Evaluation und Erweiterung von Deep Reinforcement Learning Agenten für den algorithmischen Handel

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science

vorgelegt von

Luis Kaiser
Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik X (Data Science Chair)
Prof. Dr. Andreas Hotho

Lehrstuhl für Informatik X

der Julius-Maximilians-Universität Würzburg

Bachelorarbeit in Wirtschaftsmathematik

Evaluation und Erweiterung von Deep Reinforcement Learning Agenten für den algorithmischen Handel

Evaluation and Improvement of Deep Reinforcement Learning Agents for Algorithmic Trading

Bearbeiter:	Luis Kaiser
Prüfer:	Prof. Dr. Andreas Hotho
Betreuer:	Padraig Davidson (M.Sc.)
	Julian Tritscher (M.Sc.)
Abgabe:	22. Februar 2021

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Würzburg, 22.02.2021 Luis Kaiser

Abstract

In den letzten Jahren haben sich verschiedene Varianten von deep reinforcement learning (DRL) zum state-of-the-art für das Erlernen von optimalen Strategien im algorithmischen Handel entwickelt. Dies hat zu einem großen Interesse im Finanzsektor geführt, die Anwendung und den Nutzen diverser bestehender DRL Implementierungen in komplexen und dynamischen Anlagemärkten detailliert zu evaluieren. In dieser Arbeit wird eine groß angelegte Analyse mit fünf DRL Varianten in einem realitätsnahen Evaluationssetting auf einem umfassenden Datensatz durchgeführt, um deren Performance in unterschiedlichen Marktsituationen für 70 Wertpapiere ausführlich zu untersuchen. Die Hyperparameter für jede Komposition aus DRL Variante, Wertpapier und Trainingsdurchlauf werden separat mit einer Rastersuche optimiert und deren Einfluss auf den sequentiellen Entscheidungsprozess analysiert. Insgesamt werden die Ergebnisse von 2400 Experimenten ($\approx$ 82.000 CPU-Kern-Stunden) zusammengefasst, wodurch neue Erkenntnisse über die Performance von DRL Ansätzen im algorithmischen Handel gewonnen werden können. Die Ergebnisse zeigen, dass keine der Varianten eine signifikant bessere Performance als der standardmäßige deep q learning (DQL) Algorithmus erzielt und charakterisieren die Long Short-Term Memory (LSTM) Architektur als wichtige Eigenschaft in volatilen Marktphasen. Erweiterungen der Architektur einiger DRL Algorithmen, wie beispielsweise die Verwendung von zusätzlichen Marktinformationen, führen bei den verwendeten Modellen zu keiner signifikant besseren Entscheidungsfindung. Gleichmäßige Zeitreihen erweisen sich als entscheidender Faktor für eine gute Performance und zeigen eine Wechselwirkung mit dem Marktrisiko eines Wertpapiers. Weiterhin lassen sich aus den Effekten der studierten Hyperparameter Schlussfolgerungen für deren effiziente Einstellung bei unterschiedlichen Konjunkturphasen ziehen.

Schlüsselwörter: Algorithmic Trading · deep reinforcement learning · Evaluation

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DDPG	deep deterministic policy gradient
DDPGA	deep deterministic policy gradient agent
DDPGAK	deep deterministic policy gradient agent mit Kontextdaten
DDQL	duel deep q learning
DDRQLA	duel deep recurrent q learning agent
DQL	deep q learning
DQLA	deep q learning agent
DRL	deep reinforcement learning
EDQLA	erweiterter deep q learning agent
ETF	Exchange Traded Funds
LSTM	Long Short-Term Memory
MA	moving average
MDP	Markow-Entscheidungsproblem
ML	machine learning
NN	neuronales Netz
RL	reinforcement learning
RNN	rekurrente neuronale Netze
TT	turtle trading
^VIX	Volatilitätsindex des US-amerikanischen Aktienindexes

1 Einleitung

Jüngste Fortschritte in deep reinforcement learning (DRL) durchbrechen vorherrschende Konzepte, indem die Technologie aus vielschichtigen Rohdaten komplexe Muster erkennt, was bereits zu Disruption in diversen Domänen wie dem Meistern komplexer Spiele [48, 37] führte. Diese Erfolge bestärken die Erwartung, dass DRL auch bei multi-dimensionalen Problemstellungen wie dem algorithmischen Handel profitabel eingesetzt werden kann und, entgegen vorherrschender Wirtschaftstheoreme, langfristig Gewinne mit geringem Risiko generiert.

Jedoch stellt die Finanzumgebung einige Herausforderungen für den reinforcement learning (RL) Entscheidungsprozess dar: Zum einen enthalten die Daten Sprünge, Ausreißer und Anomalien, die teilweise nicht auf rationale Zusammenhänge zurückzuführen sind und die Vorhersagbarkeit der Wertentwicklung negativ beeinflussen. Zum anderen ist der Markt nicht vollständig beobachtbar und makroökonomische Ereignisse können Marktcharakteristiken plötzlich verändern, sodass die gelernte Strategie neu angepasst werden muss. Stationäre Indikatoren wie der gleitende Durchschnitt unterstützen die DRL Agenten nur bedingt, da deren Aussagekraft in Abschwungphasen gering ist. Folglich ist das Ziel moderner Forschung, wiederkehrende Muster direkt aus historischen Kursverläufen eines Wertpapiers gewinnbringend auszunutzen und flexibel auf kontradiktorische Marktprozesse zu reagieren.

Problem. Während einige Ansätze [65, 39, 59] dabei vielversprechende Ergebnisse liefern, gibt es kein etabliertes Setting, um die Agenten detailliert zu evaluieren und langfristige Analysen zu ermöglichen. In der Literatur wird hauptsächlich das Design der Agenten diskutiert [56, 9], allerdings lassen die Ergebnisse keine konsistente Aussage über die tatsächliche Performance der Implementierungen am Wertpapiermarkt zu. Insbesondere verwenden viele Studien [56, 23, 66, 60] zu ähnliche Wertpapiere, da vorwiegend Aktien von marktführenden Unternehmen aus der Technologiebranche analysiert werden. Wertpapiere mit anderen Charakteristika werden hingegen außer Acht gelassen, weshalb repräsentative Schlussfolgerungen nur eingeschränkt möglich sind. Speziell Aktien aus der Technologiebranche wiesen in den letzten Jahren stark steigende Kursverläufe auf, sodass der Handel mit diesen Anlagen oft zwangsläufig zu Gewinnen führt. Dieser Sachverhalt ist für eine realistische Einschätzung der Performance problematisch. Weiterhin variiert die Optimierung der Agenten stark (vgl. [54, 53, 23]) und die Marktbedingungen sind häufig nicht vergleichbar, da beispielsweise Transaktionskosten [66, 56] vernachlässigt werden.

Ziel. Ein zentraler Punkt des Forschungsinteresses ist deshalb, eine einheitliche Testumgebung für die Evaluation von DRL-Agenten zu entwickeln, mit der die Performance der Agenten unter realistischeren Bedingungen zuverlässig bewertet werden kann. Mithilfe dieses Frameworks soll nachfolgend evaluiert werden, welche Implementierung sich am besten eignet, um mit Hilfe von DRL eine maximale Rendite am Aktienmarkt zu erzielen.

Diese Arbeit leistet einen ersten Beitrag zur differenzierten Analyse der Performance von DRL Algorithmen, indem eine umfassende Testumgebung zur realitätsnahen Evaluation unterschiedlicher Marktgeschehnisse, Wertpapiere und Settings entwickelt wird.

Herangehensweise und Beitrag. Diese Testumgebung umfasst (i) einen Datensatz mit 70 Wertpapieren aus diversen Sektoren mit differierenden Charakteristika, (ii) Evaluationsmetriken, (iii) einheitliche Marktbedingungen und (iv) explizit die Auswertung von Krisenzeiten. (v) Zusätzlich werden Standards in der Optimierung der Agenten aufgestellt, die unter anderem die Hyperparametersuche, das Training und das Testen spezifizieren. Für jedes Wertpapier trainiert und optimiert der Algorithmus seine Hyperparameter neu, um die besten Ergebnisse zu erzielen - wie ein realer Trader seinen Agenten einstellen würde. Trainings- und Testzeiträume sind einheitlich und lassen somit Branchen- oder Marktvergleiche zu. Es werden mehrere Jahre getrennt evaluiert, wodurch unterschiedliche Marktsituationen besser untersucht werden können.

Dieses Evaluationssetting wird verwendet, um die Stärken und Schwächen von vier bestehenden DRL Implementierungen [66, 53] differenziert zu analysieren und zu vergleichen. Dafür werden die Frameworks um ein umfangreiches Testsetting erweitert und darin mit unterschiedlichen Wertpapieren für mehrere Jahre evaluiert. Die ausgewählten Agenten sind DRL Varianten, die sich hinsichtlicher ihrer Architektur oder ihrer Eingabedaten um einen Aspekt unterscheiden. Effekte einzelner DRL Varianten und Umsetzungen können dadurch vielseitig herausgearbeitet werden. Anschließend wird die Architektur eines Agent so modifiziert, dass Einflüsse weiterer Designänderungen auf die Performance untersucht werden können. Insbesondere erhält der erweiterte Agent zusätzliche Eingabewerte, besitzt zudem einen größeren Aktionsraum und nutzt ferner eine Regularisierungsmethode, mit der die Wahrscheinlichkeit von Überanpassung verringert wird. Eine differenzierte Analyse zeigt, dass die rekurrente Variante des deep q learning (DQL) Algorithmus mit volatilen Wertpapieren die besten Ergebnisse liefert und die Modifikationen nur in wenigen Fällen die Performance steigern.

Die Bachelorarbeit ist wie folgt gegliedert: Zu Beginn wird eine kurze Übersicht über die verschiedenen Bereiche des algorithmischen Handels und dessen bedeutsame wissenschaftliche Publikationen gegeben. Abschnitt 3 enthält die theoretischen Grundlagen und behandelt formale Aspekte der Testumgebung. Im vierten Abschnitt wird der Datensatz, das Design der Agenten und das Evaluationssetting beschrieben. Die Ergebnisse der Evaluation werden in Abschnitt 5 diskutiert. Abschließend wird in Abschnitt 6 ein Fazit und ein Ausblick über zukünftige Arbeiten gegeben.

2 Bestehende Literatur

Der Begriff „algorithmischer Handel“ umfasst diverse Domänen. Dazu gehören unter anderem Sentimentanalysen [26], Hochfrequenzhandel [6] und Risikodiversifikation durch Portfoliooptimierung [41]. In modernen Studien wird die quantitative Analyse einer einzelnen Zeitreihe am häufigsten diskutiert. Weiterhin zeigen einige Metastudien [42, 45], dass DRL dabei eine bessere Vorhersagekraft als vergleichbare Methoden besitzt, weshalb diese Ansätze Gegenstand der Arbeit sind.

[38, 39] demonstriert erstmals die Vorteile von RL gegenüber herkömmlichen stochastischen oder supervised learning Verfahren zur Lösung des Markow-Entscheidungsprozesses (MDP). Moody et al. [38, 39] haben Muster im S&P 500 Stock Index nachgewiesen, welche von rekurrenten RL Algorithmen am besten ausgenutzen werden können. Die Überlegenheit dieser Algorithmen wird am deutlichsten, wenn dem Modell Transaktionskosten hinzugefügt werden, da rekurrente RL Ansätze ihre Trainingsfrequenz in diesem Fall am stärksten reduzieren. In diesem Kontext wird auch das Sharpe Ratio eingeführt, welches sich aus dem Verhältnis von Profit zu Risiko ergibt. Wird das Sharpe Ratio optimiert, repräsentiert der Agent die Risikoaversion der meisten Investoren. Die Metrik und der RL Ansatz werden seither vorrangig in der Literatur verwendet.

Erste DRL Agenten für den Wertpapierhandel werden in [56] entwickelt. Der vorgestellte Algorithmus basiert auf q learning und kann rekurrente RL Ansätze beim Handeln mit einigen Indices übertreffen. Anstatt den Markt abzubilden, konzentriert sich q learning darauf, den Nutzen der Aktion zu optimieren. Fehler, die das Marktmodell verursacht, können dadurch vermieden werden. Jedoch kann in diesem Setting die Performance für Aktien, Warentermingeschäfte oder Exchange Traded Funds[1] (ETF) nicht bestimmt werden. Ferner vergleicht [59] verschiedene Erweiterungen von DQL wie beispielsweise eine Dueling Strategie (cf. [57]). Unter den Architekturen schneidet der DQL vor Double DQL und Dueling DQL beim Handel mit zehn amerikanischen Aktien am besten ab. Wenngleich die erweiterte Architektur beim Meistern von Spielen bessere Ergebnisse liefert [19], zeigt sie in diesem MDP keine bessere Entscheidungsfähigkeit. Allerdings wird eine differenziertere Auswertung, z.B. verschiedener Branchen oder Märkte, nicht durchgeführt. Außerdem weichen die Zeiträume der Aktien stark voneinander ab, weshalb keine konsistente Evaluation möglich ist.

Differenzierter angelegte Evaluationen von DRL-Agenten, welche Wertpapiere aus diversen Branchen bzw. Märkten untersuchen, lassen einige Erweiterungen aus, die in dieser Arbeit genauer betrachtet werden. Beispielsweise analysiert [54] nur ähnliche Wertpapiere und führt keine Hyperparametersuche durch. Obwohl die Studie die Mängel anderer wissenschaftlicher Evaluationen teilweise berücksichtigt, enthält die Testumgebung zum einen nur Aktien marktführender Unternehmen, unterscheidet nicht zwischen einzelnen Märkten und besitzt

[1] Börsengehandelter Indexfonds, der die Wertentwicklung von Aktienlisten abbildet.

keine ETFs oder Warentermingeschäfte. Darüber hinaus hat eine fehlende Optimierung der Hyperparameter zur Folge, dass das Modell lediglich mit den vordefinierten Werten am besten abschneidet. Ohne eine individuelle Feinabstimmung der Parameter können Performancevergleiche mit anderen DRL Varianten in diesem Framework nicht durchgeführt werden. Der Datensatz von [65] begrenzt sich auf sehr liquide Termingeschäfte und bildet Durchschnitte einzelner Branchen, ohne die Ergebnisse genauer zu erläutern. Dabei weist unter verschiedenen DRL Ansätzen der DQL Algorithmus vor der deep policy gradient und deep deterministic policy gradient (DDPG) Methode die beste Performance auf. Während in diesem Evaluationssetting unterschiedliche Klassen an Wertpapieren verglichen werden können, fehlt eine Analyse hinsichtlich Branchen und hinsichtlich diverser Charakteristika in den Kursverläufen. Zusätzlich klammern alle Arbeiten Krisenzeiten aus und testen ihre Agenten nur auf Datensätzen mit geringer Volatilität sowie steigenden Kursverläufen. Zuverlässige Aussagen über die tatsächliche Performance der Agenten können daher ebenfalls nicht getroffen werden.

Baily et al. [3] kritisieren die wissenschaftliche Herangehensweise der bestehenden Literatur. Aufgrund der Auswahl zu geringer Stichproben und des fehlenden mathematischen Formalismus sind viele machine learning (ML) Modelle im Finanzsektor überangepasst, womit das Scheitern von algorithmischen Hedgefonds erklärt werden kann. Überanpassung, besonders bei Daten mit großem Hintergrundrauschen, führt laut Baily et al. dazu, dass Modelle bei out-of-sample Wertpapieren deutlich schlechter abschneiden.

3 Grundlagen

Unter mehreren ML Ansätzen treibt speziell DRL den state-of-the-art im Bereich des algorithmischen Handels voran. Wie in [2] hervorgehoben, unterliegt RL der Einschränkung, dass komplexe Probleme aufgrund fehlender Skalierbarkeit und Rechnerkapazität nicht effizient gelöst werden können. DRL überwindet diese Hindernisse, indem es die Entscheidungsfindung von RL mit der Fähigkeit von neuronalen Netzen (NN), eigenständig einfache Darstellungen vielschichtiger Daten zu ermitteln, verbindet.

Das Schema des DRL Prozesses beim algorithmischen Handel ist in Abbildung 3.1 veranschaulicht. Es besteht aus einem Agenten, der in Form von Sequenzen aus Zustand s, Aktion a und Belohnung r mit der Finanzumgebung interagiert. Der Agent enthält ein NN, durch das profitable Strategien $\pi(s, a, \theta)$ direkt aus multi-dimensionalen Rohdaten effizient erlernt werden können. NNe (cf. [14]) bestehen aus einer Eingabe- und Ausgabeschicht sowie einer oder mehreren verdeckten Schichten. Jede Schicht enthält künstliche Neuronen, die über gewichtete Kanten, parametrisiert durch θ, mit Neuronen aus anderen Schichten verbunden sind. Im Rahmen dieser Arbeit werden NNe mit Varianten des q learning Algorithmus trainiert, wobei der Adam Optimierer zur Aktualisierung der Gewichte θ verwendet wird. Die meisten unten stehenden Formeln sind aus [37, 29] entnommen.

3.1 Formalisierung des Markow-Entscheidungsproblems

RL erlaubt NNen sequentielle Entscheidungsprobleme durch die Ausführung von Aktionen in einer teilweise beobachtbaren Umgebung zu lösen. Formal befindet sich der Agent beim algorithmischen Handel in einem MDP, der die Entscheidung des RL Algorithmus modelliert. Dessen Kernbausteine können für jeden Zeitschritt t folgendermaßen definiert werden:

Zustand. Der Agent erzeugt eine Abstraktion der Finanzumgebung, bezeichnet als Zustand $s \in \mathcal{S}_t$, wobei $\mathcal{S}_t$ die Potenzmenge aller beobachtbaren Informationen zum Zeitschritt t darstellt und als Zustandsraum bezeichnet wird. Neben historischen Kursverläufen des gehandelten Wertpapiers können beispielsweise weitere, korrelierende Wertpapierdaten unterstützen, qualitative bzw. quantitative Ursachen für Marktveränderungen miteinzubeziehen. Die Herausforderung besteht darin, aus der multi-dimensionalen Finanzumgebung nützliche Informationen so zu abstrahieren, dass damit maximale Gewinne erzielt werden.

Aktion. Basierend auf einer Strategie π wählt der Agent unter dem Zustand s eine Aktion a, kurz $a : \mathcal{S}_t \to \mathcal{A}_t$. Der Aktionsraum $\mathcal{A}_t$ ist diskret, abhängig vom Portfolio und umfasst den Kauf bzw. Verkauf einer oder mehrerer Wertpapiere zum Zeitschritt t. Der Aktionsraum der ausgewählten Modelle beschränkt sich auf $a \in \{-1, 0, 1\} = \{$verkaufen, halten, kaufen$\}$, wobei Leerverkäufe nicht möglich sind und für $a \neq 0$ Transaktionskosten

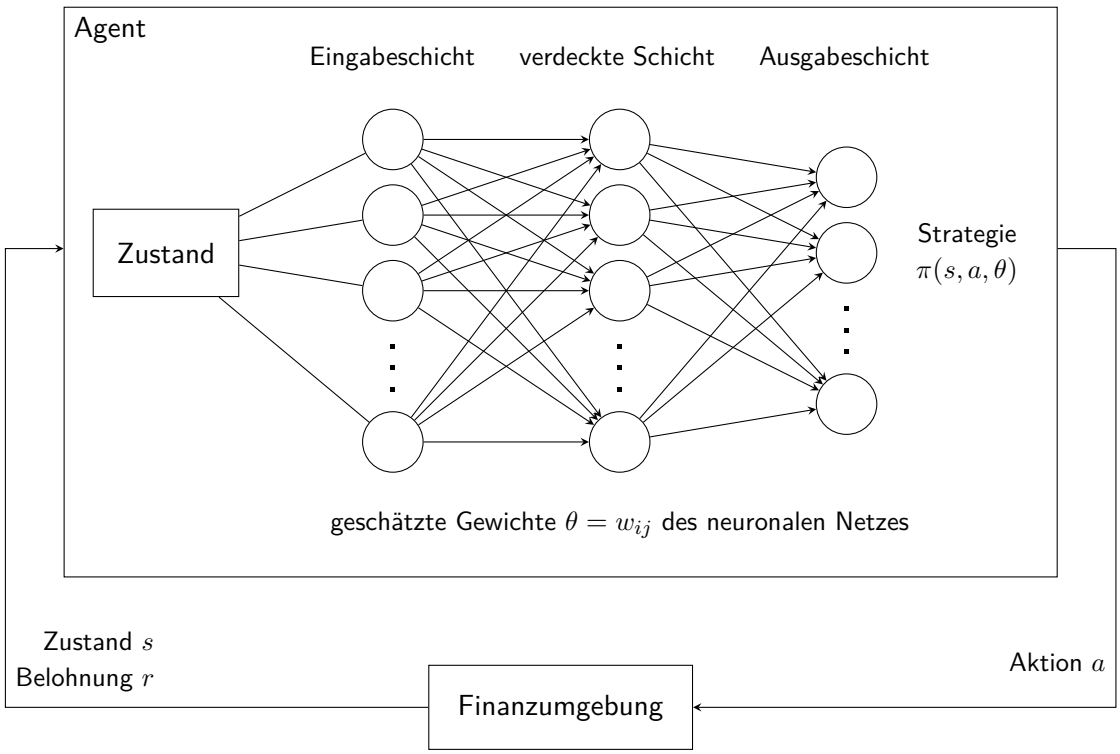

Abbildung 3.1: Kernbausteine des DRL Algorithmus. Angelehnt an [1]. In diesem Fall besteht das NN neben einer Eingabe- und Ausgabeschicht aus einer verdeckten Schicht, zwischen denen sich das Signal vorwärts gerichtet bewegt. In jedem Neuron wird der Eingabevektor mit Gewichten w_{ij} multipliziert, mit dem Verzerrungswert verrechnet und einer nicht-linearen Aktivierungsfunktion übergeben. Das Ergebnis wird zur nächsten Schicht weitergeleitet oder ausgegeben. Verwendete Abwandlungen werden in Abschnitt 3.3 vorgestellt.

anfallen.

Belohnung. Nach der Ausführung der Aktion a erhält der Agent für den Übergang von Zustand s zu Zustand s' eine Belohnung $r : \mathcal{S}_t \times \mathcal{A}_t \times \mathcal{S}_t \to \mathbb{R}$. Diese Belohnungsfunktion spezifiziert das Lernziel des Agenten. Für das algorithmische Handelsproblem ist der tägliche Ertrag des Portfolios, wie in [65, 54] beschrieben, die einfachste und profitabelste Strategie und wird daher als Belohnung in der Evaluation verwendet. Weitere Metriken folgen in Abschnitt 4.5. Das Portfolio eines Agenten besteht aus dem Gesamtwert der gehaltenen Wertpapiere und Bargeld. Zu Beginn besitzt der Agent einen fixen Betrag an Bargeld, mit dem er jeden Tag ein Wertpapier gemäß dem definierten Aktionsraum handeln kann.

Das Ziel von RL ist, eine optimale Strategie π^* zu finden, die für jeden Zeitschritt t den erwarteten diskontierten, kumulativen Gewinn $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, maximiert, wobei T den letzten betrachteten Zeitschritt darstellt. Der Diskontierungsfaktor $\gamma \in [0, 1]$ gewichtet zukünftige Gewinne; je größer der Diskontierungsfaktor ($\gamma \to 1$), desto wichtiger sind spätere Ereignisse für die aktuelle Aktion.

3.2 Deep q learning

Die Grundidee hinter q learning ist, die optimale Strategie π^* durch eine Funktion $Q^*(s', a')$ abzuschätzen, die den erwarteten Gewinn einer Aktion a in einem Zustand s optimiert. Die optimale Q-Funktion erfüllt das „Optimalitätsprinzip von Bellman“, Gleichung 3.1, welche den erwarteten Wert von $r + \gamma Q^*(s', a')$ unter Auswahl der optimalen Aktion maximiert,

$$\begin{aligned} Q^*(s, a) &= \max_{\pi} \mathbb{E}[R_t \mid s_t = s, a_t = a, \pi] \\ &= \mathbb{E}[r + \gamma \max_{a'} Q^*(s', a') \mid s, a]. \end{aligned} \tag{3.1}$$

Im Falle von DQL approximieren die Gewichte θ eines NNes die optimale Q-Funktion $Q(s, a, \theta) \approx Q^*(s, a)$. Indem die Bellmann Gleichung auf eine Sequenz aus Verlustfunktionen $L_i(\theta_i)$, genauer die Differenz der linken und rechten Seite der Gleichung 3.1, für jede Iteration i angewendet wird, kann ein DQL Algorithmus trainiert werden,

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \rho(\cdot)}[(y_i - Q(s, a, \theta_i))^2], \tag{3.2}$$

wobei $y_i = \mathbb{E}[r + \gamma \max_{a'} Q(s', a', \theta_{i-1}), a]$ die Vorhersage für Iteration i mit den Gewichten der vorherigen Iteration θ_{i-1} ist. $\rho(s, a)$ stellt die Wahrscheinlichkeitsverteilung von Sequenzen (s, a, r, s') dar, die in der Praxis häufig mit Hilfe einer ϵ-greedy Strategie berechnet wird. Bei dieser Strategie wählt der Agent mit der Wahrscheinlichkeit ϵ zufällig eine Aktion aus, wohingegen mit der Wahrscheinlichkeit $1 - \epsilon$ die optimale Aktion entsprechend dem DQL Verfahren ausgesucht wird. Für eine bessere Konvergenz startet ϵ gewöhnlich bei 1 und nimmt negativ exponentiell mit einem Faktor, referenziert als epsilon decay, ab.

Die Verlustfunktion wird durch den Gradienten bzgl. θ,

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)}[(y_i - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s, a, \theta_i)], \tag{3.3}$$

minimiert. Durch die Approximation $\nabla_{\theta_i} L_i(\theta_i) \approx (y_i - Q(s, a, \theta_i)) \nabla_{\theta_i} Q(s, a, \theta_i))$ lässt sich die optimale Verlustfunktion mit einem numerischen Verfahren [17], z.B. dem Gradientenverfahren, effizient annähern:

$$\theta_{i+1} = \theta_i + \alpha(y_i - Q(s, a, \theta_i)) \nabla_{\theta_i} Q(s, a, \theta_i)). \tag{3.4}$$

Die Lernrate α definiert wie stark die Anpassung der Gewichte des Agenten für jeden Zeitschritt sind. Eine hohe Lernrate beschleunigt den Lernprozess, kann aber zu einer suboptimalen finalen Einstellung der Gewichte oder Divergenz führen, da Minima leicht übersprungen werden. Mit Hilfe des Backpropagations-Algorithmus wird das Ausmaß des Fehlers jedes Neurons $\Delta w_{ij} = -\alpha \frac{\partial L}{\partial w_{ij}}$ berechnet und der Fehler von einer Schicht zur nächsten rückwärts weitergeleitet. Anschließend werden die Gewichte aktualisiert, sodass sich der Algorithmus iterativ einem lokalen Minimum des Fehlers nähert. Der Adam Optimierer ist eine Erweiterung des Gradientenverfahrens, der sich bei diversen Anwendungen als vorteilhaft erwiesen hat [27].

Um Korrelationen bei aufeinanderfolgenden Stichproben zu vermeiden, werden bei jedem Zeitschritt t die Erfahrungen des Agenten (s, a, r, s') in einem Wiedergabespeicher gesammelt [33], aus dem zufällig Untermengen zur Anpassung der Gewichte ausgewählt werden. Der vollständige Algorithmus mit Wiedergabespeicher ist in Algorithmus 1 abgebildet (vgl. [37]) und wird im Folgenden als DQL bezeichnet.

3.3 Varianten von deep q learning

Die in dieser Arbeit vorgestellten Anwendungen verwenden eine Reihe an DRL Architekturen, insbesondere rekurrente NN (RNN), Duel DQL (DDQL) sowie einen DDPG, die unten stehend näher erläutert werden.

3.3.1 Deep recurrent q learning

Seit einigen Jahren haben sich RNN mit Long Short-Term Memory (LSTM) [15] zum vorherrschenden Lernmodell für sequentielle Daten in Bereichen wie der Übersetzung [34, 16] oder der Sprachmodellierung [52, 36] entwickelt. In [20] führen Hausknecht und Stone den deep recurrent q learning Algorithmus mit der Intention ein, Agenten in teilweise beobachtbaren Umgebungen bessere Entscheidungen treffen zu lassen. Die Finanzbranche, bei der Millionen von individuellen Privatanlegern, Institutionen und Algorithmen gleichzeitig interagieren, zählt unter anderem zu dieser Kategorie. Indem der verdeckte Zustand des Agenten h_{t-1} dem Q-Wert übergeben wird ($Q(s, h_{t-1}, a, \theta)$), können vergangene Daten

Algorithm 1 Deep q learning Algorithmus mit Wiedergabespeicher (aus [37])

Initialisiere den Wiedergabespeicher M mit Größe m.
Initialisiere die Q-Funktion mit Ausgangsgewichten θ.
for epsiode=1 to E **do**
  Abstrahiere den initialen Zustand s der Umgebung und verarbeite ihn $\phi = \phi(s)$.
  for t=1 to T **do**
    Wähle mit Wahrscheinlichkeit ϵ eine zufällige Aktion $a \in \mathcal{A}_t$,
    oder wähle $a = \max_{a \in \mathcal{A}_t} Q(\phi(s), a, \theta_t)$.
    Führe die Aktion a_t aus; erhalte einen neuen Zustand s' und eine Belohnung r.
    Speichere die Sequenz ($\phi(s)$,a,r,$\phi(s')$) in M.
    Überschreibe den ältesten Eintrag, falls $|M| > m$.
    Wähle eine zufällige Untermenge $N \subset M$ aus Sequenzen $(\phi(s), a, r, \phi(s'))$.
    Setze $y_i = \begin{cases} r_i, & \text{falls } t+1 = T \\ r_i + \gamma \max_{a'} Q(\phi(s'), a', \theta_t), & \text{sonst.} \end{cases}$
    Wende ein numerisches Verfahren, z.B. Gl. 3.4, auf $(y_i - Q(\phi(s), a, \theta_t))^2$ an.
    Setze $s = s'$.
  end for
end for

den Lernprozess langfristiger beeinflussen und so der unvollständigen Wahrnehmung des Agenten entgegenwirken.

Konkret wird die DQL Architektur rekurrent, indem Neuronen aus verdeckten Schichten mit LSTM Zellen ersetzt werden. Im Vergleich zu vorwärtsgerichteten NN, wie in Abbildung 3.1, besteht diese LSTM Schicht aus rekurrenten Unternetzen, in denen Informationen von Signalen enthalten sind, welche bereits die Schicht durchquerten. Gewöhnliche RNN unterliegen dem „Problem des verschwindenden Gradienten“ [22]. Der Einfluss früherer Eingaben auf die verdeckte Schicht nimmt bei einfachen RNN schnell ab, während LSTM Schichten wiederkehrende Muster im Kursverlauf eines Wertpapiers besser speichern und abrufen können. Beim algorithmischen Handel haben LSTM Schichten somit die überlegene Fähigkeit, Sequenzinformationen über die Zeit zu erhalten und einen langfristigeren Lernhorizont zu ermöglichen.

3.3.2 Duel deep q learning

Die Dueling Strategie wird von Wang et al. [57] eingeführt. Die Architektur trennt explizit die Repräsentation des Zustandes $V(s,\theta) = \mathbb{E}_{a\sim\pi(s)}(Q(s,a,\theta))$ vom zustandsabhängigen Aktionsvorteil $A(s,a,\theta) = Q(s,a,\theta) - V(s,a,\theta)$. Diese Architektur berücksichtigt, dass bei manchen Entscheidungsproblemen nicht der Q-Wert jeder Aktion für jeden Zeitschritt t relevant ist. Indem man die zwei Schätzer durch eine Abspaltung im NN voneinander trennt, kann der dueling Ansatz lernen, welche Zustände wichtig sind (bzw. welche nicht), ohne den Effekt jeder Aktion für jeden Zustand zu lernen.

Die beiden Pfade ergeben den finalen Q-Wert gemäß der Definition des Aktionsvorteils $Q(s,a,\theta,\alpha,\beta) = V(s,\theta,\beta) + A(s,a,\theta,\alpha)$, wobei α und β die Parameter der getrennten Schichten darstellen. Da sich bei gegebenem Q-Wert V und A nicht bestimmen lassen, wird in der Praxis häufig der Aktionsvorteil umgeschrieben,

$$Q(s,a,\theta,\alpha,\beta) = V(s,\theta,\beta) + [A(s,a,\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a',\theta,\alpha)]. \tag{3.5}$$

Für den algorithmischen Handel ist dieser Ansatz interessant, da Aktionen nur marginalen Einfluss auf die Finanzumgebung haben.

3.3.3 Deep Deterministic Policy Gradient

[32] präsentiert erstmals den DDPG Algorithmus, der den actor-critic Algorithmus, eine Variante von RL, mit NNen verbindet. Das am häufigsten verwendete Actor-Critic Modell wurde ursprünglich von Konda und Tsitsiklis [28] beschrieben und enthält zwei Systeme: Basierend auf einer Strategie führt der Akteur Aktionen aus, während der Kritiker die Entscheidung bzgl. des vorliegenden Zustandes in Form einer Wertefunktion, z.B. der Q-Funktion, bewertet. Im Fall von DDPG werden der Akteur und der Kritiker jeweils unter Verwendung eines NNes modelliert, das die Aktionswahrscheinlichkeiten bzw. den Kritikerwert erzeugt. Für eine bessere Stabilität im Training enthält das Modell zeitverzögerte

Kopien des Akteur- und Kritiker-Netzwerks, die langsam die gelernten Netzwerke nachverfolgen. Die Gewichte dieser Netzwerke werden iterativ angepasst, $\theta' = \tau\theta + (1-\tau)\theta'$ mit $\tau < 1$. Anpassungen basieren auf der Fehlerapproximation des Kritikers, die zum einen zur Aktualisierung der Q-Funktion durch das Optimalitätsprinzip von Bellmann (Gl. 3.1) verwendet werden. Zum anderen wird die Information zum Akteur gesendet, um die Strategie $\pi(s, \theta)$ mit einem numerischen Verfahren gemäß $\max_\theta \mathbb{E}_{s\sim\rho}[Q(s, \pi(s, \theta), \theta_Q)]$ zu optimieren.

4 Evaluationsaufbau

Der Fokus dieser Arbeit ist, eine ausführliche Testumgebung zu entwickeln, mit der verschiedene DRL Varianten empirisch verglichen werden können. Aus diesem Grund sind die Experimente so entworfen, dass der Aufbau simpel, die Durchführung realitätsnah und Vergleiche fair sind. Ein zentraler Aspekt der Testumgebung ist der differenzierte Datensatz, der Wertpapiere in diverse Sektoren klassifiziert, um domänenübergreifende Eigenschaften zu analysieren. Für eine faire Evaluation muss weiterhin die Konfiguration der einzelnen DRL Varianten ähnlich sein, da unterschiedliche Varianten unterschiedliche Voreinstellungen benötigen. Aus diesem Grund werden Einstellungen wie Lernrate oder Schichtgröße individuell für jeden Agenten und jedes Wertpapier eingestellt. Diese Einstellungen werden vor dem Training des Algorithmus festgelegt, kontrollieren den Lernprozess und werden als Hyperparameter bezeichnet. Zur Vereinfachung beschränkt sich diese Arbeit auf eine Rastersuche. Dabei werden aus einer Menge an Kandidaten alle möglichen Kombinationen für jede Variante manuell evaluiert und die profitabelste Hyperparameterkombination ausgewählt.

Jeder untersuchter Agent der beiden Projektarchive [66, 53] ergänzt, entfernt oder modifiziert eine Variante aus Abschnitt 3.3 um einen Aspekt, sodass dessen Effekt isoliert betrachtet werden kann. Um genauere Schlussfolgerungen aus der Evaluation zu ziehen, fungiert eine Voranalyse (Experiment 1) dazu, den Hyperparameterraum einzugrenzen und die besten vier DRL Varianten für eine detaillierte Evaluation auszuwählen. Die ausgewählten Agenten werden anschließend um das realitätsnahe Testsetting erweitert und in die entwickelte Testumgebung eingefügt. In der umfassenden Analyse werden drei Experimente durchgeführt, mit denen die Performance in Bezug auf viele unterschiedliche Wertpapiere in mehreren Jahren (Experiment 2), Krisenzeiten (Experiment 3) und einiger Modifikationen (Experiment 4) gesondert getestet werden kann. Dadurch wird die Gefahr von Überanpassung reduziert und Unterschiede in makroökonomischen Umständen besser ausgeglichen, als wenn alle Agenten nur in einem einzigen Jahr getestet werden.

4.1 Datensatz

Alle Daten werden von Yahoo Finance bezogen und in keiner Weise bereinigt bzw. modifiziert. Diese Vorverarbeitung ist Standard im algorithmischen Handel und wird daher für eine bessere Vergleichbarkeit mit anderen DQL Agenten ausgewählt (z.B. [54, 65]). Die Aufteilungen des Datensatzes werden in Abbildung 4.1 illustriert. Mit Experiment 2 werden auf der einen Seite Performanceunterschiede auf einzelne Charakteristika der Wertpapiere zurückgeführt. Andererseits werden verschiedene DRL Varianten realitätsnah verglichen, um die Performance hinsichtlich unterschiedlicher Wertpapierklassen bzw. -sektoren differenziert einzuschätzen. Experiment 3 dient zur Sondierung der Performance

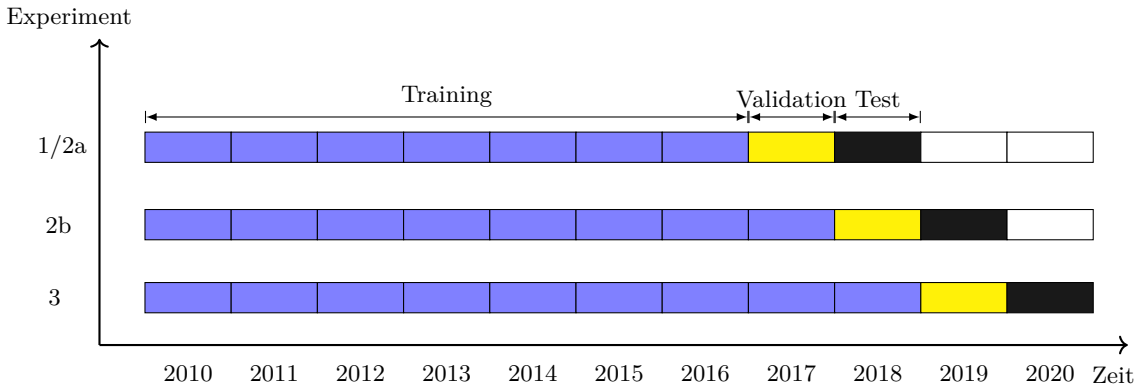

Abbildung 4.1: Aufteilung der Zeitreihe. Experiment 1 nutzt den ersten Zeitraum, Experiment 2 umfasst die Evaluation von zwei Jahren (2a und 2b). Experiment 3 verwendet den dritten Zeitraum und wird nur für eine Teilmenge der Wertpapiere durchgeführt, die im ersten Quartal 2020 starke Einbrüche erlitten haben. Experiment 4 untersucht den Effekt von Designänderungen auf die Performance, indem mit den erweiterten Agenten ebenfalls Experiment 1-3 durchgeführt werden.

der Agenten während der COVID-19 Krise, sodass die Anwendung bei volatile Marktphasen getestet werden kann. In Experiment 4 werden zwei modifizierte Agenten mit denselben drei Zeiträumen getestet, um den Effekt zusätzlicher Marktinformationen, der Regularisierungsmethode Dropout und eines erweiterten Aktionsraums auf die Entscheidungsfindung zu untersuchen.

Für optimale Ergebnisse muss der Trainingshorizont zum einen lang genug sein, damit der Kursverlauf genügend Muster enthält. Zum anderen sollten historische Werte vergleichbar sein und inbesondere keine erheblichen Marktumstellungen aufweisen, welche die Mustererkennung erschweren. Ähnlich zu anderen Evaluationen [54, 65] wird daher einen Zeitraum von elf Jahren gewählt. [30] beschreibt detailliert, wie die Aufteilungen in Trainings-, Validations- und Testdatensatz im Experiment verwendet werden. Mit unterschiedlichen Hyperparametermengen wird auf dem Trainingssatz trainiert und auf dem Validationssatz ausgewertet. Das am besten abschneidende Modell wird daraufhin neu auf dem Trainings- und Validationssatz trainiert und auf dem Testsatz getestet. Es gilt zu beachten, dass die Gewichte θ während des Testens nicht verändert werden. Dieser Vorgang wird für drei Zeiträume und drei Durchgänge mit neuen Gewichten sowie initialen Portfolioeinstellungen nach jeder Trainingsiteration wiederholt. Um gleiche Testbedingungen zu gewährleisten, wird der Testsatz auf 251 Tage beschränkt, an denen der Agent Aktionen ausführen kann.

4.1.1 Experiment 1: Vorstudie

Da die durchschnittliche Trainingszeit jedes Agenten pro Aktie inklusive des Hyperparametertunings ca. $34,2$ CPU-Kern-Stunden beträgt, werden aus Ressourcengründen neun sehr unterschiedliche Wertpapiere für die Voranalyse ausgewählt. Die Wertpapiere sind in Tabelle A2 im Anhang gekennzeichnet. Dieser Datensatz wird dazu verwendet, um in

einem initialen Testdurchlauf die vielversprechendsten vier Agenten zu bestimmen. Dafür wird das Evaluationssetting der Agenten angeglichen und die Testumgebung nur um den Datensatz sowie einheitliche Marktbedingungen (vgl. 4.4) erweitert. Für drei Durchläufe der ersten beiden Zeithorizonte aus Grafik 4.1 wird der Mittelwert des Profits gebildet, der die Performance des Agenten abbildet und die Agenten einstuft. Außerdem dient der Datensatz dazu, einige Hyperparameterkombinationen zu testen. Indem Hyperparameter ohne beobachtbare Einflüsse aus dem Suchraum ausgeklammert werden, wird die Anzahl der Dimensionen reduziert, ohne die Hyperparameteroptimierung stark zu beeinträchtigt. Das ermöglicht, die Performance der Agenten in den nachstehenden Experimenten mit mehr Wertpapieren zu untersuchen.

4.1.2 Experiment 2: Hauptevaluation

Der Datensatz für die Hauptevaluation enthält 43 Aktien, 11 ETFs und 16 Warentermingeschäfte aus dem NYSE, NASDAQ und SEHK[1]. Die Auswahl kombiniert und erweitert die Zeitreihen einiger größerer Studien [54, 65, 59]. Schwerpunktmäßig wird der amerikanische Markt (NYSE, NASDAQ) untersucht, da diese Daten in der bestehenden Literatur am häufigsten analysiert werden [43]. Aktien aus dem SEHK dienen zum Vergleich von Märkten; unterschiedliche Klassen an Wertpapieren bilden verschiedene Charakteristika im Kursverlauf ab.

Tabelle A2 kategorisiert alle gehandelten Wertpapiere hinsichtlich einiger charakteristischen Merkmale. In Bezug auf qualitative Merkmale beschränkt sich die Performancestudie auf Branchenzugehörigkeit, Unternehmensgröße, Börse und Region der Unternehmenszentrale. Als quantitative Eigenschaften werden technische Charakteristika im Kursverlauf betrachtet. Zum einen wird die Risikogewichtung einer Aktie über die Standardabweichung der Zeitreihe und den Beta-Faktor[2] bestimmt. Der Beta-Faktor zeigt die Beziehung eines Wertpapiers mit dem Finanzmarkt und weist eine positive Korrelation mit dem Gewinn eines Wertpapiers auf [13]. Zum anderen sind stark steigende und fallende Wertpapiere gruppiert, um den Effekt positiver bzw. negativer Trends auf die Performance zu untersuchen.

4.1.3 Experiment 3: Krisenevaluation

Für die Evaluation von Krisenzeiten wird eine Untermenge (siehe Tabelle A2) aus zehn Wertpapieren verwendet, die besonders starke Einbrüche im Kursverlauf aufgrund der COVID-19 Krise aufweisen. Die Auswahl basiert auf den Metriken der Studie von Chowdhury et al. [11], in der die Auswirkungen von COVID-19 auf den globalen Aktienmarkt erforscht werden. Die Pandemie sorgte im ersten Quartal 2020 für starke Einbrüche in der Weltwirtschaft [35], welche einige Herausforderungen für den algorithmischen Handel

[1] New York Stock Exchange, National Association of Securities Dealers Automated Quotations bzw. Stock Exchange of Hong Kong

[2] Der Beta-Faktor ist als $\frac{Kovarianz(r_a, r_m)}{Varianz(r_m)}$ definiert, wobei r_m die Markrendite und r_a die Rendite des Wertpapiers angeben. Berechnet wird der Beta-Faktor der letzten 10 Jahre.

darstellen. Vorwiegend weisen die Zeitreihen abrupte Schwankungen mit einer negativen Abweichung vom Mittelwert auf. Untersucht wird, wie gut der Agent seine Strategie an die hohe Volatilität neu anpassen kann.

4.1.4 Experiment 4: Erweiterte Architektur

Das Experiment erweitert den Datensatz aus Experiment 1-3 um weitere Finanzdaten, die einigen Agenten als ergänzende Marktinformationen übergeben werden. Die Auswahl der Wertpapiere basiert auf mathematisch nachgewiesenen Korrelationen: Einige Studien zeigen nicht-lineare Wechselwirkungen des Öl- bzw. Goldkurses [5, 44, 51] und Zinsschwankungen [50] mit mehreren Aktien oder ETFs. Bessere Aktienvorhersagen konnten darüber hinaus bereits bei Simulationen [31] erzielt werden, indem der Volatilitätsindex (^VIX) des US-amerikanischen Aktienmarkts in Modelle inkludiert wurde. Ein hoher ^VIX weist auf eine unsichere Marktlage hin und korreliert meist negativ mit Aktienindices. Zusätzlich sind die Zeitreihe eines weiteren Vertreters der Branche und ein Aktienindex derselben Börse im Datensatz enthalten, die ebenfalls Wechselwirkungen mit dem gehandelten Wertpapier aufweisen [40, 46]. Die ausgewählten Kontextdaten sind in Tabelle A2 aufgeführt. Eine genaue Untersuchung, welche Wertpapiere ausnutzbare Korrelationen in Bezug auf NNe aufweisen, stellt ein spannendes Forschungsfeld für weitere Arbeiten dar.

In der Evaluation werden ebenfalls die Zeiträume von Experiment 1-3 ausgewertet, wobei dasselbe, oben beschriebenen Setting inklusive einer identisch ausgeführten Hyperparametersuche verwendet wird. Indem die Agenten analogen Restriktionen unterliegen, können damit Effekte einzelner Designerweiterungen bei unterschiedlichen Marktsituationen isoliert untersucht werden.

4.2 Architektur und Training

Die zwei untersuchten Projektarchive [66, 53] enthalten insgesamt siebzehn DRL Varianten, deren Aktionsräume (siehe Abschnitt 3.1) und Konfigurationen für alle Experimente vereinheitlicht werden. Eine vollständige Liste der Agenten befindet sich in Tabelle A1 im Anhang. Die Agenten werden um das folgende Setup erweitert bzw. modifiziert:

Ein vorwärtsgerichtetes Netzwerk, bestehend aus einer verdeckten Schicht mit der ReLU Aktivierungsfunktion[3], wird mit einem Adam Optimierer verwendet, um den DQL Entscheidungsprozess aus Abschnitt 3.2 für alle Agenten zu lösen. Für eine bessere Vergleichbarkeit werden die Agenten in einigen Aspekten weiter vereinheitlicht: Epsilon decay wird mit der Exponentialfunktion skaliert $exp(-\text{epsilon decay} * \text{i})$, wobei i die i-te Iteration angibt. Durch die negativ exponentielle Skalierung kann der Trainingsprozess besser konvergieren und flexibler von Exploration zu Exploitation übergehen [63]. Das Training stoppt nach 1000 Epochen oder, falls keine Verbesserung nach fünfzehn Epochen erzielt wird. Diese Anpassung wird als early stopping bezeichnet und reduziert die Gefahr von Überanpassung.

[3] Der Rectifier ist der Positivteil eines Arguments, $f(x) \coloneqq max(0, x)$.

Als Grundlage der Implementierung dient [7]. Der Agent nutzt einen Wiedergabespeicher mit einer fixen Größe, die als Speichergröße m in Algorithmus 1 definiert ist. Zur besseren Verwaltung und Verringerung der Komplexität werden die DRL Varianten in eine Vererbungsstruktur eingegliedert.

4.2.1 Agenten ohne Kontextdaten

In der Voranalyse schneiden vier verschiedene DQL Ansätze bei neun Wertpapieren mit sehr unterschiedlichen Charakteristika am besten ab. Die in Experiment 2-3 (cf. Paragraph 4.1.2 bzw. 4.1.3) näher analysierten DRL Varianten sind:

Deep q learning Agent (DQLA). Der Agent besteht aus einem vorwärtsgerichteten Netzwerk, enthält als Eingabe die letzten 30 täglichen Schlusskurse des Wertpapiers und kann pro Tag eine Aktion aus dem in Paragraph 3.1 definierten Aktionsraum ausführen. Die Belohnung wird aus der Differenz des aktuellen und des vorherigen Wertpapierkurses berechnet. Die weiteren Varianten verändern den DQLA in den nachstehenden Aspekten.

Duel deep recurrent q learning Agent (DDRQLA). Klassische LSTM Zellen [15], die als Eingabe- bzw. Ausgabe-Aktivierungsfunktion den tanh()[4] und als rekurrente Aktivierungsfunktion die Sigmoid-Funktion[5] nutzen, werden anstelle gewöhnlicher Neuronen in der verdeckten Schicht eingesetzt. Wie in Paragraph 3.3.2 beschrieben, wird das RNN zusätzlich für eine Dueling Strategie zur Berechnung des Zustandeswertes und Aktionsvorteils aufgespalten.

Deep deterministic policy gradient Agent (DDPGA). Das Modell enthält vier vorwärtsgerichtete NNe, wobei die Q-Funktion synchron trainiert wird. An die beiden Kritiker-Netzwerke werden weiterhin eine verdeckte Schicht mit einem Ausgabeskalar angehängt, der den Kritikerwert repräsentiert.

Als Baseline dienen zwei technische Indikatoren, die in der Praxis häufig zum Einsatz kommen [58, 54]. Der erste Agent (MA) trifft seine Handlungsentscheidung entsprechend dem gleitenden 30 Tage Durchschnitt (vgl. moving average [8]). Der zweite Agent (TT) kauft ein Wertpapier, sobald der Kurs unter das globale Minimum fällt und umgekehrt (vgl. turtle trading [55]). Begründet ist diese Auswahl dadurch, dass die beiden Indikatoren entgegengesetzte Positionen einnehmen (vgl. [54]). MA folgt dem Trend, während TT gegensätzlich zum Kursverlauf handelt. Somit können unterschiedliche Kursverläufe vielseitiger technisch vorhergesagt werden.

4.2.2 Agenten mit Kontextdaten

Um den Einfluss von zusätzlichen Finanzinformationen und anderer Modifikationen auf verschiedene DRL Algorithmen zu testen, werden die folgenden Agenten in das entwickelte

[4] Der Tangens Hyperbolicus ist als $tanh(x) := \frac{sinh(x)}{cosh(x)} = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)}$ definiert.
[5] Die Sigmoid-Funktion ist als $sig(x) := \frac{1}{1+exp(-x)}$ definiert.

Evaluationssetup eingefügt und mit Experiment 4 (cf. Paragraph 4.1.4) ausgewertet:

Deep deterministic policy gradient Agent mit Kontextdaten (DDPGAK). Die Architektur des Agenten ist wie beim DDPGA aufgebaut, allerdings umfasst die Eingabe zusätzlich korrelierende Finanzmarktdaten (siehe Abschnitt 4.1.4). Anstatt dass die Eingabeschicht die letzten 30 Schlusskurse des Wertpapiers erhält, verarbeitet das NN noch weitere 150 Eingabewerte (die letzten 30 Werte von fünf korrelierenden Zeitreihen). Der Agent wird bereits in [61] in einem anderen Kontext näher untersucht, nutzt eine Vorimplementierung des Moduls OpenAI-Baselines und wird um den Aktionsraum und die Belohnungsfunktion aus Abschnitt 3.1 modifiziert.

Erweiterter deep q learning Agent (EDQLA). Um den Einfluss von Kontextdaten, geringeren Handelsbeschränkungen und Dropout auf die Performance für den DQL Lernalgorithmus zu konstantieren, wird im nächsten Schritt das Design des DQLA in drei Aspekten angepasst. Dadurch können insbesondere Schlussfolgerungen über den Nutzen eines erweiterten Zustandsraums für einen weiteren DRL Algorithmus getroffen werden. Dabei wird einerseits untersucht, wie sich das Hinzufügen von Kontextdaten im Speziellen auf die Entscheidungsfindung des DQL Algorithmus auswirkt. Andererseits wird die Architektur des Agenten in Anlehnung an state-of-the-art Ansätze [25, 54, 12] zusätzlich erweitert, um zu testen, inwiefern mit weiteren Änderungen für den DQL Ansatz bessere Ergebnisse sowohl in den Jahren 2018 und 2019, als auch im Krisenjahr erzielt werden können. Die modifizierten Aspekte sind:

(i) *Erweiterung des Zustandsraums:* Ein größerer Zustandsraum (siehe Paragraph 3.1) wird dazu verwendet, dem Agenten die Kontextdaten aus Abschnitt 4.1 als zusätzliche Eingabewerte zu übergeben. Damit sollen Korrelationen im Finanzmarkt ähnlich wie beim DDPGAK gewinnbringend ausgenutzt werden. Die Eingabeschicht wird dabei ebenfalls von 30 auf 180 erhöht, sodass der Wertpapierkurs und die korrelierenden Zeitreihen der letzten 30 Tage für jeden Zeitschritt übergeben werden.

(ii) *Implementierung von Dropout:* Zur Verringerung der Überanpassung wird die Regularisierungsmethode Dropout verwendet, die in [21] spezifiziert wird. Diese Implementierung wird bereits von anderen Studien [58, 47] im algorithmischen Handel zur Verbesserung der Modelle eingesetzt und wird für den EDQLA übernommen. Zusammengefasst werden aus der verdeckten Schicht des NNes zufällig einzelne Neuronen für die Dauer des Trainings entfernt. Das bedeutet, dass deren Beiträge zur Aktivierung der Ausgabeschicht zeitweise ignoriert und deren Gewichte währenddessen nicht angepasst werden. Die Wahrscheinlichkeit, mit der einzelne Neuronen aus dem Modell entfernt werden, wird als Dropoutrate referenziert und zusätzlich bei der Hyperparametersuche optimiert. Effektiv werden gewisse Gewichte auf null gesetzt, während die restlichen Gewichte mit $\frac{1}{1-\text{Dropoutrate}}$ so skaliert werden, dass die Summe über alle Eingaben gleich bleibt.

(iii) *Erweiterung des Aktionsraums:* Ein größerer Aktionsraum (siehe Paragraph 3.1) eröffnet dem Agenten mehr Handlungsmöglichkeiten, wenngleich daraus ein höheres Risiko und höhere Transaktionskosten resultieren. In diesem Evaluationssetting ist die

Erweiterung dadurch motiviert, dass der Agent in den Experimenten in Kapitel 4 nur einen kleinen Teil seines Bargeldes nutzen kann. Viele Portfolioallokationen können dadurch nicht angenommen werden, was eine deutliche Einschränkung im Training und Testen der Agenten darstellt. Krisenzeiten verdeutlichen, dass der Agent nicht schnell genug auf Kurseinbrüche reagiert, da er die Wertpapieranzahl nur langsam reduzieren kann. Insofern wird mit dem NN nicht die optimale Handelsentscheidung zu jedem Zeitschritt vorhergesagt. Stattdessen muss der Agent aufgrund starker Handelseinschränkungen beispielsweise frühzeitig Wertpapiere abstoßen, um hohe Verluste zu vermeiden. Die resultierenden Handlungen sind daher nicht optimal. Aus diesem Grund wird, ähnlich zu anderen Veröffentlichungen [54, 25, 12], ein stetiger Aktionsraum mit einer oberen und unteren Grenze implementiert. Konkret orientiert sich die Implementierung an der Architektur in [25]: Die Ausgabeschicht besitzt weiterhin drei Neuronen (kaufen, halten, verkaufen). Anstatt einer linearen Aktivierungsfunktion nutzt der EDQLA in der Ausgabeschicht die Softmax-Funktion (Gl. 4.2) mit einem Skalar. Der Vorteil ist, dass die Werte damit an eine normalisierte Wahrscheinlichkeitsverteilung angepasst werden. Die Zahl R_{num} der gehandelten Wertpapiere ist gegeben durch Gleichung 4.3, wobei z den Eingabevektor mit Verzerrungswerten und Gewichten angibt:

$$\sigma(z) : \mathbb{R}^n \longmapsto \{x \in \mathbb{R}^n \mid \|x\|_1 = 1,\ x_i > 0\} \tag{4.1}$$

$$\sigma(z)_i := \frac{exp(z_i)}{\sum_{j=1}^{n} exp(z_j)}, \text{ für } 1 \leq i \leq n \tag{4.2}$$

$$R_{num} = int(\max_i \sigma(z)_i) * L \tag{4.3}$$

Der Parameter n steht für die Anzahl der Neuronen in der Ausgabeschicht und die Variable L definiert die maximale Kauf- bzw. Verkaufsmenge. Analog zu [25] wird L=10 gewählt.

4.3 Hyperparametersuche

Das inititale Testsetting der Agenten aus den beiden Verzeichnissen (cf. [66, 53]) enthält keine umfassende Hyperparameteroptimierung und die Suchräume weichen teilweise voneinander ab. Da unterschiedliche Varianten bzw. Implementierungen verschiedene Hyperparameter verwenden und Vergleiche ansonsten verzerrt werden, wird die Anzahl der Parameter für alle Varianten nicht gleich gehalten. Außerdem nutzen bestimmte Lernalgorithmen ihre Hyperparameter unterschiedlich, weshalb für eine gleichwertige Konfiguration die Einstellungen individuell optimiert werden. Für jeden Hyperparameter werden zwei Werte getestet. Größere Wertebereiche würden die Agenten noch weiter verbessern. Allerdings impliziert dies eine höhere Laufzeit, was aus Ressourcengründen vermieden wird.

In der Voranalyse (siehe Paragraph 4.1.1) werden die Lernrate aus Gleichung 3.4 und, bei den DDPG Agenten, die Lernrate des Kritikers fest eingestellt. Für jeden Agenten werden drei Lernraten ($\sim \{0.01, 0.005, 0.001\}$) evaluiert. In über 70% der Fälle erzielte jeweils

eine bestimmte Lernrate die besten Ergebnisse. Deshalb wird dieser Parameter bei der Rastersuche ausgeklammert und für jeden Agenten vorher individuell eingestellt.

Wenngleich es Methoden zur genaueren Suche guter Hyperparameter gibt [30], bietet eine Rastersuche einige Vorteile für das Evaluationssetting: Sie ist einfach zu implementieren, deckt den Suchraum gut ab und ermöglicht auf effiziente Weise eine erste Einschätzung der optimalen Kombination. In allen Experimenten werden die nachstehenden Hyperparameter in einer Rastersuche individuell für jede Komposition aus Wertpapier, Agent und Durchlauf eingestellt. Insgesamt werden 1800 Rastersuchen durchgeführt (450 für jeden der vier Agenten), wobei sich die Bezeichnungen an Kapitel 3 orientieren:

- Gamma $\sim \{0.9, 0.999\}$: Eine langfristige Strategie ($y \to 1$) hat zur Folge, dass die Handelsfrequenz des Agenten aufgrund geringerer Transaktionskosten sinkt. Im Gegensatz dazu hat $y \to 0$ den Vorteil, dass unsichere, zukünftige Ereignisse weniger stark ins Gewicht fallen.
- Schichtgröße $\sim \{2^8, 2^9\}$: Die Schichtgröße definiert die Anzahl der Neuronen bzw. der LSTM Zellen in der versteckten Schicht. Mehr Neuronen in einer Schicht sorgen für ein komplexeres Modell.
- Epsilon decay $\sim \{10^{-2}, 10^{-3}\}$ bzw. epsilon $\sim \{0.5, 0.8\}$: Eine greedy Methode ($\epsilon \to 0$) kann zu geringer Exploration und Konvergenz in ein schwaches lokales Minimum führen, wohingegen rein zufällige Strategien ($\epsilon \to 1$) divergieren. Die Agenten aus [66] verwenden epsilon decay, während der DDPGAK ein konstantes ϵ einsetzt. DDPGAK enthält das Modul OpenAI-Baselines, dessen DDPG Algorithmus nur ein konstantes ϵ benutzt.
- Speichergröße $\sim \{251, 753\}$: Mögliche Speichergrößen sind ein Jahr (251 Tage) und drei Jahre (753 Tage). Für den algorithmischen Handel bietet eine optimale Einstellung für das Lernen mit einem Wiedergabespeicher, neben der Vermeidung von Überanpassung, einige Vorteile: Die Wahrscheinlichkeitsverteilung ρ wird zufällig gemittelt. Dadurch wird der Lernprozess über viele vorangegangene Sequenzen geglättet und die Varianz der Anpassungen sinkt, wodurch Ausreißer in volatilen Finanzdaten besser ausgeglichen werden. Weiterhin werden manche Zeitschritte mehrmals zur Aktualisierung der Gewichte verwendet, was zu einer besseren Datennutzung beiträgt (vgl. [37]).

Im Falle von DDPGAK werden zwei weitere Hyperparameter berücksichtigt, die bei den anderen Agenten nicht eingesetzt werden. Der erste ist die Batchgröße ($\sim \{2^7, 2^8\}$), welche die Anzahl der Zeitschritte angibt, die für jede Trainingsiteration verwendet werden. Die Batchgröße hat einen wichtigen Einfluss auf die Konvergenz des Trainings und auf die resultierende Performance des Modells. Alle anderen Agenten aktualisieren ihre Gewichte nach jedem Zeitschritt, weshalb sie eine eine fixe Batchgröße von eins besitzen. Die zweite Variable ist der Parameter $\tau \sim \{10^{-2}, 10^{-3}\}$ aus Paragraph 3.3.3. Indem die Kopien des Akteurs und des Kritikers zeitverzögert angepasst werden, stabilisiert eine optimale Wahl von τ den Lernprozess. Da der erweiterte EDQLA zur Regularisierung Dropout verwendet,

optimiert die automatische Hyperparametersuche zusätzlich für diesen Agenten noch die Dropoutrate ($\sim \{0.5, 0.8\}$) für jedes Wertpapier individuell.

4.4 Marktbedingungen

Die in dieser Arbeit entwickelte Testumgebung trifft einige Annahmen über die Finanzumgebung, welche den Entscheidungsprozess des Agenten einfach und gleichzeitig wirklichkeitsnah machen. Zum einen enthält diese Testumgebung, wie in [54, 39, 65], Transaktionskosten in Höhe von 1% der Kosten des Wertpapiers. Auswirkungen unterschiedlicher Transaktionskosten werden unter anderem in [54, 39] ausgeführt. Ähnlich zu den Vertragskonditionen führender Investmentbanken sind Depotkosten, Mindestordergebühren oder Vergünstigungen bei häufigen Handlungen nicht im Modell enthalten. Zum anderen ist das Handelsvolumen unbeschränkt und der Agent nimmt durch seine Entscheidungen keinen Einfluss auf den Marktpreis. Zur besseren Vergleichbarkeit verwenden sowohl die Voranalyse als auch die drei nachfolgenden Experimente dieselben Marktbedingungen.

4.5 Metrik

Die untersuchten, qualitativen Metriken jedes Experiments sind:

- Profit: Hauptsächlich wird die Performance der Agenten über den Profit $R_T = \sum_{t=0}^{T} r_t$ analysiert, wobei r_t die Rendite zum Zeitpunkt t darstellt. Die Rendite ist dabei die Änderung des Portfoliowerts, der in Paragraph 3.1 definiert ist.
- Volatilität: Die durchschnittliche Volatilität wird in Form der Standardabweichung der Portfolioentwicklung gemessen und zeigt das Risiko an, dem das Portfolio ausgesetzt ist.
- Ausreißer: Zur besseren Risikobetrachtung wird ebenfalls der größte Verlust des Agenten an einem Tag betrachtet.
- Sharpe Ratio: Das Sharpe Ratio [39] liefert eine risikobereinigte Rendite. Es berechnet sich aus $S_T = \frac{Durchschnitt(R_t)}{Standardabweichung(R_t)}$. Identisch zur bestehenden Literatur [65, 54, 56] wird den Agenten eine risikoneutrale Nutzenfunktion unterstellt. Das Sharpe Ratio wird verwendet, da sich damit auch risikoaverse Präferenzen darstellen lassen. Zukünftige Untersuchungen könnten auch noch Modelle miteinbeziehen, mit denen weitere Risikopräferenzen der Investoren abgebildet werden können. Um die Testumgebung so einfach wie möglich und Vergleiche mit anderen Studien fair zu gestalten, beschränkt sich dieses Setting allerdings auf eine risikoneutrale Nutzenfunktion.

Zum Vergleich der einzelnen Sektoren und Agenten in den ersten beiden Experimenten dienen Mittelwerte der vorgestellten Metriken. Ferner zeigen grafische Darstellungen die Handlungsentscheidungen des Agenten im Einzelnen, sodass die Vorhersagefähigkeit speziell bei deutlichen Kursänderungen, wie beispielsweise ausgelöst durch die COVID-19 Pandemie

in Experiment 3, auch qualitativ untersucht werden kann. Indem die Aktionen gemeinsam mit dem Portfoliowert und dem Wertpapierpreis veranschaulicht werden, lassen sich die Stärken und Schwächen zusätzlich differenzierter herausarbeiten.

5 Ergebnis und Diskussion

Jeder der insgesamt 2400 Testdurchläufe benötigte im Mittel 34,2 CPU-Kern-Stunden bei CPUs mit durchschnittlich 2.4GHz, wobei der DDPGA um 30% schneller und der DDRQLA vier Mal langsamer trainierte. Dies ergibt eine Gesamtlaufzeit auf einer CPU von etwas über 82.000 Stunden.

In diesem Kapitel werden die Ergebnisse basierend auf dem experimentellen Aufbau aus Kapitel 4 vorgestellt. Dies umfasst Einblicke in das Hyperparametertuning und Training (5.1), die Ergebnisse der Vorstudie zur Eingrenzung der DRL Varianten (5.2), die Analyse verschiedener Architekturen im Hinblick auf unterschiedliche Wertpapiercluster mit den Resultaten der Hauptevaluation (5.3), die Ergebnisse des dritten Experiments für eine Einschätzung der Agenten in Krisenzeiten (5.4), und die Performanceanalyse des vierten Experiments zur Evaluation, inwieweit einzelne Erweiterungen in der Architektur Einfluss auf DRL Strategien nehmen (5.5). Welchs t-test mit einem Signifikanzniveau von $p = 0,05$ wurde verwendet, um die Zusammenhänge einiger Charakteristika und Hyperparameter zu untersuchen.

5.1 Training und der Einfluss von Hyperparametern

Für alle Experimente können auffällig hohe Schwankungen in der Trainingsdauer verschiedener Durchläufe desselben Agenten wegen early stopping beobachtet werden. Teilweise bricht der Algorithmus nach weniger als 30 Iterationen ab, während er bei gleicher Hyperparameterkombination auf demselben Wertpapier mehr als 800 Trainingsdurchläufe iteriert. Die Endresultate der Trainingsiterationen oszillieren meistens, wodurch der Eindruck entsteht, dass der Lernalgorithmus häufig keinen stetigen Fortschritt macht. Hohe Schwankungen sind ebenfalls bei der Bestimmung der optimalen Hyperparameter für jede Kombination aus Wertpapier und Agent zu beobachten, was vermutlich auf unterschiedliche Marktmuster 2018 und 2019 zurückzuführen ist. Die Parameter weisen auch bei Vergleichen von Branchen oder Wertpapierklassen keine statistisch signifikanten Tendenzen auf, welche Einstellungen optimal sind. Nichtsdestotrotz kann durch einen Vergleich der durchschnittlichen Ergebnisse der Validation mit den Testresultaten der klare Trend beobachtet werden, dass Modelle mit einer Hyperparameteroptimierung besser abschneiden. Das bestätigt, dass die Konfiguration der Modelle trotz Marktfluktuationen ein essentieller Schritt für das Training von DRL Algorithmen im algorithmischen Handel ist.

Detaillierte Analyse. Genauere Untersuchungen zeigen jedoch, dass bei einer guten Performance einer Hyperparameterkombination im Validationsatz nicht eindeutig auch gute Ergebnisse im Testsatz zu erwarten sind. Obwohl beispielsweise eine Verringerung von epsilon decay oder gamma zu höheren Profiten bei der Validation führt, konnten damit in allen drei Testjahren schlechtere Ergebnisse erzielt werden. Genauso sorgt eine Änderung

der Speicher- oder Schichtgröße auf dem Validationssatz beim zweiten Experiment nur für einen Performanceunterschied von unter einem Prozent. Auf dem Testsatz hingegen führt eine größere verdeckte Schicht bzw. ein längerer Speicherhorizont zu hohen Verlusten in 2018, wenngleich damit 2019 und in Experiment 3 der dreifache Gewinn erzielt wird. Statistisch korreliert eine höhere Schichtgröße signifikant positiv[1] mit der Zahl der gehaltenen Wertpapiere, was das Gefälle in den Experimenten erklären könnte. Weiterhin enthält die optimale Hyperparameterkombination im Jahr 2019 etwas über 28% häufiger den geringeren Wert für die Schichtgröße. Dies ist darin begründet, dass weniger gehandeltene Wertpapiere zu weniger Verlust im Validationsjahr führen.

Die obigen Tendenzen lassen sich bis auf eine Ausnahme auf jeden Agenten individuell übertragen. Und zwar verzeichnet der DQLA bei einer Erhöhung von epsilon decay signifikant höhere Verluste, durchschnittlich ca. 35,7%. Ein langsamerer Zerfall von epsilon nimmt demnach positiven Einfluss auf die Konvergenz im Training und die Performance im Test. Analog verbessert der DDPGAK zusätzlich noch die Batchgröße und den Parameter τ. Wie erwartet ist die Batchgröße ein wichtiger Hyperparameter, der die Performance des DDPG Netzwerks beeinflusst. Der Agent trainiert bei einer höheren Batchgröße schneller, jedoch mit einer höheren Varianz in den Renditen. Der Parameter τ hingegen beeinflusst weder die Ergebnisse der Validation noch der Tests in signifikantem Maße, da bei einer Änderung von τ für keinen Agenten in keinem der Jahre eine klare Tendenz erkennbar ist.

5.2 Vorauswahl der Agenten

Die Ergebnisse der Voranalyse decken sich in größten Teilen mit der bestehenden Literatur (cf. Tabelle A1). Vier verschiedene DQL Agenten konnten sich gegenüber anderen DRL Ansätzen wie in [59, 56, 65] durchsetzen. Dueling Strategien bzw. andere Abwandlungen des DQL Algorithmus (vgl. Double DQL in [65]) zeigen analog zu [65] keine bessere Vorhersagekraft. Darüber hinaus verbessern rekurrente Architekturen die Performance im Durchschnitt (vgl. [10]) um mehr als 12%, was später in der Hauptevaluation noch deutlicher wird. Ein unerwartetes Ergebnis der initialen Studie ist, dass der einfache DQLA und DDPGA zu besseren Resultaten als dessen rekurrente Variante führen. Überdies schneidet die Kombination aus der Dueling Architektur (DDQL) mit einer LSTM Schicht entgegen der zuvor beschriebenen Erwartung unter allen Varianten am besten ab.

5.3 Hauptevaluation

Da die Agenten auf verschiedenen Wertpapierclustern unterschiedliche Resultate liefern, werden Performanceunterschiede in Bezug auf die charakteristischen Merkmale aus Abschnitt 4.1 und deren Wechselwirkungen sowohl durchschnittlich als auch individuell für jede Variante analysiert. Es gilt zu beachten, dass die Ergebnisse in Abbildung 5.1 stark von

[1] Der gemittelte Anteil der gehaltenen Wertpapiere am Portfoliowert in Bezug auf jeden Agenten für jedes Wertpapier korreliert signifikant mit der Schichtgröße mit einem p-Wert von ca. $p = 0,0477$.

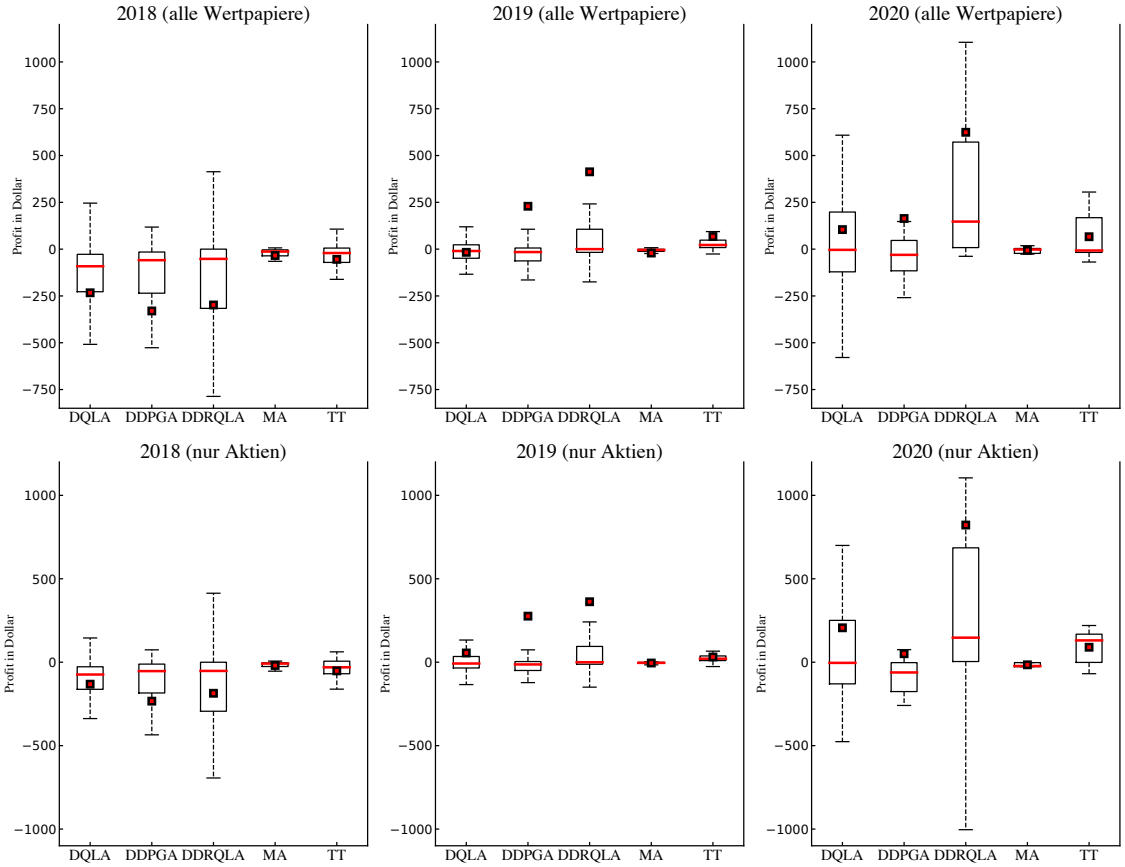

Abbildung 5.1: Ergebnisse aller 1440 Testdurchläufe der DRL Agenten ohne Kontextdaten für alle Wertpapiere (oben) und nur für Aktien (unten) für jede Variante und jedes Evaluationsjahr. Die Kästen repräsentieren die Reichweite zwischen dem 25ten und 75ten Perzentil der Werte, während die Pfeile bis auf Ausreißer die gesamte Reichweite illustrieren. Der rote Punkt zeigt den Mittelwert und die rote Linie markiert den Median der Daten. Der Mittelwert weicht im zweiten Testjahr aufgrund hoher Ausreißer nach oben stark vom Median ab.

der verwendeten Testumgebung beeinflusst sind; jede Schlussfolgerung daraus beschränkt sich folglich spezifisch auf das gewählte Setting und den Zeithorizont.

Abbildung 5.1 veranschaulicht die Verteilung der durchschnittlichen Ergebnisse jedes Agenten ohne Kontextdaten in Bezug auf deren Testjahr. Agentenübergreifend ist zu sehen, dass durchschnittlich deutlich bessere Ergebnisse 2019 (145 $) als 2018 (-185 $) erreicht werden konnten. Dieser Unterschied kann auf den längeren Trainingszeitraum und auf günstigere Marktbedingungen zurückgeführt werden. Eine erste wichtige Erkenntnis daraus ist, dass durch eine gute Wahl des Testjahres die Agenten prozentual wesentlich weniger Verlust erleiden (42,0%) als zuvor (65,3%). Da selbst dieselben Agenten in unterschiedlichen Jahren hohe Varianzen in den Ergebnissen aufweisen, sind Performancevergleiche mit anderen Studien nur bei ähnlichem Setup sinnvoll.

Im Vergleich zu anderen Evaluationen mit kongruentem Testsetting schneiden die DRL Varianten in den meisten Fällen leicht schlechter ab. Für das Jahr 2018 liegt das beste Testergebnis auf einer Technologieaktie bei ungefähr 9% Portfoliosteigerung, was weit unter den ca. 17% von Théate et al. [54] liegt, die allerdings einen viel aufwendigeren Agenten und geringere Transaktionskosten verwenden. Das beste Ergebnis im Jahr 2019 erzielte wie zu erwarten der DDRQLA mit einem Gewinn von über 12%, was auf der anderen Seite relativ nahe an die präsentierten Ergebnisse von [65, 59] herankommt. Es gilt aber zu beachten, dass nicht state-of-the-art Ergebnisse erzielt werden sollen, sondern schwerpunktmäßig faire Vergleiche einzelner Varianten und Wertpapiercluster gemacht werden.

Vergleich der Agenten. Die großen Performanceunterschiede einzelner Varianten im oberen Teil von Abbildung 5.1 zeigen, dass in der differenzierten Evaluation keine Architektur eindeutig bessere Ergebnisse liefert. Während 2019 eine überlegene durchschnittliche Performance des DDRQLA vor anderen DRL Architekturen und den beiden Baseline Agenten beobachtet werden kann, schneidet er 2018 deutlich schlechter ab. Trotz der hohen Varianz der Testergebnisse ist allerdings der Mittelwert des Sharpe Ratios der rekurrenten Variante in beiden Jahren am besten. Dahinter weisen DDPG Modelle durchschnittlich die besten Resultate auf, obwohl 2018 der DQLA leicht dominiert. Dessen bessere Ergebnisse sind auf eine geringere Standardabweichung des Portfoliowertes zurückzuführen. Dies weist auf eine niedrigere Handelsfrequenz hin, was insbesondere geringere Verluste in moderaten Abschwungphasen wie in 2018 impliziert.

Vergleich der Charakteristika. Eine unerwartete Beobachtung dieser Studie ist, dass mit Aktien höhere Gewinne erzielt werden als mit Warentermingeschäften oder ETFs (siehe den unteren Teil von Abbildung 5.1). In einer näheren Betrachtung werden Charakteristika in der Zeitreihe untersucht, mit denen die besseren Ergebnisse begründet werden können. Es zeigt sich, dass sowohl das Sharpe Ratio als auch die Performance signifikant[2] negativ mit der Volatilität einer Anlage korrelieren. Dadurch dass Aktien mit 0,68 die geringste Standardabweichung gegenüber Waren (4,85) und ETFs (0,72) aufweisen, sind diese Anlagen in diesem Evaluationssetting vorteilhaft. Genauer profitieren der DQLA und der DDPGA am meisten von der geringeren Volatilität, wohingegen der DDRQLA 2019 sogar schlechter abschneidet. Die Fähigkeit, besser auf alte Informationen durch die LSTM Architektur zuzugreifen, scheint ausschlaggebend für eine gute Performance bei instabilen Wertentwicklungen zu sein. Indes unterliegen stabile Wertpapiere weniger Spekulationen und Irregularitäten, womit die insgesamt bessere Mustererkennung erklären könnte. Nicht signifikant ins Gewicht fällt hingegen der Kursverlauf eines Wertpapiers. Erwartungsgemäß beeinflusst ein wachsender Kursverlauf die Performance leicht positiv, ist aber zumindest in diesem Kontext nicht hinreichend für bessere Ergebnisse und zeigt ferner keinen signifikanten Zusammenhang mit der Performance.

Diese Schlussfolgerungen gelten für alle Charakteristika bis auf den Vergleich einzelner Branchen. In diesem Fall schneiden die Agenten 2019 bei Aktien von Unternehmen aus

[2] Konkret wird getestet, ob der gemittelte Profit für jede Komposition aus Agent und Wertpapier signifikant mit der Volatilität zusammenhängt. Der Korrelationskoeffizient beträgt ca. -0,33.

dem Gesundheitswesen trotz einer geringeren Volatilität und einer höheren Wertsteigerung schlechter ab. Um diesen Zusammenhang zu untersuchen, wird der Einfluss des Beta-Faktors auf die Portfolioentwicklung für den Branchenvergleich zusätzlich untersucht. Da Aktien aus dem Gesundheits- oder Finanzwesen einen niedrigen Beta-Faktor besitzen, korrelieren sie weniger stark mit dem Markt als Technologieaktien. Diese Eigenschaft ist bei Aufwärtstrends wie beispielsweise im Jahr 2019 vorteilhaft, wenngleich Verluste in Abwärtstrends dadurch verstärkt werden. Allgemein lässt sich in den Ergebnissen eine statistisch signifikante Abhängigkeit zwischen der Varianz des Portfolios und dem Beta-Faktor messen. Demnach sollten Wertpapiere mit einem hohen Beta-Faktor nur bei einer risikofreudigen Nutzenfunktion eingesetzt werden. Für ein maximales Sharpe Ratio sollten jedoch Aktien gemieden werden, die sowohl eine hohe Volatilität als auch einen hohen Beta-Faktor besitzen.

5.4 Anwendung während der COVID-19 Pandemie

Die rechten Schaubilder in Abbildung 5.1 zeigen die überraschend gute Performance in Krisenzeiten. Jede DRL Variante erzielt in fast allen Fällen bessere Ergebnisse als die Baseline Agenten und weist sogar einen deutlich erhöhten Profit im Vergleich zum ersten Evaluationsjahr auf. Nur die DDPG Agenten weisen leicht schlechtere Tendenzen bei Aktien auf, speziell der DDPGAK reduziert seine Handelsfrequenz auffällig drastisch. Eine mögliche Erklärung dafür ist der stark erhöhte ^VIX, der in den Trainingsjahren häufig positiv mit Verlusten korreliert.

Die beiden Schaubilder in Abbildung 5.2 zeigen, wie sich die Handlungsentscheidungen des DDRQLA und DQLA der volatilen Krisenphase voneinander im Einzelnen unterscheiden: Die Handlungen beider Agenten bleiben verzögert hinter dem Markttrend zurück. Jedoch reagiert der DDRQLA besonders im ersten Quartal relativ früh auf die volatile Marktsituation, indem er seine Handelsfrequenz reduziert. Somit können hohe Verluste im späteren Verlauf vermieden werden. Die deutlich bessere Rendite des DDRQLA kann dadurch erklärt werden, dass anschließend im zweiten Quartal die günstige Marktsituation durch ein erhöhtes Kaufverhalten besser genutzt wird. Während der DQLA sein tendenziell konservatives Kaufverhalten beibehält, profitiert der DDRQLA von einer gierigen Strategie. Diese Erkenntnis deckt sich mit den Schlussfolderung aus Abschnitt 5.3, dass die in Paragraph 3.3.1 beschriebenen Vorteile von LSTM Schichten in volatilen Phasen deutlich die Performance verbessern. Ferner kann der rekurrente Agent die Unterstützungszone[3] im vierten Quartal besser erkennen, aber aufgrund der Einschränkung des Aktionsraums nicht schnell genug seinen Bestand erhöhen.

Trotz einer erhöhten Volatilität sind die Agenten bei großen Marktveränderungen profitabel. Allerdings kann man wegen des eher kleinen Krisen-Datensatzes nicht gesichert von einer

[3] Eine Unterstützungszone ist ein Begriff in der technischen Analyse von Finanzdaten. Er beschreibt ein Niveau, bei dem ein sinkender Wertpapierkurs dazu tendiert, nach oben abzuprallen. Wird dieses Niveau durchbrochen, ist es wahrscheinlich, dass der Wertpapierkurs weiter fällt.

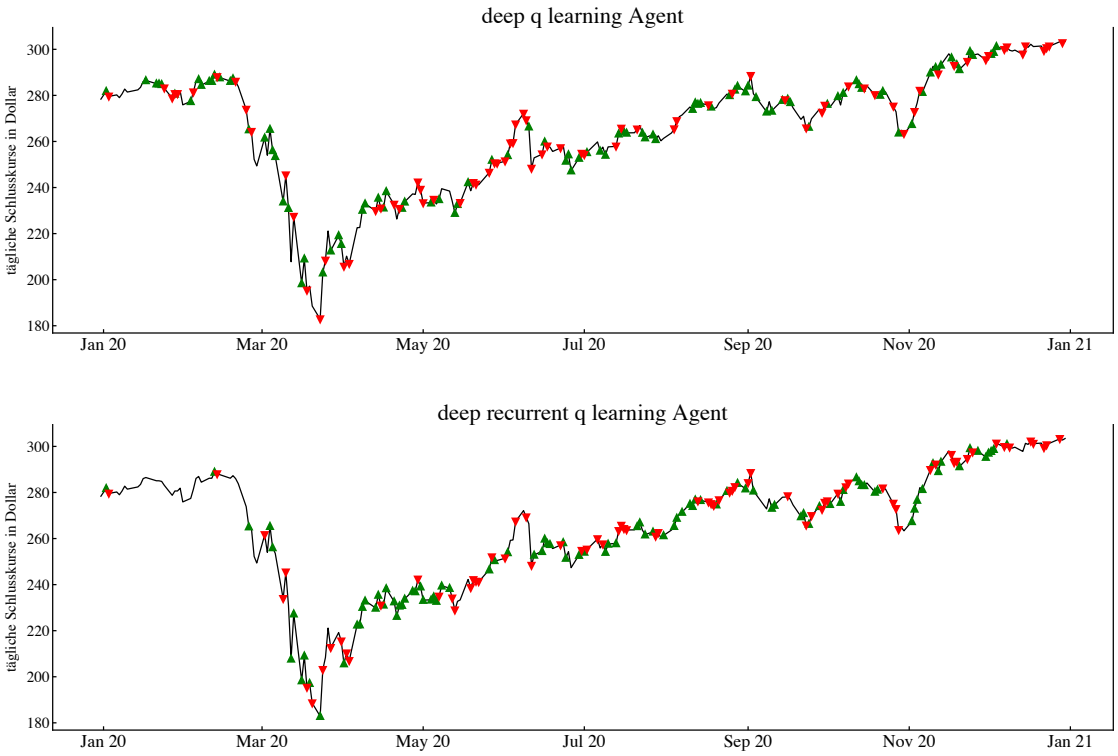

Abbildung 5.2: Ergebnisse des DQLA (oben) und des DDRQLA (unten). Die roten Pfeile signalisieren Verkaufszeitpunkte, grüne Pfeile zeigen Kaufzeitpunkte.

guten Performance in jeder Krisenzeit ausgehen. Da die Kurseinbrüche im ersten Quartal des Jahres eintreten, halten die Agenten aufgrund der strengen Handelsrestriktionen häufig nicht genügend Wertpapiere. Für eine genauere Einschätzung muss in weiteren Studien der Testzeitraum der COVID-19 Pandemie variieren, damit die Auswirkungen des Kurseinbruchs in diversen Szenarien untersucht werden können.

5.5 Einfluss der erweiterten Aspekte

Um detailliertere Erkenntnisse zu gewinnen, beschränkt sich die folgende Analyse auf Vergleiche zwischen Agenten, die denselben Trainingsalgorithmus verwenden.

Kontextdaten. Das Hinzufügen von Kontextdaten verbessert weder die Performance des DDPG Algorithmus (DDPGAK) noch des erweiterten DQL Ansatzes (EDQLA) in signifikantem Maße. Obwohl für den DDPGAK eine leicht bessere Performance in 2018 beobachtet wird, schneidet der Agent 2019 und 2020 schlechter ab. Im Vergleich dazu erleidet der EDQLA in den meisten Fällen durchschnittlich höhere Verluste, wobei allgemein ein instabileres Training mit einer signifikant längeren Laufzeit beobachtet werden kann.

Eine erhöhte Varianz sowohl bei der Validation als auch bei den Testergebnissen zeigt, dass die ergänzenden Marktinformationen in diesen simplen Modellen nicht vorteilhaft für die Entscheidungsfindung sind. Nichtsdestoweniger suggeriert eine qualitative Analyse der Handelszeitpunkte eine insgesamt geringfügig bessere Entscheidungsfindung in volatilen Phasen, die sich in einer reduzierten Handelsfrequenz bemerkbar macht. Daraus lässt sich ableiten, dass Kontextdaten bei risikoreichen Konjunkturphasen unterstützend wirken können. Da aber auch bei positivem Marktwachstum eine signifikant geringere Handelsfrequenz zu beobachten ist, können günstige Marktsituationen nur bedingt genutzt werden.

Wenngleich aufwendigere Agenten in einem anderen Setting [61, 60] mit Kontextdaten bessere Ergebnisse erzielen, können in dieser Evaluation durch korrelierende Marktinformationen keine signifikanten Unterschiede in der Performance beobachtet werden. Gegeben, dass die zusätzlichen Finanzmarktdaten in großem Maße die Anzahl der Eingabewerte erhöhen, lohnt sich diese Modifikation aufgrund der höheren Komplexität unter den Limitationen von Kapitel 4 somit nicht.

Dropout. Änderungen in der Dropoutrate ändern nicht signifikant die Performance der modifizierten DQL Variante, jedoch konnte das beste Ergebnis des Agenten in allen drei Jahren durch einen höheren Dropoutwert verbessert werden. Da die Mittelwerte der Validations- und Testergebnisse weniger stark voneinander abweichen, lässt sich ein tendenziell weniger überangepasstes Modell beobachten. Limitiert sind diese Schlussfolgerungen allerdings dadurch, dass der Effekt von Dropout nicht isoliert betrachtet werden kann. Um genauere Aussagen über die Überanpassung des Modells zu treffen, müsste in einem gesonderten Experiment der DQLA nur um die Regularisierungsmethode erweitert und den Einfluss auf die Performance statistisch analysiert werden. Das stellt ein interessantes zukünftiges Forschungsfeld dar, ist aber im Umfang dieser Bachelorarbeit nicht enthalten.

Aktionsraum. Die Erweiterung des Aktionsraums wirkt sich nicht signifikant auf die Performance des EDQLA aus, aber führt besonders im Krisenjahr aufgrund eines deutlich erhöhten Wertpapierbestandes im ersten Quartal 2020 zu schlechteren Resultaten. Trotz schwächerer Handelseinschränkungen kann der DQL Algorithmus ähnlich wie in Abbildung 5.2 den Aufwärtstrend im zweiten Quartal nicht nutzen. Stattdessen sorgt das erweiterte Framework dafür, dass wegen höherer Transaktionskosten volatile Phasen noch stärker die Performance beeinträchtigen. Statistisch kann ein erhöhter Korrelationskoeffizient um 41% für den negativen Zusammenhang der Volatilität mit der Performance nachgewiesen werden.

Allgemein investiert der Agent deutlich mehr, wodurch wie zu erwarten auch stärkere Ausreißer nach unten zu beobachten sind. Der Agent kann demnach die in Abschnitt 4.2.2 beschriebene Handelseinschränkung überwinden und einen größeren Teil seines Bargeldes nutzen, besitzt aber dadurch höhere Varianzen in den Ergebnissen. Folglich besitzt der Agent ein durchschnittlich geringeres Sharpe Ratio als der einfachere DQLA. Eine wichtige Folgerung aus diesem Experiment ist daher, dass bei einem simplen DQL Algorithmus mehr Handelsmöglichkeiten zu höherem Risiko führen. Allerdings mindern Transaktionskosten hohe Gewinne, weshalb diese Erweiterung für das verwendete Modell nicht eingesetzt

werden sollte.

6 Fazit

In dieser Arbeit wurden eine wirklichkeitsnahe Evaluationsumgebung entwickelt, um in einer groß angelegten Studie empirisch verschiedene DRL Architekturen und Implementierungen für den algorithmischen Handel differenziert zu analysieren. Das vorgestellte Testsetting stellt Standards in der Hyperparametersuche und dem Design der Agenten auf, sodass damit ausführlich Effekte einzelner Wertpapiercharakteristika und Designänderungen bei unterschiedlichen Marktverhältnissen evaluiert werden konnten.

Die Auswertung der Hyperparametereinflüsse ergab keine eindeutige Struktur. Jedoch lässt sich aus der Konfiguration der Modelle folgern, dass eine automatische Hyperparametereinstellung für jede Komposition aus Agent, Wertpapier und Zeithorizont ein wichtiger Bestandteil des Trainings darstellt. Eine zentrale Erkenntnis dabei ist, dass die Verwendung einer größeren Schichtgröße signifikant positiv mit der Handelsfrequenz korreliert, was zu besseren Ergebnissen in moderaten Aufschwungphasen führte. Überraschend erwies sich der Einsatz des Parameters τ für den DDPG Algorithmus als unwichtig in diesem Setting.

Aus den Testergebnissen lässt sich ableiten, dass die tatsächliche Performance der Agenten stark von der Wahl des Wertpapiers und des Testjahres abhängt. Insgesamt konnten sich verschiedene Varianten des DQL Algorithmus gegenüber anderen DRL Ansätzen und zwei Baseline Agenten durchsetzen. Keine DQL Variante zeigt allerdings eine signifikant bessere Performance gegenüber dem standardmäßigen DQL Ansatz. Jedoch weisen in volatilen Phasen Netzwerke mit einer LSTM Schicht eine bessere Entscheidungsfindung auf, während in Rezessionen naivere Ansätze besser abschneiden. Speziell die Kombination aus einer Dueling Strategie und einer LSTM Schicht erzielte sowohl ein besseres Sharpe Ratio als auch einen höheren durchschnittlichen Profit.

Allgemein stellt sich die Volatilität eines Wertpapiers als entscheidendste Charakteristik heraus, gefolgt vom Beta-Faktor. Während eine hohe Volatilität abträglich für eine gute Performance ist, erhöht ein größerer Beta-Faktor signifikant die Varianz des Portfolios. Bessere Ergebnisse anderer Wertpapiercluster konnten auf die Wechselwirkung dieser beiden Eigenschaften zurückgeführt werden. Modifikationen im Design wie die Eingabe zusätzlicher Marktinformationen, ein erweiterter Aktionsraum oder die Implementierung von Dropout erweitern das Modell, ohne in diesem Setting für den DDPG oder DQL Algorithmus signifikant die Performance zu verbessern. Im Speziellen reduzieren die korrelierenden Eingabewerte die Handelsfrequenz signifikant, was bei volatilen Marktgeschehnissen unterstützend, hingegen bei Aufschwungphasen hinderlich ist. Weiterhin wurde gezeigt, dass schwächere Handelsbeschränkungen durch einen größeren Aktionsraum insbesondere aufgrund hoher Transaktionskosten für die simplen Modelle zu erhöhtem Risiko und durchschnittlich mehr Verlusten führten.

Der algorithmische Handel unterliegt einer hohen Irregularität im Vergleich zu anderen Anwendungsfeldern von NNen. Dies stellt eine Hürde für Investoren dar, da die tatsächliche

Performance der Modelle stark von gewählten Zeiträumen, Wertpapieren und Konfigurationen abhängt. Mit dieser Studie wurde versucht, einige Vermutungen bzgl. Hyperparametern, Wertpapierclustern und Modifikationen empirisch für simple DRL Varianten zu untersuchen. In weiteren Studien könnten komplexere DRL Architekturen analysiert werden, um noch detaillierter die Wechselwirkungen von Wertpapiercharakteristika und den sequentiellen Entscheidungsprozess des algorithmischen Handels zu erkunden. Beispielsweise könnte der Effekt von Kontextdaten auf ein aufwendigeres Modell mit einem komplexeren NN isoliert erforscht werden. Insbesondere könnten möglicherweise andere DRL Varianten oder tiefere Netzwerke besser in der Lage sein, die multidimensionalen Muster der korrelierenden Marktinformationen zu erkennen. Es steht noch aus, Effekte von weiteren Kontextdaten wie z.B. Sentimentdaten oder fiskalpolitischen Maßnahmen experimentell zu überprüfen. Dank der entwickelten Testumgebung lassen sich diese Modifikationen vielseitig evaluieren, sodass weitere Intuitionen mit umfassenden experimentellen Ergebnissen wissenschaftlich gestützt werden können.

Während die realitätsnahe Testumgebung eine differenziertere Evaluation als zuvor beschriebene Studien ermöglicht, kann sowohl der Datensatz als auch die Konfiguration der Modelle noch umfassender verfeinert und erweitert werden. Erstens gibt es eine endlose Anzahl an Charakteristika, die man zusätzlich in die Testumgebung inkludieren könnte. Abgesehen von den untersuchten Wertpapierclustern könnten noch weitere Zusammenhänge mit Eigenschaften wie der Eigenkapitalquote, der Dividendenrendite oder des Streubesitzes zu signifikanten Unterschieden in der Performance führen. Zweitens würde ein erschöpfenderes Hyperparametertuning (vgl. [49, 4]) für bessere Ergebnisse und genauere Schlussfolgerungen sorgen. Damit wäre es insbesondere möglich, beispielsweise das fANOVA Framework [24] zu verwenden, um detailliertere Einsicht in die Bedeutung von Hyperparametern und deren Interaktionen zu erhalten. Drittens verwendet das einheitliche Framework bereits einige Implementationen, die den Lernprozess stabiler machen und gleichzeitig die Gefahr von Überanpassung reduzieren. Anpassungen in der Risikoadjustierung, etwa durch eine andere Belohnungsfunktion, könnten untersucht werden, um die starken Oszillationen im Training und die Schwankungen in den Ergebnissen zu reduzieren. Weight decay [62] oder spezifischere Regularisierungsverfahren [64] sind neben Dropout ebenfalls Methoden, mit denen bessere out-of-sample Ergebnisse erzielt werden könnten. In weiteren Studien könnte es deshalb von Interesse sein, durch diese Implementierungen homogenere Ergebnisse zu erhalten und für eine bessere Konvergenz im Training zu sorgen. Das benötigt allerdings eine aufwendige, individuelle Anpassung jedes Modells und ist daher nicht im Rahmen dieser Bachelorarbeit enthalten.

Anhang

Tabelle A1: Liste der Agenten

Zu sehen ist eine vollständige Liste aller fünfzehn verwendeter DRL Varianten und der zwei Baselines. Die ersten vier Varianten werden näher untersucht. Die Agenten sind entsprechend ihrer Performance in der Vorstudie sortiert. Deep curiosity q learning und verschiedene Modifikationen davon verwenden einen weiteren DRL Algorithmus, der unter anderem in [14] näher erläutert wird. Aufgrund der schlechteren Performance in der Vorstudie im Vergleich zu anderen Implementierungen wird auf diese Varianten daher nicht näher eingegangen. Außerdem erweist sich die Double Strategie (cf. [18]) als nicht Vorteil für den algorithmischen Handel in diesem Setting.

duel deep recurrent q learning
deep q learning
deep deterministic policy gradient
deep deterministic policy gradient mit Kontextdaten
deep recurrent q learning
double duel deep recurrent q learning
deep recurrent deterministic policy gradient
double deep recurrent q learning
duel deep q learning
duel deep recurrent deterministic policy gradient
double duel deep q learning
deep recurrent curiosity q learning
duel deep curiosity q learning
double deep q learning
duel deep deterministic policy gradient
deep curiosity q learning
turtle trading
moving average

Tabelle A2: Datensatz

Der nachstehende Datensatz umfasst alle Wertpapiere für die Vorstudie (grau hinterlegt), die Hauptevaluation und des Krisenjahres (unterstrichen). Weiterhin sind Kontextdaten enthalten, die dem DDPGAK und dem EDQLA zusätzlich übergeben werden. Die Wertpapiere werden zur besseren Übersicht in Wertpapierklassen und Börsen unterteilt. Absätze gruppieren die Wertpapiere hinsichtlich des Sektors, der Unternehmensgröße und

der Region. Hinter den geschweiften Klammern steht der weitere Vertreter des Sektors, dessen Wertentwicklung beim Handel mit Wertpapieren dieses Sektors zusätzlich in den Kontextdaten enthalten sind.

Ticker	Unternehmen		Ticker	Unternehmen	
Aktien (USA)			**Aktien (Asien)**		
ORCL	Oracle	Broadcom (AVGO)	0992.HK	Lenovo Group	Tencent (0700.HK)
AAPL	Apple		3888.HK	Kingsoft	
ACN	Accenture		0763.HK	ZTE	
MSFT	Microsoft		0939.HK	China Construction Bank	China Merchants Bank (3968.HK)
IBM	IBM		2318.HK	Ping An Insurance	
CSCO	Cisco		0998.HK	China CITIC Bank	
NVDA	Nvidia		**ETF's**		
ADBE	Adobe		DIA	Dow Jones	NYSE Composite (^NYA)
HPQ	HP		SPY	S&P 500	
INTC	Intel		QQQ	NASDAQ	
TESS	TESSCO Technologies		EWJ	Nikkei 225	HANG SENG Index (^HSI)
ASYS	Amtech Systems		EWT	iShares MSCI Taiwan	
CTG	Computer Task Group		EWY	iShares MSCI South Korea	
BELFB	Bel Fuse		EZU	FTSE 100	IBEX (IBEX)
AVNW	Aviat Networks		CAC.PA	Lyxor CAC 40	
LYTS	LSI Industries		EXS1.DE	iShares Core DAX	
JPM	JPMorgan Chase	Blackrock (BLK)	EXXY.MI	iShares Diversified Commodity	iShares S&P GSCI (GSG)
BAC	Bank of America		DBC	Invesco DB Commodity Index	
V	Visa Inc.		**Waren**		
PFE	Pfizer	AstraZeneca (AZN)	GC=F	Gold	Aluminum (ALI=F)
MRK	Merck & Co.		SI=F	Silver	
JNJ	Johnson & Johnson		PL=F	Platinum	
CAJ	Canon	China Telecom (CHA)	HG=F	Copper	
NICE	NICE		CL=F	Crude Oil	Brent Crude Oil (BZ=F)
TSM	Taiwan Semiconductor		HO=F	Heating Oil	
SNE	Sony		NG=F	Natural Gas	
UMC	United Microelectronics		RB=F	RBOB Gasoline	
CHKP	Check Point Software		HE=F	Lean Hogs	Class III Milk (DC=F)
SILC	Silicom		LE=F	Live Cattle	
GILT	Gilat Satellite Networks		GF=F	Feeder Cattle	
TSEM	Tower Semiconductor		DC=F	Class III Milk ⟶ Live Cattle (LE=F)	
LFC	China Life Insurance	Nomura Holdings (NMR)	ZC=F	Corn	Cocoa (CC=F)
SMFG	Sumitomo Mitsui		ZS=F	Soybean	
SHG	Shinhan		KC=F	Coffee	
NOK	Nokia	STMicroelectronics (STM)	KE=F	KC HRW Wheat	
ASML	ASML		**Kontextdaten**		
ERIC	LM Ericsson		^SPY	S&P 500 Trust	
SAP	SAP		^TNX	Treasury Yield	
TEL	TE Connectivity		^VIX	Volatility Index	
LOGI	Logitech		GC=F	Gold Feb 21	
HSBC	HSBC	Credit Suisse (CS)			
ING	ING				
BCS	Barclays				

Literatur

[1] `https://tex.stackexchange.com/questions/523603/latex-figure-for-deep-reinforcement-learning`.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage und A. A. Bharath. „Deep Reinforcement Learning: A Brief Survey“. In: (2017).

[3] D. H. Bailey, J. Borwein, M. L. de Prado und Q. J. Zhu. „Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance“. In: *Optimization eJournal* (2014).

[4] J. Bergstra und Y. Bengio. „Random search for hyper-parameter optimization.“ In: *Journal of machine learning research* (2012).

[5] B. S. Bernank. *The Relationship Between Stocks and Oil Prices*. 2016.

[6] I. Boukas, D. Ernst, T. Théate, A. Bolland, A. Huynen, M. Buchwald, C. Wynants und B. Cornélusse. *A Deep Reinforcement Learning Framework for Continuous Intraday Market Bidding*. 2020.

[7] R. Caruana, S. Lawrence und L. Giles. „Overfitting in Neural Nets: Backpropagation,Conjugate Gradient, and Early Stopping“. In: (2001).

[8] E. P. Chan. *Algorithmic Trading: Winning Strategies and their Rationale*. 2013.

[9] L. Chen und Q. Gao. „Application of Deep Reinforcement Learning on Automated Stock Trading“. In: *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. 2019.

[10] L. Chen und Q. Gao. „Application of Deep Reinforcement Learning on Automated Stock Trading“. In: *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. 2019.

[11] E. K. Chowdhury, I. I. Khan und B. K. Dhar. „Catastrophic impact of Covid-19 on the global stock markets and economic activities“. In: *Business and Society Review* (2021).

[12] Y. Deng, F. Bao, Y. Kong, Z. Ren und Q. Dai. „Deep Direct Reinforcement Learning for Financial Signal Representation and Trading“. In: *IEEE Transactions on Neural Networks and Learning Systems* (2017).

[13] R. Elsas, M. El-Shaer und E. Theissen. „Beta and Returns Revisited: Evidence From the German Stock Market“. In: *Journal of International Financial Markets, Institutions and Money* (2000).

[14] A. Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2017.

[15] F. Gers, J. Schmidhuber und F. Cummins. „Learning to Forget: Continual Prediction with Lstm Learning to Forget: Continual Prediction with Lstm“. In: 1999.

[16] D. Guo, W. Zhou, H. Li und M. Wang. „Hierarchical LSTM for Sign Language Translation“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2018).

[17] B. Hahn. *Numerik 1: Definitionen und Sätze*. University of Wuerzburg. 2020.
[18] H. van Hasselt, A. Guez und D. Silver. „Deep Reinforcement Learning with Double Q-Learning“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2016).
[19] H. van Hasselt, A. Guez und D. Silver. *Deep Reinforcement Learning with Double Q-learning*. 2015.
[20] M. J. Hausknecht und P. Stone. „Deep Recurrent Q-Learning for Partially Observable MDPs“. In: *CoRR* (2015).
[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever und R. R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012.
[22] S. Hochreiter, Y. Bengio, P. Frasconi und J. Schmidhuber. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.
[23] C. Y. Huang. *Financial Trading as a Game: A Deep Reinforcement Learning Approach*. 2018.
[24] F. Hutter, H. Hoos und K. Leyton-Brown. „An Efficient Approach for Assessing Hyperparameter Importance“. In: *Proceedings of the 31st International Conference on Machine Learning*. PMLR, 2014.
[25] G. Jeong und H. Y. Kim. „Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning“. In: *Expert Systems with Applications* (2019).
[26] S. Kaur. „Algorithmic Trading using Sentiment Analysis and Reinforcement Learning“. In: 2017.
[27] D. P. Kingma und J. Ba. *Adam: A Method for Stochastic Optimization*. 2017.
[28] V. Konda und J. Tsitsiklis. „Actor-Critic Algorithms“. In: *Society for Industrial and Applied Mathematics* (2001).
[29] G. Lample und D. S. Chaplot. *Playing FPS Games with Deep Reinforcement Learning*. 2018.
[30] F. Lautenschlager, M. Becker, K. Kobs, M. Steininger, P. Davidson, A. Krause und A. Hotho. „OpenLUR: Off-the-shelf air pollution modeling with open features and machine learning“. In: *Atmospheric Environment* (2020).
[31] B. LeBaron. „Forecast Improvements Using a Volatility Index“. In: *Journal of Applied Econometrics* (1992).
[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver und D. Wierstra. *Continuous control with deep reinforcement learning*. 2019.
[33] L. Lin. „Reinforcement learning for robots using neural networks“. In: 1992.
[34] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals und W. Zaremba. *Addressing the Rare Word Problem in Neural Machine Translation*. 2015.
[35] G. M, A. A, J. M, L. T, S. R und G. M. *COVID-19 and economy*. 2020.
[36] S. Merity, N. S. Keskar und R. Socher. *Regularizing and Optimizing LSTM Language Models*. 2017.
[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra und M. Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013.

[38] J. Moody und Lizhong Wu. „Optimization of trading systems and portfolios“. In: *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*. 1997.

[39] J. Moody und M. Saffell. „Reinforcement Learning for Trading“. In: *Advances in Neural Information Processing Systems*. MIT Press, 1999.

[40] G. Oh. „Grouping characteristics of industry sectors in financial markets“. In: *Physica A Statistical Mechanics and its Applications* (2014).

[41] S. Perrin und T. Roncalli. *Machine Learning Optimization Algorithms Portfolio Allocation*. 2019.

[42] L. Ryll und S. Seidens. *Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey*. 2019.

[43] L. Ryll und S. Seidens. *Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey*. 2019.

[44] J. J. Seigel. *Stocks for the long run*. MGH, 2020.

[45] O. B. Sezer, M. U. Gudelek und A. M. Özbayoglu. „Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019“. In: *CoRR* (2019).

[46] Y. Shapira, D. Kenett und E. Ben-Jacob. „The Index cohesive effect on stock market correlations“. In: *The European Physical Journal B* (2009).

[47] W. Si, J. Li, P. Ding und R. Rao. „A Multi-objective Deep Reinforcement Learning Approach for Stock Index Future's Intraday Trading“. In: *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*. 2017.

[48] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel und D. Hassabis. „Mastering the game of Go with deep neural networks and tree search“. In: *Nature* (2016).

[49] J. Snoek, H. Larochelle und R. P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012.

[50] B. Solnik, C. Boucrelle und Y. L. Fur. „International Market Correlation and Volatility“. In: *Financial Analysts Journal* (1996).

[51] S. Sumner, R. Johnson und L. Soenen. „Spillover Effects Among Gold, Stocks, and Bonds“. In: *Journal of CENTRUM Cathedra: The Business and Economics Research Journal* (2011).

[52] M. Sundermeyer, R. Schlüter und H. Ney. *LSTM neural networks for language modeling*. 2012.

[53] J. W. Tan. *Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy*. `https://github.com/AI4Finance-LLC/Deep-Reinforcement-Learning-for-Automated-Stock-Trading-Ensemble-Strategy-ICAIF-2020`. 2019.

[54] T. Théate und D. Ernst. *An Application of Deep Reinforcement Learning to Algorithmic Trading*. 2020.

[55] D. Vezeris, I. Karkanis und T. Kyrgos. „AdTurtle: An Advanced Turtle Trading System“. In: *Journal of Risk and Financial Management* (2019).

[56] Y. Wang, D. Wang, S. Zhang, Y. Feng, S. Li und Q. Zhou. *Deep Q-trading*. 2017.

[57] Z. Wang, N. Freitas und M. Lanctot. *Dueling Network Architectures for Deep Reinforcement Learning*. 2015.

[58] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia und H. Fujita. „Adaptive stock trading strategies with deep reinforcement learning methods“. In: *Information Sciences* (2020).

[59] L. Y., N. P. und V. Chang. *Application of deep reinforcement learning in stock trading strategies and stock forecasting*. 2019.

[60] H. Yang, X.-Y. Liu, S. Zhong und A. Walid. *Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy*. `https://github.com/jiewwantan/StarTrader`. 2020.

[61] H. Yang, X.-Y. Liu, S. Zhong und A. Walid. *Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy*. 2020.

[62] X. Ying. „An overview of overfitting and its solutions“. In: *Journal of Physics: Conference Series*. Bd. 1168. 2019.

[63] I. Zamora, N. G. Lopez, V. M. Vilches und A. H. Cordero. *Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo*. 2017.

[64] W. Zaremba, I. Sutskever und O. Vinyals. *Recurrent Neural Network Regularization*. 2015.

[65] Z. Zhang, S. Zohren und S. Roberts. *Deep Reinforcement Learning for Trading*. 2019.

[66] H. Zolkepli. *Stock-Prediction-Models*. `https://github.com/huseinzol05/Stock-Prediction-Models`. 2018.