. Currently, the agent is for example not aware 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 does most of the time result 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* [28]
- The agent will not clear hazards

Choice Items

As described in section 2.2.6 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 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
ToDo future matchups. As described in **(Link chapter about re-determining matchups)**, 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 lead 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 *MinMax* 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 loose all matchups against *Flying*-Type Pokémon.
While this behavior rarely effects battle, the agent failing to realize 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't 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 we added these features to our implementation, some moves are still not properly implemented. For example, the move *Counter* has a move priority **(Explain move priorities)** of -5 and works as follows: **ToDo** If the last mount of damage dealt before the use of *Counter* is greater than zero and was dealt by a *Physical*-Type 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 [55]. 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 don't 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.

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.

MinMax

The *MinMax* algorithm described in paragraph 4.4 only support 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

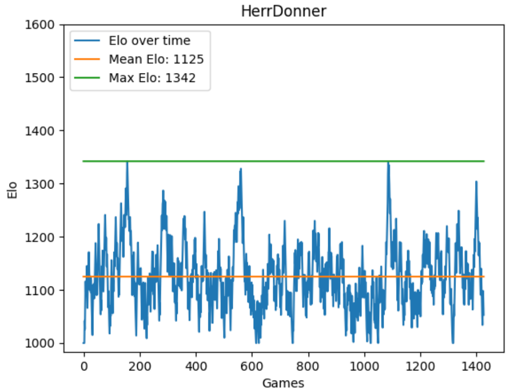

Figure 5.2.: Elo HerrDonner

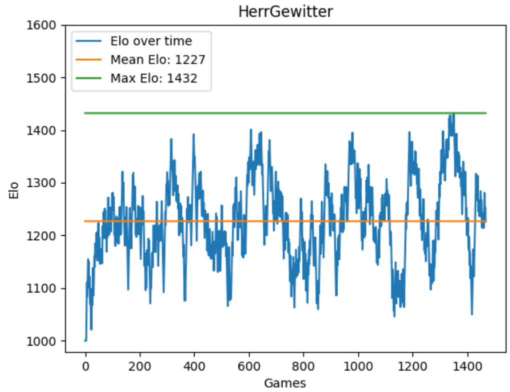

Figure 5.3.: Elo HerrGewitter

ToDo **(Print this page and check if the figures are to small)** Both agents were evaluated at the same time over the span of multiple days by playing over 1400 ranked games each against human opponents on Pokémon Showdown. Figure 5.2 shows the Elo rating of *HerrDonner* over time, while figure 5.3 displays the Elo rating of *HerrGewitter*. In order

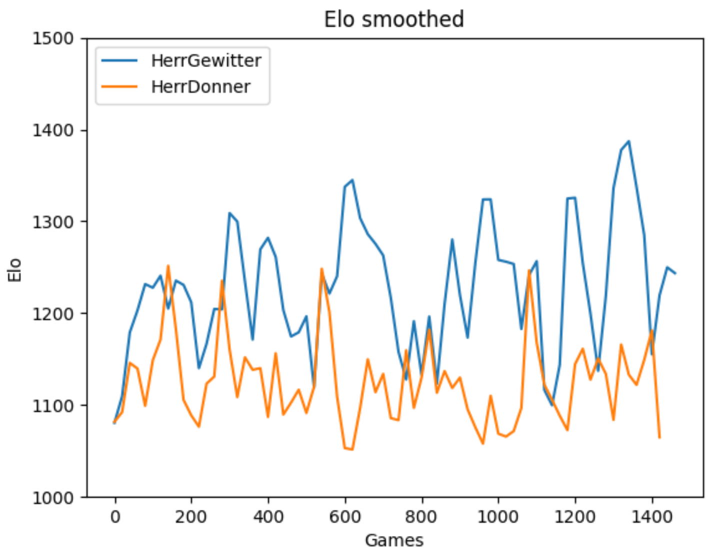

Figure 5.4.: Smoothed Elo

to make comparison of both agents more easy, figure *fig:elo-smoothed* 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 5.2: Smoothing Elo values

The first thing to note is that *HerrGewitter* has a higher mean Elo (1227) as well as a higher max Elo (1432) than *HerrDonner* who achieved an average Elo of 1125 and peaked at an Elo of 1342. **(Why is my agent worse than Pmariglia?)** Table 5.1 shows the performance **ToDo**

	Random	MaxDmg	Pmariglia [39]	DQN [49]	Donner	Gewitter
Random	N / A	-	-	995 / 5	992 / 8	993 / 7
MaxDmg		N / A	-	929 / 71	906 / 94	951 / 49
Pmariglia			N / A	612 / 388	-	273 / 727
DQN				N / A	-	-
Donner					N / A	580 / 420
Gewitter						N / A

Table 5.1.: Battle results of 1,000 games between the agents.

of different agents when directly competing against each other. **(DQN played against an older version of Pmariglia.)** 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 a multiple things to point out here: **ToDo**
While *HerrDonner* and *HerrGewitter* achieved almost similar results when battling a random agent, *HerrDonner* performed notably worse against the *MaxDmg* agent which always picks the move with the highest base power. Also, *HerrDonner* only managed to win 58% against *HerrDonner* despite notably better results against human players **(Above section how hard it is to beat Donner / Gewitter)**. **ToDo**

6. Conclusion

(Further Work: Deeper Tree = better results?) **ToDo**

List of Figures

List of Tables

Listings

Acronyms

TBD To Be Done

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

dmg damage

Bibliography

[1] P. Database, "Pokémon types & type chart." https://pokemondb.net/type, 2022.

[2] Veekun, "Parasect." https://veekun.com/dex/pokemon/parasect, 2021.

[3] Bulbapedia, "Stat." https://bulbapedia.bulbagarden.net/wiki/Stat, 2021.

[4] R. Mitarai, "Random battles competitive guide." https://www.smogon.com/articles/randbats-guide-gen8, 2021.

[5] Bulbapedia, "Charizard (pokémon)." https://bulbapedia.bulbagarden.net/wiki/Charizard, 2022.

[6] Bulbapedia, "Life orb." https://bulbapedia.bulbagarden.net/wiki/Life_Orb, 2021.

[7] Bulbapedia, "Air balloon." https://bulbapedia.bulbagarden.net/wiki/Air_Balloon, 2021.

[8] Bulbapedia, "Choice band." https://bulbapedia.bulbagarden.net/wiki/Choice_Band, 2021.

[9] Bulbapedia, "Choice scarf." https://bulbapedia.bulbagarden.net/wiki/Choice_Scarf, 2021.

[10] Bulbapedia, "Choice specs." https://bulbapedia.bulbagarden.net/wiki/Choice_Specs, 2021.

[11] Bulbapedia, "Leftovers." https://bulbapedia.bulbagarden.net/wiki/Leftovers, 2021.

[12] Bulbapedia, "Assault vest." https://bulbapedia.bulbagarden.net/wiki/Assault_Vest, 2021.

[13] Bulbapedia, "Focus sash." https://bulbapedia.bulbagarden.net/wiki/Focus_Sash, 2021.

[14] Bulbapedia, "Grounded." https://bulbapedia.bulbagarden.net/wiki/Grounded, 2021.

[15] Bulbapedia, "Damage." https://bulbapedia.bulbagarden.net/wiki/Damage, 2021.

[16] Bulbapedia, "Category:moves that deal direct damage." https://bulbapedia.bulbagarden.net/wiki/Category:Moves_that_deal_direct_damage, 2021.

[17] Bulbapedia, "Category:moves that use stats from different categories." https://bulbapedia.bulbagarden.net/wiki/Category:Moves_that_use_stats_from_different_categories, 2016.

[18] Bulbapedia, "Fire punch (move)." https://bulbapedia.bulbagarden.net/wiki/Fire_Punch_(move), 2021.

[19] Bulbapedia, "Critical hit." https://bulbapedia.bulbagarden.net/wiki/Critical_Hit, 2021.

[20] Bulbapedia, "List of moves that cause entry hazards." https://bulbapedia.bulbagarden.net/wiki/List_of_moves_that_cause_entry_hazards, 2021.

[21] Bulbapedia, "Spikes (move)." https://bulbapedia.bulbagarden.net/wiki/Spikes_(move), 2021.

[22] Bulbapedia, "Stealth rock (move)." https://bulbapedia.bulbagarden.net/wiki/Stealth_Rock_(move), 2021.

[23] Bulbapedia, "G-max stonesurge (move)." https://bulbapedia.bulbagarden.net/wiki/G-Max_Stonesurge_(move), 2021.

[24] Bulbapedia, "Sticky web (move)." https://bulbapedia.bulbagarden.net/wiki/Sticky_Web_(move), 2021.

[25] Bulbapedia, "Toxic spikes (move)." https://bulbapedia.bulbagarden.net/wiki/Toxic_Spikes_(move), 2021.

[26] Bulbapedia, "G-max steelsurge (move)." https://bulbapedia.bulbagarden.net/wiki/G-Max_Steelsurge_(move), 2021.

[27] Bulbapedia, "Rapid spin (move)." https://bulbapedia.bulbagarden.net/wiki/Rapid_Spin_(move), 2021.

[28] Bulbapedia, "Defog (move)." https://bulbapedia.bulbagarden.net/wiki/Defog_(move), 2021.

[29] Bulbapedia, "Court change (move)." https://bulbapedia.bulbagarden.net/wiki/Court_Change_(move), 2021.

[30] Bulbapedia, "Heavy-duty boots." https://bulbapedia.bulbagarden.net/wiki/Heavy-Duty_Boots, 2021.

[31] Bulbapedia, "Gyro ball (move)." https://bulbapedia.bulbagarden.net/wiki/Gyro_Ball_(move), 2021.

[32] Bulbapedia, "List of pokémon by base stats (generation vii)." https://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_base_stats_(Generation_VII), 2021.

[33] Bulbapedia, "U-turn (move)." https://bulbapedia.bulbagarden.net/wiki/U-Turn_(move), 2021.

[34] Bulbapedia, "Trick (move)." https://bulbapedia.bulbagarden.net/wiki/Trick_(move), 2021.

[35] Bulbapedia, "Switcheroo (move)." https://bulbapedia.bulbagarden.net/wiki/Switcheroo_(move), 2021.

[36] Bulbapedia, "Weakness policy." https://bulbapedia.bulbagarden.net/wiki/Weakness_Policy, 2021.

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

[38] "Programming ai for pokemon showdown + bot battle royale!," 06 2021.

[39] pmariglia, "Github/pmariglia-showdown." https://github.com/pmariglia/showdown, 2022.

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

[41] H. Sahovic, "Github/poke-env." `https://github.com/hsahovic/poke-env`, 2022.

[42] H. Sahovic, "Poke-env: baselines.py." `https://github.com/hsahovic/poke-env/blob/master/src/poke_env/player/baselines.py`, 2022.

[43] K. Chen

[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[45] alopez247, "Pokémon for data mining and machine learning." `https://www.kaggle.com/alopez247/pokemon-for-data-mining-and-machine-learning`, 2017. [Online; accessed 24-January-2022].

[46] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016.

[47] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.

[48] Honko, "Pokémon damage calculator (generation vi)." `https://calc.pokemonshowdown.com/randoms.html?gen=6&mode=randoms`, 2022. [Online; accessed 24-January-2022].

[49] D. Huang and S. Lee, "A self-play policy optimization approach to battling pokémon," in *2019 IEEE Conference on Games (CoG)*, pp. 1–4, IEEE, Aug 2019.

[50] OpenAI, "Openai five." `https://blog.openai.com/openai-five/`, 2018.

[51] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," p. 7.

[52] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

[53] "How an a.i. is becoming the world's best pokemon player," 07 2021.

[54] Smogon, "Github/damage-calc." `https://github.com/smogon/damage-calc`, 2021.

[55] Bulbapedia, "Counter (move)." `https://bulbapedia.bulbagarden.net/wiki/Counter_(move)`, 2021.

Appendix

A. First Appendix Section

ein Bild

Figure A.1.: A figure

...

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.

Place, XX. Month 20YY

. . . .

(Julian Schubert)