7 Developer Guide

Dieses Kapitel soll die Nachvollziehbarkeit des erstellten Quellcodes vereinfachen und Design Entscheidungen erläutern.

7.1 Systemarchitektur

Im Rahmen der Projektarbeit sind verschiedene Agents, Features und Reward Systeme entstanden. Um die einzelnen Implementationen schnell anpassen oder austauschen zu können, sind die Funktionalitäten in separaten Klassen abgebildet. Das vollständige Klassendiagramm befindet sich im Anhang. Für eine ausführliche Dokumentation der einzelnen Methoden, deren Parameter und Rückgabewerte empfiehlt sich die Konsultation der Inline-Dokumentation. Die folgenden Unterkapitel beschreiben die Ansätze, welche mit den jeweiligen Komponenten adressiert werden.

7.1.1 MVQLearningPlayer

Die Klasse MVQLearningPlayer beinhaltet die erste implementierte Version von Q-Learning, welche als Minimum Viable Product auf AICrowd geladen wurde. Wie im Kapitel 3.3.6 beschrieben, wurde das einfache Q-Learning Modell, und somit auch diese Klasse, nach Abschluss der ersten Realisierungseinheit nicht weiterverfolgt.
Das Präfix MV hat keine technische Bedeutung. Es setzt sich aus den ersten Buchstaben der Vornamen der beiden Teammitglieder, Michael und Vito, zusammen und dient lediglich dem Zweck, ihr Selbstbewusstsein zu stärken.

7.1.2 MVDeepQPlayer

Die Klasse MVDeepQPlayer ist der direkte Nachfolger der Klasse MVQLearningPlayer und beinhaltet die erste implementierte Version von Deep Q-Learning.
Diese Klasse verwendet noch keine eigene Basisklasse. Sämtliche Player-Klassen, die nach der Klasse MVDeepQPlayer implementiert werden, basieren auf einer einheitlichen Basisklasse.

7.1.3 MVBasePokerPlayer

Die Klasse MVBasePokerPlayer dient als Elternklasse der verschiedenen Implementationen von Player-Klassen und leitet selbst von der gegebenen BasePokerPlayer Klasse ab.

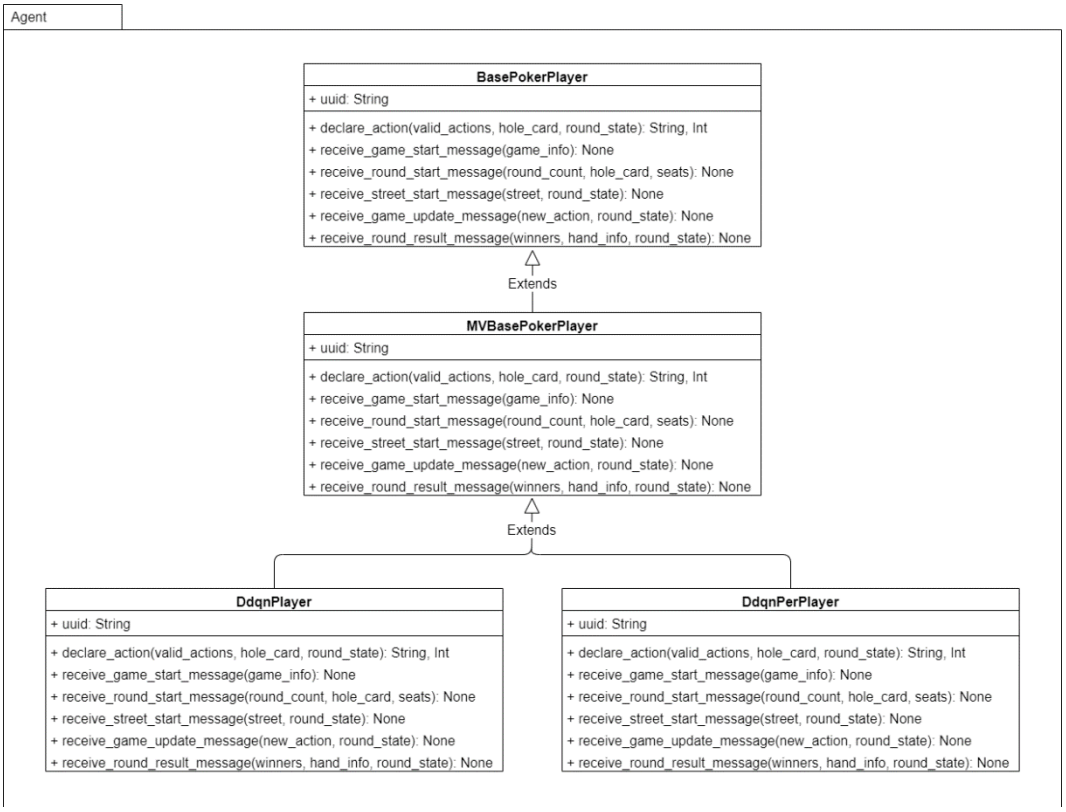

Abbildung 28 Klassendiagramm Agent Package

Abbildung 28 Klassendiagramm Agent Package zeigt diese Beziehungen. Um die Übersichtlichkeit zu gewährleisten, werden die zahlreichen Attribute der Player-Klassen weggelassen und es sind nur zwei Player-Klassen abgebildet.

Die Klasse MVBasePokerPlayer bildet die grundsätzliche Funktionalität der Player-Klassen ab. Dazu gehören das Implementieren der Methoden der Elternklasse BasePokerPlayer, das zur Verfügung stellen der Daten für den Input Layer des neuralen Netzwerks, das Erheben von Statistiken über das Vehalten der Gegner, das Bestimmen des Rewards für getroffene Entscheidungen etc.
Durch das Verwenden einer gemeinsamen Basisklasse wird die Verwendung des gleichen Feature Standes für sämtliche Player-Klassen gewährleistet. Durch die zentrale Verwaltung kann Zeit bei der Implementierung von Player-Klassen gespart werden.

7.1.4 MVDqnPlayer, MVDdqnPlayer, MVDdqnPerPlayer, MVDdqnKerasPlayer

Die vier Player-Klassen bilden jeweils eine erarbeitete Form von Deep Q-Learning ab und erben von der gemeinsamen Elternklasseklasse MVBasePokerPlayer.

Tabelle 16 Beziehung zwischen Player-Klassen und Q-Learning Modellen

Klasse	Q-Learning Form
MVDqnPlayer	Deep Q Learning
MVDdqnPlayer	Double Deep Q Learning
MVDdqnPerPlayer	Double Deep Q Learning mit Prioritized Experience Replay

MVDdqnKerasPlayer	Double Deep Q Learning, verwendet Keras Referenzimplementierung von Double Deep Q Learning

Tabelle 16 Beziehung zwischen Player-Klassen und Q-Learning Modellen zeigt das verwendete Q-Learning Modell für die jeweilige Klasse. Die Implementierung des jeweiligen Q-Learning Modells selbst befindet sich in der jeweiligen Modell-Klasse, siehe Kapitel 7.1.5.

7.1.5 DqnModel, DdqnModel, DdqnKerasModel, DdqnPerModel, DdqnPerSplitModel

Die Model-Klassen beinhalten jeweils eine Form von Q-Learning und werden durch die Player-Klassen eingebunden.

Tabelle 17 Beziehung zwischen Modell-Klassen und Q-Learning Modellen

Klasse	Q-Learning Form
DqnModel	Deep Q Learning
DdqnModel	Double Deep Q Learning
DdqnKerasModel	Double Deep Q Learning, Keras Referenzimplementierung
DdqnPerModel	Double Deep Q Learning mit Prioritized Experience Replay
DdqnPerSplitModel	Double Deep Q Learning mit Prioritized Experience Replay und separaten Netzwerken für den Preflop und die restlichen Streets

Tabelle 17 Beziehung zwischen Modell-Klassen und Q-Learning Modellen veranschaulicht, welche Klasse für die Implementierung eines spezifischen Q-Learning Modells verantwortlich ist.

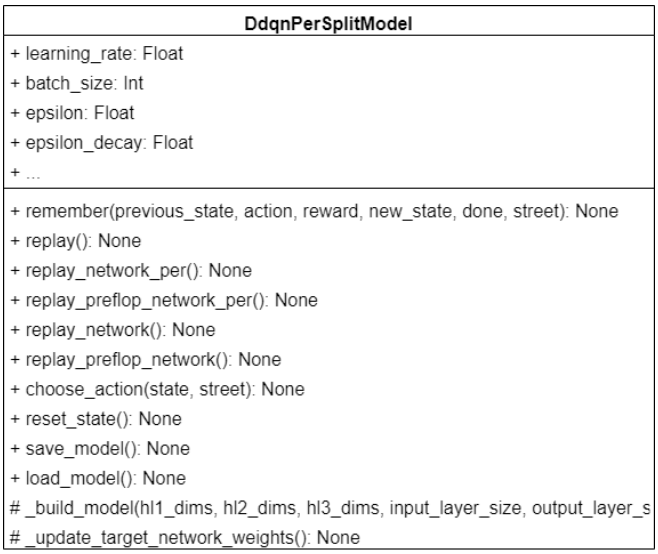

Abbildung 29 Klassendiagramm DdqnPerSplitModel

Abbildung 29 Klassendiagramm DdqnPerSplitModel zeigt beispielhaft die Methoden der Modell-Klassen. Die Klassen weisen zum Grossteil die gleichen Methoden auf, im

Gegensatz zu den Player-Klassen wurde jedoch auf eine gemeinsame Elternklasse verzichtet, da sich die Implementierungen der einzelnen Methoden unterscheiden. Lediglich die DdqnPerSplitModel-Klasse verfügt über zusätzliche Methoden, da die verschiedenen Netzwerke für den Preflop und die anderen Streets unterschiedlich bewirtschaftet werden müssen.

7.1.6 MemoryBuffer

Da der Reward für die getroffenen Aktionen einer Runde erst am Ende, genauer in der receive_round_result_message-Methode, bekannt wird, müssen sämtliche durch den eigenen Player getroffenen Entscheidungen zwischengespeichert werden. Diese Funktion wird durch die Klasse MemoryBuffer abgedeckt.

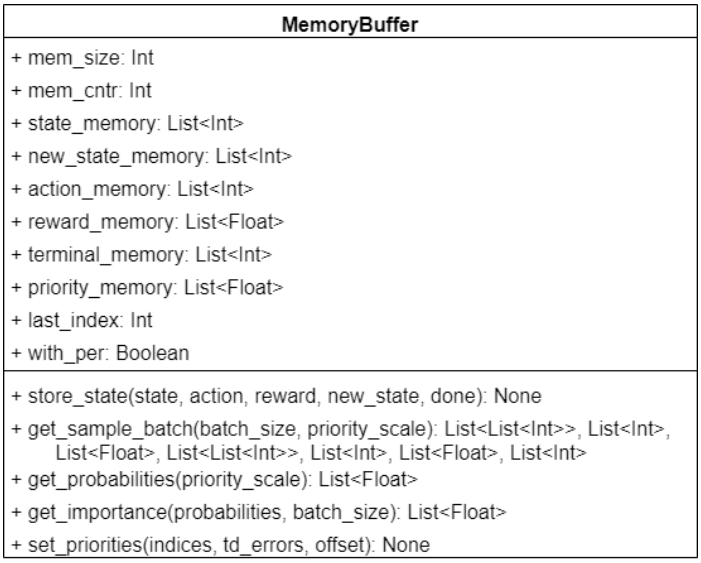

Abbildung 30 Klassendiagramm MemoryBuffer

Abbildung 30 Klassendiagramm MemoryBuffer zeigt die Eigenschaften und Methoden des Memory Buffers. Die gespeicherten Entscheidungen des eigenen Players und der dafür erhaltene Reward werden schlussendlich aus dem Memory Buffer in das neurale Netzwerk zurück gespiesen, um den Lernprozess zu ermöglichen.

7.1.7 Players

Die Klasse Players wird verwendet, um Informationen zu den Spielern am Tisch auszulesen.

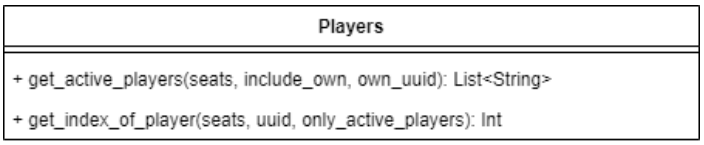

Abbildung 31 Klassendiagramm Players

Abbildung 31 Klassendiagramm Players zeigt die angebotenen Methoden, um die Plätze am Tisch nach aktiven Spielern oder deren Position zu durchsuchen.

7.1.8 Features

Die Klasse Features berechnet anhand der durch die PyPokerEngine zur Verfügung gestellten Informationen die Daten für den Input Layer des neuralen Netzwerks. Die jeweiligen Gedanken hinter den Features kann dem Kapitel 0 entnommen werden.

Abbildung 32 Klassendiagramm Features

Abbildung 32 Klassendiagramm Features zeigt anhand der zahlreichen Methoden die Features, welche im Rahmen des Projektes entstanden sind. Grundsätzlich ist die Klasse stateless gehalten, um einen möglichst hohen Grad der Entkoppelung von den Player-Klassen zu gewährleisten. Einzige Ausnahme bilden die Statistiken über das Verhalten der Gegner, welche als Dictionaries geführt und im Lauf einer Evaluation laufend aktualisiert werden.

7.1.9 HoldemCalc

Die Implementierung ist eine vereinfachte Version des Hold'em Calculators von Kevin Tseng [33]. Im Gegensatz zu der ursprünglichen Verwendungsweise werden in der angepassten Version lediglich die eigenen Hand- sowie die Gemeinschaftskarten verwendet. Die Handkarten der Gegner werden nicht berücksichtigt.

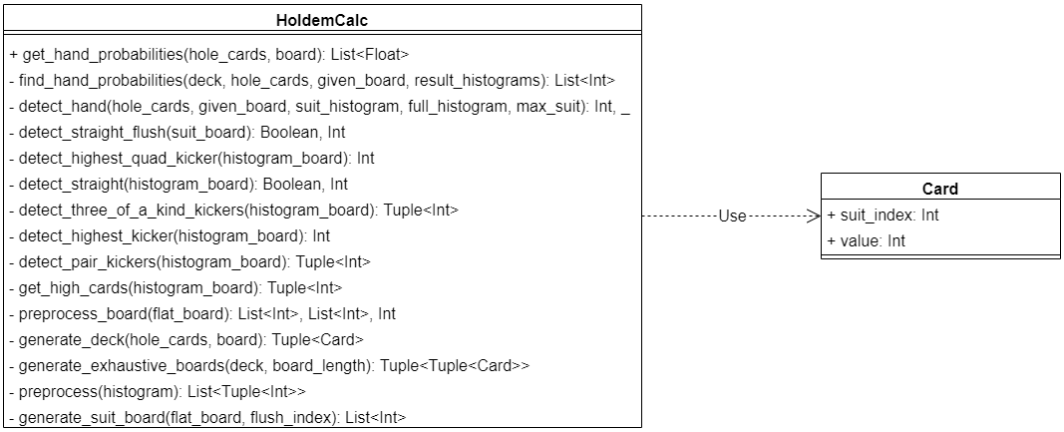

Abbildung 33 Klassendiagramm HoldemCalc

Abbildung 33 Klassendiagramm HoldemCalc zeigt die verwendete Struktur der Klasse. Die Karte Card repräsentiert eine einzelne Karte. Das Ergebnis der Methode *get_hand_probabilites* ist ein Histogramm, welches die Möglichkeiten, eine bestimmte Handstärke zu erhalten, aufzeigt.

```
Player1 Histogram:
High Card :  0.0
Pair :  0.518181818182
Two Pair :  0.384848484848
Three of a Kind :  0.0686868686869
Straight :  0.0
Flush :  0.0
Full House :  0.0272727272727
Four of a Kind :  0.0010101010101
Straight Flush :  0.0
Royal Flush :  0.0
```

Abbildung 34 Histogramm für mögliche Handstärken [33]

Abbildung 34 Histogramm für mögliche Handstärken zeigt beispielhaft ein solches Histogramm. Über die Wahrscheinlichkeiten im Histogramm kann die weitere Entwicklung der eigenen Handstärke beurteilt werden.

7.1.10 Rewards

Die erhaltenen Belohnungen für getroffene Entscheidungen sind ausschlaggebend für den Lernprozess des neuralen Netzwerkes. Das Implementieren von verschiedenen Methoden, welche jeweils ein Reward-Modell abbilden, ermöglicht sowohl das schnelle Austauschen als auch das Vergleichen der Belohnungsmodelle.

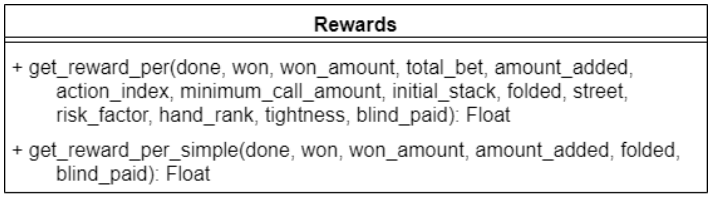

Abbildung 35 Klassendiagramm Rewards

Abbildung 35 Klassendiagramm Rewards zeigt die beiden Methoden, die derzeit im Einsatz sind. Obsolete Methoden sind im Klassendiagramm nicht mehr aufgeführt, in der Klasse jedoch noch enthalten.

7.1.11 Plots

Um das Verhalten der Agenten nachvollziehen zu können, sind grafische Auswertungen hilfreich. Die Klasse Plots bietet mehrere Formen dieser Auswertungen an.

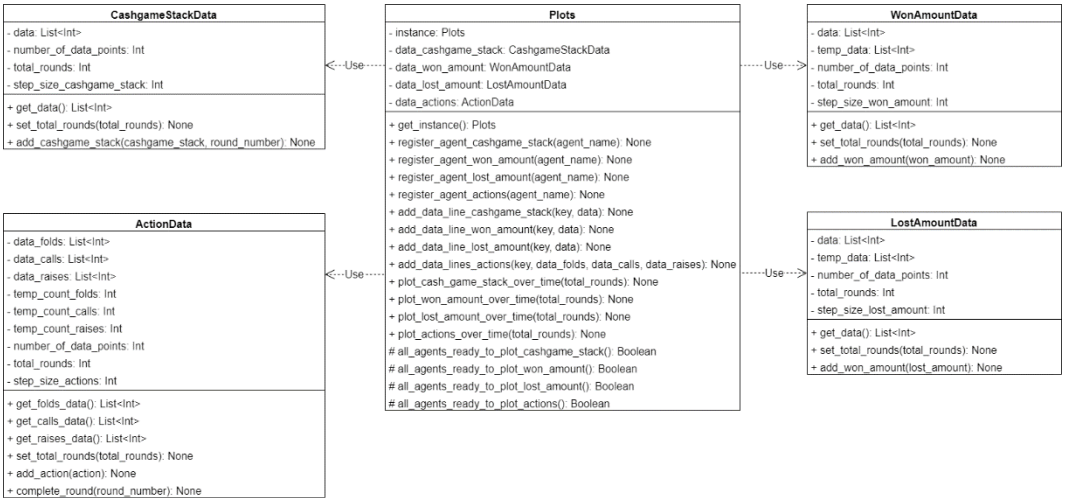

Abbildung 36 Klassendiagramm Plots

Abbildung 36 Klassendiagramm Plots zeigt die Klasse Plots und die verwendeten Data-Klassen.

Die Klasse Plots ist als Singleton implementiert. Dadurch ist gewährleistet, dass nur eine Instanz der Klasse existiert und sich mehrere Agenten registrieren und gemeinsame grafische Auswertungen erstellt werden können.

Über die register-Methoden kann sich ein Agent bei der Klasse anmelden. Die add-Methoden dienen dem Hinzufügen von Datenpunkten und über die plot-Methoden können die jeweiligen Diagramme erstellt werden. Da die Klasse von mehreren Agenten verwendet werden kann, stellen die Methoden all_agents_ready_to_plot_[...] sicher, dass sämtliche benötigten Datenpunkte zur Verfügung stehen.

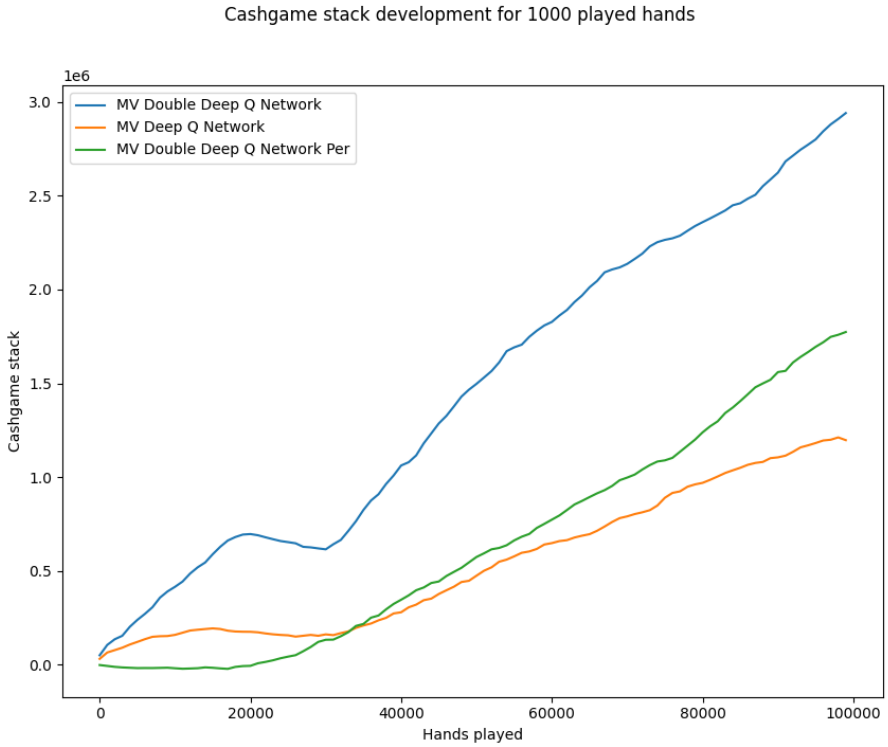

Abbildung 37 Entwicklung Cashgame Stack Plot

Abbildung 37 Entwicklung Cashgame Stack Plot zeigt einen beispielhaften Plot, an dem die Grösse des Cashgame Stacks für jeden teilnehmenden Agenten einer Evaluierung nach jeweils 1'000 gespielten Händen abgelesen werden kann.

7.1.12 CashgameStackData, WonAmountData, LostAmountData, ActionData

Die Data-Klassen beinhalten die Datenpunkte für die Plots. Sämtliche Data-Klassen bieten jeweils Methoden an, um Daten hinzuzufügen und die Anzahl zu spielender Runden zu definieren. Über die Attribute werden die anzuzeigenden Daten gespeichert.

7.1.13 Fazit

Das implementierte System erlaubt das Erstellen von neuen Agenten in einer Weise, die an einen morphologischen Kasten [42] erinnert. Im Gegensatz zu einem herkömmlichen morphologischen Kasten ist in diesem Fall eine Mehrfachauswahl des Parameters Features möglich.

Tabelle 18 Morphologischer Kasten

Parameter	Ausprägung				
Agent	Dqn	Ddqn	DdqnPer	DdqnKeras	...
Features	Pot-Odds	Street	Position	Hand Rank	...
Reward	Simple	Complex			

Tabelle 18 Morphologischer Kasten zeigt ein Beispiel, in welchem ein Double Deep Q-Learning mit Prioritized Experience Replay-Agent verwendet wird. Dieser basiert auf den Features Pot-Odds, Position, Hand Rank und berechnet den Reward über das Complex Belohnungssystem.
Dank der implementierten Systemarchitektur kann innerhalb von Minuten ein neuer Agent erstellt werden. Zudem erlauben die Plots und die Statistiken, die per Mail zugesandt werden, eine Aussage über das Verhalten des eigenen Agenten oder gar die Agenten der Gegner.
Die verschiedenen Modelle können trainiert und verglichen werden, um den idealen Kandidaten für eine Einreichung auf AICrowd zu evaluieren.