HAW HAMBURG

Bachelorarbeit

Michal Augustyn Bilinski

Konzeption und Implementierung einer Software, die Handelsinformationen von Kryptobörsen als Datenströme verarbeitet und über Arbitragemöglichkeiten informiert.

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Michal Augustyn Bilinski

Konzeption und Implementierung einer Software, die Handelsinformationen von Kryptobörsen als Datenströme verarbeitet und über Arbitragemöglichkeiten informiert.

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Martin Schultz

Eingereicht am: 28. Februar 2022

Michal Augustyn Bilinski

Thema der Arbeit

Konzeption und Implementierung einer Software, die Handelsinformationen von Kryptobörsen als Datenströme verarbeitet und über Arbitragemöglichkeiten informiert.

Stichworte

Kryptowährung, Arbitrage, Data Mining, neural network

Kurzzusammenfassung

In dieser Arbeit wird der Entwurf und die Implementierung einer Software vorgestellt, die Handelsinformationen von Kryptobörsen als Datenströme verarbeitet und Informationen über Arbitragemöglichkeiten liefert. Ein rekurrentes neuronales Netzwerk versucht, die Währungspaare mit den häufigsten Arbitragemöglichkeiten zu identifizieren, um die Datenströme des Systems zu filtern. Verschiedene Handelsdaten werden für die Vorhersage gruppiert und ausgewertet. Das Forschungsergebnis zeigt, dass die Genauigkeit des neuronalen Netzes sehr gering ist.

Michal Augustyn Bilinski

Title of Thesis

Design and implementation of software that processes trading information from crypto exchanges as data streams and informs arbitrage opportunities.

Keywords

cryptocurrencies, arbitrage, data mining, neural network

Abstract

This paper presents the design and implementation of software that processes trading information from crypto exchanges as data streams and provides information about arbitrage opportunities. A recurrent neural network attempts to identify the currency pairs

with the most frequent arbitrage opportunities to filter the system's data streams. Different types of trading data are grouped and evaluated for prediction. The research result shows that the accuracy of the neural network is very low.

Inhaltsverzeichnis

1 Einleitung und Motivation

Heutzutage ist es durchaus üblich, Daten in Form von Bildsequenzen zu finden. Mithilfe von neuronalen Netzen können Vorhersagen über das nächste Bild gemacht werden. Ziel dieser Arbeit ist es zu untersuchen, ob sich auch abstrakte Dinge als Bilder darstellen und vorhersagen lassen, in diesem Fall Börseninformationen. Es wird untersucht, ob die Technik des langen Kurzzeitgedächtnisses in faltbaren neuronalen Netzen (englisch: Convolutional LSTM, kurz: ConvLSTM) erfolgreich bei der Gestaltung eines Arbitrage-Systems eingesetzt werden kann. Das neuronale Netz soll selektiv über die Daten der Kryptobörsen operieren und nur die vielversprechendsten Währungen auswählen. Die Daten fließen als Datenstrom durch das System, was eine deklarative Verarbeitung der Daten ermöglicht.[6, vgl. Kapitel 3.1.1.]

1.1 Trianguläre Arbitrage

Unter Arbitrage versteht man im Allgemeinen die Ausnutzung einer Fehlbewertung auf dem Markt.[16, vgl. S. 234–235] Trianguläre Arbitrage (auch Dreiecksarbitrage) ist eine Handelstechnik, die darauf abzielt, von einer Preisdiskrepanz zwischen drei verschiedenen Währungspaaren an derselben Börse zu profitieren. Bei der Dreiecksarbitrage wird die Preisdifferenz zwischen einem berechneten Kreuzkurs eines Währungspaares und seinem tatsächlichen Kurs ausgenutzt. Der Kreuzkurs wird mithilfe von zwei anderen Währungspaaren berechnet. Wenn es eine Diskrepanz zwischen dem berechneten Kreuzkurs und dem tatsächlichen Wert eines Währungspaares gibt, kann der Umtausch durch alle drei Währungspaare nacheinander zu einem Gewinn führen.[1, vgl.] Eine detaillierte mathematische Erklärung findet sich in Unterabschnitt 3.2.2.

Arbitrage ist etwas, das seit Jahren auf den Devisenmärkten praktiziert wird und auch auf die Kryptowährungsmärkte angewendet werden kann. In einem effizienten Markt ist Arbitrage hart umkämpft und laut Literatur kaum oder gar unmöglich, deshalb wird sie

auch als Heiliger Gral der Finanzen bezeichnet.[16, vgl. S. 234–235] Ein effizienter Markt würde bedeutet, dass jede neue Information bei der Bildung des Marktpreises berücksichtigt ist. Der Markt hingegen sei gerade effizient genug, um die Anleger für ihre Kosten und Risiken zu entschädigen. Einer der Implikationen der Markteffizienzhypothese sei die Existenz von Arbitrage.[16, vgl. S. 233ff] Sobald die Arbitrage eingelöst wurde, besteht die Preisdiskrepanz nicht mehr und ist somit für alle anderen Marktteilnehmer nicht mehr einlösbar. Folglich kann die Arbitragemöglichkeit nur von einem Marktteilnehmer gewinnbringend aufgelöst werden.

Theoretisch ist Arbitrage risikofrei und der Arbitrageur trägt somit kein Risiko. Dreiecksarbitrage ist risikofrei, weil die Arbitrage theoretisch direkt umgesetzt wird und die Transaktionsprozesse zu erwarteten Kursen stattfinden. Die Arbitrage wird lediglich vom schnellsten Arbitrageur wahrgenommen.[24, vgl.]

1.2 Maschinelles Lernen

In dieser Thesis wird ein neuronales Netz vorgestellt, das eine Auswahl von Währungspaaren vornimmt, um Computerressourcen zu sparen und damit die Latenzzeit der Analysedaten zu verringern. Das neuronale Netz hat die Aufgabe, die Anzahl der positiven Kreuzkurse für ein zukünftiges Zeitintervall vorherzusagen. Um diese Zeitreihenanalyse möglicherweise zu verbessern, werden weitere abhängige Kursdaten als zusätzliche Merkmale zu den Eingabedaten des neuronalen Netzes hinzugefügt. Um dies zu ermöglichen, werden die Handelsdaten Zeitfenstern zugeordnet, am Beispiel von [26]. Die Handelsdaten sind jeweils mit den drei Symbolen der Dreiecksarbitrage verbunden. Ziel ist es, die gegenseitige Beeinflussung von Kreuzkursvorkommen sowie deren andere Eigenschaften unter einem räumlichen Aspekt zu untersuchen.

1.3 Datenstromverarbeitung

In dem vorgestellten System werden die Börsendaten in Datenströmen verarbeitet. Bei der Datenstromverarbeitung wird der kontinuierliche Datenstrom ohne Unterbrechung verarbeitet. Dies hat den Vorteil, dass ein Analyseprozess zeitnah auf den eingehenden

Daten durchgeführt wird, ohne dass die Daten zuvor aggregiert werden müssen. Je länger die Aktionszeit zwischen Datenentstehung bzw. Dateneingang und Datenanalyse ist, desto höher ist der Wertverlust der Analyseergebnisse.[5, vgl. S. 59–60]

Jede Form von Daten kann als Datenstrom beobachtet werden. Dabei spielt es keine Rolle, ob die Daten aus einer Datenbank, Maus-/Tastaturinteraktionen, einer statischen Konfiguration oder einer externen Quelle stammen. Der Datenerzeuger informiert den Beobachter, wenn er eine Nachricht erzeugt.[6, vgl. Kapitel 1.5.] [6, vgl. Kapitel 1.5.1.] In der Datenstromverarbeitung wird das Beobachtermuster verwendet, welches ein Entwurfsmuster der Softwareentwicklung ist. Das Beobachtermuster gehört zur Kategorie der Verhaltensmuster und wird verwendet, um Informationen an den interessierten Beobachter weiterzugeben, wenn sich das beobachtete Objekt ändert.[8, vgl.]

Deklaratives Programmierparadigma

Bei der praktischen Umsetzung dieser Arbeit wird die reaktive Programmierung verwendet. Die reaktive Programmierung folgt dem Konzept des deklarativen Programmierparadigmas. Bei diesem Stil werden die Programmanweisungen als Beschreibung dessen gegeben, was der Programmierer als Ergebnis erreichen möchte.[6, vgl. Kapitel 3.1.1.] Die reaktive Programmierung bildet den Mechanismus, um Nachrichten in einem reaktiven System zu verarbeiten und sie an die Verarbeitungskette der Anwendung weiterzuleiten. Asynchrone Kommunikation ermöglicht es den Empfängern, nur dann Ressourcen zu verbrauchen, wenn sie aktiv sind, was zu einem geringeren System-Overhead führt.[6, vgl. Kapitel 1.3.][2, vgl.] Die Reaktionsfähigkeit ist der wichtigste Aspekt. Ein reaktionsfähiges System verwendet asynchrones Nachrichtenübermittlung, um Komponenten zu entkoppeln.[6, vgl. Kapitel 1.3.4.]

1.4 Ziele

1.4.1 Automatisiertes System

Das System ermittelt, welche Symbole die höchste Wahrscheinlichkeit für eine Dreiecksarbitrage aufweisen. Die ausgewählten Symbole können dann verwendet werden, um die Echtzeitdatenströme zu filtern, noch bevor das System die Börsendaten weiter auf Arbitrage untersucht.

1.4.2 Vorschlag für die Vorhersage von Kreuzkursvorkommen

Das in [26] verwendete Deep Learning Framework hat gute Eigenschaften für die Vorhersage von Autounfällen. Erstens erkennt das ConvLSTM zeitliche Autokorrelationen und zweitens berücksichtigt der Faltungsoperator lokale räumliche Eigenschaften[26, vgl.]. In dieser Arbeit wird untersuchen, ob dies auch für eine Kreuzkursvorhersage mit den zugehörigen Symbolen und anderen Handelsdaten gilt.

1.5 Gliederung

Die Thesis ist in 7 Abschnitte aufgeteilt. Einige der Abschnitte sind wiederum in Unterabschnitte aufgeteilt.

Kapitel 2 stellt die verwandten Themen vor, auf denen dieses Werk aufbaut. Kapitel 3 stellt die gesammelten Daten und die Formalisierung des Problems vor. Kapitel 4 stellt die Eigenschaften der Eingabedaten vor, welche das künstliche neuronale Netz berücksichtigt. Kapitel 5 stellt die Lösung für die Datenstromfilterung durch Selektion vor. Die Kapitel 6 stellt die Implementierung des Systems vor. Kapitel 7 stellt die Leistung des ConvLSTM für die Vorhersage vor. Kapitel 8 zieht die Schlussfolgerung aus den Messergebnissen und beantwortet die Frage, inwieweit die Kreuzkursvorhersage als räumlich-zeitliches Problem formuliert werden kann.

2 Verwandte Arbeiten

Vorhersage von Verkehrsunfällen auf der Grundlage heterogener räumlich-zeitlicher Daten: In dieser Arbeit wird eine detaillierte Studie über das Problem der Verkehrsunfallvorhersage unter Verwendung des künstlichen neuronalen Netzes Convolutional Long-Short-Term Memory (ConvLSTM) durchgeführt. Diese Arbeit zeigt, dass Deep Learning-Techniken wie ConvLSTM vielversprechende Lösungen für die Vorhersage von Verkehrsunfällen sind, wenn einzigartige Dateneigenschaften wie die räumliche Heterogenität gut behandelt werden. Um der räumlichen Heterogenität zu begegnen, werden die räumlichen Beziehungen der Straßen, sowie andere Parameter in die Eingabedaten des neuronalen Netzes hinzugefügt.[26, vgl.]

Ein Katalog von Stromverarbeitungsoptimierungen: Dieser Artikel ist wie ein Katalog aufgebaut, um die Terminologie in Bezug zur Datenstromoptimierung zu konsolidieren, ähnlich wie ein Katalog zu Entwurfsmustern. Die Grundlage dieser Arbeit bildet das Phänomen, dass die Datenrate im Verhältnis zur Rechenleistung hoch ist und Wartezeiten bei der Datenstromverarbeitung entstehen können.[10, vgl. S. 5–6]

Um die Datenströme in der vorgeschlagenen Anwendung zu optimieren, wird das in [26] vorgeschlagene Deep Learning Framework als selektiver Operator verwendet, um die Börsendaten während der Datenstromverarbeitung zu filtern.

3 Übersicht

In diesem Abschnitt werden die gesammelten Daten vorgestellt und die Formulierung des Problems erläutert.

3.1 Quellen der Daten

Für die Herkunft der Echtzeitdaten wird die Gemini WebSocketAPI[1] verwendet. Alle Daten haben einen zugeordneten Zeitstempel.

Orderbuch: In einem Orderbuch werden alle Kauf- und Verkaufsaufträge gesammelt. Orderbücher gibt es für alle börsengehandelten Wertpapiere. Ein Orderbuch wird entweder der Ankaufs- bzw. der Verkaufsseite zugeordnet.[22, S. 415]

Balkenchart: Ein Balkenchart enthält Preisinformationen zu einem Währungspaar für ein festes Intervall (z.B.: 1-minütiges, 5-minütiges, usw.), welche den Einstiegs- und Ausgangspreis in das Intervall bzw. aus dem Intervall, sowie dem höchsten bzw. niedrigsten Wert innerhalb des Intervalls enthält.[21, vgl.]

Handelsabschluss: Ein Handelsabschluss findet genau dann statt, wenn ein neues Kauf- oder Verkaufsgeschäft auf dem Markt ausgeführt wird. In diesem Fall werden die Symbole auf dem Spotmarkt der Kryptobörse Gemini gehandelt. Einem Handel werden der Preis und die Menge sowie die Information, ob es sich um einen Kauf oder einen Verkauf handelt, zugeordnet.[15, vgl.]

Kreuzkurs: Wechselkurs zweier Währungen, der sich aus den Wechselkursen dieser beiden Währungen gegenüber einer dritten Währung ergibt; z. B. der Kurs des Euro gegenüber dem Dollar, errechnet aus den Kursen des US-Dollars gegenüber dem Euro und des US-Dollars gegenüber dem Dollar.[23, vgl.] Der Kreuzkurs ist ein aus den Orderbüchern errechneter Wert.

[1] Quelle: https://docs.gemini.com/websocket-api/ Zugriffsdatum: 10.02.2022

3.2 Formulierung des Problems

3.2.1 Vorhersage

Die mathematische Beschreibung des Problems orientiert sich an der Problemformulierung in [26]. Die Vorhersage von positiven Kreuzraten wird als ein räumlich-zeitliches Problem formuliert, bei dem sowohl die Eingabe- als auch die Ausgabedaten räumlich-zeitliche Sequenzen sind. Hierbei ist wichtig zu erwähnen, dass die eingesetzten Zeitreihenwerte in feste Intervalle überführt werden. Dies hat das Ziel, möglichst viele Daten in die Vorhersage homogen miteinzubeziehen, weil einem Intervall keine kontinuierlichen Daten zugeordnet werden können. Vorgeschlagen wird ein räumliches Raster S mit $d \times d$, in welchem jeder Rasterquadrant in S einem Symboldreieck zugeordnet wird. Ein mögliches Symboldreieck ist in Unterabschnitt 3.2.2 angegeben. Das ConvLSTM-Netz ist in der Lage, räumlich-zeitliche Korrelationen d übergreifend zu erfassen. Werden 100 Dreiecksarbitragen betrachtet, so ist das Raster beispielsweise 10×10 Quadranten groß.

In diesem Problem soll das Modell die Anzahl von positiven Kreuzkursen für jeden Rasterquadranten in S für ein zukünftiges Zeitfenster vorhersagen. Die Zeitfenster können dabei ein beliebiges Intervall haben. Der Wert von d und t ist frei wählbar. In dieser Thesis ist das Zeitfenster t jedoch als 1 Minute definiert und der Wert von d ist 2. Zeitlich-variable Merkmale, wie Kurspreise und Handelsdaten sind für das Zeitfenster t nicht verfügbar bis t abgeschlossen ist. Von daher werden nur Eigenschaften aus den Zeitfenstern vor t bis hin zu $t-1$ zur Vorhersage der Kreuzkurse in Betracht gezogen.

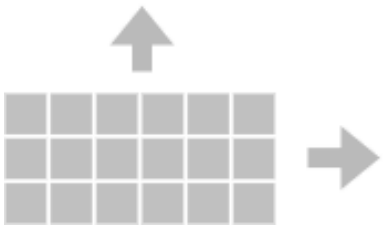


Abbildung 3.1: Diese Abbildung zeigt das Raster der Symboldreiecke. Jedem Feld ist ein Symboldreieck, das Kreuzkursvorkommen, sowie die zugehörigen Handelsdaten als zusätzliche Eigenschaften zugeordnet.

Gegeben:

- ein räumlich-temporales Feld $S \times T$, in welchem $S = s_1, s_2, ..., s_n$ ein räumliches Raster ist, und $T = t_1, t_2, ..., t_n$ das Zeitfenster für die Untersuchungsperiode ist, aufgeteilt in gleichlange Zeitfenster
- ein 3D Tensor mit positivem Kreuzkurs C, in welchem $C(s, t)$ die Anzahl der positiven Kreuzkurse $s \in S$ während des Zeitfensters ist $t \in T$
- eine Liste m Feature Tensoren $F = f_1, f_2, ..., f_m$, in welchem f_k ein dreidimensionaler Tensor ist, der das entsprechende Attribut in jedem Gitter s_i während jeden Zeitfensters t aufzeichnet
- ein Trainingsdatensatz $D_{train} = C(S, t), F(S, t)$, in welchem $t \in T_{train}$, und ein Testdatensatz $D_{test} = C(S, t), F(S, t)$, in welchem $t \in T_{test}$

zu Finden:

- ein Modell, welches $C(s, t)$ für jedes $t \in T_{test}$ vorhersagt.

Ziel:

- den Vorhersagefehler minimieren

Einschränkungen:

- Abhängigkeiten zwischen C und F variieren räumlich
- $F(S, t_i)$ ist für die Vorhersage von $C(S, t_i), t_i \in T_{test}$ nicht verfügbar

3.2.2 Kreuzkursberechnung

Im folgenden Beispiel wird die Kreuzkursberechnung für die drei Währungen USD, BTC und LTC betrachtet. Die Symbole lauten USD-LTC, USD-BTC und BTC-LTC. Gehen wir davon aus, dass ein Händler zunächst Dollar (USD) hält. Eine Möglichkeit, die Arbitrage auszunutzen, bestünde darin, eine Reihe von Transformationen durchzuführen, die diese 3 Währungen einschließen [25, vgl. S. 344,vgl. S. 174–177][7, vgl. S. 1105–1123]:

$$\text{USD} \rightarrow \text{BTC} \rightarrow \text{LTC} \rightarrow \text{USD} \tag{3.1a}$$

Die eine Möglichkeit für die drei Währungspaare wäre BTC für USD zu kaufen, dann LTC für BTC zu kaufen und anschließend LTC für USD zu verkaufen:

$$\alpha_1(t) = R_{\text{bid}}(\text{USD/BTC}, t) \cdot R_{\text{bid}}(\text{BTC/LTC}, t) \times \frac{1}{R_{\text{ask}}(\text{USD/LTC}, t)} - 1 \tag{3.2a}$$

Ist der Kreuzkurs α_1 größer als 0 und höher als die Gebühren aller Transaktionen einer Arbitrage, ist die Arbitrage profitabel.

Der alternative Ansatz wäre:

$$\text{USD} \rightarrow \text{LTC} \rightarrow \text{BTC} \rightarrow \text{USD} \tag{3.3a}$$

Die andere Möglichkeit für die drei Währungspaare wäre LTC für USD zu verkaufen, dann LTC für BTC zu verkaufen und BTC für USD zu kaufen:

$$\alpha_2(t) = \frac{1}{R_{\text{ask}}(\text{USD/LTC}, t)} \cdot \frac{1}{R_{\text{ask}}(\text{BTC/LTC}, t)} \quad \times R_{\text{bid}}(\text{USD/BTC}, t) - 1 \tag{3.4a}$$

4 Extraktion der Eigenschaften

Für jedes Zeitfenster werden die gesammelten Datensätze mit (S, t_i) gepaart, um die Liste der benötigten Eigenschaften zu erzeugen.

4.1 Zeitinvariante Eigenschaften

In diesem Fall gibt keine Zeitinvarianten Eigenschaften.

4.2 Zeitvariante Eigenschaften

Kreuzkursvorkommen-Eigenschaft (CC)**:** Die Anzahl aller positiven Kreuzkursvorkommen.

Orderbuch-Eigenschaften (OB)**:** Die Orderbuchdaten werden $OB(s_i, t)$ aus dem Durchschnitt des totalen Orderbuchwertes eines Währungspaares im Intervall t berechnet. Dies resultiert in 16 Eigenschaften.

Balkenchart-Eigenschaften (KP)**:** Balkenchart-Kerzen sind Größen, die kontinuierlich weiterlaufen. Für die Balkenchart-Eigenschaften werden die laufenden Durchschnitte der drei Währungspaare 1-minütig gemessen, auf das Zeitintervall t gemittelt und als Eigenschaft extrahiert. Dies resultiert in 15 Eigenschaften.

Handelsabschluss-Eigenschaften (T)**:** Auch hier wird der Schnitt für das Zeitintervall t für die Handelsgröße und Handelspreis in Verkaufs- und Ankaufrichtung gebildet. Dies ergibt 12 Eigenschaften.

4.3 Zusammenfassung der Eigenschaften

Es sind 46 Eigenschaften aufgestellt, welche in 4 Kategorien unterteilt sind: Kreuzkursvorkommen-Eigenschaft (CC) (1 Eigenschaft), Orderbuch-Eigenschaften (OB) (16 Eigenschaften), Balkenchart-Eigenschaften (KP) (15 Eigenschaften) und Handelsabschluss-Eigenschaften (T) (12 Eigenschaften). Jede Eigenschaft wird in einen drei-dimensionalen Tensor $2 \times 2 \times 1$ umgewandelt.

5 Ansatz der ConvLSTM basierten Datenselektion

In diesem Abschnitt wird die Lösung für die Gesamtarchitektur, sowie für die Vorhersage von Kreuzkursvorkommen vorgestellt. Zunächst wird die grundlegende Architektur erklärt, gefolgt von der Funktionsweise des ConvLSTM. Es ist wichtig zu verstehen, dass nicht die Größe des Kreuzverlaufs vorhergesagt wird, sondern die Häufigkeit des Auftretens eines positiven Kreuzverlaufs für ein zukünftiges Intervall t. Der Kreuzkurs wird dann mit der Formel berechnet, die in Unterabschnitt 3.2.2 erläutert ist.

5.1 Systemablauf

Die erste Komponente des Systems ist das Data-Mining. Auf der Grundlage der gesammelten Daten trifft das System Vorhersagen, mit wessen Resultat das Data-Mining gezielt die Börsendaten abfragen kann. Somit sind die Eingabe und auch die Ausgabe des Systems Symbole. Das Grundprinzip wird von Abbildung 5.2 abgebildet. Die Optimierung des Arbitrage-Systems erfolgt durch das Filtern von Symbolen durch das neuronale Netz.

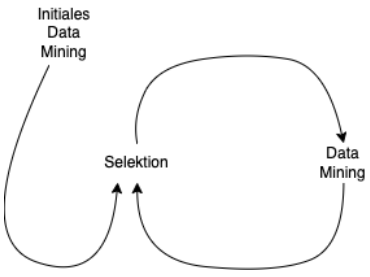


Abbildung 5.1

Wie in Kapitel 4 erwähnt, werden für jedes Symbol mehrere Daten von der Börse angefordert. Diese werden im Schritt der Datenverarbeitung zusammengefügt und zu Eingabedaten des ConvLSTM verarbeitet, wie im Schritt der Datenverarbeitung in Abbildung 5.2 zu sehen ist.

Die Eingabedaten werden zu Sequenzen zusammengefasst, um eine Minute verschoben und über die Zeit in das ConvLSTM eingegeben (siehe Abbildung 5.4). Das ConvLSTM wiederum liefert eine Sequenz mit der Vorhersage, aus der dann die profitabelsten Währungen für die Dreiecksarbitrage ausgewählt werden. In Kapitel 7 wird das Experiment für neun Währungspaare und vier Symboldreiecke durchgeführt.

5.2 Gefaltetes langes Kurzzeitgedächtnis

Ein rekurrentes neuronales Netz (Abkürzung: RNN) ist eine Art von neuronalem Netz, das sich gut für Zeitreihendaten eignet. Ein rekurrentes neuronales Netz verarbeitet Zeitreihen inkrementell und behält einen internen Zustand bei, der die Informationen, die es bisher gesehen hat, zusammenfasst.[9, vgl.]

Das kurze Langzeitgedächtnis (Abkürzung: LSTM) ist eine Art von rekurrenten neuronalen Netzen, die für ihre gute Leistung bei der Verarbeitung von Zeitreihendaten mit zeitlicher Autokorrelation bekannt ist. Eine Zelle in einem LSTM-Netz besteht aus einer Speicherzelle, einem Eingangstor, einem Merk- und Vergesstor und einem Ausgangstor. Gedächtnis- und Vergessensfähigkeit werden durch Anpassungen der zellinternen Gewichtsfunktionen an die Trainingsphase verbessert.[20, vgl.]

Das ConvLSTM-Modell ist eine Abwandlung des LSTM, um den räumlich-zeitlichen Aspekt des Problems zu berücksichtigen, das erstmals für die Niederschlagsvorhersage in [20] eingeführt wurde. Jede Eingabesequenz in das ConvLSTM-Netzwerk wird als vierdimensionaler räumlicher Vektor formuliert. Das Model ist in Gleichung 5.1 formuliert. Das $*$ kennzeichnet den Faltungsoperator und $\circ$ kennzeichnet das Hadamardprodukt.[20, vgl.]

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * h_{t-1} + b_i) \tag{5.1a}$$

$$i_t = \sigma(W_{xf} * X_t + W_{hf} * h_{t-1} + b_f) \tag{5.1b}$$

$$i_t = \sigma(W_{xo} * X_t + W_{ho} * h_{t-1} + b_o) \tag{5.1c}$$

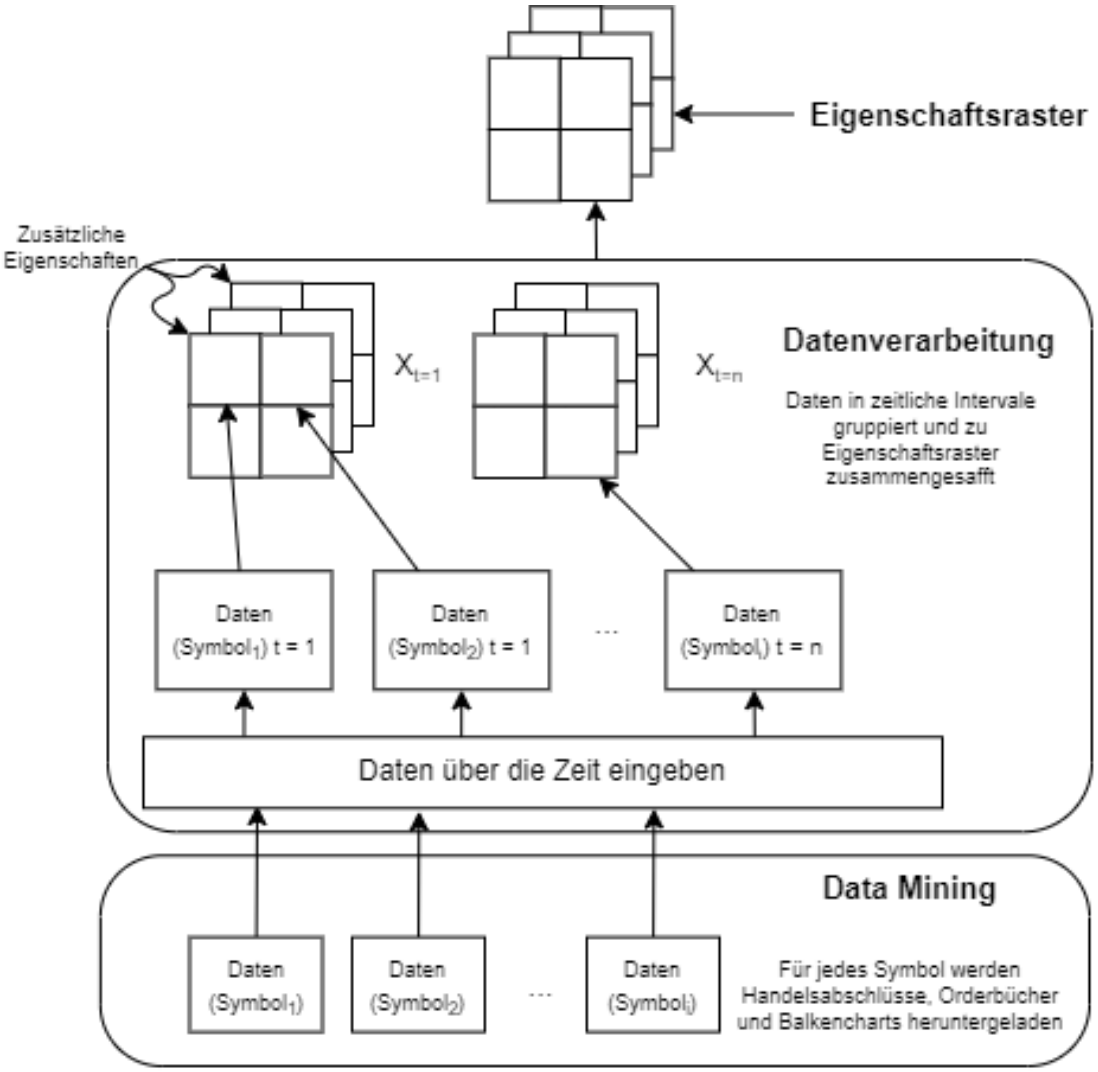


Abbildung 5.2: Systemkonzept für den Data-Mining-Teil des Systems. Aus den gesammelten Daten werden Eigenschaftsraster generiert. Diese dienen als Basis für die Erzeugung der Eingabedaten für den Teil des maschinellen Lernens.

$$C_t = f_i \circ C_{t-1} + i_t \circ tahn(W_{xc} * X_t + W_{hc} * h_{t-1} + b_c)\sigma \tag{5.1d}$$

$$h_t = o_t \circ tanh(C_t) \tag{5.1e}$$

In der Gleichung sind i_t, f_t und o_t die Ausgaben des Eingangs-, Ausgangs- und Merk- bzw. Vergesstor für den Zeitpunkt t. C_t ist die Zellenausgabe für den Zeitpunkt t. h_t ist der versteckte Zustand der Zelle zum Zeitpunkt t.

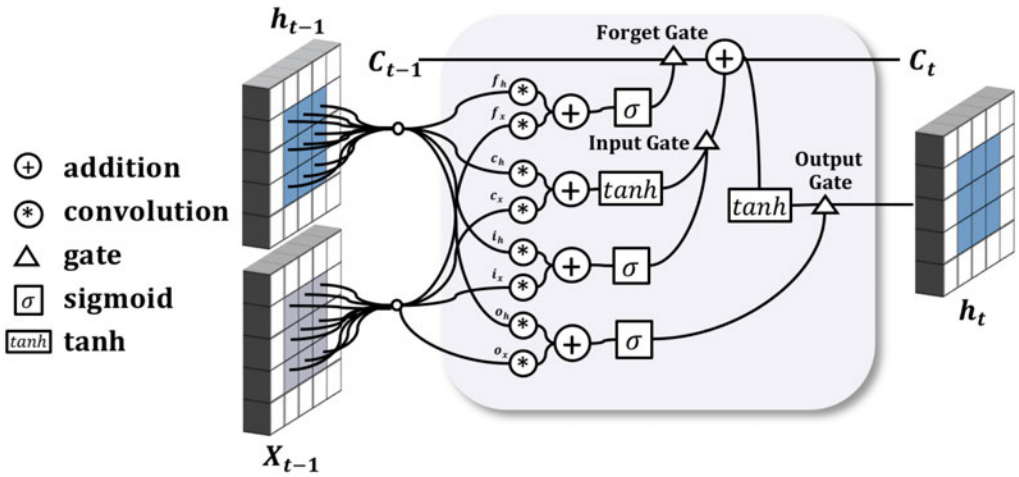


Abbildung 5.3: Die innere Struktur einer ConvLSTM Zelle. Quelle: [26].

5.3 Modell für die Vorhersage des Kreuzkursvorkommens

Bei der räumlich-zeitlichen Vorhersage, z. B. bei den Verkehrsunfällen[26], gibt es nur eine einzige abhängige Variable, aber keine anderen Merkmale. Die Eingabe X sind die historischen Werte der abhängigen Variable. In diesem Problem sind zusätzliche Merkmale eingebaut und dem Netz zugeführt. Daher sind die Eingabedaten in diesem Problem 3-dimensional.[20, vgl.] Die Parameter X_t und h_{t-1} in den obigen Formeln sind dreidimensionale Tensoren und keine zweidimensionalen Bilder wie in [20].

Das ConvLSTM-Modell besteht aus vier gestapelten ConvLSTM-Schichten, wobei jede Schicht 46 Filter enthält, um räumliche Merkmale aus den Eingabedaten und der Ausgabe der vorherigen Zeitschritte zu extrahieren. Zwischen zwei ConvLSTM-Schichten wird eine Batch-Normalisierungsschicht eingefügt, um den Trainingsprozess weiter zu beschleunigen.

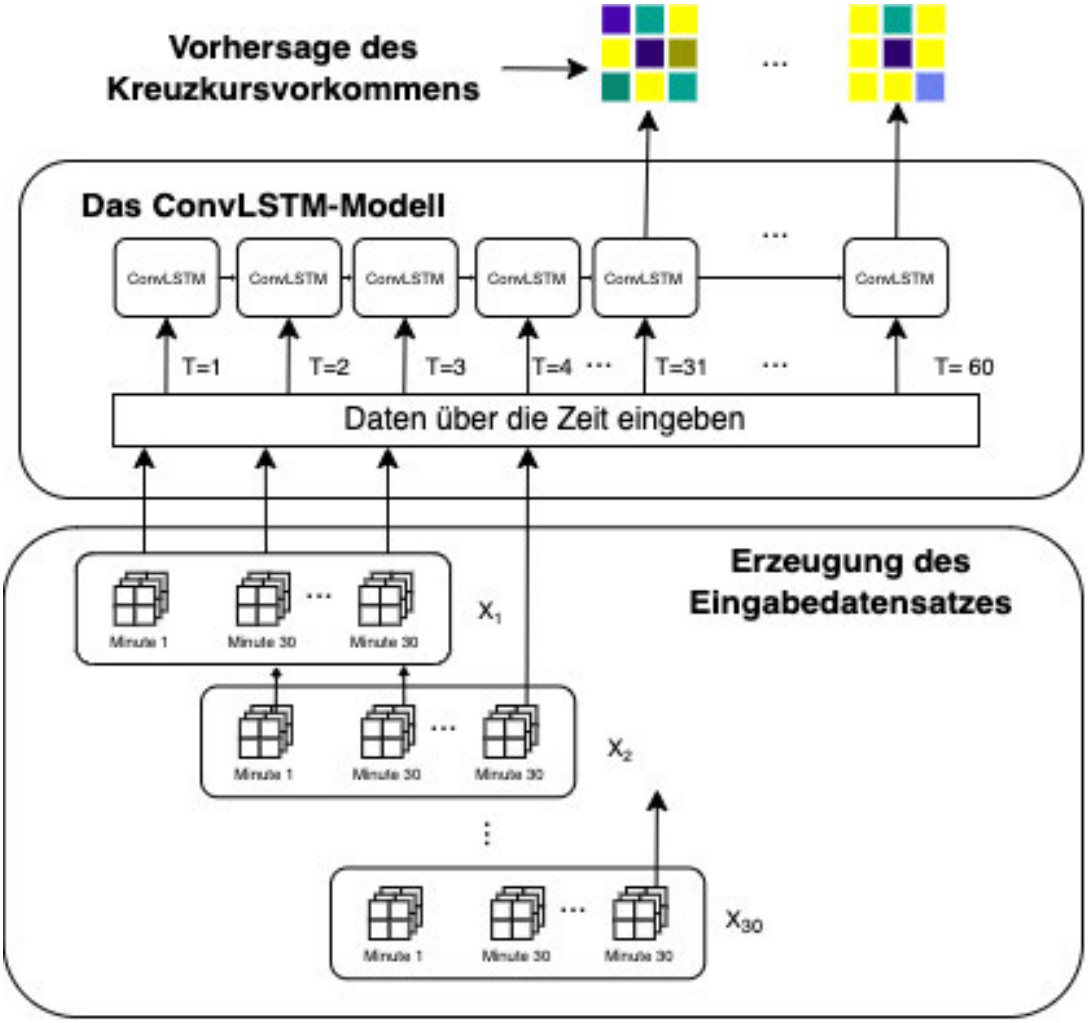


Abbildung 5.4: Die Struktur eines ConvLSTM-Modells, dessen Struktur auf [26] basiert.

Training und Testen: Jede Trainingssequenz besteht aus 60 Minuten, wobei die letzten 30 Minuten auf den Daten der ersten 30 Minuten basieren (die letzten 30 Minuten sind um 1 Minute gegenüber den ersten 30 Minuten verschoben). Alle Eigenschaftstensoren der 30 Minuten werden in das ConvLSTM eingegeben. Da die 1-minütigen-Balkencharts einen kontinuierlichen Datensatz liefern, definieren sie auch das Intervall in dieser Arbeit.

Das Modell generiert die Vorhersage für jeweils eine Minute t, basierend auf allen Eigenschaften bis zur Minute $t-1$. Anschließend wird die Grundwahrheit $C(S,t)$ als Eingabe zur nächsten Vorhersage $C(S,t+1)$ verwendet. Wichtig ist, dass wir keine Eigenschaften für Minute t benutzen, weil wir davon ausgehen müssen, dass die Eigenschaften noch nicht verfügbar sind bevor die Minute t zu Ende ist. Alle Eigenschaften, die in Kapitel 4 beschrieben sind, werden normalisiert, bevor sie verwendet werden. Als Verlustfunktion wird, wie auch in [26] die Kreuzentropie verwendet:

$$Loss = -\sum_{s,t} T_{s,t} \log P_{s,t} + (1 - T_{s,t}) \log(1 - P_{s,t})$$

wo $T_{s,t}$ und $P_{s,t}$ die Grundwahrheit und die vorhergesagte Raster an Stelle s zum Zeitpunkt t darstellen.

Zu Vergleichszwecken wird auch der mittlere quadratische Fehler in Kapitel 7 als Verlustfunktion verwendet.

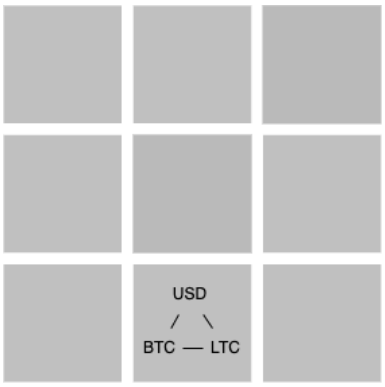


Abbildung 5.5: Jedes Feld im Raster stellt eine Kombination aus drei Währungspaaren dar. Jedes Feld besteht aus 46 Eigenschaften.

5.4 Datenstromfilterung

Das Datenaufkommen wird dadurch reduziert, dass ausschließlich die Börsendaten abgehört werden, die am Mehrteil zukünftiger Arbitragen beteiligt sind. Ermöglicht wird dies durch den Einsatz von Operator-Neuanordnung.[10, S. 5–6]

Abbildung 5.6: Ein selektiver Operator im Vorfeld, um Daten frühzeitig zu filtern. Quelle: [10, S. 5–6].

In Abbildung 5.6 steht A für die Suche nach Arbitrage über den Datenströmen und B für die ConvLSTM, welche die Datenströme in Hinblick auf das Vorkommen von Arbitrage schätzt. Wenn A vor B komm, dann verarbeitet A, unabhängig von der Selektivität von B, alle Daten und B 50 % der Daten, sodass sich die Leistung effektiv nicht ändert. Wenn B vor A kommt, dann verarbeitet B alle Daten, aber die Menge der von A verarbeiteten Daten wird durch die Selektivität von B bestimmt, und der Gesamtdurchsatz ist höher, wenn B mehr Daten ablehnt.[10, vgl. S. 5–6]

6 Implementierung

6.1 Überblick

Das Arbitrage-System besteht aus einem Data-Mining-Teil und einem Teil für das maschinelle Lernen. Der Teil für das maschinelle Lernen arbeitet mit historischen Daten, wohingegen der Data-Mining-Teil auf Echtzeitdatenströmen arbeitet. Beiden Teilen sind Aufgaben untergeordnet. Der Data-Mining-Teil besteht aus dem Abonnieren der Börsendaten, der Berechnung der Arbitrage und der Auswertung der Effizienz, sowie der Datensicherung. Der Teil für das maschinelle Lernen besteht aus der Datenaufbereitung, der Erstellung des neuronalen Netzes, des Modell-Trainings und der Modell-Tests.

Die eingesetzten Bibliotheken werden anhand der Prozessschritte genauer erläutert. Für die Implementierung wird die Sprache Python innerhalb eines Jupyter Notebooks[1] verwendet. Das Jupyter Notebook ermöglicht interaktive wissenschaftliche Datenauswertung und wissenschaftliche Berechnungen, und eignet sich deshalb sehr gut für die Umsetzung des Systems.

6.2 Prozessschritte: Data-Mining

6.2.1 Konfiguration aller Pipelines

Die Verarbeitungsdatenpipelines werden zu Beginn der Programmausführung einer festen Anzahl von Threads zugeordnet. Dies wird mit der Scheduler-Konfiguration festgelegt (siehe Listing 6.1). Für eine optimale Ressourcenzuweisung wird zunächst die Anzahl der Threads ermittelt und dann an den ThreadPoolScheduler übergeben. So wird jeder Verarbeitungspipeline mindestens ein Thread zugewiesen. Die Idee dahinter ist, dass es nie mehr Datenströme als Threads gibt, damit die sich Datenpipeline keine Threads teilen

[1] Quelle: https://jupyter.org/. Zugriffsdatum: 30.1.2022

müssen und die Verarbeitungslatenz so gering wie möglich bleibt.[4, vgl.] Da das verwendete Computersystem 22 Threads verfügt, werden nur sieben Symbole aus der Börse zur weiteren Verarbeitung verwendet. Da es im Quellcode sieben Verarbeitungspipelines gibt, werden die Ressourcen optimal auf alle Threads verteilt.

```
optimal_thread_count = multiprocessing.cpu_count()
pool_scheduler = ThreadPoolScheduler(optimal_thread_count)
```

Listing 6.1: Ermittlung der Threadanzahl und Konfiguration des Thread-Pools.

6.2.2 Auswahl der Symbole

Wenn der Teil für das maschinelle Lernen keine Symbole vorschlägt, arbeitet der Data-Mining-Teil mit den an der Börse gelisteten Symbolen.

```
def select_symbols(other_symbols):
    if other_symbols:
        return other_symbols
    else:
        base_url = "https://api.gemini.com/v1"
        response = requests.get(base_url + "/symbols")
        return response.json()

symbols = select_symbols(None)
symbols = symbols[:16]
```

Listing 6.2: Auswahl der Symbole

6.2.3 Data-Mining

Mithilfe der Funktion tri_pair_generator werden aus den Währungspaaren möglichst viele Symboldreiecke gebildet. Ein Beispiel für ein solches Dreieck findet sich in Unterabschnitt 3.2.2.

```
def tri_pair_generator(pairs):
    ...
    all_pair_combinations = list(itertools.permutations(pairs, 3))
    for permutation in all_pair_combinations:
        ...
        match = f"{a_match}{b_match}{c_match}{a_match}{c_match}{b_match}"
        if re.fullmatch(match, "".join(list(permutation)), re.IGNORECASE):
```

```
            ...
            arbtirage_tri_pair.append(tri_pair)
            ...
    return ...

generated = tri_pair_generator(symbols)
triangular_pairs =  generated["tri_pairs"]
symbols = generated["used_symbols"]
```

Listing 6.3: Generierung der Symbol-Dreiecke

Der Algorithmus erstellt zunächst alle möglichen Kombinationen aus allen Symbolen und prüft dann mittels Mustervergleich, ob die Permutation mit einer Symbolkombination für eine Dreiecksarbitrage übereinstimmt. a_{match} ist die Hauptwährung des ersten Symbols, b_{match} ist die Fremdwährung des ersten Symbols, und c_{match} ist die Hauptwährung des zweiten Währungspaares. Damit die Symbolkombination ein Dreieck bildet, muss also das folgende Muster mit der Symbolkombination übereinstimmen:

$$match = a_{match}b_{match}c_{match}a_{match}c_{match}b_{match}$$

Die gefundenen Kombinationen werden in der Variable $triangular\_pairs$ gespeichert. Die verwendeten Symbole werden zusätzlich gespeichert und überschreiben dabei die Variable *symbols*, da es möglich wäre, dass nicht alle Symbole aus Unterabschnitt 6.2.2 in den erzeugten Kombinationen aus $triangular\_pairs$ vorkommen. Folglich gibt es auch keinen Grund, Daten zu diesem Symbol bei der Börse anzufordern.

6.2.4 Abonnement von Börsendaten als Datenstrom

Jede Börse stellt eine Schnittstelle zum Abonnieren ihrer Orderbücher bereit. Die empfangenen Daten werden in ein einheitliches Datenmodell überführt und als Datenströmen veröffentlicht. Für jedes Währungspaar der Dreiecksarbitrage gibt es einen Datenstrom.

```
def order_book_listener():
    def on_subscribe(observer, scheduler=None):
        def on_message(ws, message):
            observer.on_next(message)
        ...
        ws = websocket.WebSocketApp(f"wss://api.gemini.com/v1/mulimarketdata?
    symbols={','.join(symbols)}?top_of_book=true",on_message=on_message)
        ws.run_forever(sslopt={"cert_reqs": ssl.CERT_NONE})
```

```
8           ...
9   return rx.create(on_subscribe)
10
11  def subscribe_to_order_book(symbols):
12      proxy = Subject()
13      source = order_book_listener(proxy, symbols)
14
15      return source.pipe(
16          ops.subscribe_on(pool_scheduler),
17          ops.map(json.loads),
18          ops.filter(lambda i : i["type"] == "update" and "timestamp" in i and
        "events" in i),
19          ops.map(handle_order_book),
20          ops.filter(lambda i : "side" in i)
21      )
22
23  order_book_updates = subscribe_to_order_book(symbols)
```

Listing 6.4: In diesem Ausschnitt des Quellcodes werden die empfangenen Börsendaten in Datenströme umgewandelt.

Die Methode subscribe_to_order_book(symbols) gibt einen abonnierbaren Datenstrom zurück, welcher die Marktdaten des übergebenen Symbols asynchron veröffentlicht. So kann der Datenstrom für alle Symboldreiecke verwendet werden, wobei die gewünschte Währung mithilfe von Filtern abgehört werden kann.

Die Datenpipeline in Zeile 15 in Listing 6.4 wandelt zunächst die Ergebnisse der Websocket-Verbindung in JSON[2] um. Anschließend werden alle Einträge herausgefiltert, die nicht vom Typ Handelsabschluss sind.

Analog zu subscribe_to_order_book abonniert die Methode market_data_listener die Handelsdaten und Balkencharts der Börse der übergebenen Symbole.

Speichern der Daten

Für jedes Symbol werden 3 Dateien erstellt, die den angeforderten Daten entsprechen. Die Dateinamen beginnen mit dem Symbolzeichen und enden mit dem Suffix _candles_1m_-updates.csv für Balkenchart-Aktualisierungen, _trade.csv für Handelsaktualisierungen

[2] Quelle: https://www.json.org/json-de.html, Zugriffsdatum 20.2.2022

(ausgeführte Käufe und Verkaufe) und _update.csv für Orderbuchaktualisierungen für das oberste Buch.

```
order_book_updates.pipe(
    ops.subscribe_on(pool_scheduler),
    ops.filter(lambda x : x["reason"] == "place")
).subscribe(handle_updates)

market_data_updates.pipe(
    ops.subscribe_on(pool_scheduler),
    ops.filter(lambda x : x["type"] == "candles_1m_updates")
).subscribe(handle_candle_updates)

market_data_updates.pipe(
    ops.subscribe_on(pool_scheduler),
    ops.filter(lambda x : x["type"] == "trade")
).subscribe(handle_updates)
```

Listing 6.5: In diesem Ausschnitt des Quellcodes werden die Rohdaten der Börse gespeichert.

Die drei aufgeführten Pipelines speichern die empfangenen Daten lokal.

6.2.5 Berechnung der Arbitrage

Da die Daten der drei Ströme zu unterschiedlichen Zeitpunkten empfangen werden, löst jede neu eintreffende Information eines beliebigen Datenstroms die Überprüfung auf Arbitrage erneut aus. Die Formel zu Berechnung der Arbitrage wurde in Gleichung 3.2 eingeführt.

Die Auswertung der Börsendaten zur Auffindung der Arbitrage findet auf der Verknüpfung der drei Datenströmen einer Börse statt:

$$check\_for\_arbitrage(combine\_latest((a, b, c) => d))$$

Wie in Listing 6.6 zu sehen ist, wird für jedes der drei Symbole ein neuer Datenstrom aus dem Orderbuchstrom gebildet, der die entsprechende Kauf- oder Verkaufsseite hat.

```
def compute_arbitrage(symbol_chain):
    symbols_a = order_book_updates.pipe(
```

```
        ops.filter(lambda x : x["symbol"].casefold() == symbol_chain['pair_a'
    ].casefold() and x["side"] == 'bid' and x['reason'] == 'place'),
    )
    symbols_b = ... # Analog symbols_a
    symbols_c = ... # Analog symbols_a mit x["side"] == 'ask'
    )
    rx.with_latest_from(symbols_a, symbols_b, symbols_c).pipe(
        ops.subscribe_on(pool_scheduler),
        ops.filter(check_for_timediff),
        ops.map(check_for_arbitrage),
        ops.map(convert_to_dataframe),
    ).subscribe(lambda i: to_file(i, f"data/arbitrage_opportunity/bidbidask/{
    symbol_chain['pair_a']}-{symbol_chain['pair_b']}-{symbol_chain['pair_c']}
    "), print)

for symbol_chain in triangular_pairs:
    compute_arbitrage(symbol_chain)
```

Listing 6.6: In diesem Ausschnitt des Quellcode wird ein dreier Tupel aus Symbolen auf Arbtirage untersucht.

Es wird combine_latest als Operator zum Zusammenführen der Datenströme der Börsendaten verwendet. Dieser Operator fügt jedes neue Element eines Datenstroms mit dem letzten Element der anderen Datenströme zusammen.[3, vgl.] Die Berechnung des Kreuzkurses wird dann mit Listing 6.7 implementiert.

```
def check_for_arbitrage(order_books):
    cross_rate = float(order_books[0]["price"]) * float(order_books[1]["price
    "]) * (1 / float(order_books[2]["price"]))
    return ...
```

Listing 6.7: Die Kreuzkursberechnung aus Unterabschnitt 3.2.2.

Die Untersuchungsergebnisse mit den Kreuzkursen werden unter der Kennung <symbol_a>-<symbol_b>-<symbol_c>.csv gespeichert.

6.3 Prozessschritte: Maschinellen Lernen

6.3.1 Datenaufbereitung

Datenbeschreibung

Es werden die in Abschnitt 6.2 erstellten Datensätze verwendet. Für jedes gespeicherte Symbol gibt es jeweils 3 Dateien. Die Dateinamen fangen jeweils mit dem Symbolzeichen an und enden mit dem Suffix _candles_1m_updates.csv, _trade.csv und _update.csv. Daten zu Kreuzkursen der Dreieckspaare werden unter dem Bezeichner <symbol_a>-<symbol_b>-<symbol_c>.csv geladen.

Laden der Daten

Die Zusammenfassung der Handelsdaten der Dreieckssymbole erfolgt mit der Methode create_means_dataframe, die in Listing 6.8 zu sehen ist. Dafür werden die Daten in einminütige Intervalle gruppiert (siehe Listing 6.8). Um fehlende Balkenchartdaten aufzufüllen, werden die Balkenchartspalten interpoliert. Ein Ausschnitt des zusammengefassten Datenrahmens für das Symboldreieck USD-BTC-ETH-USD ist in Abbildung 6.1 zu sehen. Kauf- und Verkaufsdaten desselben Typs werden als separate Eigenschaften behandelt.

```
1  def create_means_dataframe(triangular_pair):
2      symbols = triangular_pair.split("-")
3      result = []
4      result.append(arbitrage_data[triangular_pair].groupby([pd.Grouper(freq='1
       min') ])[['count_cross_rate']].sum())
5
6      for symbol in symbols:
7          data = crypto_data[symbol]
8          result.append(data['orderbooks'][data['orderbooks']['side'] == 'bid'
       ].groupby([pd.Grouper(freq='1min') ])[['price', 'delta', 'remaining']].
       mean())
9          result.append(data['orderbooks'][data['orderbooks']['side'] == 'ask'
       ].groupby([pd.Grouper(freq='1min') ])[['price', 'delta', 'remaining']].
       mean())
10         result.append(data['trades'][data['trades']['side'] == 'buy'].groupby
       ([pd.Grouper(freq='1min') ])[['price', 'quantity']].mean())
```

```
        result.append(data['trades'][data['trades']['side'] == 'sell'].
    groupby([pd.Grouper(freq='1min') ])[['price', 'quantity']].mean())
        result.append(data['ohlc'].groupby([pd.Grouper(freq='1min') ])[['open
    ', 'close', 'low', 'high', 'volume']].max())

    df = pd.concat(result, axis=1)
    df.fillna(0, inplace=True)
    df['open'] = df['open'].interpolate()
    df['close'] = df['close'].interpolate()
    df['low'] = df['low'].interpolate()
    df['high'] = df['high'].interpolate()
    df['volume'] = df['volume'].interpolate()
    df = df.astype({"count_cross_rate": int})
    return df

...

create_means_dataframe(['btcusd-ethbtc-ethusd'])
```

Listing 6.8: Verarbeitungsfunktionen

	count_cross_rate	price	delta	remaining	price	delta	remaining	price	quantity	price	quantity	open	close	low	high	volume	price
2022-02-24 11:49:00	0	0.0	0.0	0.0	0.0	0.0	0.0	35477.50	0.194942	35476.28	0.051860	35452.02	35412.58	35412.58	35477.50	0.421696	0.0
2022-02-24 11:50:00	0	0.0	0.0	0.0	0.0	0.0	0.0	35421.44	0.026722	35422.11	0.051860	35412.58	35422.11	35392.04	35422.11	0.137841	0.0
2022-02-24 11:51:00	0	0.0	0.0	0.0	0.0	0.0	0.0	35427.61	0.002200	35420.57	0.002964	35422.11	35425.93	35406.38	35427.61	0.013873	0.0
2022-02-24 11:52:00	0	0.0	0.0	0.0	0.0	0.0	0.0	35403.32	0.069228	35407.20	0.004000	35425.93	35343.63	35343.63	35425.93	0.135446	0.0
2022-02-24 11:53:00	0	0.0	0.0	0.0	0.0	0.0	0.0	35353.09	0.026003	35343.63	0.789986	0.00	0.00	0.00	0.00	0.000000	0.0

Abbildung 6.1: Auszug aus dem zusammengeführten Datensatz.

Auf die Datensätze werden Verarbeitungsfunktionen angewendet, um die Daten in die in Kapitel 4 erläuterte Form zu bringen. Wie in Abbildung 6.1 zu sehen ist, wird für jede Minute eine Beobachtung aufgeführt. Das bedeutet, dass es für eine Stunde 60 Beobachtungen gibt. Dementsprechend enthält ein einzelner Tag 1440 (60x24) Beobachtungen.

6.3.2 Erzeugung des Rasters und Normalisierung von Daten

Als nächsten Schritt werden die Datensätze der assoziierten Symboldreiecke zu einem einzelnen Datensatz zusammengeführt. Dafür wird in Listing 6.9 die Funktion generate_grid verwendet.

0	0	0	0	0	0	0	0	0	0	0	...
0	0	0	0	0	0	0	0	1	1	1	...
0	0	0	0	1	1	1	1	0	0	0	...
0	1	2	3	0	1	2	3	0	1	2	...
1	2	3	4	5	6	7	8	9	10	11	...

Tabelle 6.1: Beispiel *arr* für das Lesen von Elementen aus den Eingabedaten nach Index.

```
1 def generate_grid(data_arrays):
2     return np.stack(([n[:4096] for n in data_arrays]), axis=1)
3
4 result = generate_grid(all_dataframes)
5 result = result.reshape((16384, 46))
6 scaler = preprocessing.MinMaxScaler()
7 result = scaler.fit_transform(result)
8 result = result.reshape((4096, 2, 2, 46))
```

Listing 6.9: Der folgende Quellcode zeigt das Stappeln der Datensätze und die Änderung der Datensatzform.

Die Daten werden für eine bessere Leistung des neuronalen Netzes normalisiert und in die Eingabeform transformiert. Die Form $(4096, 2, 2, 46)$ des Arrays steht für die Anzahl Zeitfenster, x-Koordinate auf dem Raster, y-Koordinate auf dem Raster und die Anzahl der Eigenschaften.

In Tabelle 6.1 ist ein Beispiel für ein Array *arr* der Form $(2, 2, 2, 4)$ mit einer Menge von 32 Elementen aufgeführt. In der untersten Zeile sind die Elemente des Arrays aufgelistet. Der Zugriff auf die untersten Elemente erfolgt also von der ersten bis zur letzten Zeile über den Index. Die Tatsache, dass die Elemente in diesem Beispiel sortiert sind, spielt keine Rolle. Auf das dritte Element ($n = 2$) des ersten Zeitfensters ($t = 0$) an der Position $x = 1,\ y = 0$ kann nun mit $arr[0][0][1][2] = 7$ zugegriffen werden. Da die Datensätze in Zeile 2 zeilenweise im Reißverschlussverfahren zusammengeführt wurden, kann das Array in Zeile 8 in die notwendige Form konvertiert werden (siehe Listing 6.9).

Es ist wichtig, die Merkmale vor dem Training eines neuronalen Netzes zu skalieren. Eine gängige Methode zur Durchführung dieser Skalierung ist die Normalisierung, bei der der Mittelwert von jedem Element des Datensatzes abgezogen und durch die Standardabweichung jedes Merkmals geteilt wird. Die Skalierung wird durchgeführt, um die Leistung des ConvLSTM zu verbessern.[19, vgl.] Der Einfachheit halber wird hier die Methode

sklearn.preprocessing.MinMaxScaler verwendet, die die Werte des Datensatzes in einem Bereich von [0,1] neu skaliert. [18, vgl.]

Aufteilung von Daten

Die Methode sklearn.model_selection.train_test_split wiederum teilt den normalisierten Datensatz in die Trainings-, Validierungs- und Testdatensätze auf.[17, vgl.]

```
train_ratio = 0.5
validation_ratio = 0.15
test_ratio = 0.10

def preprocess(data_array, train_ratio: float, validation_ratio: float,
    test_ratio: float):
    train_array, test_array = train_test_split(data_array, test_size=1 -
    train_ratio)
    val_array, _ = train_test_split(train_array, test_size=test_ratio/(
    test_ratio + validation_ratio), random_state=1)
    return train_array, val_array, test_array

train_array, val_array, test_array = preprocess(result, train_ratio,
    validation_ratio, test_ratio)

...
```

Listing 6.10: Die Datensätze werden in Trainings-, Validierungs- und Testdatensätze aufgeteilt.

	0	1	2	3	4	5	6	7	8	9	10	11
16379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.199546
16380	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.717173	0.001649	0.0	0.0	0.199110
16381	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.005741
16382	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.789628
16383	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.199546

Abbildung 6.2: Ausschnitt aus dem Eingabedatensatz mit den normalisierten Werten.

Die Methode create_tf_dataset konvertiert die aufgeteilten Daten in TensorFlow-Datensätze, nach dem in Abbildung 5.4 erläuterten Schema. Die Methoden preprocess und create_tf_dataset sind mit kleinen Änderungen aus dem Beispiel [14] übernommen.

6.3.3 Netzwerkarchitektur

Das Modell zur Vorhersage der Kreuzkursvorkommen besteht aus einer Faltungsschicht und vier LSTM-Schichten.

LSTM plus Faltung

```
epochs = 160
filters = 46
features = 46

model = keras.Sequential()
model.add(layers.Input(shape=(None, 2, 2, features)))
model.add(layers.ConvLSTM2D(filters, (2,2), padding="same", return_sequences=
    True, activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.ConvLSTM2D(filters, (2,2), padding="same", return_sequences=
    True, activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.ConvLSTM2D(filters, (2,2), padding="same", return_sequences=
    True, activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.ConvLSTM2D(filters, (2,2), padding="same", return_sequences=
    True, activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.Conv3D(filters, (3,3,3), activation="sigmoid", padding="same
    "))

model.compile(
    optimizer=tf.keras.optimizers.Adam(
        learning_rate=0.001,
        beta_1=0.9,
        beta_2=0.999,
        epsilon=1e-07,
        amsgrad=False,
        name="Adam"
    ),
    #loss=keras.losses.binary_crossentropy,
    loss='mse',
    metrics=['accuracy']
)

```

```
model.summary()
```

Listing 6.11: LSTM plus Faltung

binary_crossentropy: Der Zweck von Verlustfunktionen besteht darin, die Größe zu berechnen, die ein Modell während des Trainings zu minimieren versuchen sollte.[12, vgl.]

6.3.4 Modell-Training

```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience
    =5)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience
    =10)

history = model.fit(
    train_dataset,
    validation_data=(val_dataset),
    epochs=epochs,
    callbacks=[early_stopping, reduce_lr],
)
```

Listing 6.12: Im diesem Ausschnit des Quellcodes wird das Modell trainiert.

Folgend erklärt sind die eingesetzten Callbacks EarlyStopping und ReduceLROnPlateau.

EarlyStopping: EarlyStopping ist verantwortlich für die vorzeitige Beendigung des Trainings, wenn sich die überwachte Kennzahl nicht mehr verbessert.[11, vgl.]

ReduceLROnPlateau: ReduceLROnPlateau reduziert die Lernrate, wenn sich eine Metrik nicht mehr verbessert. Dieser Callback überwacht eine Metrik, und wenn für eine Anzahl von Epochen keine Verbesserung zu sehen ist, wird die Lernrate reduziert.[13, vgl.]

6.3.5 Vorhersagen auf dem Testsatz machen

In Listing 6.13 ist zu sehen, wie die Vorhersagen visualisiert werden. Nur die letzte Sequenz von Testdaten wird dabei ausgegeben. Die Ausgabe wird so umgewandelt, dass

nur die vorhergesagten Kreuzkursvorkommen angezeigt werden. Die normalisierten Werte werden vor der Ausgabe zurück skaliert.

Um das Auftreten des Kreuzkursvorkommens in Relation zueinander zu betrachten, werden die Quadranten auf der Grundlage ihrer relativen Werte zueinander farblich kodiert. Je gelblicher die Farbe ist, desto höher ist der relative Anteil von Kreuzkursvorkommen. Je bläulicher die Farbe ist, desto weniger Kreuzkursvorkommen sind für das Symboldreieck zu erwarten. Für die Auswertung muss der Systembenutzer das Ergebnis interpretieren und sich für ein Symboldreieck entscheiden.

Abbildung 6.3: Eine farbige und kontrastreiche Kennzeichnung der Werte zur besseren Interpretation.

```
predicted = model.predict(test_dataset)
for i in range(60):
    gprediction = predicted[-1, i, ::, ::, ::]
    prediction_inverse =  scaler.inverse_transform(gprediction.reshape((4,46)
    ))
    fig, ax = plt.subplots()
    shw = ax.imshow(prediction_inverse[::,0].reshape((2,2)))
    ...
```

Listing 6.13: Dieser Ausschnitt aus dem Quellcode zeigt, wie die Vorhersage visualisiert wird.

7 Untersuchungsergebnisse

7.1 Einstellungen

7.1.1 Datenaufbereitung

Die folgenden Symbole wurden für die Datenabfrage ausgewählt: BTCUSD, ETHBTC, ETHUSD, BTCEUR, ETHEUR, BTCSGD, ETHSGD, BTCDAI und ETHDAI. Mindestens vier Dreieckspaarungen sind damit möglich. Die Daten für die Symbole wurden mindestens drei Tage lang aufgezeichnet, aus welchen der Eingabedatensatz für die Vorhersage erstellt wird.

Ziel ist die Vorhersage von Kreuzkursvorkommen in den nächsten 30 Minuten auf der Grundlage der letzten 30 Minuten. Daher wird jede Trainings- und Testprobe als eine Sequenz von 60 Zeitfenstern (30 Zeitfenster für das Training und die nächsten 30 Zeitfenster für die Vorhersage) erstellt. Aus den gesammelten Daten werden die ersten 4096 Minuten für das Erstellen der Trainingsdaten gewählt. Alle Daten werden in 2 Gruppen aufgeteilt, wobei die Daten der ersten Gruppe als Trainingsmenge und die zweite Gruppe als Testmenge verwendet werden. 10% des Trainingdatensatzes werden als Validierungsdatensatzes ausgewählt.

7.1.2 Evaluation

Die Experimente sollen folgende Fragen beantworten: (1) Wie genau ist das neuronale Netz bei der Vorhersage von Kreuzkursvorkommen? Wie nützlich sind die Vorhersageergebnisse? (2) Welchen Einfluss haben die aggregierten Merkmale auf das Kreuzkursvorkommen?

7.1.3 Evaluierungsmetriken

Die Genauigkeit der Modelle wird anhand des mittleren quadratischen Fehlers (MSE) und der Kreuzentropie (CE) bewertet. Für jede Verlustfunktion wird das Experiment einmal durchgeführt.

7.1.4 Parameter Konfiguration

Trainiert wird das vorgeschlagene ConvLSTM-Modell durch Minimierung des Kreuzentropieverlustes, sowie auch des mittleren quadratischen Fehlers, mithilfe des Adam-Optimierers mit den Einstellungen: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ und $\epsilon = 10^{-7}$. Während des Trainings wird auch die Technik des frühen Stoppens ebenfalls verwendet.

7.1.5 Plattform

Die Experimente wurden auf einem hochperformanten Computersystem der HAW durchgeführt. Das System hat 16GB RAM und einen Rechenknoten mit einem 4-Kern-CPU. Für das Training der Deep-Learning-Modelle wird eine NVIDIA V100-Grafikkarte mit Unterstützung der Tensorflow-Bibliothek verwendet[1].

7.2 Auswertung der Versuchsergebnisse

7.2.1 Messergebnisse der Vorhersage

Abbildung 7.1 beschreibt, wie die Symboldreiecke in der Visualisierung angeordnet sind.

Abbildung 7.2 zeigt 2 von 30 Diagrammen, die durch Listing 6.13 erstellt wurden. Jedes Diagramm steht für eine Minute des 30-minütigen Zeitfensters. Die vollständige Vorhersage ist in A aufgeführt.

[1] Quelle: https://icc.informatik.haw-hamburg.de/docs/services/jupyterhub/. Zugriffsdatum: 15.2.2022

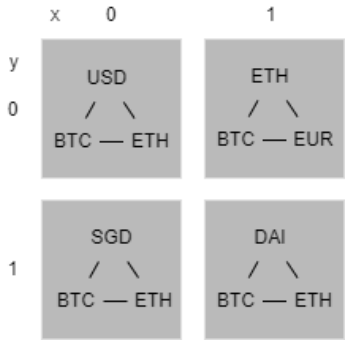
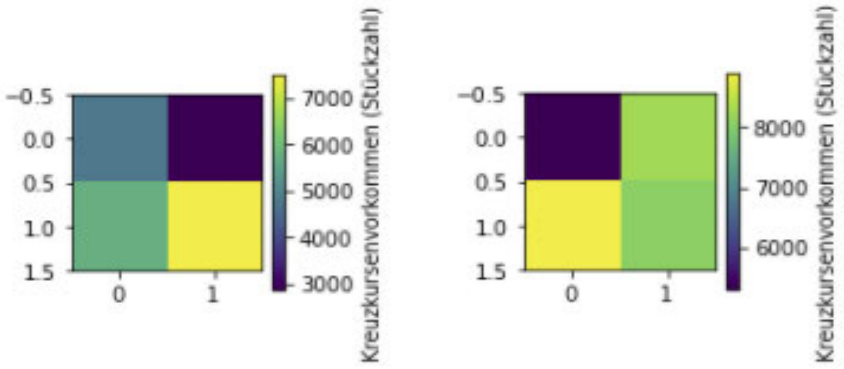


Abbildung 7.1: Symbolanordnung des Rasters.



(a) $t = 31$ (b) $t = 60$

Abbildung 7.2: Vorhersageergebnisse der Kreuzkursvorkommen für die Zeitstempel $t = 31$ und $t = 60$.

7.2.2 Diagramm zur Genauigkeit

Aus den Genauigkeits-Diagrammen Abbildung 7.3 und Abbildung 7.4 ist ersichtlich, dass das Modell nicht weiter trainiert werden sollte, da der Trend der Genauigkeit in beiden Datensätzen in den letzten Epochen nicht weiter ansteigt. Es ist zu erkennen, dass das Modell keine Überanpassung aufweist. In beiden Datensätzen ist eine vergleichbar niedrige Genauigkeit zu sehen, und zwar jeweils mit beiden Verlustfunktionen.

7.2.3 Verlustfunktionsauswertung

Aus dem Verlustdiagramm geht hervor, dass das Modell sowohl in den Trainings- als auch in den Validierungsdatensätzen vergleichbare Leistungen erbringt. Das Training

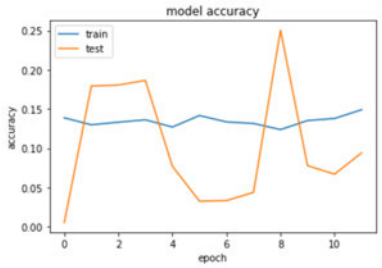


Abbildung 7.3: Verwendet Kreuzentropie als Verlustfunktion. Die Testgenauigkeit liegt bei ca. 15%.

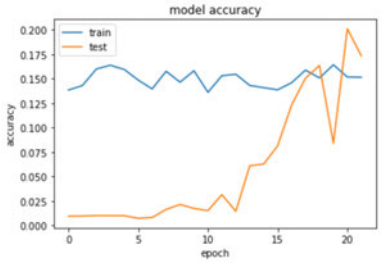


Abbildung 7.4: Verwendet die Verlustfunktion der mittleren quadratischen Abweichung. Die Testgenauigkeit liegt bei ca. 15%.

wird bei beiden Verlustfunktionen früher unterbrochen (siehe Abbildung 7.5 und Abbildung 7.6).

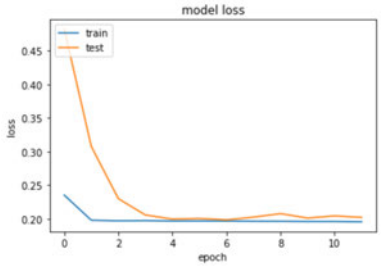


Abbildung 7.5: Das Modell verwendet hier die Kreuzentropie als Verlustfunktion.

7.2.4 Untersuchung der Datenkorrelation

Um festzustellen, warum die Genauigkeit des neuronalen Netzes so gering ist, wird die Datenkorrelation für das Symboldreieck USD-BTC-ETH-USD untersucht. Je näher der Wert bei 1 liegt, desto stärker ist die Korrelation in Abbildung 7.7. Je näher der Wert

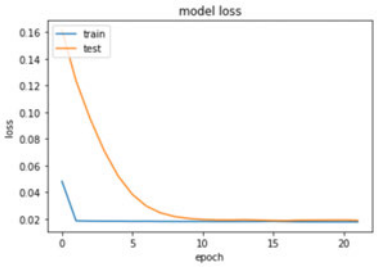


Abbildung 7.6: Das Modell verwendet hier die Verlustfunktion der mittleren quadratischen Abweichung.

bei 0 liegt, desto schwächer ist die Korrelation in Abbildung 7.7. Das Gleiche gilt für Abbildung 7.8.

Weder in Abbildung 7.7 noch in Abbildung 7.8 korreliert das Kreuzkursvorkommen (Index 0) mit den übrigen Eigenschaften. Wie bereits erwähnt, handelt es sich bei den Eingabedaten um verknüpfte Datensätze aus verschiedenen Handelsdaten. Die Verknüpfungen wiederholen sich, und dies ist in der Korrelationsmatrix in Abbildung 7.7 zu sehen. Die Daten des Balkenchart-Typs mit den Indizes 11-15, 26-30 und 41-45 zeigen eine hohe Korrelation untereinander. Die Daten des Typs Orderbuch, mit den Indizes 1-6, 16-21 und 31-36, zeigen eine hohe Korrelation zwischen den Spalten der Orderbucheigenschaften des zugehörigen Symbols und zwischen den Orderbucheigenschaften der beiden anderen Symbole.

Daten aus verschiedenen Datentypen korrelieren kaum oder gar nicht (siehe Abbildung 7.7).

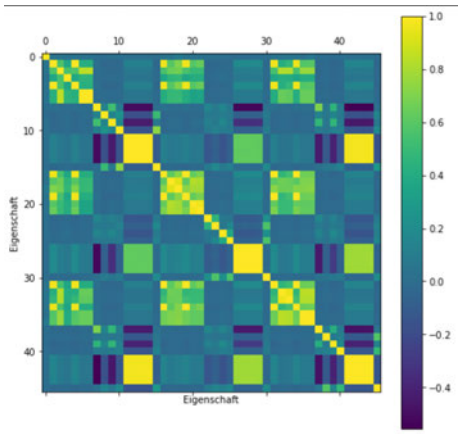


Abbildung 7.7

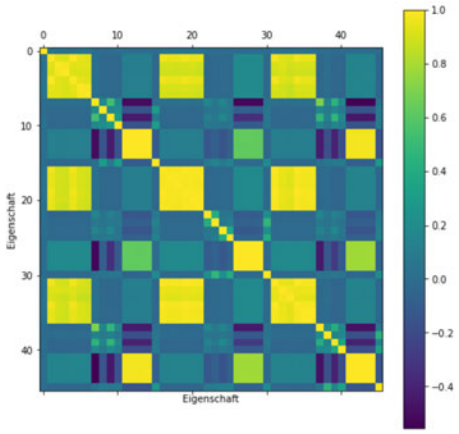


Abbildung 7.8

7.2.5 Untersuchung des Kreuzkursvorkommens

Sowohl bei Abbildung 7.9 als auch bei Abbildung 7.10 bewegt sich die Linie des Diagramms überwiegend in der Nähe des Wertes 0 auf der x-Achse. Sowohl bei Abbildung 7.9, als auch bei Abbildung 7.10 sind sehr starke Artefakte zu erkennen.

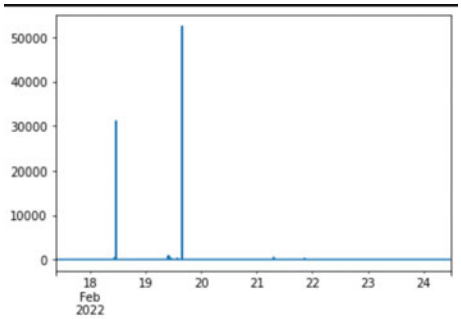


Abbildung 7.9: Kreuzkursvorkommen über die Zeit für das Symbol-Dreieck USD-BTC-ETH-USD.

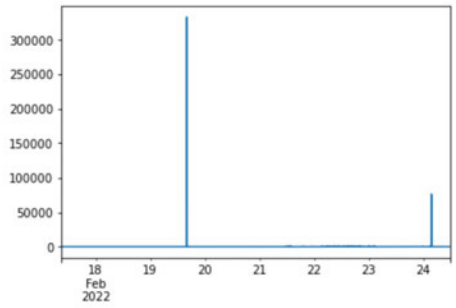


Abbildung 7.10: Kreuzkursvorkommen über die Zeit für das Symbol-Dreieck EUR-BTC-ETH-EUR

8 Fazit

In diesem empirischen Forschungsprojekt wird ein Arbitrage-System vorgestellt, bei welchem der Teil für das maschinelle Lernen nachweislich keine guten Vorhersagen macht.

Unter der Annahme, dass die Vorhersage der ConvLSTM genau ist, kann aus den Messergebnissen in A eine Filterstrategie für die nächsten 30 Minuten erstellt werden. Dazu werden die Zeiträume und das Symboldreieck mit dem höchsten Kreuzkursvorkommen ermittelt und dem Zeitraum entsprechend zugeordnet. Für die Messergebnisse aus A ergibt sich die Tabelle 8.1. Der Quellcode müsste in dieser Hinsicht erweitert werden, um die Filterung automatisch an die entsprechenden Zeitfenster anzupassen.

Aus dem Unterabschnitt 7.2.2 ist erkennbar, dass das ConvLSTM-Modell eine sehr schlechte Vorhersagekraft von etwa 15% hat (1). Ein sehr wichtiger Grund dafür ist die geringe Korrelation des Kreuzkursvorkommens (Index 0) mit allen anderen Eigenschaften (Index 1-45) (siehe Unterabschnitt 7.2.4). Woraus sich schließen lässt, dass die Aggregation der Daten nach der Umformung in Eingabedaten eine erhebliche Ungenauigkeit in die Vorhersagekraft des Modells einbringt. Aus diesem Grund hat keine der verwendeten Eigenschaften einen guten Einfluss auf die Genauigkeit des neuronalen Netzes (2) und folglich kann das Auftreten von Kreuzkursvorkommen deshalb nicht aus den zusätzlichen Eigenschaften abgeleitet werden.

Anfang des Zeitfensters	Ende des Zeitfensters	Symboldreieck
31	31	USD-BTC-ETH-USD
32	34	ETH-EUR-BTC-ETH
35	38	SGD-BTC-ETH-SGD
39	49	DAI-BTC-ETC-DAI
50	59	USD-BTC-ETH-USD
60	60	SGD-BTC-ETH-SGD

Tabelle 8.1: Die Tabelle mit den selektierten Symbolen der nächsten 30 Minuten anhand der Vorhersagen in A.

Das Resultat ergibt, dass die Gruppierung von Börsendaten zu festen Intervallen unangemessen ist, um Kreuzkursvorkommen vorherzusagen. Bei Arbeiten, die sich auf diese Arbeit stützen, sollte darauf geachtet werden, dass die abhängigen Daten eine moderate Korrelation mit der untersuchten Variable aufweisen.

Literaturverzeichnis

[1] Aiba, Yukihiro ; Hatano, Naomichi ; Takayasu, Hideki ; Marumo, Kouhei ; Shimizu, Tokiko: Triangular arbitrage as an interaction among foreign exchange rates. In: *Physica A: Statistical Mechanics and its Applications* 310 (2002), Nr. 3, S. 467–479. – URL https://www.sciencedirect.com/science/article/pii/S0378437102007999. – ISSN 0378-4371

[2] Boner, Jonas ; Farley, Dave ; Kuhn, Roland ; Thompson, Martin: *The Reactive Manifesto*. 2014. – URL https://www.reactivemanifesto.org/

[3] Brattli, Dag: *Operators*. – URL https://rxpy.readthedocs.io/en/latest/reference_operators.html#rx.operators.combine_latest. – Zugriffsdatum: 27.02.2022

[4] Campbell, Lee: *PART 4 - Concurrency*. – URL http://introtorx.com/Content/v1.0.10621.0/15_SchedulingAndThreading.html. – Zugriffsdatum: 25.02.2022

[5] Dorschel, Joachim (Hrsg.): *Praxishandbuch Big Data*. Wiesbaden : Springer Gabler, 2015. – ISBN 978-3-658-07288-9

[6] Dresher, Tamir: *Rx.NET in Action*. In: *Rx.NET in Action*, Manning, 2017. – URL https://www.manning.com/books/rx-dot-net-in-action. – ISBN 9781617293061

[7] Drozdz, Stanislaw ; Kwapien, Jaroslaw ; Oswiecimka, Pawel ; Rak, Rafal: The foreign exchange market: return distributions, multifractality, anomalous multifractality and the Epps effect. In: *New Journal of Physics* 12 (2010), oct, Nr. 10, S. 105003. – URL https://doi.org/10.1088/1367-2630/12/10/105003

[8] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John M.: *Design Patterns: Elements of Reusable Object-Oriented Software*. 1. Addison-Wesley Professional, 1994. – URL http://www.amazon.com/Design-Patterns-

Elements-Reusable-Object-Oriented/dp/0201633612/ref=ntt_at_ep_dpi_1. – ISBN 0201633612

[9] Hewamalage, Hansika ; Bergmeir, Christoph ; Bandara, Kasun: Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. In: *International Journal of Forecasting* 37 (2021), Jan, Nr. 1, S. 388–427. – URL http://dx.doi.org/10.1016/j.ijforecast.2020.06.008. – ISSN 0169-2070

[10] Hirzel, Martin ; Soulé, Robert ; Schneider, Scott ; Gedik, Buğra ; Grimm, Robert: A Catalog of Stream Processing Optimizations. In: *ACM Comput. Surv.* 46 (2014), mar, Nr. 4. – URL https://doi.org/10.1145/2528412. – ISSN 0360-0300

[11] keras.io: *EarlyStopping*. – URL https://keras.io/api/callbacks/early_stopping/. – Zugriffsdatum: 25.02.2022

[12] keras.io: *Probabilistic losses*. – URL https://keras.io/api/losses/probabilistic_losses/#binary_crossentropy-function. – Zugriffsdatum: 25.02.2022

[13] keras.io: *ReduceLROnPlateau*. – URL https://keras.io/api/callbacks/reduce_lr_on_plateau/. – Zugriffsdatum: 25.02.2022

[14] Khodadadi, Arash: *Traffic forecasting using graph neural networks and LSTM*. 2021. – URL https://keras.io/examples/timeseries/timeseries_traffic_forecasting/. – Zugriffsdatum: 27.02.2022

[15] nextmarkets.com: *Was ist ein Trade?*. – URL https://www.nextmarkets.com/de/handel/glossar/was-ist-ein-trade. – Zugriffsdatum: 25.02.2022

[16] Pedersen, L.: Efficiently Inefficient. In: *How Smart Money Invests and Market Prices Are Determined*, 2015

[17] scikit: *sklearn.model_selection.train_test_split*. – URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. – Zugriffsdatum: 27.02.2022

[18] scikit: *sklearn.preprocessing.MinMaxScaler*. – URL https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. – Zugriffsdatum: 27.02.2022

[19] Shanker, M. ; Hu, M.Y. ; Hung, M.S.: Effect of data standardization on neural network training. In: *Omega* 24 (1996), Nr. 4, S. 385–397. – URL https://www.sciencedirect.com/science/article/pii/0305048396000102. – ISSN 0305-0483

[20] Shi, Xingjian ; Chen, Zhourong ; Wang, Hao ; Yeung, Dit-Yan ; Wong, Wai-kin ; Woo, Wang-chun: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA : MIT Press, 2015 (NIPS'15), S. 802–810

[21] tradistats.com: *Was sind OHLC- oder Balkencharts ?*. – URL https://tradistats.com/ohlc-charts/. – Zugriffsdatum: 25.02.2022

[22] Wiesbaden, Springer F.: *Kompakt-Lexikon Wirtschaft*. Gabler Verlag, 2014. – ISBN 978-3-658-05791-6

[23] wirtschaftslexikon24.com: *Kreuzkurs*. – URL http://www.wirtschaftslexikon24.com/d/kreuzkurs/kreuzkurs.htm. – Zugriffsdatum: 25.02.2022

[24] wirtschaftslexikon24.com: *Devisenarbitrage*. 2021. – URL http://www.wirtschaftslexikon24.com/d/devisenarbitrage/devisenarbitrage.htm. – Zugriffsdatum: 15.09.2021

[25] Y., Aiba ; N., Hatano ; H., Takayasu ; K., Marumo ; T., Shimizu: Triangular Arbitrage in the Foreign Exchange Market. (2004). – URL https://link.springer.com/chapter/10.1007/978-4-431-53947-6_2. ISBN 978-4-431-67961-5

[26] Yuan, Zhuoning ; Zhou, Xun ; Yang, Tianbao: Hetero-ConvLSTM: A Deep Learning Approach to Traffic Accident Prediction on Heterogeneous Spatio-Temporal Data. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA : Association for Computing Machinery, 2018 (KDD '18), S. 984–992. – URL https://doi.org/10.1145/3219819.3219922. – ISBN 9781450355520

A Messergebnisse der Vorhersage

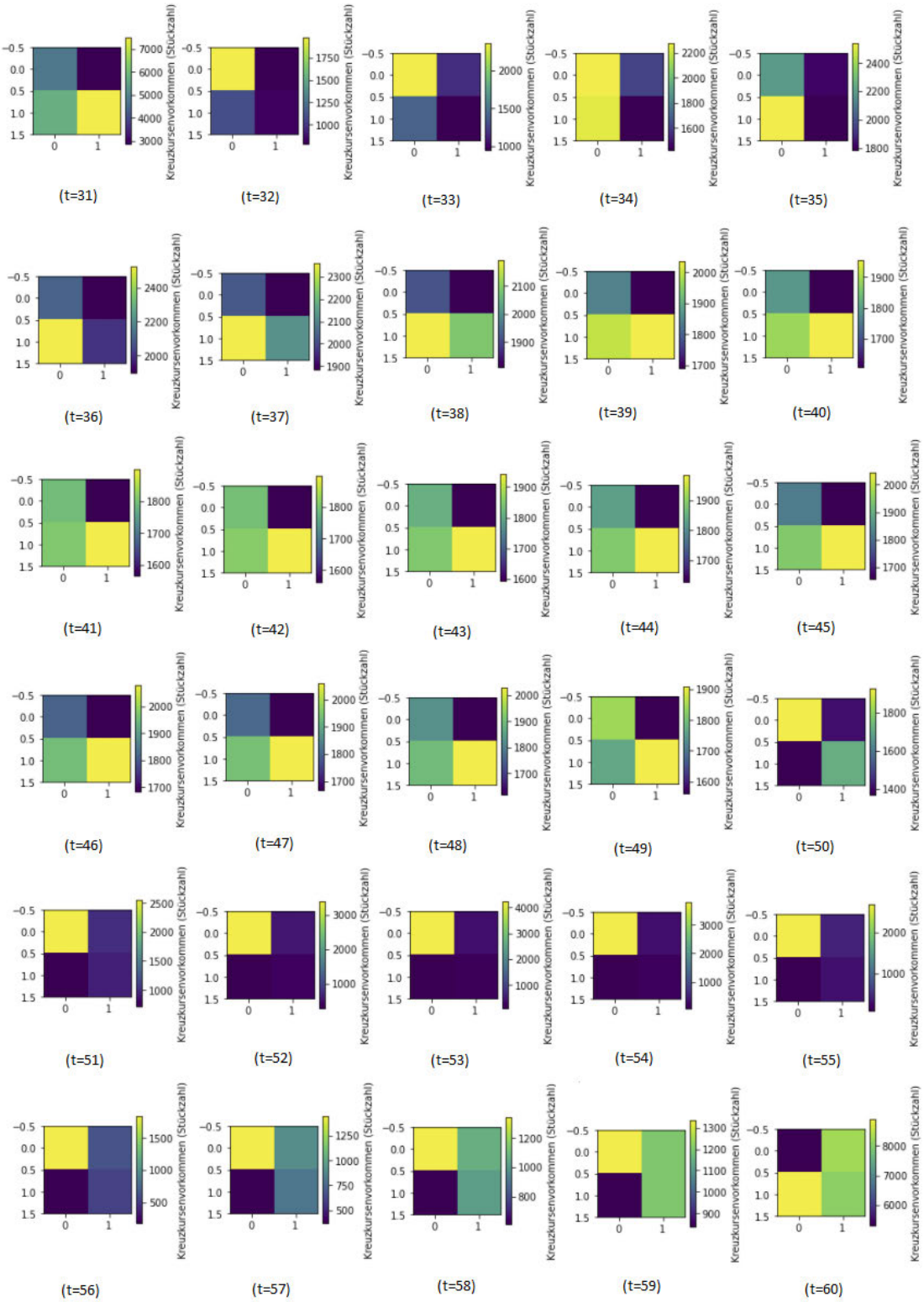


Abbildung A.1: Alle 30 vorhergesagten Messergebnisse.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original