CARL VON OSSIETZKY universität OLDENBURG

Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

OFFIS INSTITUT FÜR INFORMATIK

Studiengang: Fachmaster Informatik

Masterarbeit

# Realisierung einer Kurzfrist-Prognose des Intraday-Markts mittels verschiedener Machine-Learning-Techniken

vorgelegt von

Thies de Graaff

Erster Gutachter: Prof. Dr. Oliver Kramer
Zweiter Gutachter: Dr. Martin Tröschel

Oldenburg, 11.05.2018

## Zusammenfassung

Angetrieben durch den Klimaschutz und eine absehbare Verknappung von natürlichen Ressourcen, wird viel in den kontinuierlichen Zubau von Erzeugungsanlagen auf Basis von regenerativen Energiequellen investiert. Diese Veränderung in der Erzeugungsstruktur von elektrischem Strom macht einen Wandel in vielen Aspekten der Energiewirtschaft notwendig. Der notwendige Ausbau des deutschen Stromnetzes zur deutschlandweiten Verteilung des in küstennähe produzierten Windstroms wurde in den letzten Jahren immer wieder breit diskutiert. Die Abhängigkeit der Stromerzeugung einer Erneuerbare-Energie-Anlage von den Umweltbedingungen erlaubt nur relativ kurzfristige Prognosen der zu einem Zeitpunkt zur Verfügung stehenden Leistung. Das ist ein drastischer Unterschied zu der langfristigen Planbarkeit von Großkraftwerken, die bis zu mehrere Jahre in die Zukunft möglich ist. Da es das langfristige Ziel ist, diese Großkraftwerke durch die erneuerbaren Energien zu ersetzen, findet eine Verschiebung an den Strommärkten statt. Energieversorgungsunternehmen und andere Strombezieher müssen immer kurzfristiger ihren Strombedarf abdecken. Zu diesem Zweck wurden Strombörsen geschaffen, die einen sehr kurzfristigen Handel von Strom erlauben. An dem Intraday-Markt der EPEX SPOT kann Strom noch bis kurz vor der physikalischen Lieferung gehandelt werden. Der Preis für eine gehandelte Strommenge ist eine flexible Größe, die sich aus den Geboten von Käufer und Verkäufer bestimmen.

In dieser Arbeit wird durch den Einsatz von Maschinellem Lernen ein Prognosemodell entwickelt, das die Preisentwicklung am Intraday-Markt vorhersagen soll. Hierfür werden verschiedene Datenquellen herangezogen, deren Rohdaten in einem ersten Schritt zunächst aufbereitet werden. Die so erhaltenen Daten werden als Eingabe für verschiedene Machine Learning Modelle eingesetzt. Dabei kommen Lineare Regressionen, k-Nearest Neighbor Regressionen, Random Forest Regressionen, Support Vector Regressionen und verschiedenste neuronale Netze zum Einsatz. Durch eine Parameteroptimierung werden diese Modelle spezieller an das Problem angepasst, sodass dann eine Evaluation der Prognosequalität vorgenommen werden kann.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Unsere Energieversorgung vollzieht aktuell einen Paradigmenwechsel. Während früher vor allem auf zentrale Großkraftwerke, wie z. B. Braunkohlekraftwerke oder Atomkraftwerke, gesetzt wurde, wird heutzutage vermehrt in den Ausbau von regenerativen Energiequellen investiert. Dieser Trend wird durch politische, ökologische und auch ökonomische Interessen angetrieben. Einerseits sind sich Wissenschaftler heute einig, dass die Verbrennung fossiler Brennstoffe und die dadurch entstehenden Treibhausgase eine klimaschädliche Wirkung haben, andererseits wird die Förderung eben dieser Ressourcen auch immer schwieriger, da bereits erschlossene Quellen versiegen werden und die Erschließung neuerer Quellen oftmals nur durch umweltschädliche Gewinnungstechnologien rentabel wird. Auch Zwischenfälle, wie beispielsweise die Reaktorkatastrophe in Fukushima (2011), haben das öffentliche Interesse und die Politik stark beeinflusst. Als Reaktion hat die deutsche Bundesregierung das Atom-Moratorium beschlossen [Bun11]. Im Zuge dessen wurden alle deutschen Kernkraftwerke einer Sicherheitsüberprüfung unterzogen und auch einige Alt-Meiler vom Netz genommen. Anschließend ging die Bundesregierung noch einen Schritt weiter und beschloss den Atomausstieg bis zum Jahre 2022. Die bereits heruntergefahrenen Kraftwerke sind seitdem nicht mehr am Stromnetz [Buna].

Dieser Wegfall von Großkraftwerken soll durch den Ausbau von regenerativen Energiequellen kompensiert werden. Andererseits muss neben der aktuellen Erzeugungsstruktur auch der zukünftige Stromverbrauch betrachtet werden. Als ein Beispiel sei an dieser Stelle die steigende Elektromobilität angeführt. Diese ist aus ökologischer Sicht sinnvoll, da so die Verbrennung fossiler Ressourcen und somit auch die Emission schädlicher Treibhausgase reduziert wird. 2015 verbrauchte der deutsche Straßenverkehr laut dem Umweltbundesamt knapp 69 Milliarden Liter Kraftstoff [Umw17a]. Daher investiert die Politik in die Forschung der Elektromobilität und versucht verschiedene Anreize für die Käufer von Elektrofahrzeugen zu setzen [Bunb]. Das ambitionierte Ziel der Bundesregierung von 1 Millionen zugelassener Elektrofahrzeuge bis 2020 liegt dennoch in weiter Ferne, da 2014 nur 12.156 Elektroautos auf deutschen Straßen fuhren und ihr Anteil an den Neuzulassungen laut dem Umweltbundesamt nicht einmal 1% beträgt (2012) [Umw15]. Entgegen den aktuellen Zahlen wird die Elektromobilität langfristig betrachtet trotzdem eine immer größere Rolle spielen und entscheidend am Stromverbrauch beteiligt sein. Darüber hinaus könnten sich die Akkus der Elektroautos auch als elektrische Energiespeicher nutzen lassen, wenn das Auto ungenutzt in der Ladestation steht und ein Energiedefizit im Stromnetz ausgeglichen werden muss. Es ist also ein klarer Trend erkennbar, der auf die Ausnutzung der nachhaltig erzeugten, sauberen Energie abzielt.

Tagtäglich sind die Energieerzeuger und die Versorgungsunternehmen mit der Herausforderung konfrontiert, den durch die verschiedensten Verbraucher induzierten Stromverbrauch zu decken. Da die Versorgungsunternehmen den benötigten Energiebedarf für den nächsten Tag nicht genau kennen, müssen sie anhand von Erfahrungswerten eine Prognose erstellen, wie viel Strom sie zu den jeweiligen Tageszeiten benötigen. Daraus werden dann Fahrpläne für die verschiedenen Kraftwerke abgeleitet, sodass der Energiebedarf bestenmöglich gedeckt werden kann. Das bisherige System von wenigen Großerzeugern samt der Wirkleistungsplanung war erprobt, wobei die Betriebsmittel kostenoptimal eingesetzt wurden. Doch durch den eingangs beschriebenen Umschwung auf erneuerbare Energien gestaltet sich diese Aufgabe als deutlich komplexer. Die bisher eingesetzten Planungsalgorithmen sind nicht auf die stetig wachsende Menge von regenerativen Energiequellen skalierbar, ein Berechenbarkeitsproblem ist die Folge. Darüber hinaus erweist sich eine Verplanung von regenerativen Energiequellen als schwierig, da die Leistung von Photovoltaik- oder Windkraftanlagen durch das Wetter bestimmt ist. Diese können zwar in einem gewissen Rahmen prognostiziert werden, doch bei Abweichungen führt es zu unvorhergesehenen Einspeisungen.

Aus oben genannten Gründen wird der Handel von Energie immer bedeutsamer, insbesondere der kurzfristige Handel. An den sogenannten Strombörsen haben Akteure verschiedene Möglichkeiten, um mit Stromprodukten zu handeln. Bei diesen Stromprodukten spricht man meist von Verträgen, da beide Parteien (Verkäufer und Käufer) verpflichtet sind, das Stromprodukt zu liefern bzw. anzunehmen. In diesen Verträgen ist das Volumen (Arbeit in MWh), das Zeitintervall, in dem das Volumen bereitgestellt wird, und der Preis (€/MWh) festgelegt. Außerdem haben die Verträge eine bestimmte Laufzeit. Die längsten Verträge werden über mehrere Jahre abgeschlossen, die kürzesten Verträge können eine Laufzeit von nur 15 Minuten haben. So haben die Versorgungsunternehmen die Möglichkeit, sich durch langfristige Verträge eine gewisse Basisversorgung zu einem relativ günstigen Preis zu sichern. Tägliche Schwankungen können dann durch kurzfristige Verträge kompensiert werden. Die langfristigen Verträge werden an sogenannten Terminmärkten gehandelt, für Deutschland ist die European Energy Exchange (EEX) zuständig. Kurzfristige Produkte werden an Spotmärkten gehandelt, wobei die European Power Exchange SPOT (EPEX SPOT) der deutsche Ansprechpartner ist. An der EPEX SPOT lassen sich noch zwei verschiedene Teilmärkte unterscheiden, der Day-Ahead-Markt und der Intraday-Markt. Am Day-Ahead-Markt können Teilnehmer Verträge mit einer Mindestlaufzeit von einer Stunde für den nächsten Tag aushandeln. Entscheidungen für solche Geschäfte werden von den Versorgungsunternehmen durch Prognosen des Energiebedarfs getroffen. Sollte ein Versorger an dem entsprechenden Tag aber feststellen, dass er für gewisse Zeitspannen zu wenig oder zu viel Strom zur Verfügung hat, so kann er kurzfristig noch am Intraday-Markt handeln. An dem Intraday-Markt können Verträge mit einer Mindestlaufzeit

von 15 Minuten noch bis zu 5 Minuten vor der eigentlichen Lieferung gehandelt werden. Diese Form des Handels gewinnt durch die eingangs beschriebene steigende Energieerzeugung aus regenerativen Energiequellen immer mehr an Bedeutung. Denn durch die starke Abhängigkeit der Erzeugungsanlagen an den Umweltbedingungen ist die zu einem gewissen Zeitpunkt erzeugte Energie nur relativ kurzfristig genauer prognostizierbar.

Der Preis bestimmt sich durch Angebot und Nachfrage. Ist das Angebot sehr groß, weil beispielsweise das Wetter die Erzeugung von Strom durch Windkraft- oder Photovoltaikanlagen begünstigt, so sinkt der Preis. Auch der Tageszeitpunkt beeinflusst die Kosten, da beispielsweise nachts die Nachfrage geringer ist als tagsüber. Durch das steigende Handelsvolumen am Intraday-Markt ist es aus ökonomischer Sicht von Vorteil, wenn die Preisentwicklung am Intraday-Markt abgeschätzt werden kann. Der Intraday-Markt ist durch die Möglichkeit des Handels kurz bis vor der Lieferung sehr flexibel. Ein Akteur kann schon Stunden vor der Lieferung einen Handel eingehen, oder er wartet erst bis zum Schluss ab. Der Preis für den gleichen Vertrag kann aber je nach Zeitpunkt des Vertragsabschlusses unterschiedlich hoch ausfallen. Mit Hilfe einer Prognose der Preisentwicklung können also Zeitpunkte abgeschätzt werden, an denen der Kauf oder Verkauf eines gewissen Stromprodukts am rentabelsten ist.

## 1.2 Problemstellung

Das zuvor motivierte Ziel dieser Masterarbeit soll es sein, ein Prognosemodell zu entwickeln, mit dem die Preisentwicklung des Intraday-Markts der EPEX SPOT abgeschätzt werden kann. Als Hauptdatengrundlage dient ein von der EWE Trading GmbH bereitgestellter Datensatz, der alle Transaktionen (abgeschlossene Verträge) am Intraday-Markt der EPEX SPOT von Januar 2015 bis Dezember 2017 umfasst. Eine solche Transaktion besteht aus

- dem Zeitpunkt des Vertragsabschlusses,
- dem Handelsvolumen und dem vereinbarten Preis,
- dem Zeitintervall, in dem das gehandelte Volumen bereitgestellt wird und
- den zwei Marktzonen, von der bzw. in die der gehandelte Strom geliefert wird.

Aus diesen Daten lässt sich ablesen, wie sich an jedem Tag der Preis pro Megawattstunde für die verschiedenen Produkte entwickelt hat. In Abbildung 1 ist ein solcher Verlauf für die Viertelstunde von 18:00 bis 18:15 am 03. Mai 2017 dargestellt. Es ist erkennbar, dass lange Zeit vor der Lieferung nur vereinzelt gehandelt wird. Erst wenige Stunden vor der Lieferung findet der überwiegende Teil statt, wodurch sich deutliche Preisschwankungen bemerkbar machen. Insgesamt lässt sich in diesem

Beispiel eine deutliche Verminderung des Preises erkennen, je näher der Zeitpunkt des Vertragsabschlusses der Lieferung des Stromprodukts entgegenrückt. Dies hat zur Folge, dass bei dem ersten Handel noch 34,40 €/MWh bezahlt wurde, während bei den letzten Verträgen der Preis nur noch bei 14 €/MWh lag.

Es soll nicht das Ziel dieser Masterarbeit sein, diesen Verlauf exakt vorherzusagen. Stattdessen sollen Durchschnittspreise vorhergesagt werden, wie sie für das vorherige Beispiel in Abbildung 2 zu sehen sind. Hier wurden solche Durchschnittspreise über Intervalle mit einer Länge von 30 Minuten gebildet. Für den anfänglichen Verlauf der Kurve scheint das auch eine vertretbare Länge zu sein. Da aber gegen Ende der Auktion deutlich mehr Dynamik in der Preisentwicklung auftritt, könnte solch ein 30 minütiges Intervall zu lang sein. Jedoch ist es fraglich, ob der Durchschnittspreis über kürzere Zeiträume noch zuverlässig prognostiziert werden kann. Eine solche Evaluation soll daher auch Thema der Arbeit sein.

## 1.3 Aufbau der Arbeit

Die Struktur dieser Arbeit gliedert sich wie folgt: Zunächst werden in Kapitel 2 diverse Grundlagen eingeführt, die für das Verständnis dieser Ausarbeitung essenziell sind. Dazu zählt eine Beschreibung der deutschen bzw. europäischen Energiewirtschaft, die in Hinsicht auf die Energieerzeugung, die Energieübertragung und den Energiemarkt genauer beschrieben wird. Nachfolgend werden die Grundlagen des Maschinellen Lernens eingeführt, damit die Realisierung des Prognosemodells technisch nachvollzogen werden kann. Dabei werden verschiedene Verfahren eingehend erläutert.

In Kapitel 3 wird auf die Konzeption des Prognosemodells eingegangen. Hier wird zunächst beschrieben, welche Daten verwendet werden, und wie diese aufbereitet werden müssen. Anschließend werden die verschiedenen Prognosemodelle erklärt, die im Rahmen dieser Arbeit entworfen wurden. Darüber hinaus wird die Parameteroptimierung beschrieben, die für das Feintuning der Modelle verwendet wurde. Die Ergebnisse der Prognosemodelle werden in Kapitel 4 evaluiert. Hierbei werden die verschiedensten Parametrisierungen in Hinsicht auf die Güte der Pro-

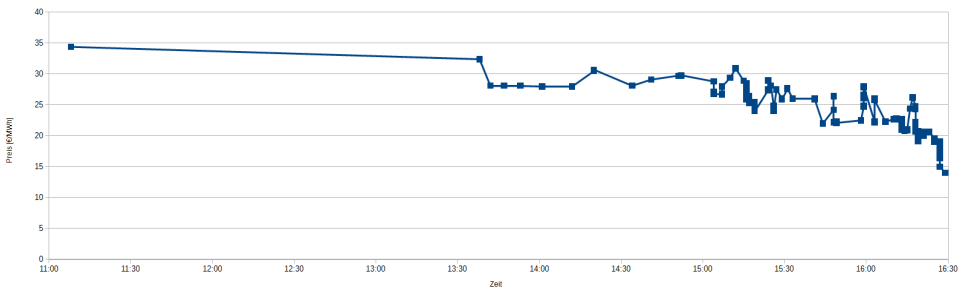



Abbildung 1: Gehandelter Preis pro MWh je nach Zeitpunkt des Vertragsabschlusses für die Viertelstunde von 18:00 bis 18:15 am 03.05.2017.

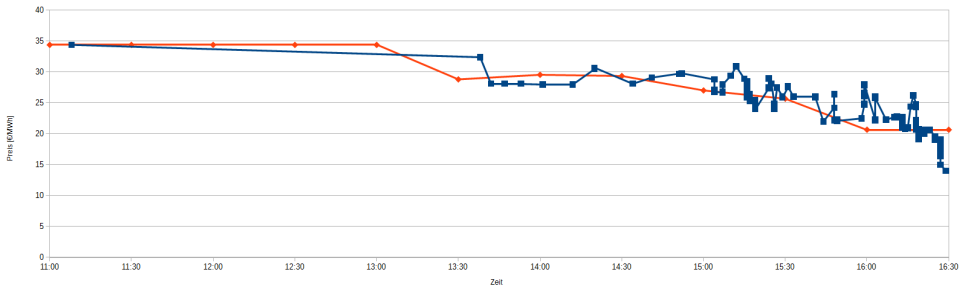



Abbildung 2: Gehandelter Preis pro MWh je nach Zeitpunkt des Vertragsabschlusses für die Viertelstunde von 18:00 bis 18:15 am 03.05.2017. Die blaue Kurve zeigt jeden abgeschlossenen Handel, die rote Kurve zeigt den Durchschnitt für 30 minütige Intervalle.

gnosen verglichen. Abschließend wird in Kapitel 5 eine Zusammenfassung und ein Ausblick gegeben.

# 2 Grundlagen

Wie bereits in Kapitel 1.2 beschrieben, ist es das Ziel dieser Masterarbeit ein Modell zu entwickeln, mit dem die Preisentwicklung am Intraday-Markt der EPEX SPOT prognostiziert werden kann. Bevor aber auf die Konzeption eingegangen werden kann, müssen zunächst einige Grundlagen geschaffen werden. Dafür wird in Kapitel 2.1 zunächst ein breites Verständnis über die gesamte Energiewirtschaft, insbesondere im Hinblick auf die Energieerzeugung, -übertragung und -vermarktung, gegeben. Darauf aufbauend kann für die Realisierung eines Prognosemodells eine große Vielfalt von Verfahren eingesetzt werden. Einige dieser Verfahren basieren auf einfachsten mathematischen Überlegungen, während andere eine deutlich komplexere mathematische Struktur aufweisen. In Kapitel 2.2 werden verschiedene Verfahren eingeführt und ihre mathematische Funktionsweise genauer erläutert. Hierfür ist es zum Teil auch erforderlich, dass gewisse mathematische Definitionen und Grundlagen eingeführt werden, die für das Verständnis notwendig sind.

## 2.1 Energiewirtschaft

Elektrische Energie ist für die heutige moderne Welt unabdingbar. Haushaltsgeräte, Lampen, Computer, Smartphones, Industrieanlagen, Verkehrsampeln etc. – die Liste aller Geräte, die von einer Energieversorgung abhängig sind, ist schier endlos. Viele verschiedene Akteure sind daran beteiligt, dass diese Versorgung sichergestellt ist. Auf der einen Seite muss zunächst der Strom erzeugt werden, der von den Verbrauchern benötigt wird. Es gibt viele verschiedene Stromerzeugungsanlagen, die in das deutsche Stromnetz einspeisen. Auf der anderen Seite muss eben diese Energieübertragung von den Stromerzeugern zu den Verbrauchern gewährleistet werden. Dabei gilt es viele verschiedene Faktoren zu berücksichtigen, damit Stromausfälle oder Beschädigungen von Netzkomponenten vermieden werden. Die ganzen Prozesse werden im Folgenden genauer beschrieben.

### 2.1.1 Energieerzeugung

In Deutschland gibt es eine Vielfalt an unterschiedlichen Stromerzeugungsanlagen. Allgemein lassen sich diese Anlagen in die drei Kategorien *Fossile Energie*, *Kernenergie* und *Erneuerbare Energie* einteilen. Dabei zählen zu den Erzeugern aus fossilen Rohstoffen hauptsächlich Braunkohle-, Steinkohle- und Erdgaskraftwerke. Die erneuerbaren Energien setzen sich aus Windenergie, Photovoltaik, Biomasse und Wasserkraft zusammen. Als Kernenergie wird der Strom bezeichnet, der durch Kernspaltung gewonnen wird.

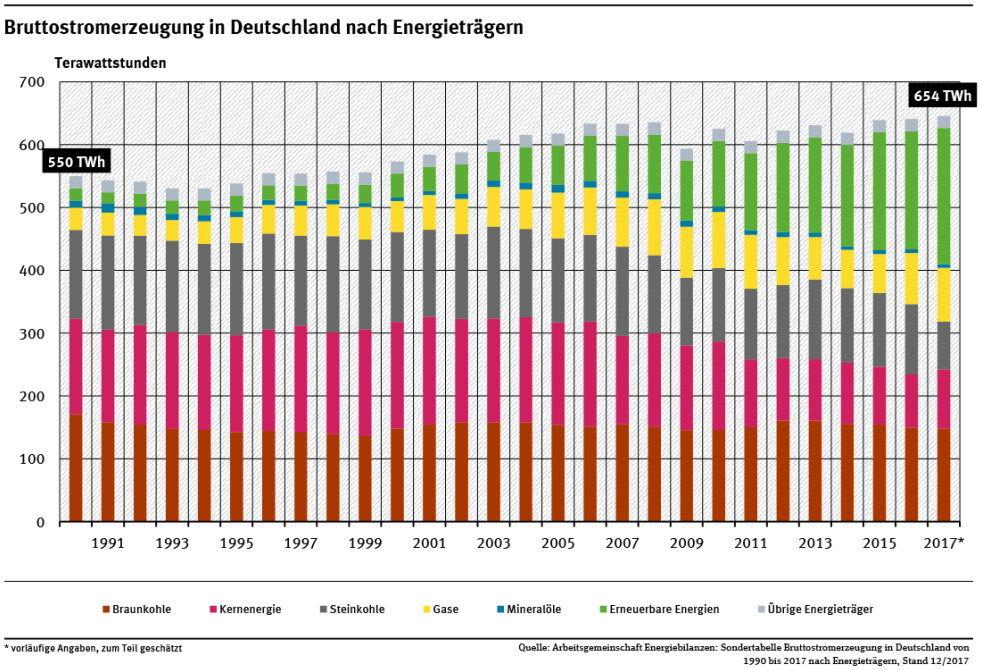



Abbildung 3: Bruttostromerzeugung in Deutschland nach Energieträgern von 1990 bis 2017 [Umw18, CC BY-NC-ND 4.0].

### Wandel der Energieerzeugung

Während früher vor allem auf zentrale Großkraftwerke, wie z. B. Braunkohlekraftwerke oder Atomkraftwerke, gesetzt wurde, wird heutzutage vermehrt in den Ausbau von regenerativen Energiequellen investiert. In Abbildung 3 ist die Entwicklung der Bruttostromerzeugung in Deutschland von 1990 bis 2017 nach den unterschiedlichen Energieträgern dargestellt. Es ist gut zu erkennen, dass die Anteile von Steinkohle und Kernenergie zurückgehen und durch das Wachstum der erneuerbaren Energien kompensiert werden. Die Produktion aus Braunkohle und Erdgas ist hingegen seit einigen Jahren konstant geblieben. 2017 steuerten die erneuerbaren Energien mit 217,9 TWh einen Anteil von etwa 33,3% zu der gesamten deutschen Stromproduktion von 654 TWh bei. Davon wird der größte Teil von 48,9% durch Windkraftanlagen, 23,5% aus Biomasse, 18,3% durch Photovoltaikanlagen und 9,1% aus Wasserkraft produziert [Umw18].

### Probleme der erneuerbaren Energien

Die fortschreitende Konzentration auf die Nutzung regenerativer Energiequellen bringt aber auch einige Probleme mit sich, die es zu bewältigen gilt. Im Folgenden werden einige dieser Probleme skizziert.

**Dezentralisierte Erzeugungsstruktur** Ein großes Problem stellt die steigende Dezentralisierung der Erzeugungsstruktur dar. Durch die stetig wachsende Anzahl

von Windkraft-, Photovoltaik-, Wasserkraft- und Biomasseanlagen muss der zu deckende Energiebedarf, statt auf eine überschaubare Menge von Großerzeugern, auf eine viel größere Menge von Kleinerzeugern verteilt werden. Die Struktur des deutschen Stromnetzes ist hierfür aber nicht konzipiert worden (siehe Kapitel 2.1.2). Photovoltaik-Anlagen und kleinere Blockheizkraftwerke (BHKW) speisen ihren erzeugten Strom hauptsächlich im Niederspannungsnetz ein; Windkraftanlagen, größere BHKW oder Solarparks wiederum speisen überwiegend im Mittelspannungsnetz ein. Die Folge ist ein in Stärke als auch Richtung kontinuierlich schwankender Stromfluss. Trotzdem müssen technische Rahmenbedingungen wie das Spannungsband und der maximale Stromfluss eingehalten werden, um Ausfälle zu vermeiden [Bre15].

**Einsatzplanung der Kraftwerke** Als noch auf konventionelle Weise die Stromversorgung durch eine überschaubare Menge von Großerzeugern sichergestellt wurde, war eine optimale Einsatzplanung dieser Kraftwerke einfach möglich. Man unterteilt dabei die verschiedenen Kraftwerke in Grund-, Mittel- und Spitzenlastkraftwerke.

Die Grundlastkraftwerke (Braunkohlekraftwerke und Kernkraftwerke) sind sehr unflexibel, d. h. dass sie nur in einem durchgängigen Betrieb kostenoptimal arbeiten können und ein An- und Abfahren dieser Anlagen aufgrund ihrer Funktionsweise nur langsam möglich ist. Daher werden diese Kraftwerke möglichst den ganzen Tag über unter Volllast betrieben, um kostengünstig das Grundlastprofil des deutschen Stromverbrauchs decken zu können.

Über den Tag ergeben sich durch den Tag-Nacht-Rhythmus periodische Schwankungen des Stromverbrauchs. Dazu kommen noch Zeiträume, in denen die Menschen vermehrt Strom verbrauchen. Anhand von Erfahrungswerten lassen sich diese Schwankungen gut prognostizieren und durch die Mittellastkraftwerke (Steinkohlekraftwerke) abdecken. Mittellastkraftwerke sind deutlich flexibler als die Grundlastkraftwerke und können innerhalb von Stunden an- und abgefahren werden, daher eignen sie sich besonders gut für diesen Einsatzzweck. Der Nachteil ist, dass der Strom nicht so kostengünstig erzeugt werden kann, wie mit den Grundlastkraftwerken.

Erfolgen starke Lastanstiege im Netz, weil bspw. Stromverbrauch und -erzeugung zu stark voneinander abweichen, muss entsprechend schnell reagiert werden. Die Grund- und Mittellastkraftwerke sind für diesen Zweck aber nicht flexibel genug, sodass dann Spitzenlastkraftwerke (Gaskraftwerke und Pumpspeicherkraftwerke) eingesetzt werden. Diese können innerhalb von Sekunden bis Minuten voll hochgefahren werden, sodass entsprechend schnell gegengeregelt werden kann. Diese Stromerzeugung durch Spitzenlastkraftwerke ist aber auch entsprechend teurer, daher sollen sie möglichst nur einige Stunden am Tag eingesetzt werden.

Mit der immer weiter steigenden Anzahl an regenerativen Energieerzeugern wird dieses Verfahren jedoch auf die Probe gestellt. Durch das Erneuerbare-Energien-Gesetz (EEG) sind Netzbetreiber weitestgehend verpflichtet, den Strom aus erneuerbaren Energien vorrangig abzunehmen und zu verteilen [17]. Windkraft- und Photovoltaikanlagen sind jedoch nur geringfügig steuerbar. Entweder sie speisen ein, oder sie werden in Problemsituationen zwangsabgeschaltet. Dementsprechend müssen die Fahrpläne der anderen Kraftwerke angepasst werden, je nachdem wie hoch die Einspeisung aus erneuerbaren Energien zu erwarten ist. Glücklicherweise lassen sich Wind und Sonne relativ gut prognostizieren, sodass eine Planung in gewissem Maße möglich ist. Stärkere Fluktuationen lassen sich jedoch nicht langfristig einplanen, sodass ggf. Spitzenlastkraftwerke einspringen müssen. Je abhängiger die Stromversorgung von den regenerativen Energien ist, desto größer wird also auch der Bedarf an flexiblen Kraftwerken oder Energiespeichern.

**Regionale Verteilung** Der Ausbau von Photovoltaik- und Windkraftanlagen ist stark regional abhängig. Bedingt durch die Küstennähe ist im Norden Deutschlands (Niedersachsen und Schleswig-Holstein) der überwiegende Anteil aller Windkraftanlagen installiert. Dazu kommen ebenfalls noch die Offshore-Anlagen in der Nord- und Ostsee. Im Gegensatz dazu macht der Süden Deutschlands (Bayern und Baden-Württemberg) einen deutlichen Anteil der Solarenergie aus [Umw17b; Umw17c].

Bereits jetzt wird ca. 50% der regenerativen Energieerzeugung durch Windkraft erzeugt. Werden nun bis 2022 alle Kernkraftwerke stillgelegt, so muss dieser Wegfall ausgeglichen werden. Hierfür müssen aber auch entsprechende Stromleitungen existieren, die den Strom aus dem Norden in den Süden transportieren können. Da das deutsche Stromnetz dieser Herausforderung bisher nicht gewachsen ist, müssen zusätzliche Stromtrassen gebaut werden. Die Umsetzung dieses Vorhabens gestaltet sich aber als durchaus schwierig, da die Streckenführung dieser Trassen oft mit den Interessen der Ortsangehörigen in Konflikt steht.

### 2.1.2 Energieübertragung

In Kapitel 2.1.1 wurde bereits beschrieben, wie die aktuelle Situation der Energieerzeugung ist. Von den Stromerzeugern muss der Strom über das Stromnetz an die Verbraucher verteilt werden. Im Folgenden soll der Aufbau des deutschen Stromnetzes und die daran beteiligten Akteure genauer erläutert werden. Darüber hinaus wird auf neue Herausforderungen für das Stromnetz eingegangen, die ihre Ursache in der Energiewende finden.

### Aufbau des Stromnetzes

Das deutsche Übertragungsnetz wird von den vier Unternehmen *TransnetBW GmbH*, *TenneT TSO GmbH*, *Amprion GmbH* und *50Hertz Transmission GmbH* betrieben. In Abbildung 4 ist diese Unterteilung dargestellt. Die unterschiedlichen Zonen der einzelnen Netzbetreiber werden als *Kontroll-* oder *Regelzonen* bezeichnet. Diese sind miteinander und auch über die Landesgrenze hinweg mit den Stromnetzen der deutschen Nachbarstaaten verbunden. Um eine solche Interoperabilität zwischen den verschiedenen Netzen zu gewährleisten, sind viele europäische Übertragungsnetzbetreiber Mitglied der *European Network of Transmission System Operators for Electricity* (ENTSO-E). Diese sieht eine Unterteilung des Stromnetzes in die verschiedenen Spannungsebenen *Höchstspannung* (380 kV und 220 kV), *Hochspannung* (110 kV), *Mittelspannung* (10-20 kV) und *Niederspannung* (0,4 kV) vor. Dabei wird Dreiphasenwechselstrom mit einer Frequenz von 50 Hz übertragen. Bedingt durch die physikalische Gesetzmäßigkeit für elektrische Leistung

$$P = U \cdot I$$

kann durch Erhöhung der Spannung (U) die Stromstärke (I) entsprechend gesenkt werden, wobei die zu übertragende Leistung (P) gleich bleibt. Durch eine Senkung der Stromstärke kann der Leitungsquerschnitt reduziert werden, ohne dass der ohmsche Verlust zu groß wird. Daher eignen sich hohe Spannungen deutlich besser für die Übertragung über große Entfernungen, da ansonsten die Leitungen zu schwer und zu teuer werden würden. Auf den niedrigeren Spannungsniveaus findet dann die regionale Vernetzung statt. Aus diesen Gründen wird das Höchstspannungsnetz auch *Übertragungsnetz* genannt, während Hoch-, Mittel- und Niederspannungsnetz zu dem sogenannten *Verteilnetz* zusammengefasst werden. Der Transfer von Strom zwischen den verschiedenen Spannungsebenen findet in Umspannwerken mit Hilfe von Transformatoren und Schaltanlagen statt.

### Ausbau des Stromnetzes

Traditionell ist das Netz darauf ausgelegt, dass die Stromerzeuger eher in den höheren Spannungsebenen einspeisen, während die Stromabnehmer über die niedrigeren Spannungsebenen versorgt werden. Wie in Kapitel 2.1.1 beschrieben, nimmt die Dezentralisierung der Erzeugungsstruktur durch den Wechsel zu erneuerbaren Energien kontinuierlich zu. Windkraftanlagen sind überwiegend in Norddeutschland konzentriert, wobei dieser Ausbau sich teilweise schneller vollzieht, als der dafür auch notwendige Netzausbau. Als Beispiel dafür lassen sich einige Offshore-Windparks nennen, die zwar bereits fertiggestellt, jedoch noch nicht mit dem Festland vernetzt waren, um den Strom zu übertragen [Zei13]. Dazu kommen die fehlenden Stromtrassen aus dem Norden in den Süden Deutschlands, um den Windstrom

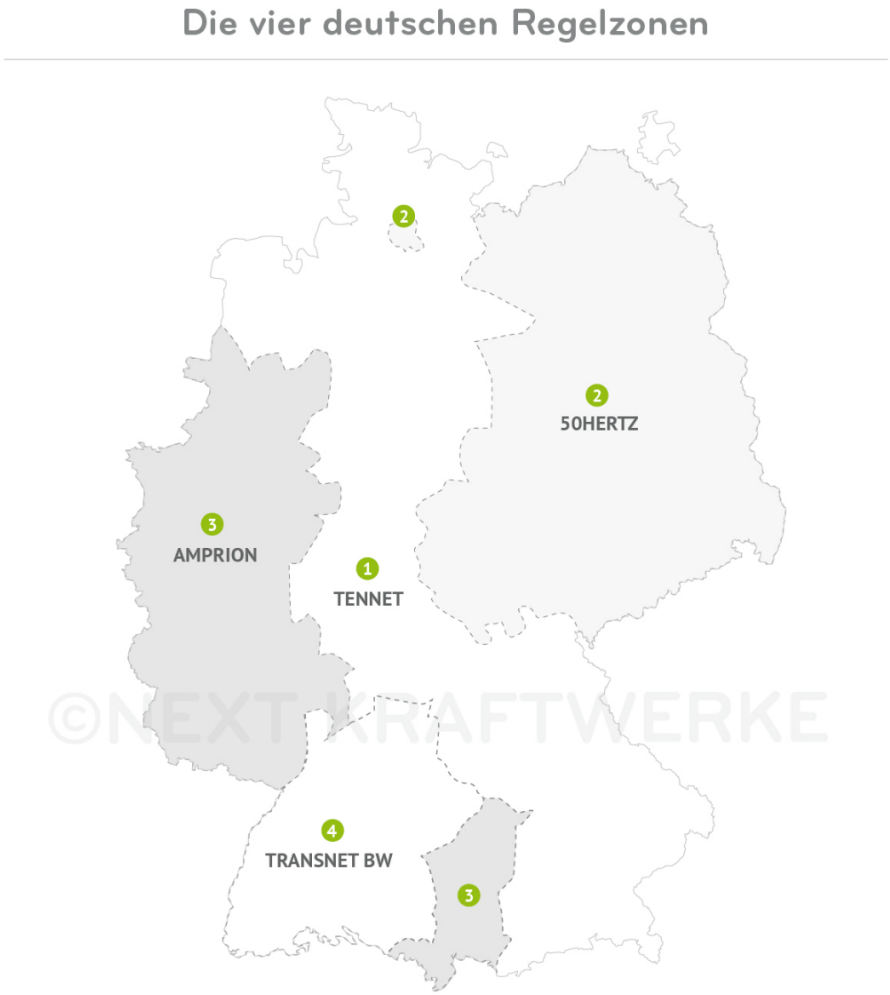



Abbildung 4: Die vier deutschen Regelzonen, betrieben durch die Übertragungsnetzbetreiber TransnetBW GmbH, TenneT TSO GmbH, Amprion GmbH und 50Hertz Transmission GmbH [Krab, Verwendung der Grafiken durch Next Kraftwerke autorisiert].

deutschlandweit zu verteilen (vgl. Kapitel 2.1.1). Auch speisen nahezu alle Windkraftanlagen (ca. 96%) nicht in das Übertragungs-, sondern in das Verteilnetz ein [Win]. Da der Betriebszustand des Verteilnetzes historisch bedingt nicht kontinuierlich überwacht und gesteuert wird, kann es bei großen Einspeisevolumina durch Windkraftanlagen zu kritischen Situationen kommen [Wis15]. Im Rahmen des Einspeisemanagements werden Anlagen durch den Netzbetreiber abgeregelt, um das Stromnetz zu stabilisieren. Jedoch können auch Netzengpässe im Übertragungsnetz zu solchen Maßnahmen führen [Win15]. Dadurch entgehen den Anlagenbetreibern die Erlöse, die durch den Verkauf der Energie erzielt worden wären. Nach EEG §15 (1) sind die Netzbetreiber verpflichtet, eine entsprechende Entschädigung auszuzahlen. Darüber hinaus muss der Wegfall des durch die Windkraftanlagen produzierten Stroms durch umweltschädlichere Kraftwerke ausgeglichen werden. Daher ist ein kontinuierlicher Netzausbau geboten und auch durch die Politik und verschiedene Netzbetreiber angestrebt. Basis für eine erfolgreiche Realisierung solcher Projekte ist eine breite gesellschaftliche Akzeptanz, die durch Transparenz seitens der Netzbetreiber und Bundesnetzagentur, sowie durch Beitragsmöglichkeiten für die Bürger gewonnen werden soll [50H13].

### Stabilisierung des Stromnetzes

Wie im vorherigen Abschnitt angedeutet wurde, können im Stromnetz Situationen auftreten, die zu einer Beschädigung oder Zerstörung physikalischer Komponenten führen würden. Um dies zu vermeiden, gibt es verschiedene Maßnahmen, die eingesetzt werden können, um die verschiedenen Netzparameter zu beeinflussen. Mit diesen sogenannten *Systemdienstleistungen* kann gezielt gegengesteurt werden, um die Parameter in ihren Normbereich zu bringen. Die *Deutsche Energie-Agentur* (dena) differenziert vier verschiedenen Systemdienstleistungen:

- *Frequenzhaltung*: Halten der Netzfrequenz auf ihrem Sollwert von 50 Hz.
- *Spannungshaltung*: Halten der Netzspannung auf ihrem Sollwert, der je nach Spannungsebene unterschiedlich ist.
- *Versorgungswiederaufbau*: Wiederaufbau der Stromversorgung nach einem kompletten oder großräumigen Ausfall.
- *Betriebsführung*: Überwachung und Steuerung des Stromnetzes sowie der daran angeschlossenen Erzeuger und Verbraucher für einen sicheren Betrieb des Gesamtsystems.

Da gewisse Maßnahmen im Rahmen der Frequenzhaltung auch Einfluss in diese Masterarbeit finden, soll darauf noch genauer eingegangen werden. Auf eine detailliertere Beschreibung der anderen Systemleistungen wird dagegen verzichtet.

**Frequenzhaltung** In einer Regelzone wird Strom eingespeist und auch entnommen. Werde alle Einspeisungen zusammengefasst und davon die Entnahmen abgezogen, erhält man den *Regelzonensaldo*. Im Idealfall liegt dieser bei null, d. h. es wird genau soviel Strom erzeugt, wie auch verbraucht wird. Das ist aber in der Regel nicht genau möglich, d. h. es liegt ein Ungleichgewicht vor. Dieses führt dann auch zwangsläufig zu einer Veränderung der Netzfrequenz. Liegt die Erzeugungsleistung über der Verbrauchsleistung (Überversorgung), so kommt es zu einer Frequenzerhöhung. Im Gegensatz dazu fällt die Frequenz ab, falls mehr Strom verbraucht als erzeugt wird (Unterversorgung). Um dieses Problem zu beseitigen, muss der Ursache entgegengewirkt werden. Bei einer Überversorgung kann dementsprechend die Erzeugung reduziert oder die Last erhöht werden. Analog gelten die beiden Maßnahmen bei einer Unterversorgung umgekehrt. Für die Umsetzung dieses Prinzips werden verschiedene Mechanismen mit unterschiedlicher Priorität eingesetzt, die unter dem Begriff Frequenzhaltung zusammengefasst werden.

Als erste Maßnahme wird die *Momentanreserve* eingesetzt. Unter der Momentanreserve versteht man die Aufnahme bzw. Abgabe von kinetischer Energie durch die Trägheit der rotierenden Massen von Generatoren konventioneller Kraftwerke [den14], da diese Generatoren direkt an das Netz gekoppelt sind. Sie wird europaweit über das gesamte Verbundnetz aufgebracht und kann bei Ausfällen von bis zu 3.000 MW die Netzfrequenz im Toleranzbereich von 50 Hz $\pm$ 0,8 Hz (kurzzeitig/dynamisch) bzw. 50 Hz $\pm$ 0,2 Hz (stationär) ausgleichen [den14]. Windkraft- oder Photovoltaikanlagen können ohne Weiteres nicht direkt zu der Momentanreserve beitragen, da sie nicht direkt mit dem Stromnetz gekoppelt sind. Es gibt aber Entwicklungen, die die Nutzung der Rotationsenergie von Windkraftanlagen für die Momentanreserve möglich machen. Auch kann durch Drosselung der dezentralen Anlagen oder auch Batteriespeicher Momentanreserve aufgebracht werden. Es wird davon ausgegangen, dass durch entsprechende Investition in diesen Bereichen Deutschland auch zukünftig seinen Teil zu der Momentanreserve beitragen kann, trotz der stetigen Ersetzung konventioneller Kraftwerke durch erneuerbare Energien.

Kann die Frequenzänderung nicht allein durch die Momentanreserve aufgefangen werden, so wird im zweiten Schritt die *Regelleistung* eingesetzt. Diese Regelleistung lässt sich auf zwei Arten unterteilen.

Zunächst unterscheidet man zwischen positiver und negativer Regelleistung. Positive Regelleistung wird bei einer zu niedrigen Netzfrequenz eingesetzt, also wenn eine Unterversorgung vorliegt. Entweder wird aufseiten der Erzeuger zusätzliche Leistung bereitgestellt, oder es werden Lasten durch Drosselung oder Abschaltung reduziert. Ist die Netzfrequenz hingegen durch eine Überversorgung zu hoch, so wird negative Regelleistung erbracht. In diesem Fall werden Kraftwerke gedrosselt oder ganz heruntergefahren, aber auch eine Zuschaltung zusätzlicher Lasten ist möglich.

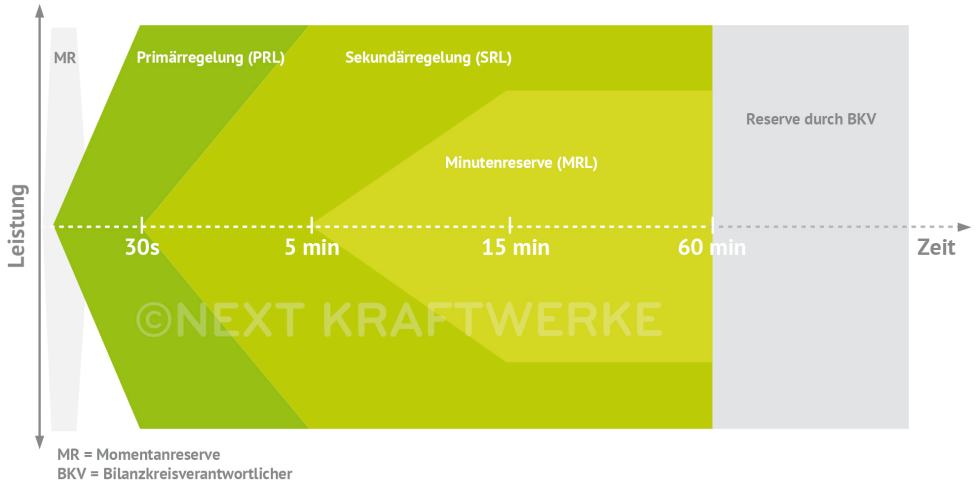



Abbildung 5: Die Abbildung zeigt, wie der Abruf der verschiedenen Arten von Regelleistungen zeitlich gestaffelt ist [Kraa].

Die zweite Art der Unterteilung von Regelleistung ist die Kategorisierung in *Primärregelleistung*, *Sekundärregelleistung* und *Minutenreserve* (auch Tertiärregelleistung genannt). Der Unterschied zwischen diesen Produkten ist die Zeit, die vergehen darf bis die gesamte Leistung erbracht werden muss. Konkret heißt das, dass an der Primärregelleistung partizipierende Kraftwerke innerhalb von 30 Sekunden ihre gesamte Primärregelleistung bereitstellen können. Bei der Sekundärregelleistung beträgt diese Zeitspanne 5 Minuten, bei der Minutenreserve sind es 15 Minuten. Daraus ergibt sich eine Reihenfolge, in der die drei Regelleistungsprodukte aktiviert werden. Sie lösen sich dadurch nacheinander ab, sodass die wieder freigewordenen Ressourcen für neue Störungen einsetzbar sind. Dieses Prinzip ist in Abbildung 5 visualisiert. Nicht jeder Kraftwerkstyp kann die zeitlichen Anfoderungen einhalten, insbesondere die der Primärregelleistung. Daher erbringen hauptsächlich Großkraftwerke Primärregelleistung, indem mit Turbinenreglern die erzeugte Leistung verändert wird [Zim17]. Für Sekundärregelleistung und auch Minutenreserve sind Spitzenlastkraftwerke, wie bspw. Gasturbinen- oder Pumpspeicherkraftwerke geeignet. Mit einem Beschluss der Bundesnetzagentur wird es ab dem Sommer 2018 möglich sein, dass auch Erzeuger aus erneuerbaren Energien entsprechende Leistungen erbringen können.

Die Akteure, die das Ungleichgewicht zwischen eingespeister und entnommener Leistung verursacht haben, müssen für die Erbringung von Regelleistung bezahlen. Der Preis für eine MWh Regelleistung wird als *Ausgleichsenergiepreis* bezeichnet. Die Berechnung des Preises ist durch die Bundesnetzagentur vorgegeben und für alle vier deutschen Regelzonen gleich [Bun12]. Diejenigen Akteure, die helfen das Ungleichgewicht auszugleichen, werden mit dem gleichen Ausgleichsenergiepreis vergütet.

Sollte es auch mit der Regelleistung nicht möglich sein, das Netz in einen stabilen Zustand zu steuern, muss der Netzbetreiber Teile des Stromnetzes abschalten, um eine Entlastung zu erreichen. Im Falle einer Unterversorgung erfolgt dies nach dem 5-Stufen-Plan, der in Tabelle 1 skizziert ist.

| | | |
|---|---|---|
| **Stufe 1:** | 49,8 Hz | Alarmierung des Personals und Einsatz der noch nicht mobilisierten Erzeugungsleistung auf Anweisung des Übertragungsnetzbetreibers, Abwurf von Pumpen. |
| **Stufe 2:** | 49,0 Hz | Unverzögerter Lastabwurf von 10 - 15 % der Netzlast. |
| **Stufe 3:** | 48,7 Hz | Unverzögerter Lastabwurf von weiteren 10 - 15 % der Netzlast. |
| **Stufe 4:** | 48,4 Hz | Unverzögerter Lastabwurf von weiteren 15 - 20 % der Netzlast. |
| **Stufe 5:** | 47,5 Hz | Abtrennen aller Erzeugungsanlagen vom Netz. |

Tabelle 1: 5-Stufen-Plan als Mittel gegen Störungen in der Netzfrequenz [Net07]

### 2.1.3 Energiemarkt

In den vorherigen Kapiteln wurde erläutert, wie Strom in Deutschland erzeugt und übertragen wird. Gewerbliche und private Endkunden beziehen ihren Strom in der Regel aber nicht direkt von den Energieerzeugern, sondern von Energieversorgungsunternehmen. Damit diese Versorger all ihre Kunden mit elektrischer Energie versorgen können, müssen sie zunächst den zu erwartenden Energiebedarf prognostizieren, um davon ausgehend die benötigte Leistung von Energieerzeugern einkaufen zu können. Für diesen Stromhandel gibt es verschiedene Märkte, an denen unterschiedliche Stromprodukte angeboten werden. Im Folgenden soll erklärt werden, wie solche Stromprodukte genau definiert sind, und wie der Handel an den verschiedenen Märkten genau funktioniert.

#### Stromprodukte

Definitionsgemäß setzt sich ein Stromprodukt aus einer elektrischen Leistung, einem Zeitintervall und einem Erbringungsort zusammen. Das Zeitintervall beschreibt dabei den Produktlieferzeitraum, in dem die Leistung durchschnittlich erbracht werden muss. Als Erbringungsort wird üblicherweise die Regelzone des Erzeugers angegeben [Wis15]. Es ist also gänzlich unabhängig davon, durch welchen Kraftwerkstyp der Strom produziert wurde, auch physisch ist diese Eigenschaft nicht nachweisbar. Aufgrund ökonomischer Interessen haben Energieversorger dennoch eine künstliche Differenzierung zwischen herkömmlichem Strom auf Basis von fossiler und nuklearer Ressourcen sowie Strom aus erneuerbaren Energien geschaffen.

Letzterer wird als *Grünstrom* oder *Ökostrom* vermarktet, wodurch umweltbewusste Kunden angesprochen werden sollen. Realisiert wird dies durch Herkunftsnachweise, die in Deutschland durch das Umweltbundesamt für eine gewisse Menge an elektrischer Energie aus regenerativen Energiequellen ausgestellt werden, falls der Strom nicht bereits durch die EEG-Umlage gefördert wird. Energieversorger können diese Herkunftsnachweise erwerben und entsprechend viel Strom als Grünstrom verkaufen. Dieser Vorgang wird dann dem Umweltbundesamt gemeldet, sodass eine Entwertung des Nachweises vorgenommen wird. Die Herkunftsnachweise dienen damit als Nachweisinstrument, dass nicht mehr Strom als Grünstrom ausgewiesen wurde, als tatsächlich eingespeist wurde [Umw12].

Stromprodukte können am *Terminmarkt* und am *Spotmarkt* gehandelt werden, die sich durch verschiedene Lieferfristen unterscheiden. Sie haben aber gemeinsam, dass sie sich in Börsenhandel und sogenannten Over-the-counter-Handel (OTC-Handel) unterscheiden lassen. Der OTC-Handel stellte ursprünglich ein bilaterales Geschäft zwischen Käufer und Verkäufer dar, die dann die Konditionen unter sich aushandeln. Inzwischen haben sich aber Plattformen etabliert, die als Broker zwischen den beiden Parteien fungieren. Die verhandelten Volumina und Preise sind nur den Vertragspartnern bekannt und werden nicht öffentlich gemacht. Im Gegensatz dazu steht der Börsenhandel, an dem standardisierte Stromprodukte, sogenannte Kontrakte, gehandelt werden können. Die Vertragspartner kennen sich gegenseitig nicht, nur die beteiligte Strombörse hat diese Information zur Durchführung des Handels. Lieferzeitraum und -ort, sowie Volumen und Preis dieser Geschäfte werden öffentlich dokumentiert. Daher können sich andere Marktteilnehmer für eine Preisfindung ihrer Kaufs- bzw. Verkaufsgebote an diesen Daten orientieren. Doch auch über die Strombörse hinaus beeinflusst dies die Preise am OTC-Markt, da dortige Partizipanten nicht gewillt sind, deutliche Preisunterschiede in Kauf zu nehmen. Für einen der beiden Vertragspartner wäre dies ein Verlustgeschäft [Deu08].

Im Folgenden werden Terminmarkt und Spotmarkt eingehender beschrieben. Dabei werden die allgemeinen Eigenschaften, sowie die zuständige Strombörse genauer beschrieben. Auf den OTC-Handel wird nicht weiter eingegangen.

### Terminmarkt

Der Terminmarkt bietet den Händlern die Möglichkeit eines mittel- bis langfristigen Stromverkaufs bzw. -einkaufs. Mit einem solchen Termingeschäft einigen sich die Vertragspartner schon lange vor dem eigentlichen Lieferzeitraum auf feste Konditionen für eine bestimmte Menge an Strom. So können sich Käufer schon frühzeitig einen gewissen Grundbedarf für einen festen Preis sichern. Das Gleiche gilt auch für Verkäufer wie bspw. Stromerzeuger, die durch die frühzeitige Veräußerung ihrer geplanten Stromproduktion eine gewünschte Marge sichern können. Die Alternative

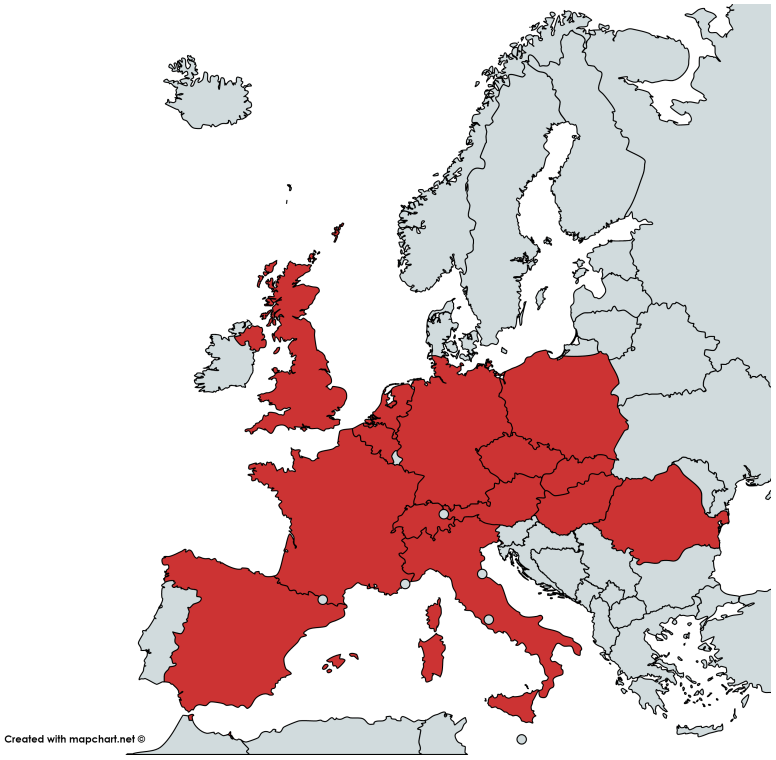



Abbildung 6: Marktgebiet der European Energy Exchange (EEX)

wäre, dass ohne diese Termingeschäfte jeder den Preisschwankungen am Spotmarkt (siehe Spotmarkt) ausgesetzt wäre. Dadurch sind zwar Gewinne im Vergleich zu einem Termingeschäft mit festgelegtem Preis möglich, jedoch auch entsprechende Verluste. Ein Termingeschäft eliminiert dieses Risiko für beide Parteien, sodass sie mit festen Einnahmen bzw. Ausgaben rechnen können.

Für Deutschland bietet die *European Energy Exchange* (EEX) eine Handelsbörse für Termingeschäfte an. Weitere Länder, in denen ebenfalls an der EEX gehandelt werden kann, sind in Abbildung 6 dargestellt. Neben Strom können bei der EEX auch bspw. Emissionsberechtigungen, Kohle oder Milchprodukte gehandelt werden. Da diese Produkte aber für diese Ausarbeitung unerheblich sind, wird im Folgenden die EEX als eine reine Strombörse angesehen.

An der EEX werden Stromprodukte in einer standardisierten Form gehandelt, den Kontrakten. Dabei wird zwischen *Futures* und *Options* unterschieden. Futures sind unbedingte Termingeschäfte, d. h. dass beide Vertragspartner verpflichtet sind, das Handelsvolumen in dem entsprechenden Zeitraum für den vereinbarten Preis zu kaufen bzw. zu verkaufen. Genauer gesagt handelt es sich bei diesen Futures um *finanzielle* Futures, da sie bei Fälligkeit zu keiner physischen Stromlieferung (*physische* Futures) verpflichten, sondern eine Garantie für einen späteren Kauf bzw. Verkauf darstellen. Da an der EEX derzeit keine physischen Futures gehandelt werden [EEX18c], wird im Folgenden keine explizite Differenzierung zwischen den Begriffen „Futures“ und „finanzielle Futures“ vorgenommen. Options sind be-

| Lieferzeit | Tag | Uhrzeit | Lieferzeitraum |
|---|---|---|---|
| Base | Montag - Sonntag | 00:00 - 24:00 | - |
| Peak | Montag - Freitag | 08:00 - 20:00 | Day/Week/Month/Quarter/Year |
| | Samstag - Sonntag | 08:00 - 20:00 | Day/Weekend |
| Off-Peak | Montag - Freitag | 00:00 - 08:00 | - |
| | | 20:00 - 24:00 | - |
| | Samstag - Sonntag | 00:00 - 24:00 | - |

Tabelle 2: Die Tabelle zeigt die genauen Eigenschaften der Lieferzeiten Base, Peak und Off-Peak an der EEX. Bei Peak hängen die Liefertage von dem Lieferzeitraum des Kontraktes ab. Informationen entnommen aus [EEX18c].

dingte Termingeschäfte, bei denen der Käufer (Kaufoption) oder der Verkäufer (Verkaufoption) das Recht hat, am letzten Handelstag die ausgehandelte Strommenge für den festgelegten Preis zu kaufen bzw. zu verkaufen. Im Gegensatz zu den Futures ist er aber nicht dazu verpflichtet, dieses Recht auch tatsächlich in Anspruch zu nehmen. Daher sind Options für den Vertragspartner mit einem entsprechenden Risiko verbunden. Tatsächlich machen Options mit 22,3 TWh auch nur einen geringen Anteil der 274,3 TWh aus, die im Februar 2018 an der EEX gehandelt wurden [EEX18b].

Die EEX unterscheidet Futures zunächst nach dem Marktgebiet, in dem es gehandelt wird und im nächsten Schritt wird nach der Lieferzeit differenziert. Es gibt die drei Lieferzeiten *Base*, *Peak* und *Off-Peak*, die genau beschreiben, zu welchen Tagen und welcher Tageszeit die Leistung bereitgestellt wird. Die genauen Definitionen der drei Lieferzeiten sind aus Tabelle 2 zu entnehmen. Nach dieser Unterteilung bietet die EEX in Deutschland die Futures Phelix-DE-Base- und Peak-Futures, sowie auch Phelix-DE/AT-Base, Peak- und Off-Peak-Futures an [EEX18c]. Phelix (Abkürzung für **Ph**ysical **El**ectricity **I**nde**x**) bezeichnet dabei den Stromindex der Marktgebiete Deutschland und Österreich. Bei dem Lieferzeitraum differenziert die EEX zwischen Day, Week, Weekend, Month, Quarter, Season (Winter oder Sommer) und Year. Abhängig vom Lieferzeitraum ist begrenzt, wie weit dieser Zeitpunkt in der Zukunft liegen darf. Year-Futures dürfen bis maximal 6 Jahre in die Zukunft abgeschlossen werden, für die Anderen gelten kürzere Zeitspannen. Für eine noch detailliertere Beschreibung sei auf [EEX18c] verwiesen.

Ein Futurekontrakt beschreibt immer eine genaue Leistung von 1 MW, die zu jedem Zeitpunkt der Lieferzeit bereitgestellt werden muss. Will ein Erzeuger aber bspw. 30 MW veräußern, so muss er 30 der entsprechenden Kontrakte verkaufen. Eine feinere Unterteilung als 1 MW ist am Terminmarkt der EEX nicht möglich.

Der Handel am Terminmarkt der EEX ist werktäglich von 8 bis 18 Uhr möglich. Es handelt sich dabei um eine Kombination aus Auktion und fortlaufendem Handel. Die Auktion findet um 8 Uhr statt. Bevor die Börse öffnet, haben die Marktteilnehmer die Möglichkeit, Kauf- und Verkaufgebote für bestimmte Kontrakte einzu-

stellen. Käufer geben dabei an, wie viel sie maximal bereit sind zu zahlen, während die Verkäufer angeben, wie viel sie mindestens einnehmen wollen. Wenn die Börse öffnet, wird nach dem Meistausführungsprinzip der Preis bestimmt, bei dem die meisten Kauf- und Verkaufgebote einen Zuschlag erhalten. Zu genau diesem sogenannten *Clearingpreis* werden dann die Geschäfte zwischen Käufern und Verkäufern abgeschlossen. Die Gebote, deren Minimal- bzw. Maximalpreis nicht mit dem Clearingpreis vereinbar sind, werden in das Orderbuch geschrieben, sodass sie im fortlaufenden Handel berücksichtigt werden können. Die Auktion wird damit abgeschlossen und der fortlaufende Handel wird eröffnet. Werden nun neue Gebote eingestellt, so wird überprüft ob diese mit einem bereits bestehenden Gebot im Orderbuch vereinbar sind, um dann ggf. ein entsprechendes Geschäft abzuschließen. Gebote, die auch nach Handelsschluss nicht ausgeführt wurden, können entweder durch den Einsteller bearbeitet oder gelöscht werden. Andernfalls verbleiben sie im Orderbuch und werden am nächsten Tag wieder miteinbezogen. [Alt; Kar02; EEX18a]

### Spotmarkt

Über den Spotmarkt können kurzfristig lieferbare Strommengen gehandelt werden. Im Gegensatz zu dem Terminmarkt sind mit den Kontrakten am Spotmarkt auch tatsächlich physische Stromlieferungen verbunden. D. h., dass Verkäufer und Käufer verpflichtet sind, den Strom zu liefern bzw. abzunehmen. Allgemein handeln die Stromerzeuger und -abnehmer am Spotmarkt die Strommengen, die sie noch nicht am Terminmarkt abgedeckt haben. Die Energieversorger müssen bspw. zusätzlichen Strom einkaufen, wenn die aktuellste Prognose des Stromverbrauchs der Endkunden höher liegt als die bereits eingekaufte Strommenge. Auf der anderen Seite müssen Energieerzeuger bspw. zusätzlichen Strom verkaufen, falls Produktionsvolumina ihrer Anlage noch nicht am Terminmarkt veräußert wurden. Diese Möglichkeit ist insbesondere für Erzeuger auf Basis erneuerbarer Energien wichtig, da genaue Wetterprognosen und die damit verbundenen Prognosen der Erzeugungsleistung nur kurzfristig möglich sind.

Für Deutschland bietet die *European Power Exchange* (EPEX SPOT) verschiedene Handelsbörsen für Spotgeschäfte an. Unterschieden wird hierbei zwischen dem *Day-Ahead-Markt* und dem *Intraday-Markt*, die im Folgenden genauer erklärt werden. Die Teilnehmer an der EPEX SPOT sind in Abbildung 7 dargestellt.

**Day-Ahead-Markt** Am Day-Ahead-Markt können verschiedene Kontrakte für den folgenden Tag gehandelt werden. Handelbare Kontrakte beziehen sich dabei entweder auf konkrete Einzelstunden oder spezifische Blöcke, die sich aus mehreren Stunden zusammensetzen. Diese Blöcke können entweder standardisiert sein (bspw. Base, Peak, Night, Morning, etc.) oder beliebig durch den Handelsteilneh-

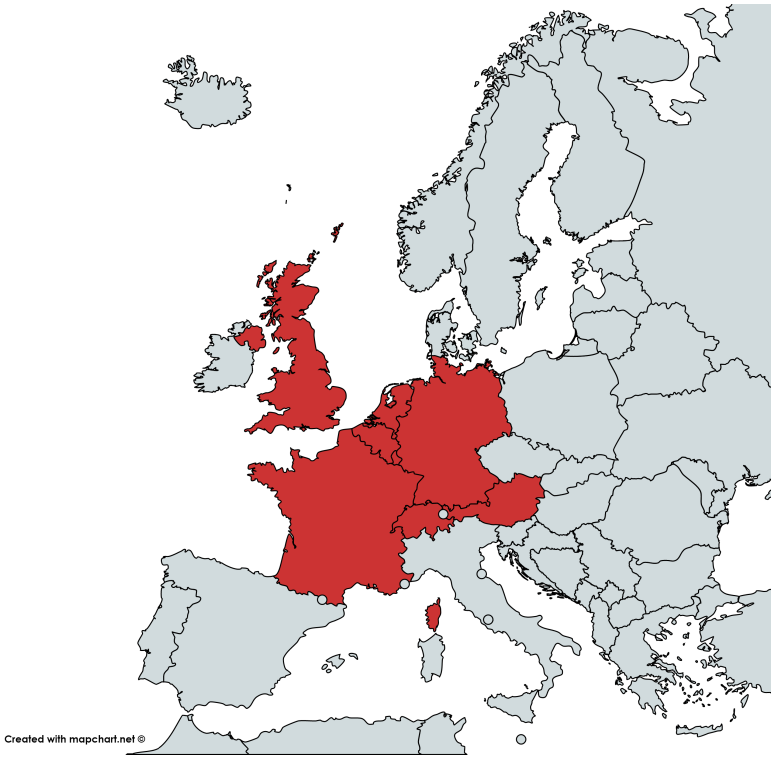



Abbildung 7: Marktgebiet der European Power Exchange SPOT (EPEX SPOT)

mer konfiguriert werden. Das Mindestvolumen eines Kontraktes beträgt 0,1 MW, unabhängig davon, ob es sich um eine Einzelstunde oder einen Block handelt. Im Gegensatz zum Terminmarkt ist der Day-Ahead-Markt nur eine reine Auktion, die an jedem Tag im gesamten Jahr um 12 Uhr ausgeführt wird. Der zugrunde liegende Preisbildungsmechanismus unterscheidet sich aber von dem Meistausführungsprinzip, da am Day-Ahead-Markt das *Market Coupling* zum Einsatz kommt. Unter Market Coupling versteht man die Verknüpfung verschiedener Marktgebiete, zwischen denen Kapazitäten ausgetauscht werden können. Realisiert wird dies durch Grenzkuppelstellen, über die der Strom von einem Marktgebiet zum Anderen fließen kann. So nähern sich die Strompreise in den gekoppelten Marktgebieten einander an. Durch die Kapazität der Grenzkuppelstellen ist die maximal übertragbare Strommenge von einem Marktgebiet in das Andere begrenzt. Daher muss es nicht zu einer vollständigen Harmonisierung der Preise kommen. Die EPEX SPOT hat zusammen mit sechs weiteren Strombörsen das Market Coupling mit der Initiative *Price Coupling of Regions* (PCR) vorangetrieben. Das Ziel ist ein harmonisierter europäischer Strommarkt. Mittlerweile sind die Märkte von Belgien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Italien, Lettland, Litauen, Luxemburg, den Niederlanden, Norwegen, Österreich, Polen, Portugal, Rumänien, Schweden, der Slowakei, Slowenien, Spanien, Tschechien, Ungarn und dem Vereinigten Königreich miteinander gekoppelt [SPOb]. Abzüglich Norwegen als Nicht-Mitgliedstaat der Europäischen Union, decken diese Länder gemeinsam etwa 96% des Strom-

verbrauchs der EU-28 ab [CIA]. Diese Dimension zeigt auf, dass die Preisbildung am Day-Ahead-Markt ein recht komplexes Problem ist, da viele verschiedene Rahmenbedingungen eingehalten werden müssen. Hierfür hat sich die PCR-Initiative gemeinsam auf den Preisbildungsalgorithmus *EUPHEMIA* geeinigt. Wie dieser Algorithmus funktioniert kann in [EPE16] eingesehen werden.

**Intraday-Markt** Am Intraday-Markt können neben Einzelstunden- und Blockkontrakten auch 15-Minuten-Kontrakte gehandelt werden. Da der Intraday-Markt ein fortlaufender bzw. kontinuierlicher Handel ist, kann zu jedem Zeitpunkt zwischen den verschiedenen Börsenteilnehmern gehandelt werden. Dabei startet der Handel für die Einzelstunden- und Blockkontrakte eines Tages am Vortag um 15 Uhr, während die 15-Minuten-Kontrakte ab 16 Uhr handelbar sind. Die Einführung der 15-Minuten-Kontrakte bietet zusätzliche Flexibilität in der kurzfristigen Portfoliooptimierung der Börsenteilnehmer, die insbesondere im Bezug auf Strom aus regenerativen Energiequellen wichtig ist. Diese Flexibilität wird auch durch die kurzen Vorlaufszeiten verstärkt, die am Intraday-Markt möglich sind. Die Vorlaufszeit bezeichnet die Zeitspanne, die zwischen einem Handel und der physikalischen Erfüllung liegen muss, und variiert je nach den Marktgebieten, zwischen denen eine Stromlieferung stattfinden soll. Deutschlandweit beträgt diese Vorlaufzeit 30 Minuten, seit Juni 2017 innerhalb einer der vier deutschen Regelzonen sogar nur 5 Minuten [SPO17b]. Zwischen gekoppelten Marktgebieten beträgt die Vorlaufzeit hingegen 60 Minuten, wobei diese Kopplung am Intraday-Markt unabhängig von dem Market Coupling am Day-Ahead-Markt ist. Denn im Gegensatz dazu ist am Intraday-Markt derzeit nur ein Handel zwischen folgenden Ländergrenzen möglich: NL-BE, BE-FR, FR-DE, FR-CH, DE-CH, DE-AT [SPOa]. Diese Kopplung soll aber im Rahmen des *Cross-Border Intraday Market Projects* (XBID) auf deutlich mehr Länder ausgeweitet werden. Das Ziel dieses Projekts ist die Verbindung der Intraday-Märkte der verschiedenen europäischen Strombörsen zu einem einzelnen grenzübergreifenden Intraday-Markt. Momentan befindet sich dieses System in der Testphase und soll voraussichtlich im zweiten Quartal 2018 gestartet werden.

Da der Intraday-Markt ein kontinuierlicher Handel ist, funktionieren die Preisbildungsmechanismen des Terminmarkts der EEX oder des Day-Ahead-Markts nicht, weil diese für ein auktionsbasiertes Verfahren gedacht sind. Im folgenden Schema wird das Zusammenfinden von Käufer und Verkäufer vereinfacht dargestellt. Für detaillierte Einblicke sei auf [SPO17a] verwiesen.

- Ein Marktteilnehmer stellt eine *Order* in das Handelssystem ein. Diese Order beinhaltet hauptsächlich die Informationen, ob es sich um ein Verkaufsgebot oder Kaufsgebot handelt, das zu (ver-)kaufende Volumen und ein Preislimit. Das Preislimit gibt dabei an, wie viel der Einsteller bereit ist zu zahlen bzw. wie viel er mindestens erhalten möchte.

- Diese Order wird in ein Orderbuch eingetragen, in dem alle Orders nach Typ (Kaufgebot oder Verkaufgebot), Preislimit und dem Zeitpunkt der Einstellung angeordnet sind.
- Gibt es ein Kaufgebot, dessen Preislimit über dem Preislimit eines Verkaufgebots liegt, so findet ein Handel zwischen diesen beiden Partnern statt. Dies kann in zwei Fällen eintreten:
  1. Ein Käufer stellt ein Gebot ein, das höher als ein bereits im Orderbuch befindliches Verkaufgebot ist.
  2. Ein Verkäufer stellt ein Gebot ein, das niedriger als ein bereits im Orderbuch befindliches Kaufgebot ist.
- Der Preis wird nach dem *pay-as-bid*-Verfahren ermittelt, d. h. der Ausführungspreis entspricht dem Preis des ausgeführten Gebots. Im ersten Fall bedeutet das, dass der Verkäufer nur das Geld bekommt, das er mindestens gefordert hat, obwohl der Käufer bereit gewesen wäre, auch mehr zu zahlen. Im zweiten Fall verhält es sich genau umgekehrt. Hier erhält der Verkäufer den von dem Käufer maximal gebotenen Preis, obwohl der Verkäufer auch mit weniger zufrieden gewesen wäre. [KV16]

Durch diesen Preisbildungsmechanismus können am Intraday-Markt für gleiche Stundenprodukte unterschiedliche Preise entstehen. Vor allem zwischen Eröffnung des Handels und dem Handelsschluss für das Stromprodukt sind oftmals signifikante Preisunterschiede zu beobachten (siehe bspw. Abbildung 1).

## 2.2 Maschinelles Lernen

Wie eingangs bereits beschrieben, ist es das Ziel dieser Masterarbeit, die Preisentwicklung am Intraday-Markt für die verschiedenen Produkte zu prognostizieren. Es gibt eine Vielzahl verschiedener Verfahren, die für solche Prognosen eingesetzt werden können. Sie werden dem Forschungsbereich des *Maschinellen Lernens* zugeordnet, der sich im Allgemeinen mit der Gewinnung von Wissen aus Daten beschäftigt. Dabei wird zunächst zwischen *überwachtem Lernen* und *unüberwachtem Lernen* differenziert.

Die Idee des überwachten Lernens ist stets die Gleiche: Ausgehend von einem Datensatz, der aus Eingabedaten und den dazugehörigen erwarteten Zielwerten besteht, soll ein Modell erstellt werden, das auch für unbekannte Eingabedaten verlässliche Zielwerte liefert. Dieser Prozess wird im Allgemeinen als *Training* des Modells bezeichnet, analog dazu wird der zugrunde liegende Datensatz *Trainingsdatensatz* genannt. Grob teilt man die verschiedenen Verfahren des überwachten Lernens in die Kategorien *Klassifikation* und *Regression* ein. Bei der Klassifikation soll ein Modell den Eingabedaten einen Zielwert aus einer endlichen diskreten Menge zuordnen. Ein Beispiel wäre die Objekterkennung in Bildern, bei der dem Bild

bspw. das zu sehende Tier zugeordnet wird. Bei der Regression wird den Eingabedaten hingegen ein reeller Zielwert zugeordnet. Diese Aufgabe entspricht daher bspw. einer reellwertigen Prognose.

Im Gegensatz zu dem überwachten Lernen sind bei dem unüberwachten Lernen die Zielwerte nicht bekannt. Daher sollen solche Verfahren alleine in den Eingabedaten verschiedene Strukturen oder Muster finden, die zusätzliche Informationen über die Daten preisgeben. Bekannteste Vertreter des unüberwachten Lernens sind Clusteringverfahren, die die Eingabedaten anhand ihrer Ähnlichkeit in verschiedene Gruppen (*Cluster*) einteilen.

In den folgenden Kapiteln sollen einige wichtige Regressionsverfahren vorgestellt werden, die auch im praktischen Teil dieser Masterarbeit Einzug finden. Da Regressionsverfahren oftmals eine Adaption von ähnlichen Klassifikationsverfahren darstellen, werden diese ggf. zusätzlich eingeführt. Ein gewisses mathematisches Basiswissen wird für die folgenden Kapitel vorausgesetzt.

### 2.2.1 Definitionen und Begriffserklärungen

Bevor auf die verschiedenen Verfahren eingegangen wird, sollen in diesem Kapitel zunächst noch einige Definitionen und Begriffserklärungen eingeführt werden, die für alle folgenden Kapitel gelten.

Sei $X \subset \mathbb{R}^d$ der Datenraum, der alle möglichen Eingabedaten beschreibt. $x \in X$ wird *Muster* oder *Pattern* genannt und stellt eine konkrete Ausprägung von Eingabedaten dar. Jedes Pattern besteht aus $d$ verschiedenen *Merkmalen* bzw. *Features*.

Die Menge $Y$ beschreibt die möglichen Zielwerte. Im Falle einer Klassifikation für $k$ Klassen gilt $Y = \{1, ..., k\}$, für eine Regression gelte im Allgemeinen $Y = \mathbb{R}$. Eine konkrete Ausprägung $y \in Y$ wird generell als *Zielwert* oder *Target* bezeichnet, wobei bei einer Klassifikation spezieller von einem *Label* gesprochen wird.

$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ beschreibt den Trainingsdatensatz bestehend aus $n$ Patterns, auf dessen Basis die verschiedenen Modelle trainiert werden. Die $(x_i, y_i)$-Paare ordnen dabei jedem Pattern ein konkretes Target zu. Dieses Target ist der Referenzwert für das entsprechende Pattern und sollte daher durch ein Modell möglichst gut angenähert werden. Dieser Vorhersagewert eines Modells wird in Anlehnung an den Referenzwert $y \in Y$ als $\hat{y} \in Y$ genannt.

### 2.2.2 Lineare Regression

Eine Lineare Regression ist eines der einfachsten Modelle, die im Bereich des Maschinellen Lernens eingesetzt werden können. Es handelt sich dabei lediglich um eine Funktion der Gestalt $\hat{y} = w \cdot x + b$, die im eindimensionalen Fall $X = \mathbb{R}$ eine einfache lineare Funktion darstellt. $w$ kann als ein Gewichtungsvektor angesehen werden, der den Einfluss der einzelnen Features von $x$ auf $\hat{y}$ beschreibt. Je größer

die Differenz zwischen $w_j$ und $0$ ist, desto stärker ist der Einfluss von Feature $x_j$ auf das Ergebnis von $\hat{y}$. Dabei kann $w_j$ positive als auch negative Werte annehmen, sodass $\hat{y}$ entweder größer oder kleiner wird. Ist $w_j = 0$, so hat das Feature $x_j$ keinen Einfluss auf das Ergebnis. $b$ ist eine Verschiebungskonstante, die den Wert von $\hat{y}$ unabhängig von dem aktuellen $x$ beeinflusst. [GBC16]

Das Training eines Linearen Regressors besteht darin, die Parameter $w$ und $b$ so einzustellen, dass der Fehler zwischen Referenzwert und Vorhersagewert möglichst klein wird. Die Residuenquadratsumme (RSS) ist ein Fehlermaß, das die Abweichung quadratisch aufsummiert: $RSS = \sum_{i=1}^{n} (y_i - \hat{y_i}) = \sum_{i=1}^{n} (y_i - w \cdot x_i + b)$. Gesucht ist nun die Parametrisierung für $w$ und $b$, für die dieser Fehler minimal wird, d. h. die Regressionsgerade, die sich am Besten an die Datenpunkte anpasst. Dieses Vorgehen wird auch als Methode der kleinsten Quadrate bezeichnet. Mathematisch lässt sich das Finden des Minimums durch eine Nullstellenbestimmung der Ableitungsfunktion durchführen. Für eine eindimensionale lineare Regression lassen sich die optimalen Werte für $w$ und $b$ durch folgende Gleichungen bestimmen:

$$w = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$b = \bar{y} - w \cdot \bar{x}$$

Dabei bezeichnen $\bar{x}$ und $\bar{y}$ jeweils die arithmetischen Mittel aller $x_i$ bzw. $y_i$. [HTF09]

Solche lineare Modelle sind durch die geringe Komplexität sehr gut nachvollziehbar. Aus dem Gewichtungsvektor $w$ kann der Einfluss eines Features $x_j$ auf den Wert von $\hat{y}$ abgelesen werden, unter der Annahme, dass die verschiedenen Features unkorreliert sind. Liegt stattdessen aber eine Korrelation zwischen mehreren Features vor, so hat die Änderung eines Features auch Einfluss auf das Andere. Daher gilt stets zu untersuchen, ob solche Interpretationen auch wirklich sinnvoll sind. Außerdem können lineare Modelle nur lineare Zusammenhänge modellieren, was in vielen Fällen nicht ausreichend ist. Beispielsweise könnte ein Feature auch polynomiellen Einfluss auf das Target haben, oder auch das Zusammenwirken mehrerer Features. Dennoch werden lineare Regressoren gerne eingesetzt, um andere Verfahren damit zu vergleichen. Schließlich ist der Einsatz komplexer Verfahren nicht gerechtfertigt, wenn bereits durch ein lineares Modell gleich gute Ergebnisse erzielt werden können.

### 2.2.3 K-Nearest Neighbor Regression

Die K-Nearest Neighbor Regression basiert auf dem K-Nearest Neighbor Algorithmus, der für die Klassifikation von Daten eingesetzt wird. Daher wird dieser zunächst im Folgenden beschrieben, sodass danach eine Modifikation beschrieben werden kann, die aus dem Klassifikations- ein Regressionsverfahren macht.

### K-Nearest Neighbor Algorithm

Wie viele andere Verfahren auch basiert der K-Nearest Neighbor Algorithm auf dem Lokalitätsprinzip. D. h., dass Datenpunkte, die im Datenraum nah beieinander liegen als ähnlich eingestuft werden. Dementsprechend liegt die Vermutung nahe, dass diese Datenpunkte auch das gleiche Label tragen. Der K-Nearest Neighbor Algorithm verfolgt daher die Idee, dass für ein neues unbekanntes Muster die Trainingsdaten nach den $k$ nächsten Nachbarn durchsucht werden. Das $k$ ist dabei ein durch den Anwender festzulegender Parameter. Die Metrik zur Bestimmung der Distanz zwischen zwei Datenpunkten kann beliebig gewählt werden, wobei aber der Einsatz der euklidischen Distanz am gewöhnlichsten ist. Nachdem die $k$ nächsten Nachbarn bestimmt wurden, werden sie nach den Labels aggregiert. Dann kann per Mehrheitsentscheid das Label bestimmt werden, das dem neuen unbekannten Muster zugeordnet werden soll. Bei diesem Prozess ist auch eine Gewichtung der verschiedenen Nachbarn in Abhängigkeit von ihrer Distanz zu dem unbekannten Datenpunkt möglich. So tragen die näheren Nachbarn mehr zu dem Mehrheitsentscheid bei, als weiter entfernte Nachbarn. In Abbildung 8 ist beispielhaft die Klassifikation mit dem K-Nearest Neighbor Algorithm für unterschiedliche Parametrisierung von $k$ dargestellt.

Technisch kann der K-Nearest Neighbor Algorithm unterschiedlich umgesetzt werden. Die einfachste Variante speichert einfach alle Datenpunkte in einer Liste ab, was im Prinzip die Trainingsphase des Algorithmus ist. Bei der Klassifikation muss dann die gesamte Liste nach den $k$ nächsten Nachbarn durchsucht werden. Diese Suche muss für jeden zu klassifizierenden Datenpunkt durchgeführt werden, was bei einem großen Trainingsdatensatz zu erheblichen Performanzproblemen führen kann. Daher investieren zwei andere Ansätze deutlich mehr Aufwand in die Trainingsphase, um dadurch schnellere Klassifikationen zu ermöglichen. Diese Idee ist durchaus sinnvoll, da das Training nur einmal durchgeführt werden muss, mit dem Modell dann aber beliebig oft klassifiziert werden kann. Der eine Ansatz basiert auf der Konstruktion eines Suchbaums (genauer einem k-d-Baum), mit dem die Traingsmenge schneller durchsucht werden kann [Pan08]. Locality-Sensitive Hashing hingengen versucht, mit Hilfe spezieller Projektionen die Nachbarn zu finden [SC08].

### Umwandlung in ein Regressionsverfahren

Bisher wurde nur gezeigt, wie der K-Nearest Neighbor Algorithm zur Klassifikation eingesetzt werden kann. Die Umwandlung in ein Regressionsverfahren ist dabei denkbar einfach. Wie auch bei dem K-Nearest Neighbor Algorithm werden für einen unbekannten Datenpunkt zunächst die $k$ nächsten Nachbarn in den Trainingsdaten gesucht. Anstatt nun einen Mehrheitsentschluss zu fassen, wird stattdessen das arithmetische Mittel der mit den Nachbarn verbundenen Targets gebildet. Der so

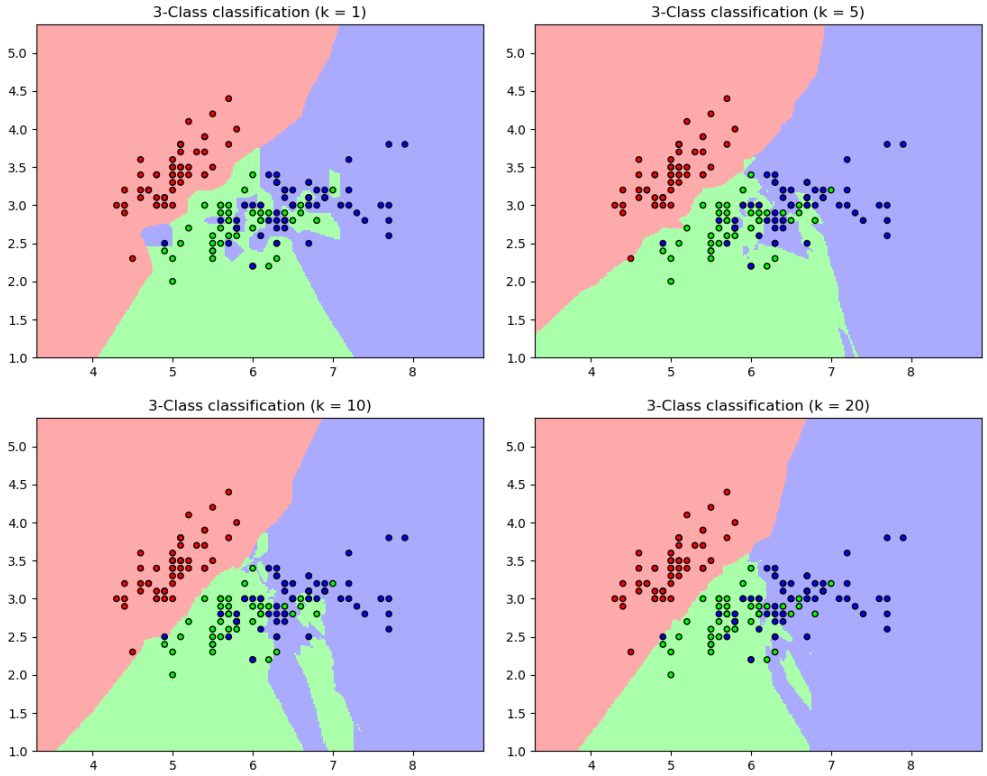



Abbildung 8: K-Nearest Neighbor Algorithm für das gleiche Klassifikationsproblem mit unterschiedlicher Parametrisierung von $k$.

entstehende Wert entspricht der Vorhersage der K-Nearest Neighbor Regression für das eingegebene Pattern. Wie auch bei der Klassifikationsvariante kann die Distanz der Nachbarn zur Gewichtung eingesetzt werden.

### 2.2.4 Random Forest Regression

Ein Entscheidungsbaum (*Decision Tree*) ist ein sehr leicht interpretierbares Verfahren, mit dem ein Eingabedatum verarbeitet wird. In jedem Knoten des Baums wird eine Merkmal des Eingabedatums untersucht und entschieden, welcher Nachfolger als nächstes betrachtet werden soll (siehe Abbildung 9). So wird ein Pfad von der Wurzel bis in eines der Blätter des Baumes verfolgt. Dieses Blatt ist der Output des Verfahrens. Da solch ein Entscheidungsbaum nur eine endliche Menge an Blättern haben kann, ist entsprechend die Anzahl der verschiedenen Outputs begrenzt. Die Regressionskurve ist daher eine Treppenfunktion, statt eine stetigen Kurve. Darüber hinaus ist ein Entscheidungsbaum auch recht anfällig für verrauschte Daten, da in jedem Knoten auf Basis eines einzelnen Attributs die Entscheidung für den Nachfolgeknoten getroffen wird.

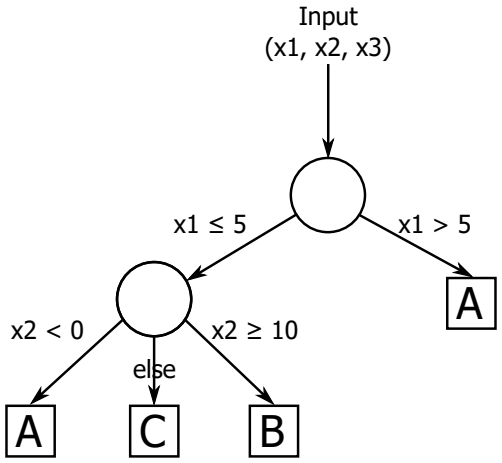



Abbildung 9: Ein Entscheidungsbaum mit dreidimensionalem Input. Anhand der Merkmale wird das entsprechende Blatt ausgewählt und der entsprechende Wert als Output geliefert. In diesem Beispiel kann der Entscheidungsbaum die drei verschiedenen Werte A, B und C zurückgeben. Der Input x3 wird nicht verwendet, hat also keinen Einfluss auf das Ergebnis des Entscheidungsprozesses.

Diese Probleme lassen sich durch einen *Random Forest* lösen. Ein Random Forest repräsentiert eine Vielzahl von unterschiedlichen Entscheidungsbäumen, die anhand der Trainingsdaten gelernt wurden. Sie unterscheiden sich, da bei dem Training auch Zufallseffekte eingehen. Ein neues Eingabedatum wird nun auf jeden Baum des Random Forests angesetzt. Soll der Random Forest für die Klassifikation eingesetzt werden, so wird die Vorhersage durch Mehrheitsentscheid getroffen. Für eine Regression kann beispielsweise das arithmetische Mittel aller Entscheidungen gebildet werden.

Der Classification and Regression Trees (CART) Algorithmus ist eine gängige Methode, um Entscheidungsbäume zu trainieren. Der Algorithmus erzeugt immer einen Binärbaum, d. h. von jedem Knoten des Baums gehen immer genau zwei Kindknoten aus (mit Ausnahme der Blätter). Der Algorithmus startet zunächst mit der Erzeugung eines Wurzelknotens, dem alle Trainingsdaten zugeordnet sind. Dieser Knoten soll nun in zwei Kindknoten aufgeteilt werden, sodass unterschiedliche Targets möglichst gut voneinander getrennt werden. Hierfür muss eine Trennungsregel (*Splitting Rule*) eingeführt werden, die auf Basis eines Attributes eine Unterscheidung zwischen linkem und rechtem Kindsknoten trifft. Nach gewissen Kriterien (*Splitting Criterion*) wird die Entscheidung getroffen, welches Attribut getrennt werden muss, um den größtmöglichen Informationsgewinn zu erhalten. Nachdem ein Knoten so getrennt wurde, wird entsprechend mit den Kindsknoten fortgefahren. Ein Knoten wird nicht weiter getrennt, wenn spezielle Stoppregeln (*Stopping Rules*) auf diesen Knoten zutreffen. Dies kann bspw. sein, wenn ein Knoten nur noch Muster enthält, die alle mit dem gleichen Target versehen sind. Für eine detaillierte Beschreibung des Algorithmus sei auf [Ste09] verwiesen.

### 2.2.5 Support Vector Regression

#### Definitionen

Bevor das Prinzip von Support Vector Maschinen (SVM) erläutert werden kann, müssen zunächst einige grundlegende Definitionen gegeben werden, die zentrale Bestandteile des Verfahrens sind.

Ein Hilbertraum $\mathcal{H}$ ist ein Vektorraum mit einem Skalarprodukt. Dazu gibt es eine Featuremap $\Phi$, die Elemente aus einer nichtleeren Menge $X$ in diesen Hilbertraum abbildet:

$$\Phi : X \to \mathcal{H}$$

Die Bildpunkte werden Features genannt und sollen $x \in X$ möglichst gut charakterisieren, sodass mehr Struktur zur Beschreibung der Daten als in $X$ entsteht [Bre15]. Zu dieser Featuremap $\Phi$ existiert ein sogenannter Kern $k : X \times X \to \mathbb{R}$ mit

$$k(x, z) := \langle \Phi(x), \Phi(z) \rangle_{\mathcal{H}} .$$

Mit Hilfe eines Kerns lässt sich also das Skalarprodukt in dem zugehörigen Hilbertraum berechnen.

#### Grundprinzip einer SVM

Der Grundansatz für SVMs wurde durch den Generalized Portrait Algorithm von Vapnik & Lerner gegeben [VL63]. Dieser trennt eine widerspruchsfreie Trainingsmenge durch eine Hyperebene. In der Literatur wird dieses Verfahren oft auch als *Lineare SVM* bezeichnet.

Seien $X$ und $D$ nach Kapitel 2.2.1, sowie $Y = \{-1, 1\}$, für die ein Vektor $w \in \mathbb{R}^d$ mit $\|w\|_2 = 1$ und eine reelle Zahl $b \in \mathbb{R}$ existieren, sodass gilt

$$\begin{aligned} \langle w, x_i \rangle + b > 0 \quad & \forall i \text{ mit } y_i = +1 \\ \langle w, x_i \rangle + b < 0 \quad & \forall i \text{ mit } y_i = -1. \end{aligned}$$

$\langle w, x_i \rangle + b$ definiert nun eine Hyperebene, die die Trainingsdaten optimal trennt. Hier zeigt sich aber auch gleich die große Schwäche des Verfahrens, denn es wird vorausgesetzt, dass die Daten linear trennbar sind. Dies ist aber für echte Anwendungen oftmals nicht der Fall, sodass sich hier der Generalized Portrait Algorithm als ungeeignet erweist.

Aufgrund dieser Probleme wurde das Verfahren erweitert und stellt in seiner Gesamtheit die heute bekannte SVM oder auch *Kernel SVM* dar [Vap13]. Hierfür bildet man die Trainingsdaten mittels einer Featuremap in einen höherdimensionalen Hilbertraum $\mathcal{H}_0$ ab. Durch diese Abbildung der ursprünglichen, nicht-linear

trennbaren Trainingsdaten in den höherdimensionalen Raum lässt sich eine lineare Trennbarkeit erreichen, um anschließend den Generalized Portrait Algorithm in diesem Featureraum anzuwenden. So erhält man eine trennende Hyperebene in dem Hilbertraum, die zurückabgebildet in den Ursprungsraum die Klassen optimal trennt. Um nun zusätzlich noch die gewollte Fehlertoleranz umzusetzen, wird das Verfahren um Schlupfvariablen erweitert. Das gesamte Optimierungsproblem für eine SVM sieht dann wie folgt aus:

$$\begin{aligned} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i \to min \quad & ,w \in \mathcal{H}_0, b \in \mathbb{R}, \xi \in \mathbb{R}^n \\ \text{u. d. N. } y_i\left(\langle w, \Phi(x_i)\rangle + b\right) \geq 1 - \xi_i \quad & ,1 \leq i \leq n \\ \xi_i \geq 0 \quad & ,1 \leq i \leq n \end{aligned}$$

Um zu vermeiden, dass zu viele Berechnungen in dem Hilbertraum vorgenommen werden müssen, nutzt man die Lagrangemethode um die folgende duale Problemstellung zu erhalten:

$$\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j)\rangle \to max \quad \alpha \in [0, C]^n$$

$$\text{u. d. N. } \sum_{i=1}^{n} y_i \alpha_i = 0,$$

wobei die $\alpha_{i/j}$ die Lagrange-Multiplikatoren sind. Lediglich der Term $\langle \Phi(x_i), \Phi(x_j)\rangle$ muss noch in $\mathcal{H}_0$ berechnet werden, doch mit Hilfe eines Kerns $k$ kann auch dieses Skalarprodukt in $\mathbb{R}^d$ berechnet werden, sogar ohne Kenntnis der eigentlichen Featuremap.

Das Ergebnis dieses Verfahrens ist eine Entscheidungsfunktion $f_D(x)$, die für den Vektor $x$ bestimmt, auf welcher Seite der Hyperebene er liegt, d. h. zu welcher Klasse er gehört.

$$\begin{aligned} & f_D(x) = sgn\left(\langle w_D, \Phi(x)\rangle + b^*\right) \\ \text{mit} \quad & w_D = \sum_{i=1}^{n} \alpha_i^* y_i \Phi(x_i) \\ \text{und} \quad & b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i k(x_i, x_j) \\ \Rightarrow \quad & f_D(x) = sgn\left(\sum_{i=1}^{n} \alpha_i^* y_i k(x_i, x) + y_j - \sum_{i=1}^{n} \alpha_i^* y_i k(x_i, x_j)\right) \end{aligned}$$

Die Berechnung von $w_D$ lässt erkennen, dass nur Trainingsvektoren mit $\alpha_i^* \neq 0$ in die Berechnung der Hyperebene mit einfließen. Daher werden gerade diese Vek-

toren die *Supportvektoren* genannt und sind auch namensgebend für das gesamte Verfahren.

#### Regressionsverfahren

Die Herleitung einer Support Vector Regression funktioniert analog zur Herleitung einer herkömmlichen SVM. Anstatt die Hyperebene so zu legen, dass die Trainingsdaten optimal separiert werden, soll stattdessen eine Hyperebene gefunden werden, deren Distanz zu den erwarteten Ergebnissen kleiner als ein Grenzwert $\epsilon$ ist. Dieses Problem lässt sich folgendermaßen definieren:

$$\begin{aligned} &\text{minimiere} \frac{1}{2} \langle w, w \rangle \\ &\quad \text{u. d. N.} \, |y_i - (\langle w, x_i \rangle + b)| \leq \epsilon \end{aligned}$$

Auch hier können dann wieder Schlupfvariablen eingeführt werden, durch die eine Verletzung der Bedingung möglich wird. Andernfalls ist nicht garantiert, dass die Hyperebene überhaupt existiert. Anschließend kann mit der Lagrangemethode wieder die duale Problemstellung hergeleitet werden. Daraus folgt dann die Regressionsfunktion:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} (\alpha_n - \alpha_n^*) \langle x_i, x \rangle + b \\ b &= \begin{cases} y_i - \langle w, x_i \rangle - \epsilon & , \alpha_i \in (0, C) \\ y_i - \langle w, x_i \rangle + \epsilon & , \alpha_i^* \in (0, C) \end{cases} \\ w &= \sum_{i=1}^{n} (\alpha_n - \alpha_n^*) \, x_i \end{aligned}$$

Für eine genauere Herleitung sei auf [SS98] verwiesen. Auch bei der Support Vector Regression können Kerne eingesetzt werden, um Nichtlinearität zu erreichen. Dadurch verändert sich die Regressionsfunktion zu:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} (\alpha_n - \alpha_n^*) \, k(x_i, x) + b \\ w &= \sum_{i=1}^{n} (\alpha_n - \alpha_n^*) \, \Phi(x_i) \end{aligned}$$

### 2.2.6 Neuronale Netze

Ein künstliches neuronales Netz ist ein Netzwerk von vielen Knoten, die in Schichten (*Layern*) angeordnet und durch gerichtete Kanten miteinander vernetzt sind (siehe Abbildung 10). Dabei unterscheidet man zwischen dem Input Layer, der die Eingabedaten annimmt, den Hidden Layern, die den Output ihrer vorherigen

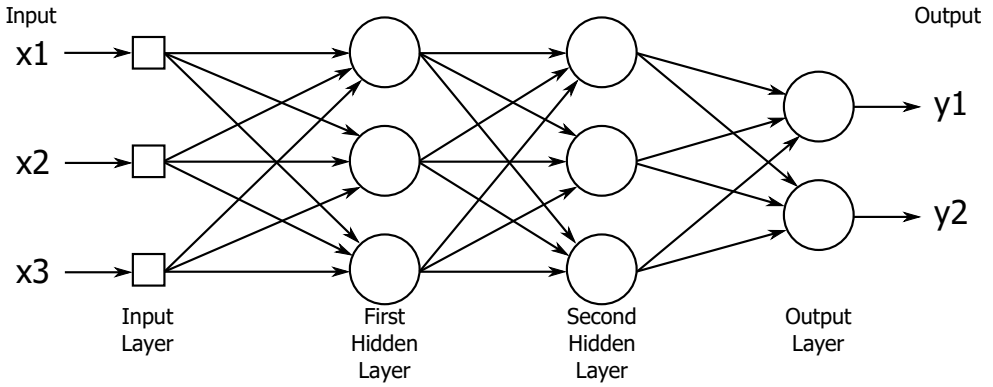



Abbildung 10: Ein feedforward Netz, in dem jeder Knoten mit allen Knoten der Vorgänger- und Nachfolgerschicht verbunden ist (fully-connected). Ein dreidimensionaler Input-Vektor $(x1, x2, x3)^T$ wird in das neuronale Netz eingegeben und durch zwei Hidden Layer verarbeitet. Der Output Layer erzeugt eine zweidimensionale Ausgabe $(y1, y2)^T$.

Schicht weiterverarbeiten, und dem Output Layer, der die Ausgabe erzeugt. Während der Input Layer die Daten lediglich an die nächste Schicht weitergibt, finden in den Knoten der anderen Schichten Berechnungen statt. Durch die Berechnungen in den Hidden Layern sollen die Eingabedaten so transformiert werden, dass der Output Layer möglichst präzise die gewünschten Ergebnisse produzieren kann.

Die bisher eingeführte Art von neuronalen Netzen bezeichnet man als *Feedforward Neural Network*, also ein Netz, in dem die berechneten Outputs der jeweiligen Knoten immer nur vorwärts in die nächste Schicht propagiert werden. Es gibt aber auch diverse Variationen dieser Architektur, die für bestimmte Zwecke besser geeignet sind. Diese sollen im Folgenden genauer vorgestellt werden. Zuvor wird aber noch ein detaillierterer Einblick in die genaue Funktionsweise der Berechnungsknoten, der Netzwerkachitektur und den damit verbunden Trainingsalgorithmen gegeben.

### Berechnungsknoten

Die Grundstruktur eines Berechnungsknotens ist in Abbildung 11 zu sehen. Diese Berechnungsfolge lässt sich durch folgende Gleichungen beschreiben:

$$\begin{aligned} u_k &= \sum_{j=1}^{m} w_{kj} \cdot i_j & & & v_k &= \sum_{j=0}^{m} w_{kj} \cdot i_j \\ v_k &= u_k + b_k & & \Leftrightarrow & & \\ o_k &= \varphi(v_k) & & & o_k &= \varphi(v_k) \end{aligned}$$

$k$ bezeichnet hierbei den $k$-ten Knoten und $m$ die Anzahl der Eingaben dieses Knotens. Die verschiedenen Eingaben $i_j$ werden mit ihren entsprechenden Gewichten $w_{kj}$ multipliziert und aufsummiert. Zu dieser Summe $u_k$ wird noch ein *Bias*

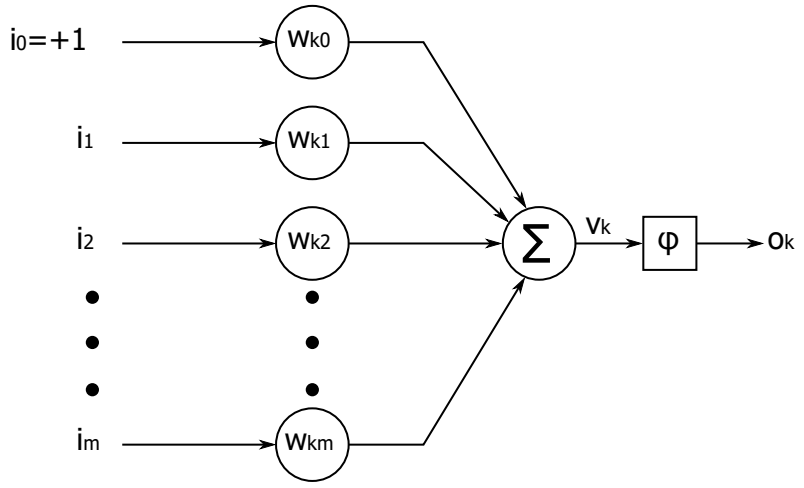



Abbildung 11: Schematische Darstellung eines Berechnungsknotens

$b_k$ addiert und anschließend als Eingabe $v_k$ für die Aktivierungsfunktion $\varphi(\cdot)$ verwendet. Das Ergebnis $o_k$, repräsentiert die Ausgabe des Knotens und wird dann an die Knoten der nächsten Schicht propagiert. Das Bias $b_k$ wird eingeführt um die Ausdrucksstärke eines Berechnungsknotens zu erhöhen, da dadurch eine Wertverschiebung von $u_k$ möglich wird. Um die Einführung dieses Bias zu vereinfachen, wird eine neue Eingabe $i_0$ eingeführt, die den konstanten Wert $i_0 = +1$ hat. Dazu kommt ein neues Gewicht $w_{k0}$, mit der das Bias $b_k = i_0 \cdot w_{k0}$ gebildet wird. Auf diese Weise wird das Bias wie ein Eingabesignal mit entsprechendem Gewicht modelliert, analog zu den anderen Eingaben des Berechnungsknotens. Die Aktivierungsfunktion kann eine beliebige differenzierbare Funktion sein, die ausgehend von der Eingabe eine reellwertige Ausgabe produziert. Einige ausgewählte Aktivierungsfunktionen sind in Abbildung 12 dargestellt. Die hier gewählten Bezeichnungen $i_j$ und $o_k$ weichen von den in der Literatur gebräuchlichen Bezeichnungen $x_j$ und $y_k$ ab, um diese klar von der Eingabe $x$ und Ausgabe $y$ für ein Modell zu differenzieren.

### Netzwerkarchitektur

Wie eingangs bereits beschrieben, setzt sich ein künstliches neuronales Netz im Allgemeinen aus einem Input Layer, einem Output Layer und beliebig vielen Hidden Layern zusammen. Die Anzahl der Knoten in dem Input Layer entspricht genau der Anzahl an Features, die zusammen ein Eingabemuster ausmachen. Die Anzahl der Hidden Layer, sowie die Anzahl an Knoten in jedem dieser Layer ist durch den Anwender festzulegen. Das *universal approximation theorem* besagt, dass neuronale Netze mit nur einem Hidden Layer jede messbare Funktion approximieren können, sofern das Netz aus ausreichend vielen Knoten besteht [GBC16]. Da aber die Anzahl der benötigten Knoten sehr groß werden könnte, eignet sich ein Netz mit mehreren Hidden Layern, dafür aber weniger Knoten, ggf. besser. Daher wird die Netzwerkarchitektur meist experimentell festgelegt. Erfahrungswerte für gut

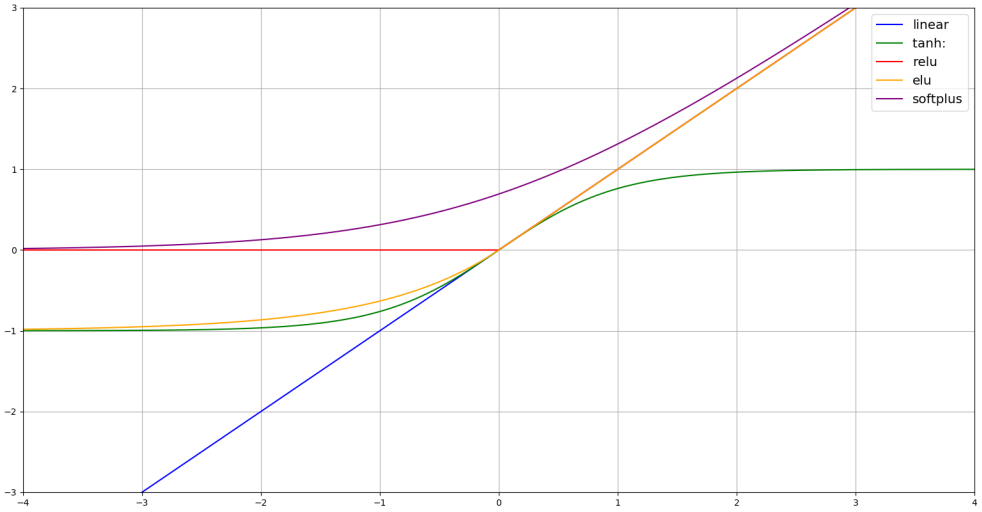



Abbildung 12: Die Abbildung zeigt die Kurvenverläufe verschiedener Aktivierungsfunktionen. *linear*: $f(x) = x$; *tanh*: $f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$; *relu*: $f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$; *elu*: $f(x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & x \geq 0 \end{cases}$; *softplus*: $f(x) = ln(1+e^x)$. Da $f(x) = x, x > 0$ bei den Aktivierungsfunktionen linear, relu und elu gilt, überlagern sich die Graphen in diesem Wertebereich.

funktionierende Netzwerkarchitekturen lassen sich innerhalb einer Problemdomäne oftmals recht gut übertragen. Das Gleiche muss aber nicht für Probleme aus völlig unterschiedlichen Domänen gelten.

Der Output Layer bestimmt die Ausgabe des neuronales Netzes, d. h. die Anzahl der Knoten in dem Output Layer entspricht auch der Anzahl an Ausgaben des Netzes. Je nachdem, ob das neuronale Netz als Klassifikator oder Regressor eingesetzt werden soll, müssen für die Knoten in dem Output Layer entsprechende Aktivierungsfunktionen eingesetzt werden. Um einen Klassifikator zu modellieren, ist es eine gängige Variante, dass das neuronale Netz genauso viele Ausgabeknoten besitzt, wie es unterschiedliche Ausgaben gibt. Den Knoten wird dann eine Aktivierungsfunktion zugewiesen, deren Ausgabe sich auf Werte in dem Intervall $(0, 1)$ beschränkt. Der einzelne Wert eines jeden Knotens kann dann als Wahrscheinlichkeit interpretiert werden, dass das Eingabemuster zu dem entsprechenden Label passt. Im Falle einer Regression wird stattdessen üblicherweise eine lineare Funktion verwendet, da diese den Wertebereich nicht einschränkt. Soll ein Wert vorhergesagt werden, so muss auch nur ein Knoten verwendet werden. Es können aber auch mehrere Ausgabeknoten eingesetzt werden, um mehrere Werte zu prognostizieren.

### Trainingsalgorithmus

Durch das Training des Netzes sollen die freien Parameter so eingestellt werden, dass das neuronale Netz möglichst gute Vorhersagen für die Eingabedaten trifft. Da die Netzwerkarchitektur inklusive der Aktivierungsfunktionen bereits vor dem Training festgelegt werden, sind diese Parameter nicht mehr einstellbar. Die einzigen freien Parameter bei einem künstlichen neuronalen Netz stellen die Kantengewichte zwischen den Knoten der verschiedenen Schichten, sowie das zu jedem Knoten zugehörige Bias dar. Um diese Gewichte optimieren zu können, muss eine Fehlerfunktion der Ergebnisse in Abhängigkeit der Gewichte definiert werden. Diese Fehlerfunktion soll dann durch den Trainingsprozess minimiert werden.

Die Ergebnisse des künstlichen neuronalen Netzes werden durch die Knoten in dem Output Layer definiert. Der Fehler des $j$-ten Outputknotens für das Trainingspaar $(x_i, y_i) \in D$ ist dann definiert durch

$$e_j(x_i) = y_{ij} - \hat{y}_j(x_i),$$

wobei $\hat{y}_j$ die Berechnungen vom Input Layer bis zum $j$-ten Knoten des Output Layers beschreibt. In diese Funktion fließen implizit die Kantengewichte des Netzes ein. Summiert man diese Fehler für alle Ausgabeknoten quadratisch auf, so erhält man die übliche Fehlerfunktion

$$\mathscr{E}(x_i) = \frac{1}{2} \sum_{j \in C} e_j^2(x_i) = \frac{1}{2} \sum_{j \in C} (y_{ij} - \hat{y}_j(x_i))^2$$

für ein einzelnes Trainingsmuster $x_i$. $C$ beschreibt die Menge aller Knoten in dem Output Layer und der Skalierungsfaktor $\frac{1}{2}$ wird eingeführt, um weitergehende analytische Herleitungen zu vereinfachen [Hay09]. Diese Fehlerfunktion hängt nur von einem einzigen Trainingsmuster ab und eine Minimierung kann auch nur für eben dieses Trainingsmuster durchgeführt werden. D. h. für jedes Trainingsmuster wird diese Fehlerfunktion einzeln minimiert und die Gewichte entsprechend angepasst. Dieses Verfahren wird als *online learning* bezeichnet. Wird das arithmetische Mittel der Fehlerfunktion über alle Trainingsbeispiele gebildet, so erhält man:

$$\mathscr{E}_{av}(D) = \frac{1}{|D|} \sum_{(x_i, y_i) \in D} \mathscr{E}(x_i) = \frac{1}{2 \cdot |D|} \sum_{(x_i, y_i) \in D} \sum_{j \in C} e_j^2(x_i)$$

Mit dieser Fehlerfunktion kann eine Minimierung auf Basis aller Trainingsbeispiele durchgeführt werden. Diese Trainingsvariante wird *batch learning* genannt. Durch einfache Modifikation kann $\mathscr{E}_{av}$ auch so angepasst werden, dass nur ein Bruchteil aller Trainingsbeispiele ausgewertet werden kann. Dies stellt einen Kompromiss aus *online learning* und *batch learning* dar und nennt sich *mini-batch*.

Bisher wurde lediglich davon gesprochen, dass die Fehlerfunktion minimiert werden soll. Aufgrund der Komplexität von neuronalen Netzen, sowie den üblicherweise nichtlinearen Aktivierungsfunktionen, ist eine analytische Bestimmung des Minimums impraktikabel. Stattdessen wird der *Backpropagation*-Algorithmus eingesetzt, der im Folgenden genauer erläutert wird.

**Backpropagation** Der Backpropagation-Algorithmus ist ein iteratives Verfahren, das sukzessiv die Gewichte eines neuronalen Netzes optimiert, um die Fehlerfunktion zu minimieren. Der Einfachheit halber werden die folgenden Schritte nur in Bezug auf online learning beschrieben.

Nachdem eine Initialisierung stattgefunden hat, wird in jeder Iteration eine Reihe von Berechnungen durchgeführt, deren Ergebnis ein Hinweis dafür ist, wie die Gewichte angepasst werden müssen. Diese Korrektur wird durch die *delta rule* beschrieben:

$$\Delta w_{pq}(x_i) = -\eta \frac{\partial \mathscr{E}(x_i)}{\partial w_{pq}(x_i)}$$

$w_{pq}$ beschreibt hier das Kantengewicht des Knotens $p$ und seiner Eingabe $q$. Durch Differenzierung von $\mathscr{E}(x_i)$ und weiteren Umformungen folgt

$$\Delta w_{pq}(x_i) = \eta \delta_p(x_i) \cdot o_p(x_i).$$

$\eta$ ist ein frei einstellbarer Parameter, der als *Lernrate* bezeichnet wird. $\delta_p(x_i)$ beschreibt den lokalen Gradienten und $o_p(x_i)$ die Ausgabe des Berechnungsknotens $p$. Die Berechnung des lokalen Gradienten ist abhängig davon, ob der Knoten im Output Layer oder einem der Hidden Layer liegt.

$$\delta_p^{(l)}(x_i) = \begin{cases} e_p^{(L)}(x_i) \cdot \varphi_p'(v_p^{(L)}(x_i)) & \text{Berechnungsknoten p liegt in Output Layer L} \\ \varphi_p'(v_p^{(l)}(x_i)) \cdot \sum_k \delta_k^{(l+1)}(x_i) \cdot w_{kp}^{(l+1)}(x_i) & \text{Berechnungsknoten p liegt in Hidden Layer l} \end{cases}$$

Die Berechnungsvorschrift für einen Knoten im Hidden Layer von $\delta_p^{(l)}(x_i)$ beschreibt eine rekursive Berechnung, da die lokalen Gradienten in der Folgeschicht ausgewertet werden müssen. Dies gilt aber nicht für den Output Layer, dessen lokale Gradienten direkt aus den Fehlern und der Ableitung der Aktivierungsfunktion bestimmt werden können. Daher wird in der praktischen Ausführung des Backpropagation-Algorithmus zunächst die Gewichtskorrektur des Output Layers bestimmt, wobei die hierfür berechneten lokalen Gradienten dann für die nächst vorherige Schicht weiterverwendet werden können.

Im Folgenden ist der Ablauf des Algorithmus genauer dargestellt.

1. *Initialisierung*: Den Gewichten aller Kanten müssen Startwerte zugewiesen werden. Hierfür gibt es verschiedene Möglichkeiten. Entweder wird allen Ge-

wichten ein konstanter Wert zugewiesen (bspw. 0 oder 1), oder die Startwerte werden mit Hilfe einer Wahrschheinlichkeitsverteilung ermittelt.

2. *Iteration*: Eine Iteration des Backpropagation-Algorithmus wird *Epoche* genannt. In jeder Epoche wird das Netzwerk mit den Trainingsdaten konfrontiert, um daraus die Anpassung der Gewichte abzuleiten. Innerhalb einer Epoche werden für jeden Batch zwei Schritte durchgeführt, die Vorwärtsberechnung (*forward pass*) und die Rückwärtsberechnung (*backward pass*). Die Batchgröße wird durch den Anwender definiert und kann zwischen eins (online learning) und $|D|$ (batch learning) liegen. Der Algorithmus läuft so lange, bis ein Stoppkriterium erreicht wurde. Dies kann bspw. eine Begrenzung der Anzahl an Epochen sein, oder aber falls keine wirkliche Verbesserung des Fehlers zwischen den verschiedenen Epochen erreicht wird.

   2.1 *Vorwärtsberechnung*: Es wird ein Trainingsmuster als Eingabe an das Netzwerk angelegt. Aus der Eingabe berechnen die Knoten der ersten Schicht ihre Ausgabe und geben diese an die nächste Schicht weiter. So wird Schicht für Schicht die Ausgabe der Knoten berechnet und weitergegeben, bis der Output Layer die Ergebnisse berechnet hat. Für die Knoten des Output Layers können die Fehler $e_j(x_i)$ zu den erwarteten Ausgaben berechnet werden.

   2.2 *Rückwärtsberechnung*: Nun wird mit Hilfe der Berechnungsvorschrift von $\Delta w_{pq}(x_i)$ die Gewichtskorrektur für den Output Layer berechnet. Dies wird dann sukzessive für die nächst vorherige Schicht durchgeführt, bis die erste Schicht des Netzwerks erreicht wurde. Da nun für jeden Knoten aller Schichten die Gewichtskorrektur vorgenommen wurde, können die Gewichte nun entsprechend angepasst werden.

Es gibt diverse Erweiterungen des Backpropagation-Algorithmus, die bspw. die Lernrate während des Trainings anpassen. Für eine detaillierte Beschreibung solcher Verfahren sei auf [RB93; KB14] verwiesen.

### 2.2.7 Convolutional Neural Network

Vom Prinzip her sind *Convolutional Neural Networks* einfache feedforward Netze, die sich aber dadurch auszeichnen, dass mindestens einer der Hidden Layer (typischerweise der Erste) eine sogenannte *Convolution* berechnet. Die Idee der Convolution ist, dass ein Teilbereich der Eingabedaten (das sogenannte *Receptive Field*) mit einem *Kernel* verrechnet wird und so einen einzelnen Wert produziert. Dann wird der Kernel verschoben, sodass ein neuer Teilbereich der Eingabedaten überdeckt wird. Das Ergebnis dieses Prozesses wird als *Feature Map* bezeichnet und ist von niedrigerer Dimension als die Eingabe. Diese Feature Map wird dann als Eingabe für die nächste Schicht des neuronalen Netzes verwendet. Ein Beispiel

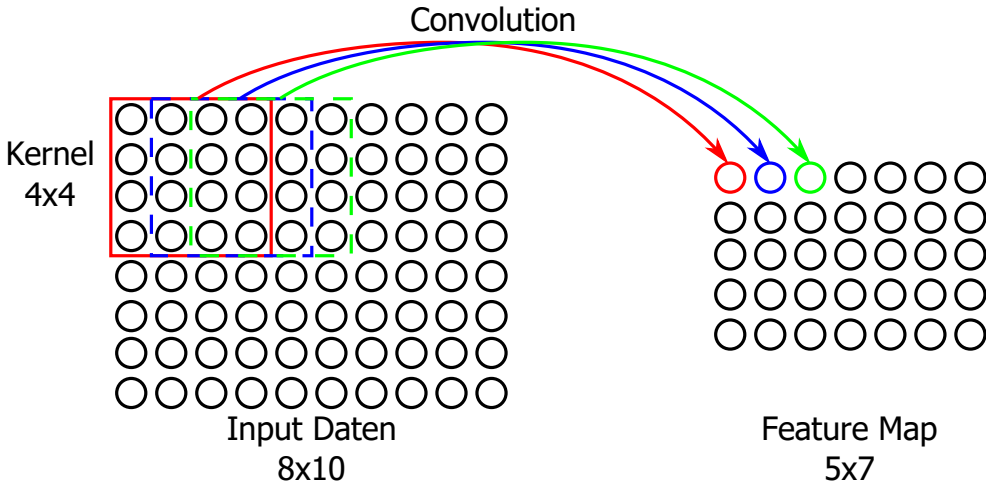



Abbildung 13: Die Abbildung zeigt beispielhaft die ersten drei Schritte eines Convolution-Prozesses. Die Eingabedaten sind dabei auf einem zweidimensionalen Gitter angeordnet, wie es bei Bilddaten beispielsweise üblich ist. Der Zusammenhang der verschiedenen Receptive Fields und den daraus resultierenden Punkten in der Feature Map ist farblich dargestellt.

solch einer Convolution ist in Abbildung 13 dargestellt. Der Kernel entspricht einer Matrix, deren Einträge wiederum als Gewichte bezeichnet werden. Diese Gewichte werden aber nicht zu Beginn festgelegt, sondern während des Optimierungsprozesses bestimmt. Das Prinzip einer Convolution hat daher eine starke Ähnlichkeit zu den normalen Hidden Layern eines feedforward Netzes. Sie unterscheiden sich aber in zwei wesentlichen Dingen:

- Jeder Punkt der Feature Map hängt nur von dem zugehörigen Receptive Field ab, zu den anderen Eingabeknoten besteht hingegen keine Verbindung.
- Alle Punkte der Feature Map teilen sich den gleichen Kernel, d. h. es gibt deutlich weniger freie Parameter. Das Convolutional Neural Network kann daher zwar weniger komplexe Zusammenhänge modellieren, jedoch erhöht sich dadurch auch gleichzeitig die Fähigkeit des Modells, gut zu generalisieren. [Hay09]

Durch die Convolution lernt das neuronale Netz eine Operation, die eine aussagekräftige Transformation der Eingabedaten ist. Es ist also eine Art von Vorverarbeitung, die bei ursprünglichen Ansätzen selbst umgesetzt werden musste. Da ein Kernel aber nur ein Feature aus den Eingabedaten extrahieren kann, werden in der Regel eine Vielzahl von Convolutions parallel durchgeführt. Dadurch erhöht sich die Anzahl der erlernbaren Features.

Nach einer Convolution wird oftmals ein *pooling layer* eingefügt. Das Pooling beschreibt eine Operation, die eine Gruppe von Eingabedaten zu einem Wert zusammenfasst. Das kann bspw. durch das Bilden des Maximums, Minimums oder Durchschnitts geschehen. Dadurch soll das Convolutional Neural Network weniger

anfällig für Rauschen in den Eingabedaten werden. Außerdem verringert sich auf diese Weise auch die Menge an Daten, die in der nächsten Schicht verarbeitet werden müssen [Hay09].

Nachdem mehrmals Convolutions und Pooling durchgeführt wurden, werden die Ausgaben der letzten Schicht an ein feedforward Netz weitergegeben. Dieses kann auch nochmal aus einigen Hidden Layern bestehen, bis abschließend durch den Output Layer eine Ausgabe erzeugt wird. Daher wird ein Convolutional Neural Network oftmals in die zwei Bestandteile *Feature Extraction* (Convolution und Pooling Layer) und *Klassifikation/Regression* (Feedforward Netz) unterteilt.

### 2.2.8 Recurrent Neural Network

In bisherigen feedforward Netzen wird der Output eines Knotens nur in die nächste Schicht propagiert. Ein Knoten kann daher sein Verhalten für die nächste Berechnung nicht beeinflussen. Diese Idee ist aber von großem Interesse, wenn zwischen den Eingabedaten Zusammenhänge bestehen. Bei der Analyse von Text ist es beispielsweise essenziell, dass Wörter im Kontext der Nachbarwörter, des Satzes oder gar des Absatzes betrachtet werden. Wird das Wort *Decke* nur für sich betrachtet, so kann nicht unterschieden werden, ob eine *Bettdecke* oder eine *Zimmerdecke* gemeint ist.

Damit ein Knoten seine Berechnung für kommende Eingabedaten in Bezug auf das aktuelle Eingabedatum beeinflussen kann, werden Zyklen in das neuronale Netz eingebaut. Durch diese Zyklen beeinflusst der Output eines Knotens auch die Berechnung im nächsten Schritt. Dadurch sind Reccurent Neural Networks mit endlicher Größe in der Lage, jede Funktion zu berechnen, die durch eine Turingmaschine berechenbar ist [GBC16]. Es gibt verschiedene Arten, wie diese Zyklen umgesetzt werden (siehe beispielsweise Abbildung 14). Jedoch leiden viele Varianten unter dem *vanishing and exploding gradient problem*. Dieses Problem beschreibt die Tatsache, dass in vielen Regionen des Parameterraums die Gradienten entweder fast 0 (vanishing) werden oder sehr große Werte annehmen (exploding). Wird ein Gradient fast 0, so kommt das Training des neuronalen Netzes fast zum erliegen, da die Gewichte nur noch sehr langsam angepasst werden. Im anderen Fall wird das Training sehr instabil, wenn die Gradienten „explodieren“. Gerade in Bezug auf Langzeitabhängigkeiten leiden Recurrent Neural Networks stark unter verschwindenden Gradienten. Da bei Langzeitabhängigkeiten die zu verknüpfenden Informationen schon so weit auseinander liegen, werden diese durch kurzfristigere Abhängigkeiten überlagert. Es kann daher sehr lange dauern oder gar unmöglich sein, diese Verknüpfung zu lernen. Dieses Problem kann durch *Long short-term memory*-Netze (LSTM) behoben werden.

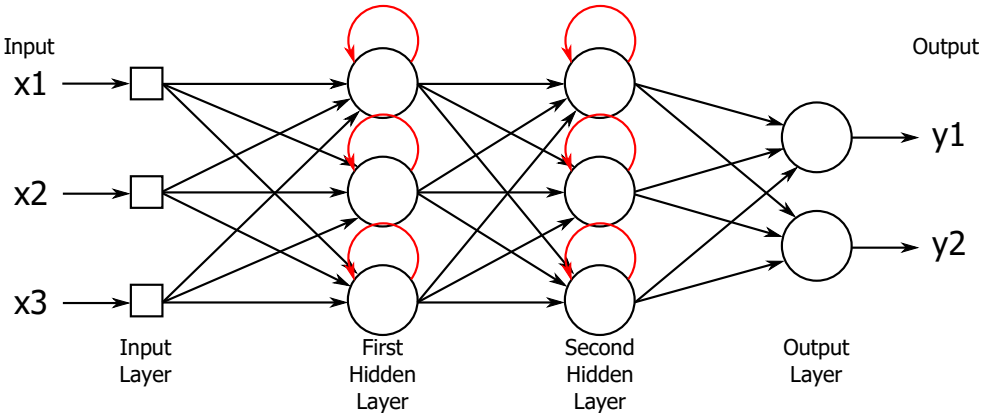



Abbildung 14: Ein Recurrent Neural Network, in dem jeder Knoten eine Schleife zu sich selbst hat (rot dargestellt). So ist der Output eines Knotens bei der $i$-ten Berechnung auch Input für die $i+1$-te Berechnung.

### Long short-term memory-Netze

Ein Long short-term memory-Netz (LSTM) ist eine spezielle Struktur, in der ein klassischer Berechnungsknoten durch weitere Strukturen erweitert wird. Ein stark vereinfachtes Schema ist in Abbildung 15 dargestellt. Kernelement einer LSTM-Zelle ist die *State Unit*, die Werte über längere Zeit speichern kann. Daher wird sie auch als Gedächtnis (memory) bezeichnet, was unter anderem auch namensgebend für das Verfahren ist. Neben der State Unit gibt es noch ein *Input Gate*, ein *Forget Gate* und ein *Output Gate*. Diese Gates kontrollieren den Informationsfluss innerhalb der LSTM-Zelle. Ihr Aufbau ist dabei genau gleich und in Abbildung 16 dargestellt. Als Eingabe zum Zeitpunkt $t$ hat ein Gate das aktuelle Eingabemuster $x_t$ und den letzten Wert der State Unit $s_{t-1}$. Diese werden mit Gewichtsmatrizen $U$ und $W$, sowie einem Biasvektor $b$ verrechnet. Das Ergebnis wird durch eine Aktivierungsfunktion auf das Intervall $(0, 1)$ abgebildet. Hierfür wird üblicherweise eine Sigmoidfunktion, also eine Funktion mit S-förmigem Kurvenverlauf verwendet. Der einzige Unterschied zwischen den verschiedenen Gates ist die Weiterverwendung ihrer Ausgangssignale. Dies wird im Folgenden genauer erläutert. [GBC16]

**Input Gate** Der Ausgangswert des Input Gates berechnet sich durch

$$i_t = \sigma\left(b^i + U^i x_t + W^i h_{t-1}\right),$$

wobei $\sigma$ eine Sigmoidfunktion und $h_{t-1}$ die Ausgabe der LSTM-Zelle für das letzte Eingabedatum beschreiben.

**Forget Gate** Der Ausgangswert des Forget Gates berechnet sich durch

$$f_t = \sigma\left(b^f + U^f x_t + W^f h_{t-1}\right).$$

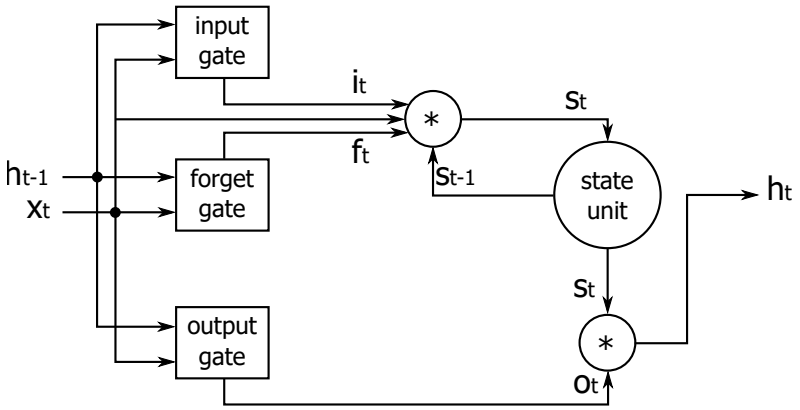



Abbildung 15: Die Abbildung zeigt einen vereinfachten Aufbau einer LSTM-Zelle. Die verschiedenen Knoten führen hierbei mehrere Berechnungen durch.

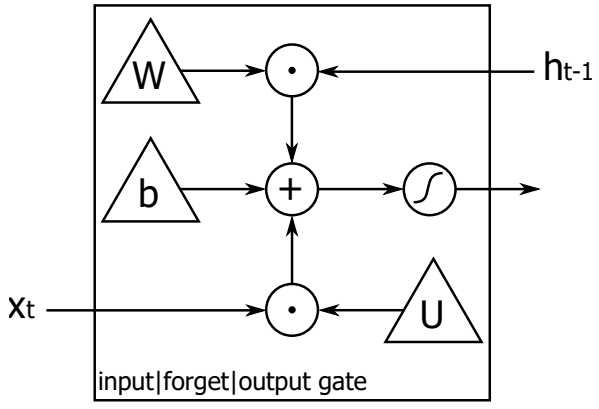



Abbildung 16: Die Abbildung zeigt den Aufbau eines Gates einer LSTM-Zelle.

**State Unit** Der neue Wert $s_t$ der State Unit berechnet sich in Abhängigkeit von dem Input Gate und dem Forget Gate. Das Input Gate steuert, wie stark das neue Eingabemuster $x_t$ den Wert der State Unit $s_t$ beeinflusst. Das Forget Gate hingegen kann die aktuellen Zustände der State Unit löschen. Die Berechnung sieht folgendermaßen aus:

$$s_t = f_t \circ s_{t-1} + i_t \circ \sigma\left(b^s + U^s x_t + W^s h_{t-1}\right),$$

wobei der Operator $\circ$ das Hadamard-Produkt beschreibt, also das elementweise Produkt der einzelnen Vektoreinträge. So können die Gates für die einzelnen Einträge der Vektoren entscheiden, wie stark diese Werte „durchgelassen“ werden. Durch eine 0 würde das korrespondierende Signal komplett ausgelöscht werden, während es durch eine 1 gar nicht beeinflusst wird.

**Output Gate** Der Ausgangswert des Output Gates berechnet sich durch

$$o_t = \sigma\left(b^o + U^o x_t + W^o h_{t-1}\right).$$

Das Ausgabesignal $o_t$ beeinflusst direkt das Ausgabesignal der gesamten LSTM-Zelle $h_t$. Dieses berechnet sich durch

$$h_t = o_t \circ \sigma\left(s_t\right).$$

Durch das Training eines LSTM-Netzes müssen nun die verschiedenen Gewichtsmatrizen $U$ und $W$, sowie die Biasvektoren $b$ für die Gates und die State Unit so angepasst werden, dass die Fehlerfunktion minimal wird. Für ein konkretes Problem bedeutet das, dass die Abhängigkeiten zwischen den Eingabedaten möglichst gut erkannt werden. Durch das Forget Gate soll wiederum gewährleistet werden, dass die State Unit auch zuvor gesehene Eingabedaten wieder vergessen kann. Im Bezug auf das eingangs beschriebene Beispiel mit dem Wort *Decke* ist dies bspw. nötig, falls in zwei aufeinander folgenden Absätzen einmal von dem Wort *Bettdecke* und dann dem Wort *Zimmerdecke* die Rede ist.

# 3 Konzeption

## 3.1 Datenquellen

Das zu realisierende Prognosemodell soll zu einem Zeitpunkt $t$ Prognosen für die Preisentwicklung am Intraday-Markt auf Basis von Daten, die zu diesem Zeitpunkt zur Verfügung stehen, liefern. Dabei ist es sehr wichtig, dass die Eingabedaten auch wirklich live zur Verfügung stehen, da das Prognosemodell sonst nicht in Echtzeit verwendbar wäre.

Im Folgenden werden verschiedene Datenquellen beschrieben, die als Eingabedaten für das zu entwickelnde Modell genutzt werden sollen. Die Transaktionslogs, Prognosen für Solar- und Windenergieerzeugung sowie historische Daten über die Auktionspreise am Day-Ahead-Markt wurden durch das OFFIS und die EWE Trading GmbH zur Verfügung gestellt. Die historischen Daten des Ausgleichsenergiepreises und des Regelzonensaldos sind frei erhältlich [50H]. Für alle Datenquellen liegen die historischen Daten des Zeitraums vom 1. Januar 2015 bis zum 31. Dezember 2017 vor.

### 3.1.1 Transaktionslogs des Intraday-Markts

#### Beschreibung der Rohdaten

Die wichtigste Datenquelle stellen die Transaktionslogs des Intraday-Markts von der EPEX SPOT dar, die für das deutsch-österreichische Marktgebiet vorliegen. In den Logs werden alle Transaktionen notiert, die in Bezug auf das entsprechende Marktgebiet am Intraday-Markt stattfinden. Eine solche Transaktion besteht dabei aus den folgenden Informationen [EPE17]:

- *Date*: Das Datum des Tages, an dem die Stromlieferung stattfinden soll.
  Format: DD/MM/YYYY
- *Market Area Buy*: Eine Kennzeichnung des Landes, aus dem der Käufer agiert, d. h. in das der Strom geliefert werden soll.
  Format: zweistelliger Ländercode
- *Market Area Sell*: Eine Kennzeichnung des Landes, aus dem der Verkäufer agiert, d. h. aus dem der Strom geliefert werden soll.
  Format: zweistelliger Ländercode
- *Hour from*: Eine Kennzeichnung der Stunde bzw. Halbstunde bzw. Viertelstunde, ab der der Strom geliefert werden soll.
  Format: [1-24] für Stundenbezeichnung, [1-24]hh[1-2] für Halbstundenbezeichnung, [1-24]qh[1-4] für Viertelstundenbezeichnung

| Date | Market Area Buy | Market Area Sell | Hour from | Hour to | Volume (MW) | Price (EUR) | Time Stamp |
|---|---|---|---|---|---|---|---|
| | | | | ⋮ | | | |
| 31/12/2017 | DE | AT | 3 | 3 | 1.4 | 0.10 | 30/12/2017 22:26:00 |
| 31/12/2017 | DE | DE | 1qh3 | 1qh3 | 0.5 | -2.00 | 30/12/2017 22:26:00 |
| 31/12/2017 | DE | DE | 15qh1 | 15qh1 | 0.7 | -1.40 | 30/12/2017 22:26:00 |
| 31/12/2017 | DE | DE | 17qh1 | 17qh1 | 1.0 | -0.60 | 30/12/2017 22:26:00 |
| 31/12/2017 | DE | AT | 3 | 3 | 1.4 | 0.30 | 30/12/2017 22:26:00 |
| 31/12/2017 | DE | DE | 24qh4 | 24qh4 | 0.2 | -19.40 | 30/12/2017 22:26:00 |
| | | | | ⋮ | | | |

Tabelle 3: Auszug aus dem Transaktionslog des Intraday-Markts der EPEX SPOT für das Marktgebiet Deutschland/Österreich vom 31. Dezember 2017. Die Trade ID-Spalte wird nicht dargestellt.

- *Hour to*: Eine Kennzeichnung der Stunde bzw. Halbstunde bzw. Viertelstunde, bis zu der der Strom geliefert werden soll.
  Format: siehe Format von Hour from
- *Volume (MW)*: Die Leistung, die pro Stunde geliefert werden soll.
  Format: Dezimalzahl
- *Price (EUR)*: Der Preis, der pro MWh bezahlt wird.
  Format: Dezimalzahl
- *Time Stamp*: Ein Zeitstempel, der den Zeitpunkt beschreibt, an dem Käufer und Verkäufer den Vertrag abgeschlossen haben.
  Format: DD/MM/YY hh:mm:ss
- *Trade ID*: Eine Identifikationsnummer, die für jede Transaktion eindeutig ist.
  Format: Ganzzahl

In Tabelle 3 ist ein Auszug des Transaktionslogs vom 31. Dezember 2017 zu sehen, an dem beispielhaft die folgenden Erklärungen nachvollzogen werden können. Jede Zeile beschreibt eine einzelne Transaktion.

Die Trade ID ist für die zu lösende Problemstellung eine völlig irrelevante Information, da sie der EPEX SPOT lediglich zur Abwicklung interner Prozesse dient. Daher werden die Trade IDs nicht als Eingabedaten verwendet.

Die Informationen Market Area Buy und Market Area Sell sind prinzipiell interessante Informationen, da sie die Stromflüsse innerhalb eines oder zwischen verschiedenen Marktgebieten beschreiben. Insbesondere ist der Transfer zwischen verschiedenen Marktgebieten durch die Kapazität der Grenzkoppelstellen begrenzt. Sollte eine Grenzkoppelstelle voll ausgelastet sein, so sind zwischen den Marktgebieten keine weiteren Transaktionen möglich [EPE18]. Dies könnte dann auch

entsprechende Auswirkungen auf den Strompreis haben. Da aber auch Stromflüsse des Day-Ahead-Markts und des Terminmarkts über die Grenzkoppelstellen geleitet werden, ist es ohne weitere Informationen nicht möglich, nur anhand der Daten des Intraday-Markts Rückschlüsse auf die verbleibenden Kapazitäten der Grenzkoppelstellen zu schließen. Darüber hinaus haben eigene Untersuchungen der Transaktionslogs ergeben, dass in dem Zeitraum vom 1. Januar 2015 bis zum 31. Dezember 2017 ca. 87,68% aller Intraday-Transaktionen innerhalb des deutschen Marktgebiets stattgefunden haben. Transaktionen zwischen verschiedenen Marktgebieten sind daher eher in der Minderheit. Aus diesen Gründen werden auch die Informationen Market Area Buy und Market Area Sell nicht berücksichtigt. Auf diese Entscheidung wird in Kapitel 5.2 noch genauer eingegangen.

Die Informationen Hour from und Hour to beschreiben zusammengefasst, um welches konkrete Stromprodukt es sich handelt. Wie oben bereits beschrieben, verwendet die EPEX SPOT ein eigenes Format, um Stunden, Halbstunden und Viertelstunden zu bezeichnen. Die 24 Stunden eines Tages werden dabei durch die Zahlen 1 bis 24 beschrieben, wobei „1“ den Zeitraum von 0 Uhr bis 1 Uhr repräsentiert. Soll eine Halbstunde beschrieben werden, so werden der Stundenbezeichnung die Buchstaben „hh“ (half hour) und die Zahl 1 oder 2 angehängt. 1 beschreibt die erste halbe Stunde, 2 folglicherweise die zweite halbe Stunde. „1hh2“ repräsentiert also den Zeitraum von 0:30 Uhr bis 1:00 Uhr. Für die feinere Unterteilung in Viertelstunden werden der Stundenbezeichnung die Buchstaben „qh“ (quarter hour) und die Zahl 1, 2, 3 oder 4 angehängt. „24qh4“ beschreibt so den Zeitraum von 23:45 Uhr bis 0:00 Uhr.
Für Stunden-, Halbstunden- und Viertelstundenprodukte sind die Werte Hour from und Hour to immer gleich. Nur bei Blockprodukten unterscheiden sich die beiden Werte. Eigene Analysen der Transaktionslogs haben gezeigt, dass ca. 47,97% aller Transaktionen dem Handel von Stundenprodukten und ca. 51,83% dem Handel von Viertelstundenprodukten zuzuschreiben sind. Die verbleibenden ca. 0,2% entfallen auf alle Block- und Halbstundenprodukte. Aufgrund dieser sehr dünnen Datengrundlage kann für solche Produkte kein zuverlässiges Prognosemodell entwickelt werden. Daher werden im Folgenden nur noch die Stunden- und Viertelstundenprodukte betrachtet.

Anhand des Time Stamp kann eine chronologische Historie aller Transaktionen abgeleitet werden und tatsächlich sind die Transaktionlogs bereits nach aufsteigendem Zeitstempel sortiert. Da aber zu jedem Zeitpunkt für jedes noch handelbare Stromprodukt ein Gebot abgegeben werden kann, sind die Transaktionen für verschiedene Stromprodukte vermischt (wie auch in Tabelle 3 zu sehen ist). Durch Filtern der Daten anhand von Date, Hour from und Hour to kann die Transaktionshistorie für ein bestimmtes Stromprodukt rekonstruiert werden.

### Transformation der Zeitreihen

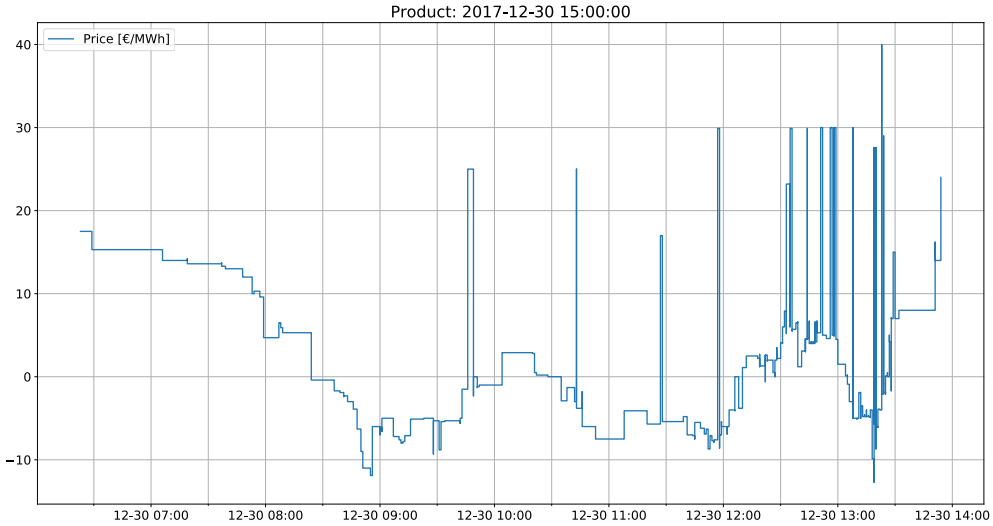
Die Rohdaten in ihrer aktuellen Form stellen ein spezielles Problem in der Analyse von Zeitreihen dar. In Abbildung 17 sind die Zeitreihen für das 15-Uhr-Stundenprodukt des 30. und 31. Dezembers 2017 gegenüber gestellt. Die beiden Grafiken zeigen dabei einige interessante Details:

- Der Preis, den der Käufer des Stroms bezahlen muss, kann negativ werden. Er bekommt also Geld von dem Verkäufer dafür, dass er diesen Strom abnimmt. Das ist in der Regel dann der Fall, wenn das Angebot am gesamten Markt deutlich über der Nachfrage liegt, die Verkäufer aber ihre verbleibenden Kapazitäten noch veräußern müssen. In der Tat war an beiden Tagen die Energieproduktion durch Windkraftanlagen sehr hoch, was diesen Effekt hervorgerufen haben könnte.

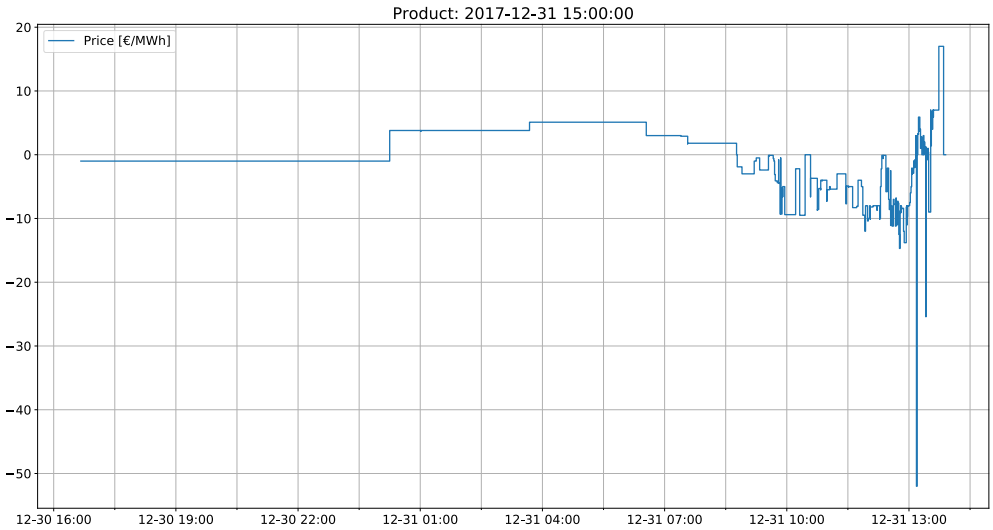
- Die Preisentwicklung für das gleiche Stundenprodukt kann an zwei aufeinander folgenden Tagen völlig unterschiedlich sein. Dies symbolisiert unter Berücksichtigung der zeitlichen Korrelation und ähnlichen Wetterverhältnissen die Volatilität des Marktes, für den trotzdem ein möglichst gutes Prognosemodell entwickelt werden soll.

Der wichtigste Punkt ist aber, dass die Transaktionen der beiden Zeitreihen zu völlig unterschiedlichen Zeitpunkten auftreten. In Abbildung 17a kam die erste Transaktion knappe 9 Stunden vor der Stromlieferung zustande. Im Gegensatz dazu beträgt diese Zeitspanne in Abbildung 17b fast 23 Stunden. Auch die letzten Transaktionen beider Beispiele liegen zeitlich etwa 20 Minuten auseinander. Auch die zeitlichen Abstände zwischen den Transaktionen fallen völlig unterschiedlich aus. In Abbildung 17b beträgt der Abstand zwischen zwei Transaktionen zu Beginn mehrere Stunden, gegen Ende reduziert sich dieser jedoch in den Minutenbereich. Eine solche Zeitreihe mit unregelmäßigen Abständen wird auch als eine nicht-äquidistante Zeitreihe bezeichnet. Dadurch unterscheiden sich die beiden Beispiele auch in der gesamten Anzahl an Transaktionen, die für das entsprechende Stromprodukt zustande gekommen sind.

Analysen auf nicht-äquidistanten Zeitreihen lagen eine lange Zeit nicht im Fokus der Forschungsarbeit, da Analysen von äquidistanten Zeitreihen in Bezug auf den Rechenaufwand einfacher sind [Eck14]. Daher sind viele gängige Verfahren auf die Analyse von äquidistanten Zeitreihen beschränkt und für das hier vorliegende Problem ungeeignet. Um diese Methoden trotzdem einsetzen zu können, müssen die nicht-äquidistanten Zeitreihen in äquidistante Zeitreihen umgewandelt werden. Das geschieht durch Zerlegung der Zeitreihe in Bereiche (auch *bin* genannt), die alle ein gleich großes Zeitintervall abdecken. Alle Transaktionen, die in dem gleichen Bin liegen, werden aggregiert, indem z. B. der Mittelwert des Preises gebildet wird. Die zeitliche Auflösung der Bins kann dabei frei gewählt werden. Durch ei-




(a)




(b)

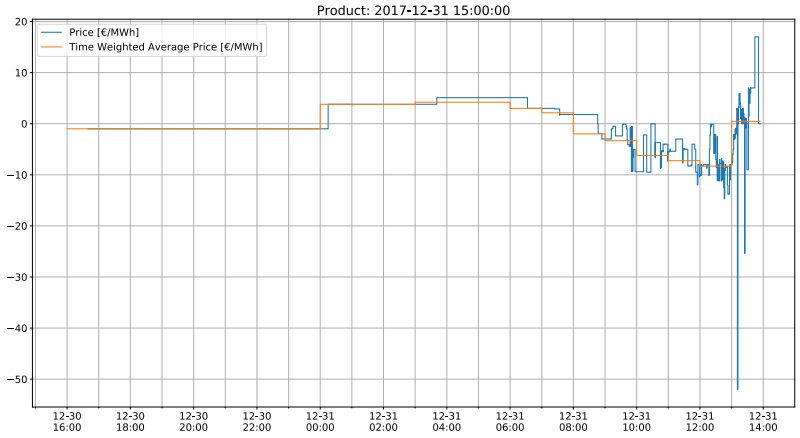
Abbildung 17: Entwicklung des absoluten Preises pro MWh für das 15-Uhr-Stundenprodukt des **(a)** 30. Dezembers 2017 und des **(b)** 31. Dezembers 2017.

ne größere Auflösung gehen viele Details des originalen Kurvenverlaufs verloren, während durch kleinere Auflösungen „Löcher“ entstehen können. Mit dem Begriff „Loch“ ist hier ein Bin gemeint, in dem keine Transaktion stattgefunden hat, also auch keine Aggregation vorgenommen werden kann. Diesen Löchern muss aber ein Wert zugewiesen werden, da die äquidistante Zeitreihe sonst nicht als Eingabe für ein Modell geeignet wäre. Eine Möglichkeit wäre es, den Löchern einen konstanten Wert (bspw. 0) zuzuweisen. Oder aber den Löchern wird der Wert des vorherigen Bins zugewiesen (ggf. rekursiv, falls mehrere Löcher hintereinander). Experimente haben gezeigt, dass die zweite Variante zu besseren Ergebnissen führt. Das lässt sich möglicherweise dadurch begründen, dass durch die Zuweisung eines konstanten Wertes der Eindruck entsteht, dass dort tatsächlich Transaktionen mit entsprechenden Werten aufgetreten sind. Die Ergebnisse dieses Verfahrens sind für jeweils unterschiedliche Auflösungen der Bins in Abbildung 18 dargestellt.

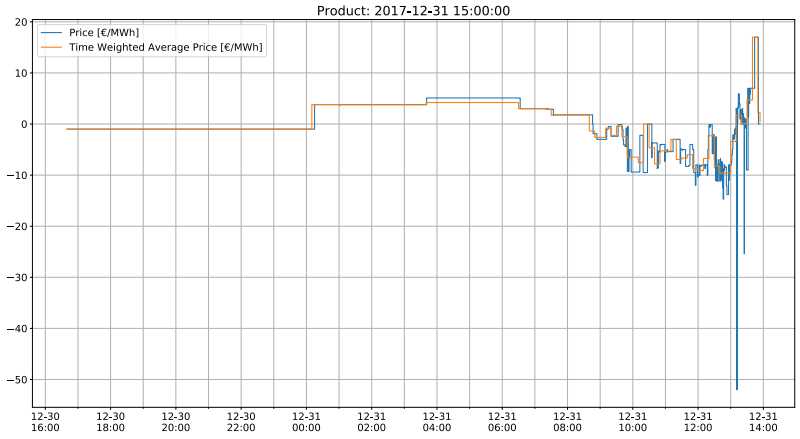
Um möglichst viele Informationen bei der Aggregation zu einem Bin zu erhalten, werden verschiedenste statistische Merkmale berechnet. Diese werden im Folgenden genauer beschrieben.

- *Count*: Die Anzahl der Transaktionen, die in dem entsprechenden Bin aggregiert wurden.
- *First*, *Last*: Der Wert der ersten bzw. letzten Transaktion des entsprechenden Bins.
- *Min*, *Max*: Der minimalste bzw. maximalste Wert aller Transaktionen des entsprechenden Bins.
- *Sum*: Die Summe der Werte aller Transaktionen des entsprechenden Bins.
- *Average*: Der Durchschnitt der Werte aller Transaktionen des entsprechenden Bins.
- *Std*: Die Standardabweichung der Werte aller Transaktionen des entsprechenden Bins.
- *Autocorrelation*: Die Autokorrelation der Werte aller Transaktionen des entsprechenden Bins. Die Autokorrelation beschreibt die Korrelation einer Zeitreihe mit sich selbst, verschoben um einen einstellbaren Zeitschritt.

Bis auf Count werden alle Features auf die Zeitreihen des Volumens und des Preises berechnet. Darüber hinaus wird für jeden Bin der Preiszeitreihe auch noch der volumengewichtete Durchschnittspreis gebildet. Dieser ergibt sich, indem für jede Transaktion zunächst bestimmt wird, wie viel Anteil ihr Volumen an dem Volumen aller Transaktionen des entsprechenden Bins hat. Somit erhält man eine Gewichtung der Transaktionen, die dann anschließend mit ihren Preisen multipliziert werden. Die Ergebnisse werden summiert und stellen dann den volumengewichteten




(a)




(b)

Abbildung 18: Entwicklung des absoluten Preises pro MWh für das 15-Uhr-Stundenprodukt des 30. Dezembers 2017. In **(a)** wurde der Durchschnitt über 1-stündige, in **(b)** über 10-minütige Intervalle gebildet.

Durchschnittspreis dar. Mathematisch lässt sich das ausdrücken durch:

$$vwap(bin_i) = \sum_{t \in T(bin_i)} p(t) \cdot \frac{v(t)}{v(T(bin_i))}$$

$T(bin_i)$ beschreibt eine Menge, die alle Transaktionen des $i$-ten Bins enthält, $p(\cdot)$ ist der Preis einer Transaktion und $v(\cdot)$ ist das Volumen einer bzw. mehrerer Transaktionen. Somit erhält man analog zu der Preismittelwertskurve aus eine weitere Kurve, die im Verlauf ähnlich ist, jedoch einige kleinere Abweichungen aufweist. Sollten in einem Bin jedoch viele Transaktionen mit sehr geringem Volumen und Preis, sowie wenige Transaktionen mit sehr großem Volumen und Preis abgeschlossen worden sein, so würde der volumengewichtete Durchschnittspreis deutlich von dem normalen Durchschnittspreis abweichen. In dieser Hinsicht ist der volumengewichtete Durchschnittspreis aussagekräftiger, da er den Durchschnittspreis für das gehandelte Volumen besser beschreibt.

### 3.1.2 Prognosen der Solar- und Windenergieerzeugung

Aufgrund der verbreiteten Vermarktung von regenerativen Energien an dem Intraday-Markt ist anzunehmen, dass ein kausaler Zusammenhang zwischen der aktuell und zukünftig erbrachten Leistung von Photovoltaik- und Windkraftanlagen und dem Strompreis besteht. In Kapitel 3.1.1 wurde im Bezug auf Abbildung 17 bereits darauf eingegangen, dass sehr viel Windenergie erzeugt wurde, als der Preis ein sehr niedriges Niveau erreicht hat. Aufgrund dieser Vermutung werden Prognosen über die Solar- und Windenergieerzeugung herangezogen, die die EWE Trading GmbH durch die energy & meteo systems GmbH bezieht.

Alle 15 Minuten wird eine solche Prognose erstellt, die für die nächsten 9 Tage die deutschlandweit erbrachte Photovoltaikleistung respektive Windkraftleistung vorhersagt. Die zeitliche Auflösung der Prognosen beträgt dabei ebenfalls 15 Minuten, d. h. die nächsten 9 Tage werden in je 96 Intervalle eingeteilt, denen dann die jeweils vorhergesagte Leistung zugeordnet wird. Seit dem 15. September 2015 wird außerdem eine Vorhersageunsicherheit durch eine untere und obere Schranke angegeben. Da diese Informationen aber nicht für alle Daten zur Verfügung stehen, werden sie nicht mit einbezogen. In Tabelle 4 ist ein Auszug der Windkraftprognose für den 31. Dezember 2017 (analog zu Abbildung 17) dargestellt.

### 3.1.3 Day-Ahead-Preis

In die Preisbildung für ein Stromprodukt am Intraday-Markt fließt auch der Preis ein, der zuvor für ein äquivalentes Produkt am Day-Ahead-Markt per Auktionsverfahren ermittelt wurde (siehe Kapitel 2.1.3). Da Blockprodukte nicht berücksichtigt werden (siehe Kapitel 3.1.1), werden lediglich die Day-Ahead-Preise für die Stundenprodukte berücksichtigt. Diese liegen als eine Liste vor, in der jedem

| timestamp[utc] | power[MW] | power min[MW] | power max[MW] |
|---|---|---|---|
| 2017-12-31 12:00:00 | 36791 | 34616 | 38456 |
| 2017-12-31 12:15:00 | 36838 | 34662 | 38250 |
| 2017-12-31 12:30:00 | 36884 | 34729 | 38062 |
| 2017-12-31 12:45:00 | 36850 | 34804 | 37793 |
| 2017-12-31 13:00:00 | 36819 | 34892 | 37525 |
| 2017-12-31 13:15:00 | 36785 | 34864 | 37443 |
| 2017-12-31 13:30:00 | 36754 | 34840 | 37368 |
| 2017-12-31 13:45:00 | 36689 | 34838 | 37422 |
| 2017-12-31 14:00:00 | 36619 | 34839 | 37480 |

Tabelle 4: Auszug aus der Windleistungsprognose vom 31. Dezember 2017.

Stundenprodukt der durch das Auktionsverfahren ermittelte Preis zugeordnet ist. Am Day-Ahead-Markt werden keine Auktionen für Viertelstundenprodukte durchgeführt, doch auch für diese soll in ein Prognosemodell entwickelt werden. Daher wird als Annäherung einem Viertelstundenprodukt der Day-Ahead-Preis des umfassenden Stundenprodukts zugeordnet.

### 3.1.4 Regelzonensaldo und Ausgleichsenergiepreis

Der Regelzonensaldo und der Ausgleichsenergiepreis können das Verhalten der Marktteilnehmer beeinflussen. Ist der Regelzonensaldo negativ, so liegt ein Energieüberschuss vor. Falls ein Stromabnehmer seinen geplanten Bedarf noch nicht komplett durch Einkauf von Energie gedeckt hat, so könnte er nun entscheiden, die verbleibende Differenz nicht am Strommarkt zu begleichen. Die Folge ist, dass der Stromabnehmer mehr Strom aus dem Netz entnimmt, als er eingekauft hat. Da er dadurch aber das Regelzonensaldo positiv beeinflusst, wird er mit dem Ausgleichsenergiepreis für die entnommene Energie vergütet. Ist der Regelzonensaldo hingegen positiv, so liegt ein Energiedefizit vor. Sollte ein Stromverkäufer noch freie Kapazitäten haben, dann müsste er diese nun nicht mehr verkaufen. Der überschüssige Strom würde dem Energiedefizit entgegenwirken, wofür der Stromverkäufer auch mit dem Ausgleichsenergiepreis vergütet wird. Diese Szenarieren machen für die Akteure dann Sinn, wenn durch den Ausgleichsenergiepreis ein besserer Preis als am Strommarkt erzielt werden kann.

Regelzonensaldo und Ausgleichsenergiepreis werden für jede Viertelstunde bestimmt und veröffentlicht. Für den Regelzonensaldo gibt es eine Fülle an unterschiedlichen Informationen, insbesondere werden bei Monatsende qualitätsgesicherte Daten ergänzt, die ggf. von den ursprünglich genannten Werten abweichen. Da diese aber nicht in Echtzeit zur Verfügung stehen, werden sie nicht als Teil der Eingabedaten verwendet.

## 3.2 Gewinnung der Features und Targets aus den Rohdaten

In Kapitel 3.1 wurde genau erklärt, welche verschiedenen Datenquellen zur Verfügung stehen. Es folgt nun die Beschreibung, wie die verschiedenen Daten transformiert werden, sodass sie als Eingabedaten für die zu entwickelnden Prognosemodelle verwendet werden können. Außerdem müssen aus den Rohdaten auch die Zielwerte extrahiert werden, die das Modell für die verschiedenen Eingabedaten vorhersagen soll.

### 3.2.1 Beschreibung des Prognoseproblems

Zunächst einmal muss beschrieben werden, welchem Zeitraum die Eingabedaten entstammen sollen, und was das eigentliche Prognoseziel ist. Bisher wurde nur darauf eingegangen, welche Datenquellen es gibt und dass ein Durchschnittspreis in der Zukunft prognostiziert werden soll. Es wurde aber noch nicht konkret beschrieben, zu welchem Zeitpunkt eine Prognose durchgeführt, auf welche Daten zurückgeblickt und wie weit die Preisentwicklung in die Zukunft vorhergesagt werden soll.

In Abbildung 19 sind diese Informationen schematisch dargestellt. Die *physical delivery* beschreibt den Zeitpunkt, an dem das Stromprodukt geliefert werden muss. Die *gate closure* entspricht dem Handelsschluss für dieses Stromprodukt. Wie aus Kapitel 2.1.3 hervorgeht, ist der Handel am Intraday-Markt innerhalb einer deutschen Regelzone bis zu 5 Minuten vor Lieferung möglich, zuvor lag dieses Limit bei 30 Minuten. Durch Mitarbeiter der EWE Trading GmbH wurde aber bestätigt, dass diese Möglichkeit durch die Händler seit der Einführung im Juni 2017 noch eher wenig wahrgenommen wird. Es sei zwar ein steigender Trend zu beobachten, dennoch ließe sich keine signifikante Verlagerung von Transaktionen aus dem Zeitbereich $t < physical\_delivery - 30m$ in den Zeitbereich $t < physical\_delivery - 5m$ feststellen. Aus diesen Gründen wird der Handelsschluss auch für den Handel nach dem Juni 2017 als 30 Minuten vor der Lieferung festgesetzt. *Wake-up time* beschreibt den Zeitpunkt, an dem das Modell die Vorhersage starten soll. Hierfür wird ein gewisser Zeitraum, das sogenannte *look back* window, in die Vergangenheit geschaut, und die darin enthaltenen Daten werden für die Gewinnung der Features verwendet. Wie weit in die Zukunft vorhergesagt werden soll, wird durch das *forecast window* festgelegt. Darüber hinaus kann die zeitliche Auflösung bei der Transformation der nicht-äquidistanten Zeitreihe in eine äquidistante Zeitreihe für beide Intervalle unterschiedlich gewählt werden (*look back resolution* und *forecast resolution*). Somit erhält man 5 frei konfigurierbare Parameter, die möglichst so eingestellt werden sollen, dass die Qualität der Prognose möglichst gut wird. Im Folgenden der Ausarbeitung wird die *wake-up time* aber konstant auf eine Stunde vor dem Handelsschluss, die Länge des *forecast windows* ebenfalls auf eine Stunde und die *forecast resolution* auf 20 Minuten festgesetzt.

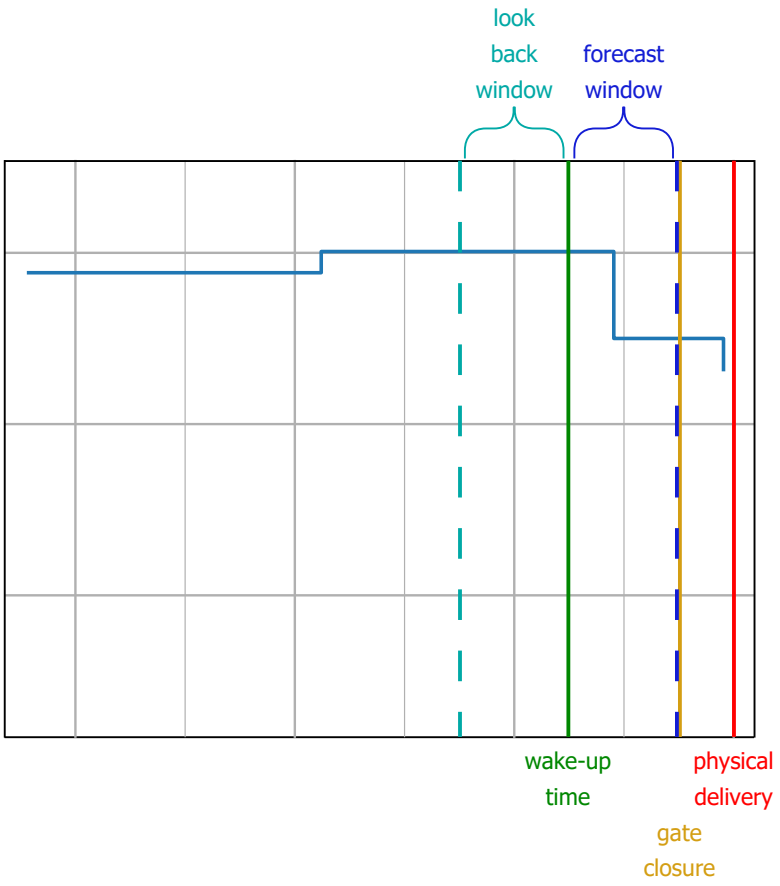



Abbildung 19: Schematische Darstellung der verschiedenen Zeitpunkte und Intervalle, die für die Bestimmung der Features und Targets berücksichtigt werden müssen.

Dies ermöglicht eine Vergleichbarkeit der Ergebnisse mit dem Modell, das von der BTC AG erstellt wurde (siehe Kapitel 4). Somit verbleiben als frei einstellbare Parameter die Größe und die zeitliche Auflösung des *look back windows*.

### 3.2.2 Gewinnung der Targets

Wie in Kapitel 3.2.1 beschrieben, wird die Länge des *forecast windows* auf eine Stunde und die *forecast resolution* auf 20 Minuten festgelegt. In Kapitel 3.1.1 wurde bereits erklärt, wie eine nicht-äquidistanten Zeitreihe in eine äquidistante Zeitreihe transformiert werden kann. Zielwert der Prognose soll der volumengewichtete Durchschnittspreis sein, da dieser eine bessere Aussagekraft als der einfache Durchschnittspreis hat (siehe Kapitel 3.1.1). Das Transformationsverfahren wird nun für jedes Stromprodukt aller Tage in den vorliegenden Transaktionslogs auf dem Intervall $[\text{wake-up time}, \text{wake-up time} + 60m)$ durchgeführt, wobei das Intervall in drei Bins zerlegt werden soll. Im Folgenden werden diese drei Bins auch als *Early*, *Mid*, und *Late* bezeichnet. Das Prognosemodell soll daher auf Basis der Eingabedaten

drei unterschiedliche Werte vorhersagen, die die Preisentwicklung der kommenden Stunde beschreiben.

### 3.2.3 Gewinnung der Features

#### Transaktionslogs

Aus den Transaktionslogs können viele verschiedene Features gewonnen werden. In Kapitel 3.1.1 wurde bereits detailliert erläutert, welche Features sich aus den Zeitreihen des Preises und des Volumens bestimmen lassen. Die Größe des *look back windows*, sowie dessen zeitliche Auflösung, kann frei eingestellt werden, wobei die minimalste Auflösung einer Minute entspricht. Diese Grenze entsteht dadurch, dass der Zeitstempel in den Rohdaten lediglich minutengenau ist.

Außerdem können noch weitere Features aus den anderen Daten abgeleitet werden, die das Stromprodukt, für das eine Prognose getroffen werden soll, etwas genauer beschreiben. Die folgende Auflistung gibt einen genauen Überblick.

- *DEL_MONTH*: Eine Ganzzahl, die den Monat beschreibt, in dem das Stromprodukt geliefert wird.
- *DEL_INSTANT*: Ein Dezimalzahl, die die Stunde bzw. Viertelstunde beschreibt, zu der das Stromprodukt geliefert werden soll. Stundenprodukte werden als Ganzzahl kodiert, während die Viertelstundenprodukte durch einen Dezimalanteil davon abgegrenzt werden. So wird numerisch die Beziehung zwischen einem Stundenprodukt und den dazugehörigen Viertelstundenprodukten beibehalten.
- *DEL_SEASON_CATEGORICAL*: Eine Ganzzahl, die die meteorologische Jahreszeit des Tages beschreibt.
- *DEL_SEASON_SINE_WAVE*: Eine Dezimalzahl, die die meteorologische Jahreszeit des Tages anhand einer Sinuskurve beschreibt. Diese Sinuskurve hat eine Periodizität von 365 Tagen und wird so verschoben, dass ihr Höhepunkt dem meteorologischen Höhepunkt des Sommers (16. Juli) entspricht.

Diese Informationen könnten für das Prognosemodell hilfreich sein, da sie die Rahmenbedingungen für ein Stromprodukt beschreiben. Beispielsweise ist es denkbar, dass im Winter andere Strompreise als im Sommer zustande kommen.

#### Solar- und Windprognosen

Im Gegensatz zu den anderen Daten, die Zustände aus der Gegenwart bzw. Vergangenheit beschreiben, blicken die Solar- und Windprognosen in die Zukunft. Von Interesse sind hierbei die Energieprognosen für das *forecast window*, dessen Preisentwicklung prognostiziert werden soll. Darüber hinaus erscheint es sinnvoll,

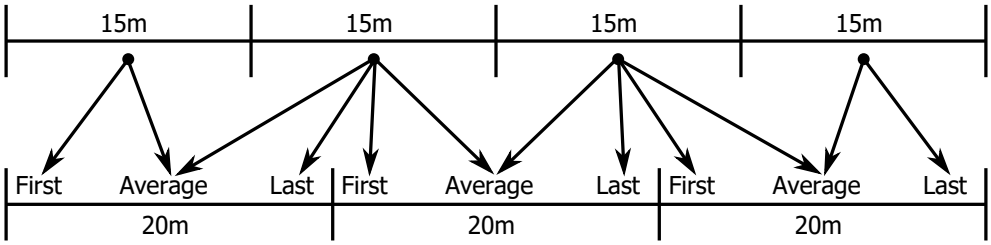



Abbildung 20: Schematische Darstellung der Transformation von Energieprognosen mit 15-minütiger Auflösung zu einer 20-minütigen Auflösung.

eine Kopplung der Energieprognosen mit den Bins *Early*, *Mid* und *Late* zu erzeugen. Dies wird umgesetzt, indem die zeitliche Auflösung der Energieprognosen auf die *forecast resolution* angepasst wird. Da die Energieprognosen mit 15-minütiger Genauigkeit aufgelöst sind und die *forecast resolution* 20 Minuten beträgt, wäre jedoch ein Informationsverlust zu erwarten. Dies kann aber vermieden werden, indem (analog zu den Transaktionslogs) die Features *First*, *Last* und *Average* berechnet werden. Jeder Prognosewert aus einem der 15-Minuten Intervalle findet sich exakt in einem *First* oder *Last* Attribut wieder (siehe Abbildung 20). Darüber hinaus bietet der *Average* auch noch den Mehrwert, dass die zu einem Bin zugehörigen Daten aus zwei Eingabeintervallen aggregiert werden.

#### Day-Ahead-Preis

Für ein Stromprodukt kann der korrespondierende Day-Ahead-Preis ganz einfach aus der Preisliste ausgelesen und als Feature eingebunden werden.

#### Regelzonensaldo und Ausgleichsenergiepreis

Auch der Regelzonensaldo und der Ausgleichsenergiepreis beschreiben ihre Entwicklung durch eine Zeitreihe. Im Gegensatz zu den Transaktionslogs sind diese Zeitreihen aber bereits äquidistant, mit zeitlichen Abständen von 15 Minuten. Für eine einfachere Integration sollen diese Auflösung ebenfalls an die *look back resolution* angepasst werden. Daher erfolgt eine Berechnung von Features auf Basis der gleichen Bins, die auch bei den Transaktionslogs genutzt werden. Für jeden Bin werden die Features *First*, *Last* und *Average* (analog zu den Transaktionslogs) für den Regelzonensaldo respektive den Ausgleichsenergiepreis berechnet.

## 3.3 Formatierung der Eingabedaten

Jedes Machine Learning Modell erwartet die Eingabedaten in einer speziellen Form. Daher wird im Folgenden genau beschrieben, wie die Eingabedaten für die verschiedenen Verfahren angepasst werden müssen. Darüber hinaus ist nicht jedes Verfahren dazu in der Lage, mehrere Targets vorherzusagen. Daher wird eine Methode vorgestellt, durch die dies möglich wird.

### 3.3.1 LR, kNN, RF, SVR

Lineare Regression, k-Nearest Neighbors, Random Forest und Support Vector Regression; alle vier Verfahren teilen die gleichen Anforderungen an das Format der Eingabedaten und sind nur dazu in der Lage, ein einzelnes Target auszugeben. Dabei ist das Eingabeformat denkbar einfach: Ein Pattern wird einfach als eine Liste von Features erwartet. Werden demnach alle Features, wie in Kapitel 3.2.3 beschrieben, extrahiert und in einer Liste gesammelt, so kann diese als Eingabe für die Modelle fungieren. Um auch die drei Targets *Early*, *Mid* und *Late* vorhersagen zu können, werden anstatt eines einzelnen Modells einfach drei Modelle trainiert, die sich jeweils auf eines der Targets spezialisieren.

### 3.3.2 Feedforward Netz

Auch ein einfaches feedforward Netz erwartet die Eingabedaten als eine Liste von Features. Der Unterschied zu den in Kapitel 3.3.1 beschriebenen Modellen ist jedoch, dass ein feedforward Netz mehrere Ausgaben erzeugen kann. Dies erreicht man dadurch, dass die letzte Schicht des Netzes aus mehreren Berechnungsknoten besteht. Jeder dieser Knoten repräsentiert eine Ausgabe des neuronalen Netzes, sodass für die Targets *Early*, *Mid* und *Late* drei Berechnungsknoten im Output Layer notwendig sind.

### 3.3.3 LSTM und CNN

Wie auch ein einfaches feedforward Netz können Long short-term Memory-Netze und Convolutional Neural Networks multiple Ausgaben erzeugen. Sie unterscheiden sich aber von den bisherigen Modellen in dem Format der Eingabedaten, denn in beiden Fällen werden Zeitreihen erwartet. Die Erzeugung dieser Zeitreihen ist in Kapitel 3.1 und Kapitel 3.2 ausreichend erläutert worden. In dieser Form können sie einfach in die Netze eingegeben werden. Es gibt aber auch Features, die sich nicht auf Zeitreihen beziehen, wie bspw. der Day-Ahead-Preis für das Stromprodukt. Wie mit diesen Informationen umgegangen wird, ist in Kapitel 3.4 beschrieben.

## 3.4 Topologie der neuronalen Netze

Die Eingabedaten für die Modelle lassen sich in die Kategorien *Metadaten* und *Zeitreihen* einordnen. Die Metadaten beschreiben Informationen, die keiner Zeitreihe zuzuordnen sind. Stattdessen beschreiben sie als einzelner Wert einen Sachverhalt für das gesamte Stromprodukt. Dagegen stehen die Features der Zeitreihen, deren Ausprägungen zu unterschiedlichen Zeitpunkten einen zeitlichen Verlauf darstellen. Dabei werden die Zeitreihen noch in *look back*-Zeitreihen und *forecast*-Zeitreihen unterschieden, da diese eine unterschiedliche zeitliche Auflösung haben können.

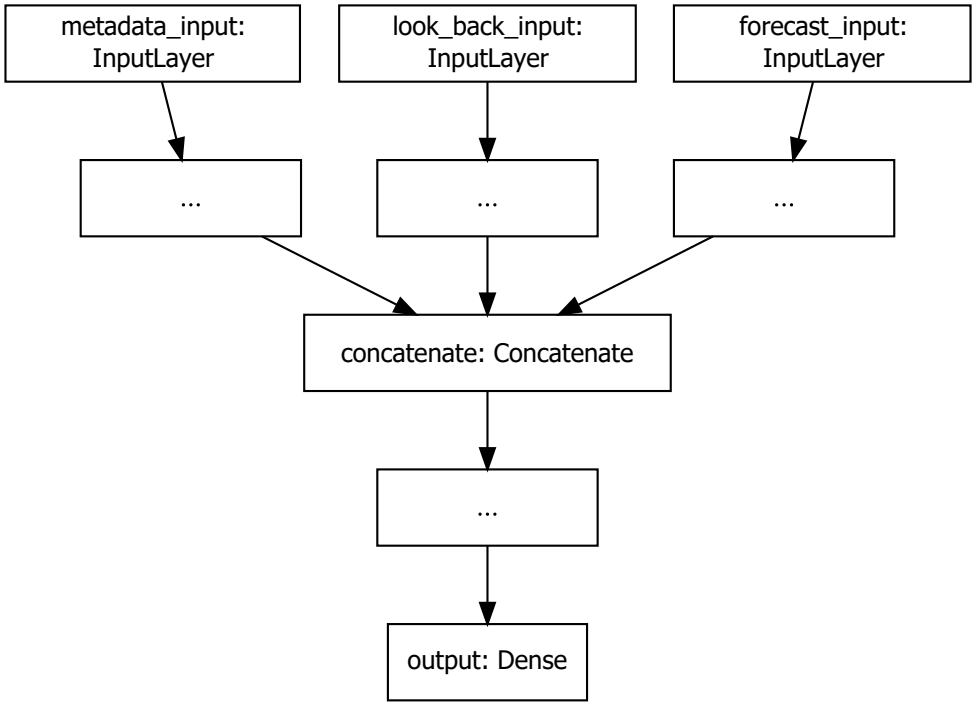



Abbildung 21: Allgemeiner Aufbau aller neuronalen Netze, die in dieser Arbeit entwickelt werden.

Um diese verschiedenen Eingabedaten in einem Modell zu vereinen, wurde eine spezielle Netztopologie entwickelt, die in Abbildung 21 dargestellt ist. Dabei werden die Metadaten, *look back*-Zeitreihen und *forecast*-Zeitreihen zunächst unabhängig voneinander verarbeitet. Somit sind für die Verarbeitung der Zeitreihen sowohl fully-connected Schichten, als auch LSTM- oder Convolution-Schichten einsetzbar. Im Anschluss werden die verschiedenen Zweige zusammengeführt, sodass dann weitere Berechnungsschichten auf den kombinierten Zwischenergebnissen angewendet werden können. Die Ausgabe des Netzes wird durch einen einfachen fully-connected Layer erzeugt.

Auf Basis dieser Architektur wurden verschiedenste Netztopologien entwickelt, die in Kapitel A.2 aufgelistet sind.

## 3.5 Parameteroptimierung

Die verschiedenen Modelle zeichnen sich alle dadurch aus, dass sie eine Menge an freien Parametern haben, die die Güte der Prognoseergebnisse beeinflussen. Daher müssen die Parameterräume der Modelle durchsucht werden, um die Parametrisierung mit der größten Güte zu finden. Dieses Problem ist nicht trivial lösbar, denn wenn bereits ein einziger Parameter durch einen kontinuierlichen Wert dargestellt wird, entstehen unendlich viele mögliche Parametrisierungen. Es ist daher ein gängiges Verfahren, dass jedem Parameter eine Auswahl an möglichen Werten zugewiesen wird. Dadurch wird das Problem diskretisiert und es muss nur eine

endliche Anzahl an Parameterkombinationen getestet werden. Diese Methode wird als *Grid Search* bezeichnet. Die so erhaltene beste Parametrisierung muss aber nicht die bestmögliche Parametrisierung für das zu lösende Problem sein, da durch die Diskretisierung nur ein Bruchteil des eigentlichen Parameterraumes durchsucht wird. Es ist aber dennoch ein vertretbarer Kompromiss aus der Qualität der Lösung und dem benötigten Rechenaufwand. Bei der Parameteroptimierung werden zum Training die Daten des gesamten Jahres 2015 verwendet, validiert werden die Modelle auf den Daten des Jahres 2016.

Die Parameteroptimierung wird in zwei Phasen durchgeführt. Zur klareren Differenzierung wird die erste Phase als Modellparameteroptimierung und die zweite Phase als Datensatzoptimierung bezeichnet. In der Modellparameteroptimierung werden zunächst die optimalen Parameter für die einzelnen Modelle bestimmt. Dies geschieht anhand zweier Datensätze, deren Eingabedaten grob bzw. fein aufgelöst sind. So wird Rechenaufwand gespart und dennoch kann bereits die Tendenz erkannt werden, ob ein Modell besser auf grob bzw. fein aufgelösten Eingabedaten performt. In der Datensatzoptimierung werden die zuvor ermittelten Modellparameter beibehalten, dann aber Datensätze mit unterschiedlichster Auflösung getestet. So kann evaluiert werden, welche Auflösung am besten für ein Modell geeignet ist.

Im Folgenden wird zunächst auf alle freien Parameter eingegangen, um dann konkrete Ausprägungen zu definieren. Die Anzahl zu testender Parameterkombinationen kann dadurch genau angegeben werden.

### 3.5.1 Parameter für die Verarbeitung der Rohdaten

Wie in Kapitel 3.2.1 erläutert wurde, kann die Größe des *look back windows* und die dazugehörige *look back resolution* frei gewählt werden. Um aber die Anzahl an Parameterkombinationen zu reduzieren, wird für die Größe des *look back windows* ein fester Wert von 60 Minuten festgelegt.

In der Modellparameteroptimierung wird die *look back resolution* auf die Werte 20 Minuten und 5 Minuten beschränkt. Mit diesen zwei Datensätzen soll für die verschiedenen Modelle getestet werden, ob sie eher auf einer groben oder einer feinen Auflösung gut performen. Außerdem soll die dazugehörige beste Modellparametrisierung bestimmt werden. In der Datensatzoptimierung wird die *look back resolution* auf die Werte 20, 15, 10, 5, 2 und 1 Minute(n) erweitert. Mit den bereits bestimmten Parametrisierungen aus dem ersten Schritt kann dann die Auflösung bestimmt werden, die am allerbesten funktioniert.

In der Modellparameteroptimierung muss die Parameteroptimierung also nur auf zwei Datensätzen durchgeführt werden, in der Datensatzoptimierung sind es 6 Datensätze. Im ersten Schritt hängt die Anzahl der zu evaluierenden Parameterkombinationen von dem jeweiligen Modell ab. Im zweiten Schritt sind die Modell-

parameter aber feste Werte, sodass jedes Modell einmal auf allen 6 Datensätzen evaluiert werden muss.

### 3.5.2 Modellparameter

#### Lineare Regression

Vom Prinzip her hat eine Lineare Regression keine freien Parameter. Die Parameter $w$ und $b$ werden durch das Verfahren selbst bestimmt. Jedoch kann der Anwender entscheiden, ob die Verschiebungskonstante $b$ tatsächlich berechnet werden soll, oder aber gleich null gelassen wird. In diesem Fall verläuft die Lineare Regression durch den Ursprung des Koordinatensystems.

Daher ergeben sich für die Lineare Regression lediglich zwei unterschiedliche Parametrisierungen. In der Modellparameteroptimierung werden daher nur insgesamt $2 \cdot 2 = 4$ Parameterkombinationen getestet.

#### k-Nearest Neighbor Regression

Bei der k-Nearest Neighbor Regression werden zwei Parameter optimiert. Das ist einerseits der Wert $k \in \mathbb{N}$, der die Anzahl der nächsten Nachbarn angibt, die für ein Eingabemuster untersucht werden sollen. Der zweite Parameter ist die Gewichtungsfunktion, die den Einfluss der einzelnen Nachbarn steuern kann. Entweder haben alle Nachbarn den gleichen Einfluss (*uniform*), oder es findet eine Gewichtung in Abhängigkeit von der Distanz statt (*distance*).

Es werden folgende Modellparameter getestet:

- $k$: 1, 3, 5, 10, 20, 30, 50, 75 und 100, also insgesamt 9 verschiedene Ausprägungen.
- Gewichtsfunktion: uniform und distance, also insgesamt 2 verschiedene Ausprägungen.

In der Modellparameteroptimierung werden daher insgesamt $2 \cdot 9 \cdot 2 = 36$ Parameterkombinationen getestet.

#### Random Forest Regression

Bei einem Random Forest kann der Aufbau der Entscheidungsbäume durch viele Parameter beeinflusst werden. Um diese Menge zu reduzieren, wurde die Optimierung auf die wichtigsten Parameter *max depth* und *minimum samples per leaf* beschränkt. *Max depth* gibt eine obere Schranke für die Tiefe aller Entscheidungsbäume des Random Forests an. *Minimum samples per leaf* entscheidet, wie viele Datenpunkte der Trainingsdaten in einem Blattknoten minimal vorhanden sein müssen. Durch beide Parameter kann eine zu starke Spezialisierung auf die Traingsdaten vermieden werden.

Es werden folgende Modellparameter getestet:

- *max depth*: 3, 5, 10, 20 und $\infty$, also insgesamt 5 verschiedene Ausprägungen.
- *minimum samples per leaf*: 1, 3, 5, 7 und 10, also ebenfalls 5 verschiedene Ausprägungen.

In der Modellparameteroptimierung werden daher insgesamt $2 \cdot 5 \cdot 5 = 50$ Parameterkombinationen getestet.

### Support Vector Regression

Die Support Vector Regression bietet die Möglichkeit, die Parameter $C$, $\epsilon$ und den verwendeten *Kern* einzustellen. Der Parameter $\epsilon$ steuert die Breite des Bereichs um die Hyperebene herum, in der die erwarteten Ergebnisse liegen sollen. Durch $C \in (0, 1]$ können Verletzungen gegen diese Bedingung zugelassen werden. Die Verwendung eines Kerns macht die Durchführung des Verfahrens in einem hochdimensionalen Featureraum möglich. Hierfür stehen die Kernel *linear*, *rbf* und *sigmoid* zur Verfügung.

Es werden folgende Modellparameter getestet:

- $C$: 1.0, 0.8, 0.6, 0.5 und 0.3, also insgesamt 5 verschiedene Ausprägungen.
- $\epsilon$: 0.01, 0.05, 0.1, 0.2 und 0.3, also ebenfalls 5 verschiedene Ausprägungen.
- Kern: linear, rbf und sigmoid, also insgesamt 3 verschiedene Ausprägungen.

In der Modellparameteroptimierung werden daher insgesamt $2 \cdot 5 \cdot 5 \cdot 3 = 150$ Parameterkombinationen getestet.

### Neuronale Netze

Neuronale Netze besitzen eine Vielzahl an freien Parametern. Der erste Parameter ist die Topologie des Netzes. Da aber unendlich viele verschiedene neuronale Netze möglich sind, wird diese Auswahl eingeschränkt (siehe Kapitel A.2). Die Anzahl an Knoten in jeder Schicht, sowie die verwendeten Aktivierungsfunktionen können ebenfalls eingestellt werden. Bei dem Training des Netzwerks können unterschiedliche Optimierungsalgorithmen eingesetzt werden, bei denen die *learning rate* eingestellt werden kann. Auch die *batch size* muss angegeben werden, also die Anzahl an Eingabedaten, die dem Netzwerk gezeigt werden, bevor die Gewichte angepasst werden. Dann können noch Regularisierungsparameter eingestellt werden, die verhindern sollen, dass sich das Modell zu sehr an die Trainingsdaten anpasst und somit besser generalisiert. Hierfür wird eine Kombination aus *Dropout* und einer *max-norm Regularisierung* verwendet. Der Dropout beschreibt eine prozentuale Wahrscheinlichkeit für einen Knoten, ob dieser im aktuellen Trainingsschritt verwendet oder ignoriert wird. Durch die max-norm Regularisierung kann ein

Grenzwert für die Kantengewichte im neuronalen Netz gesetzt werden, der nicht überschritten werden kann.

Es werden folgende Modellparameter getestet:

- Netztopologie: Es wurden insgesamt 17 verschiedene Netztopologien entworfen (siehe Kapitel A.2).
- Anzahl der Neuronen in jedem Layer: 16, 32, 64, 128 und 256, also insgesamt 5 verschiedene Ausprägungen.
- Aktivierungsfunktion: relu, elu, softplus und linear, also insgesamt 4 verschiedene Ausprägungen. tanh wurde im Vorhinein ausgeschlossen, da Experimente bereits gezeigt haben, dass tanh sehr schlecht für das zu lösende Problem funktioniert.
- Optimierungsalgorithmus: Adam, Stochastic Gradient Descent und RMSprop, also insgesamt 3 verschiedene Ausprägungen.
- Learning Rate: 0.001, 0.01 und 0.1, also insgesamt 3 verschiedene Ausprägungen.
- Batch Size: 16, 32, 64 und 128, also insgesamt 4 verschiedene Ausprägungen.
- Dropout: 0.0, 0.2, 0.5 und 0.7, also insgesamt 4 verschiedene Ausprägungen.
- Max-norm Regularisierung: None, 2, 3, 4, also insgesamt 4 verschiedene Ausprägungen. None bedeutet, dass es keine Beschränkung der Gewichte gibt.

In der Modellparameteroptimierung müssten daher insgesamt $2{\cdot}17{\cdot}5{\cdot}4{\cdot}3{\cdot}3{\cdot}4{\cdot}4{\cdot}4 = 391.680$ Parameterkombinationen getestet werden. Das ist in Hinsicht auf die benötigte Rechenzeit aber nicht praktikabel. Aus diesem Grund wird dieser Schritt der Parameteroptimierung in drei Teilschritte zerlegt. Als Erstes wird für jede Netztopologie die Anzahl der Neuronen und die Aktivierungsfunktion optimiert. Danach wird auch wieder für jede Netztopologie der Optimierungsalgorithmus, sowie die Learning Rate und die Batch Size optimiert. Und als Letzes wird dann der Dropout und die max-norm Regularisierung optimiert. Dadurch zerlegt sich das Problem in $2 \cdot 17 \cdot (5 \cdot 4 + 3 \cdot 3 \cdot 4 + 4 \cdot 4) = 2.448$ zu testende Parameterkombinationen. Wie auch schon bei der Grid Search an sich, garantiert dieses Verfahren aber nicht, dass die beste Parameterkombination gefunden wird.

# 4 Evaluation

Nachdem nun detailliert beschrieben wurde, wie die Rohdaten aufbereitet werden und die Modelle konzipiert sind, soll nun eine Evaluation für die Ergebnisse der verschiedenen Prognosemodelle vorgenommen werden. Zunächst wird hierbei auf die Parameteroptimierung eingegagen, durch die möglichst gute Parametrisierungen für die verschiedenen Modelle gefunden werden sollen. Im Anschluss werden diese Parametrisierungen verwendet, um die Genauigkeit der Modelle auf unterschiedlichen Datensätzen zu untersuchen. Um die Güte der Modelle vergleichen zu können, wird ihre Genauigkeit im Bezug auf die Validierungsmenge betrachtet. Für jedes Validierungsbeispiel wird mit dem Modell eine Vorhersage der Targets *Early*, *Mid* und *Late* erstellt. Diese Vorhersagen werden mit den erwarteten Werten verglichen, indem für jedes Target der Root Mean Square Error (RMSE, dt. Wurzel des mittleren quadratischen Fehlers) berechnet wird. Die Güte eines Modells wird nun durch den Durchschnitt dieser drei Fehlerwerte ausgedrückt, d. h. je kleiner der gemittelte Fehler, desto größer ist die Güte.

Falls nicht explizit benannt, beziehen sich alle Analysen immer auf Datensätze, in denen nur die Stundenprodukte betrachtet werden.

## 4.1 Parameteroptimierung

Im Folgenden werden die Modellparameteroptimierung und die Datensatzoptimierung für die verschiedenen Verfahren dargestellt. Da je nach Modell die Anzahl an Parameterkombination sehr groß werden kann, werden immer nur die zehn besten Parameterkombinationen angegeben. Die Tabellen sind dabei absteigend nach der Ergebnisgüte sortiert. Eine größere Anzahl aller getesteten Parameterkombinationen kann in Kapitel A.3 eingesehen werden, jedoch sind auch diese auf die jeweils 50 Besten beschränkt, da die Liste der 2.448 für die neuronalen Netze zu testenden Parameterkombination viel zu lang wäre.

### 4.1.1 Lineare Regression

#### Modellparameteroptimierung

Wie schon in Kapitel 3.5.2 beschrieben wurde, kann bei einer Linearen Regression einzig darüber entschieden werden, ob eine Verschiebungskonstante $b$ berechnet werden soll oder nicht ($b = 0$). Die Ergebnisse sind in Tabelle 5 dargestellt. Es zeigt sich, dass dieser Parameter für die Auflösung von 20 Minuten unerheblich ist. Jedoch führt die Verwendung dieser Verschiebungskonstante bei dem 5-minütigen Datensatz zu einer enormen Verschlechterung der Validierungsergebnisse. Aus diesem Grund wird die Verschiebungskonstante im Folgenden abgeschaltet.

| look back resolution | $b = 0$ | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|
| | | Early | Mid | Late | Early | Mid | Late |
| 20min | Ja | 2.347 | 4.384 | 7.172 | 1.89 | 3.175 | 5.881 |
| 20min | Nein | 2.347 | 4.384 | 7.172 | 1.89 | 3.175 | 5.881 |
| 5min | Nein | 2.293 | 4.276 | 7.007 | 2.319 | 3.455 | 6.159 |
| 5min | Ja | 2.297 | 4.764 | 7.418 | 1.022e+12 | 1.447e+13 | 1.678e+14 |

Tabelle 5: Ergebnisse der Modellparameteroptimierung für die Lineare Regression.

#### Datensatzoptimierung

Tabelle 6 zeigt die Ergebnisse der Linearen Regression auf den verschieden aufgelösten Datensätzen. Durch die Verfeinerung der Auflösung auf 15 Minuten wird im Mittel ein noch etwas besseres Ergebnis erzielt, als es zuvor schon für die 20-minütige Auflösung der Fall war. Durch eine noch feinere Auflösung werden die Ergebnisse wieder deutlich schlechter, wobei ab einer Auflösung von unter 2 Minuten extreme Abweichungen entstehen.

| look back resolution | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| 15min | 2.333 | 4.369 | 7.147 | 1.901 | 3.235 | 5.805 |
| 20min | 2.347 | 4.384 | 7.172 | 1.89 | 3.175 | 5.881 |
| 10min | 2.322 | 4.35 | 7.114 | 1.936 | 3.672 | 5.742 |
| 5min | 2.293 | 4.276 | 7.007 | 2.319 | 3.455 | 6.159 |
| 2min | 2.246 | 4.119 | 6.792 | 6.78e+7 | 8.79e+7 | 1.15e+8 |
| 1min | 2.14 | 3.904 | 6.438 | 1.51e+7 | 2.33e+8 | 4.99e+8 |

Tabelle 6: Ergebnisse der Datensatzoptimierung für die Lineare Regression.

### 4.1.2 k-Nearest Neighbor Regression

#### Modellparameteroptimierung

In Tabelle 7 sind die besten Ergebnisse zu sehen, die mit den verschiedenen Parametrisierungen einer k-Nearest Neighbor Regression erzielt werden konnten. Es zeigt sich, dass das Verfahren in erster Linie eine feinere Auflösung favorisiert. Die Anzahl der zu betrachtenden Nachbarn pendelt sich bei um die 20 Stück ein. Dabei liefert die distanzabhängige Gewichtung immer leicht bessere Ergebnisse, als wenn keine solche Gewichtung vorgenommen wird. Soll eine Vorhersage für einen durch das Training bereits bekannten Datenpunkt gemacht werden, wird eben dieser als nächster Nachbar mit einer Distanz von null ausgewählt. Folglich wird dieser Nachbar bei der distanzabhängigen Gewichtung zu 100% gewichtet, während die Anderen nicht mit in die Berechnung einfließen. Aus diesem Grund ist bei dieser Variante der RMSE auf den Trainingsdaten immer gleich null.

| look back resolution | $k$ | weight | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|
| | | | Early | Mid | Late | Early | Mid | Late |
| 5min | 20 | distance | 0.0 | 0.0 | 0.0 | 5.419 | 5.984 | 7.633 |
| 5min | 20 | uniform | 5.575 | 6.787 | 8.786 | 5.443 | 6.005 | 7.648 |
| 5min | 10 | distance | 0.0 | 0.0 | 0.0 | 5.375 | 5.987 | 7.738 |
| 5min | 10 | uniform | 5.161 | 6.341 | 8.298 | 5.393 | 6.003 | 7.75 |
| 5min | 30 | distance | 0.0 | 0.0 | 0.0 | 5.55 | 6.09 | 7.667 |
| 5min | 30 | uniform | 5.874 | 7.087 | 9.056 | 5.581 | 6.117 | 7.687 |
| 5min | 5 | distance | 0.0 | 0.0 | 0.0 | 5.524 | 6.196 | 8.123 |
| 5min | 5 | uniform | 4.704 | 5.782 | 7.684 | 5.538 | 6.208 | 8.133 |
| 5min | 50 | distance | 0.0 | 0.0 | 0.0 | 5.834 | 6.336 | 7.88 |
| 5min | 50 | uniform | 6.256 | 7.459 | 9.404 | 5.878 | 6.375 | 7.912 |

Tabelle 7: Ergebnisse der Modellparameteroptimierung für die k-Nearest Neighbor Regression.

#### Datensatzoptimierung

Tabelle 8 zeigt die Ergebnisse der k-Nearest Neighbor Regression auf den verschieden aufgelösten Datensätzen. Es ist zu beobachten, dass die Regression immer genauer wird, je feiner die Daten aufgelöst sind. Daher werden bei folgenden Prognosen mit Hilfe einer k-Nearest Neighbor Regression die Datensätze immer mit der minimal möglichen Auflösung von einer Minute verwendet.

| look back resolution | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| 1min | 0.0 | 0.0 | 0.0 | 4.91 | 5.525 | 7.312 |
| 2min | 0.0 | 0.0 | 0.0 | 5.067 | 5.629 | 7.37 |
| 5min | 0.0 | 0.0 | 0.0 | 5.419 | 5.984 | 7.633 |
| 10min | 0.0 | 0.0 | 0.0 | 6.252 | 6.757 | 8.276 |
| 15min | 0.0 | 0.0 | 0.0 | 6.276 | 6.772 | 8.327 |
| 20min | 0.0 | 0.0 | 0.0 | 7.165 | 7.636 | 9.019 |

Tabelle 8: Ergebnisse der Datensatzoptimierung für die k-Nearest Neighbor Regression.

### 4.1.3 Random Forest Regression

#### Modellparameteroptimierung

Wie in Tabelle 9 zu sehen ist, ist es für die Güte der Modelle tatsächlich von Vorteil, wenn die Entscheidungsbäume bei ihrem Wachstum eingeschränkt werden. Ein guter Wert scheint bei einer maximalen Tiefe von 5 zu liegen, während durch eine Begrenzung von minimal 3 Datenpunkten pro Blatt eine möglichst gute Generalisierbarkeit erreicht wird.

| look back resolution | max depth | min. samples per leaf | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|
| | | | Early | Mid | Late | Early | Mid | Late |
| 5min | 5 | 3 | 2.394 | 4.273 | 6.834 | 2.237 | 3.363 | 5.836 |
| 5min | 5 | 7 | 2.465 | 4.327 | 6.926 | 2.233 | 3.351 | 5.856 |
| 5min | 5 | 5 | 2.428 | 4.285 | 6.864 | 2.236 | 3.345 | 5.868 |
| 20min | 5 | 3 | 2.4 | 4.304 | 6.846 | 2.171 | 3.466 | 5.823 |
| 5min | 10 | 7 | 1.885 | 3.355 | 5.524 | 2.164 | 3.358 | 5.95 |
| 5min | 10 | 3 | 1.667 | 3.099 | 5.111 | 2.156 | 3.387 | 5.948 |
| 5min | 5 | 10 | 2.61 | 4.465 | 7.037 | 2.284 | 3.316 | 5.904 |
| 20min | 5 | 1 | 2.391 | 4.268 | 6.815 | 2.171 | 3.496 | 5.837 |
| 5min | 10 | 5 | 1.777 | 3.205 | 5.294 | 2.18 | 3.356 | 5.973 |
| 20min | 10 | 7 | 1.921 | 3.447 | 5.639 | 2.113 | 3.453 | 5.951 |

Tabelle 9: Ergebnisse der Modellparameteroptimierung für die Random Forest Regression.

#### Datensatzoptimierung

Tabelle 10 zeigt die Ergebnisse der Random Forest Regression auf den verschieden aufgelösten Datensätzen. Interessanterweise erzielt das Modell weder auf zu grob noch auf zu fein aufgelösten Daten gute Ergebnisse. Die besten Ergebnisse wurden mit einer *look back resolution* von 10 Minuten erreicht, daher werden bei folgenden Prognosen mit Hilfe einer Random Forest Regression immer Datensätze mit dieser Auflösung eingesetzt.

| look back resolution | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| 10min | 2.403 | 4.284 | 6.841 | 2.152 | 3.376 | 5.833 |
| 5min | 2.394 | 4.273 | 6.834 | 2.237 | 3.363 | 5.836 |
| 20min | 2.4 | 4.304 | 6.846 | 2.171 | 3.466 | 5.823 |
| 15min | 2.392 | 4.27 | 6.832 | 2.202 | 3.389 | 5.901 |
| 2min | 2.398 | 4.282 | 6.795 | 2.249 | 3.35 | 5.932 |
| 1min | 2.401 | 4.262 | 6.799 | 2.203 | 3.383 | 5.976 |

Tabelle 10: Ergebnisse der Datensatzoptimierung für die Random Forest Regression.

### 4.1.4 Support Vector Regression

#### Modellparameteroptimierung

Tabelle 11 zeigt die besten Parametrisierungen für die Support Vector Regression. Es fällt sofort auf, dass es nur Modelle mit linearem Kern unter die besten Ergebnisse geschafft haben. Tatsächlich zeigt sich bei der Betrachtung aller Ergebnisse, dass alle Modelle mit linearem Kern besser performen, als Modelle mit nichtlinearem Kern. Die besten Werte für $C$ und $\epsilon$ zeigen, dass mehr Fehler im Bezug auf

die Trainingsdaten zugelassen werden müssen, um eine bessere Generaliserbarkeit zu erhalten.

| look back resolution | $C$ | $\epsilon$ | Kern | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | 0.6 | 0.3 | linear | 2.38 | 4.447 | 7.243 | 1.9 | 3.125 | 5.636 |
| 5min | 0.5 | 0.01 | linear | 2.388 | 4.453 | 7.256 | 1.903 | 3.122 | 5.637 |
| 5min | 0.5 | 0.3 | linear | 2.387 | 4.454 | 7.251 | 1.905 | 3.12 | 5.637 |
| 5min | 0.6 | 0.01 | linear | 2.382 | 4.446 | 7.247 | 1.898 | 3.13 | 5.634 |
| 5min | 0.6 | 0.2 | linear | 2.38 | 4.447 | 7.244 | 1.899 | 3.129 | 5.635 |
| 5min | 0.5 | 0.05 | linear | 2.389 | 4.454 | 7.256 | 1.903 | 3.125 | 5.638 |
| 5min | 0.5 | 0.2 | linear | 2.387 | 4.453 | 7.254 | 1.904 | 3.125 | 5.638 |
| 5min | 0.6 | 0.05 | linear | 2.382 | 4.447 | 7.248 | 1.899 | 3.134 | 5.634 |
| 5min | 0.5 | 0.1 | linear | 2.389 | 4.454 | 7.255 | 1.904 | 3.127 | 5.638 |
| 5min | 0.8 | 0.3 | linear | 2.372 | 4.437 | 7.234 | 1.892 | 3.137 | 5.643 |

Tabelle 11: Ergebnisse der Modellparameteroptimierung für die Support Vector Regression

#### Datensatzoptimierung

Tabelle 12 zeigt die Ergebnisse der Support Vector Regression auf den verschieden aufgelösten Datensätzen. Es ist zu beobachten, dass ausgehend von einer 20-minütigen Auflösung die Ergebnisse sich durch immer feiner werdende Auflösungen verbessern. Bei einer Auflösung von 5 Minuten ist jedoch ein Grenzwert erreicht und durch eine weitere Verfeinerung werden die Ergebnisse wieder deutlich schlechter.

| look back resolution | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| 5min | 2.38 | 4.447 | 7.243 | 1.9 | 3.125 | 5.636 |
| 10min | 2.4 | 4.463 | 7.282 | 1.942 | 3.138 | 5.628 |
| 15min | 2.419 | 4.492 | 7.322 | 1.988 | 3.135 | 5.664 |
| 2min | 2.38 | 4.437 | 7.218 | 2.003 | 3.263 | 5.691 |
| 20min | 2.425 | 4.509 | 7.342 | 2.04 | 3.211 | 5.749 |
| 1min | 2.364 | 4.373 | 7.125 | 2.048 | 3.26 | 5.819 |

Tabelle 12: Ergebnisse der Datensatzoptimierung für die Support Vector Regression.

### 4.1.5 Neuronale Netze

#### Modellparameteroptimierung

**Anzahl Neuronen, Aktivierungsfunktion** Im ersten Teilschritt der Modellparameteroptimierung für die neuronalen Netze werden die Anzahl der Neuronen pro Layer, sowie die verwendete Aktivierungsfunktion optimiert. Bei der Betrachtung

der Ergebnisse in Tabelle 13 fällt sofort auf, dass für die meisten Netztopologien die Aktivierungsfunktionen der Hidden Layer am besten linear sein sollten. Dieses Verhalten lässt sich also nicht nur bei der Support Vector Regression, sondern auch bei den neuronalen Netzen beobachten. Das ist ein überraschendes Ergebnis, denn aufgrund der Komplexität des zu lösenden Problems war initial vermutet worden, dass nichtlineare Modelle besser geeignet wäre.

Die Anzahl der Neuronen pro Layer variiert von Modell zu Modell. Es ist kein klarer Trend erkennbar, dass Modelle mit weniger Neuronen bessere Ergebnisse erzielen, als Modelle mit vielen Neuronen.

| look back resolution | Netz | $\varphi(x)$ | Anzahl Neuronen | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN3 | linear | 128 | 2.399 | 4.411 | 7.165 | 1.904 | 3.062 | 5.591 |
| 5min | CNN1 | linear | 16 | 2.367 | 4.388 | 7.155 | 1.888 | 3.117 | 5.624 |
| 5min | CNN2 | linear | 32 | 2.374 | 4.38 | 7.146 | 1.912 | 3.113 | 5.606 |
| 5min | CNN1 | linear | 256 | 2.407 | 4.412 | 7.173 | 1.932 | 3.11 | 5.615 |
| 5min | CNN2 | linear | 64 | 2.37 | 4.391 | 7.147 | 1.938 | 3.098 | 5.623 |
| 5min | CNN3 | linear | 32 | 2.368 | 4.375 | 7.12 | 1.92 | 3.109 | 5.631 |
| 5min | CNN3 | linear | 64 | 2.376 | 4.385 | 7.155 | 1.929 | 3.115 | 5.628 |
| 5min | CNN1 | linear | 128 | 2.376 | 4.385 | 7.151 | 1.926 | 3.122 | 5.625 |
| 5min | CNN2 | linear | 16 | 2.366 | 4.389 | 7.158 | 1.952 | 3.109 | 5.616 |
| 5min | MLP5 | linear | 128 | 2.412 | 4.415 | 7.2 | 1.969 | 3.113 | 5.601 |

Tabelle 13: Ergebnisse der Modellparameteroptimierung (Anzahl Neuronen, Aktivierungsfunktion) für die neuronalen Netze.

**Optimierungsalgorithmus, Learning Rate, Batch Size** Die Ergebnisse aus Tabelle 14 zeigen, dass durch die Optimierungsalgorithmen Adam und RMSprop die besten Ergebnisse erzielt werden können. Die initiale Learning Rate tendiert dabei zwischen 0.01 und 0.001. Die Batch Size sollte in den meisten Fällen eher klein gehalten werden, einzelne Ausnahmen sind aber auch möglich.

| look back resolution | Netz | Optimierer | Learning Rate | Batch Size | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN3 | rmsprop | 0.001 | 16 | 2.422 | 4.421 | 7.185 | 1.961 | 3.093 | 5.582 |
| 5min | CNN1 | rmsprop | 0.001 | 16 | 2.381 | 4.4 | 7.187 | 1.931 | 3.101 | 5.613 |
| 5min | CNN3 | adam | 0.001 | 16 | 2.382 | 4.371 | 7.121 | 1.924 | 3.115 | 5.622 |
| 5min | MLP1 | rmsprop | 0.01 | 32 | 2.389 | 4.407 | 7.188 | 1.949 | 3.088 | 5.63 |
| 20min | CNN2 | adam | 0.01 | 32 | 2.456 | 4.496 | 7.232 | 1.962 | 3.094 | 5.612 |
| 5min | CNN1 | adam | 0.001 | 64 | 2.376 | 4.402 | 7.203 | 1.935 | 3.114 | 5.619 |
| 5min | CNN2 | adam | 0.01 | 64 | 2.393 | 4.398 | 7.161 | 1.961 | 3.099 | 5.61 |
| 5min | CNN1 | rmsprop | 0.01 | 64 | 2.379 | 4.383 | 7.166 | 1.937 | 3.105 | 5.631 |
| 5min | CNN2 | adam | 0.001 | 128 | 2.409 | 4.424 | 7.205 | 1.962 | 3.103 | 5.612 |
| 5min | MLP2 | adam | 0.01 | 32 | 2.413 | 4.414 | 7.215 | 1.966 | 3.086 | 5.627 |

Tabelle 14: Ergebnisse der Modellparameteroptimierung (Optimierungsalgorithmus, Learning Rate, Batch Size) für die neuronalen Netze.

**Dropout, Max-norm Regularisierung** Die Regularisierungsmaßnahmen zeigen sich im größten Sinne als nicht sehr wirkungsvoll. Das Ursache hierfür liegt darin begründet, wie die Netze trainiert wurden. Anstatt ein Netz für eine gewisse Anzahl an Epochen zu trainieren, dann zu stoppen und die Validierung durchzuführen, wurde das Training dauerhaft überwacht. D. h. dass nach jeder Trainingsepoche die Validierung durchgeführt wurde und anschließend das Modell mit seinen aktuellen Gewichten und den Validierungsergebnissen abgespeichert wurde. Nach Beendigung des Trainings werden dann alle Epochen miteinander verglichen, und diejenige bestimmt, die die besten Ergebnisse erzeugt hat. Dann kann das dazugehörige Modell ausgewählt werden. Dieses Vorgehen wird als *Early Stopping* bezeichnet und stellt für sich auch schon eine Art der Regularisierung dar. Im Kontext dieses Problems scheint diese Methode besser zu funktionieren, als der Einsatz eines Dropouts. Im Gegensatz dazu kann die max-norm Regularisierung noch einige kleine Verbesserungen erzielen.

| look back resolution | Netz | Dropout Rate | max-norm | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN3 | 0.0 | None | 2.397 | 4.409 | 7.167 | 1.93 | 3.093 | 5.595 |
| 5min | CNN1 | 0.0 | 2 | 2.399 | 4.416 | 7.214 | 1.925 | 3.106 | 5.607 |
| 5min | CNN1 | 0.0 | 3 | 2.369 | 4.384 | 7.179 | 1.925 | 3.114 | 5.622 |
| 5min | CNN1 | 0.0 | None | 2.377 | 4.391 | 7.188 | 1.927 | 3.114 | 5.623 |
| 5min | CNN3 | 0.0 | 2 | 2.384 | 4.416 | 7.216 | 1.919 | 3.104 | 5.643 |
| 5min | MLP5 | 0.0 | None | 2.377 | 4.403 | 7.203 | 1.943 | 3.11 | 5.618 |
| 5min | CNN3 | 0.0 | 4 | 2.368 | 4.391 | 7.141 | 1.889 | 3.154 | 5.636 |
| 5min | CNN2 | 0.0 | None | 2.396 | 4.389 | 7.174 | 1.954 | 3.113 | 5.617 |
| 5min | CNN3 | 0.0 | 3 | 2.37 | 4.403 | 7.175 | 1.912 | 3.096 | 5.682 |
| 5min | MLP4 | 0.0 | 2 | 2.43 | 4.442 | 7.246 | 1.962 | 3.102 | 5.628 |

Tabelle 15: Ergebnisse der Modellparameteroptimierung (Dropout Rate, max-norm Regularisierung) für die neuronalen Netze.

### Datensatzoptimierung

Tabelle 16 zeigt die Ergebnisse der neuronalen Netze auf den verschieden aufgelösten Datensätzen. Auch hier ist zu beobachten, dass die neuronalen Netze für die mittlere Auflösungen die besten Ergebnisse liefern. Wie auch schon bei der Modellparameteroptimierung beobachtet wurde, stechen die Convolutional Neural Networks am meisten hervor. Ihre Ergebnisse stehen aber auch in direkter Konkurrenz zu denen der einfachen feedforward Netze. Die Long short-term memory-Netze konnten auch durch die verschiedenen Auflösungen nicht aufholen. Jedoch ist in Kapitel A.4 zu sehen, dass die verschiedenen LSTM-Netze ihre besten Ergebnisse zum Teil auf den sehr feinen aufgelösten Datensätzen erreichen.

| look back resolution | Netz | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|
| | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN1 | 2.386 | 4.393 | 7.177 | 1.932 | 3.097 | 5.614 |
| 15min | CNN1 | 2.387 | 4.41 | 7.205 | 1.913 | 3.12 | 5.642 |
| 5min | CNN3 | 2.372 | 4.403 | 7.186 | 1.925 | 3.12 | 5.639 |
| 10min | CNN3 | 2.372 | 4.4 | 7.173 | 1.926 | 3.131 | 5.629 |
| 10min | CNN1 | 2.363 | 4.388 | 7.168 | 1.924 | 3.131 | 5.635 |
| 5min | MLP5 | 2.369 | 4.388 | 7.173 | 1.928 | 3.133 | 5.633 |
| 5min | MLP3 | 2.433 | 4.43 | 7.215 | 2.001 | 3.089 | 5.606 |
| 15min | MLP2 | 2.399 | 4.441 | 7.232 | 1.965 | 3.095 | 5.638 |
| 20min | CNN2 | 2.379 | 4.458 | 7.227 | 1.947 | 3.141 | 5.62 |
| 5min | MLP4 | 2.426 | 4.424 | 7.196 | 1.96 | 3.113 | 5.639 |

Tabelle 16: Ergebnisse der Datensatzoptimierung für die verschiedenen neuronalen Netze. Die Ergebnisse für alle Netzwerktopologien ist in Kapitel A.4 zu sehen.

## 4.2 Evaluation auf ausgewählten Datensätzen

Mit Hilfe der Parameteroptimierung konnte für jedes Modell festgelegt werden, welche Parameter für bestmögliche Ergebnisse sorgen, und welche zeitliche Auflösung die Eingabedaten dafür haben sollten. Diese Ergebnisse werden nun auf die Evaluation von speziell ausgewählten Datensätzen übertragen, die wie folgt definiert sind:

1. Training der Modelle auf den Daten des Jahres 2015 und
   - 1.1 Validierung auf den Daten des Jahres 2016,
   - 1.2 Validierung auf den Daten des Jahres 2017.
2. Training der Modelle auf den Daten des Jahres 2015 und 2016, sowie Validierung auf den Daten des Jahres 2017.
3. Training der Modelle auf den Daten vom 01.01.2015 bis 31.03.2016, sowie Validierung auf den Daten vom 01.04.2016 bis 31.07.2016.
4. Training der Modelle auf den Daten des Jahres 2015 und 2016, sowie
   - 4.1 Validierung auf Daten des Winters 2017,
   - 4.2 Validierung auf Daten des Frühlings 2017,
   - 4.3 Validierung auf Daten des Sommers 2017,
   - 4.4 Validierung auf Daten des Herbstes 2017.

Durch Szenario 1 soll evaluiert werden, wie sich die Güte eines Modells verhält, wenn zwischen den Trainingsdaten und den Daten, für die eine Prognose erstellt werden soll, eine größere zeitliche Distanz entsteht. Durch Szenario 2 soll anschließend analysiert werden, ob durch zusätzliche, zeitlich nähere Informationen eine Verbesserung der Modellgüte erreicht werden kann. Das dritte Szenario wurde ausgewählt, um einen Vergleich zu dem Prognosemodell der BTC AG herstellen zu können. Dieses wurde auf den gleichen Zeiträumen trainiert und validiert. Mit Hilfe des letzen Szenarios soll abschließend analyisiert werden, wie sich das Prognosemodell für die unterschiedlichen Jahreszeiten verhält.

### 4.2.1 Training 2015; Validierung 2016 und 2017

In Tabelle 17 und Tabelle 18 werden die Ergebnisse gegenübergestellt, die sich durch eine Verschiebung des Validierungszeitraumes ergeben. In beiden Fällen wurden die verschiedenen Modelle auf den Daten des Jahres 2015 trainiert. Bei der Validierung auf den Daten des Jahres 2016 zeigt sich, dass die die Support Vector Regression und das Convolutional Neural Network am besten abschneiden. Beide sind aber dicht gefolgt von der Linearen Regression, deren Ergebnis für das Early-Target sogar besser ist, als das des CNN. Die Random Forest Regression liefert auch

noch passable Ergebnisse, sie sind aber schon etwas schlechter als die Ergebnisse der drei Spitzenreiter. Das Schlusslicht bildet die k-Nearest Neighbor Regression, deren Ergebnisse deutlich hinter den anderen zurückliegen.

Im Vergleich dazu stehen die Ergebnisse für die Validierung auf den Daten des Jahres 2017. Nun hat sich die Lineare Regression für das Early-Target auf den ersten Platz geschoben. Im Vergleich zu den Ergebnissen aus Tabelle 17 ist eine deutliche Verschlechterung der Prognosequalität erkennbar. Diese zieht sich durch alle drei Targets fort, wobei insbesondere das Late-Target um einiges an Genauigkeit verloren hat. Diese Ergebnisse zeigen also, dass ein Modell auf veralteten Daten deutlich an Genauigkeit verliert.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.333 | 4.369 | 7.147 | 1.901 | 3.235 | 5.805 |
| KNN | 0.0 | 0.0 | 0.0 | 4.910 | 5.525 | 7.312 |
| RF | 2.388 | 4.269 | 6.842 | 2.151 | 3.373 | 5.876 |
| SVR | 2.380 | 4.447 | 7.243 | **1.900** | 3.125 | 5.636 |
| CNN3 | 2.37 | 4.391 | 7.165 | 1.909 | **3.113** | **5.618** |

Tabelle 17: Training der verschiedenen Modelle auf Daten des Jahres 2015. Die Validierung wurde auf den Daten des Jahres 2016 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.333 | 4.369 | 7.147 | **2.967** | 4.427 | 7.971 |
| KNN | 0.0 | 0.0 | 0.0 | 8.753 | 9.146 | 11.117 |
| RF | 2.394 | 4.251 | 6.849 | 6.001 | 7.036 | 9.700 |
| SVR | 2.380 | 4.447 | 7.243 | 3.057 | **4.302** | 7.755 |
| CNN3 | 2.385 | 4.424 | 7.189 | 3.055 | 4.33 | **7.724** |

Tabelle 18: Training der verschiedenen Modelle auf Daten des Jahres 2015. Die Validierung wurde auf den Daten des Jahres 2017 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

### 4.2.2 Training 2015 + 2016; Validierung 2017

In Kapitel 4.2.1 wurde gezeigt, dass die Ergebnisqualität der Prognosemodelle mit größerer Distanz zwischen Trainings- und Validierungszeitraum abnimmt. Tabelle 19 zeigt, wie sich die Ergebnisse verändern, wenn weitere Daten zum Training hinzugenommen werden, die dem Validierungszeitraum zeitlich näher sind. Dabei ist zwar eine leichte Verbesserung der Ergebnisse zu erkennen, jedoch fällt diese geringer aus, als ursprünglich erwartet wurde. Daher scheint es so, als hätte sich im Jahr 2017 eine deutliche Veränderung des Marktverhaltens ergeben, die von

dem Prognosemodell nicht erfasst werden kann. Daher wird im Folgenden das Jahr 2017 genauer untersucht werden.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.140 | 3.878 | 6.548 | **2.934** | 4.263 | 7.693 |
| KNN | 0.0 | 0.0 | 0.0 | 8.111 | 8.588 | 10.691 |
| RF | 2.237 | 3.911 | 6.357 | 5.490 | 6.748 | 8.982 |
| SVR | 2.197 | 3.942 | 6.627 | 3.014 | **4.255** | 7.735 |
| CNN3 | 2.223 | 3.932 | 6.581 | 2.998 | **4.255** | **7.684** |

Tabelle 19: Training der verschiedenen Modelle auf Daten des Jahres 2015 und 2016. Die Validierung wurde auf den Daten des Jahres 2017 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

### 4.2.3 Training 2015 + 2015; Validierung Jahreszeiten 2017

In Tabelle 20, Tabelle 21, Tabelle 22 und Tabelle 23 sind die Validierungsergebnisse für die vier Jahreszeiten des Jahres 2017 dargestellt. Dabei wurden die meteorologischen Jahreszeiten verwendet, d. h.

- Frühling: 01. März bis 31. Mai,
- Sommer: 01. Juni bis 31. August,
- Herbst: 01. September bis 30. November,
- Winter: 01. Dezember bis 28./29. Februar.

Theoretisch verläuft der Winter also über zwei Jahre. Da die Daten für 2018 aber nicht vorliegen, werden stattdessen Januar, Februar und Dezember des Jahres 2017 als Winter zusammengefasst. Das Training der verwendeten Modelle wurde auf den Daten von 2015 und 2016 durchgeführt, da in Kapitel 4.2.2 bereits gezeigt werden konnte, dass sich durch die Hinzunahme der 2016-Daten eine Verbesserung der Prognosegüte ergibt.

Die Ergebnisse zeigen, dass die Prognosen für den Frühling und Sommer deutlich verlässlicher sind, als für den Herbst und den Winter, denn dort steigen die Fehler in der Prognose signifikant an. Aus diesem Ergebnis sind daher zwei Schlussfolgerungen möglich

1. Die Qualität der Prognosen hängt signifikant von der Jahreszeit bzw. allgemein von dem Zeitpunkt im Jahr ab.
2. Im Verlauf des Jahres 2017 hat sich das Marktverhalten derart verändert, sodass die Abweichung zu der Trainingsgrundlage der Modelle immer größer wurde. Diese Schlussfolgerung ist aber nur möglich, da im Winter der Anfang und das Ende des Jahres zusammengefasst wurden.

Zur Untersuchung der zweiten Schlussfolgerung wurde noch eine separate Validierung nur auf den Daten des Januars und Februars durchgeführt (siehe Tabelle 24). Tatsächlich sind diese Ergebnisse noch schlechter, als wenn der Dezember mit in die Validierungsmenge aufgenommen wurde. Daher kann diese Hypothese abgelehnt werden.

Es zeigt sich also, dass der Zeitpunkt, für den eine Prognose durchgeführt werden soll, einen sehr starken Einfluss auf die Prognosemodelle hat.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.140 | 3.878 | 6.548 | **1.798** | 3.128 | **5.461** |
| KNN | 0.0 | 0.0 | 0.0 | 5.252 | 5.998 | 7.495 |
| RF | 2.230 | 3.913 | 6.362 | 2.359 | 3.804 | 6.074 |
| SVR | 2.197 | 3.942 | 6.627 | 1.884 | 3.247 | 5.578 |
| CNN3 | 2.188 | 3.931 | 6.581 | 1.800 | **3.096** | 5.470 |

Tabelle 20: Training der verschiedenen Modelle auf Daten des Jahres 2015 und 2016. Die Validierung wurde auf den Daten des Frühlings von 2017 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.140 | 3.878 | 6.548 | **1.778** | 2.841 | 5.510 |
| KNN | 0.0 | 0.0 | 0.0 | 4.827 | 5.479 | 7.178 |
| RF | 2.230 | 3.891 | 6.361 | 1.934 | 3.100 | 5.592 |
| SVR | 2.197 | 3.942 | 6.627 | 1.840 | 2.908 | 5.570 |
| CNN3 | 2.187 | 3.907 | 6.586 | 1.800 | **2.827** | **5.431** |

Tabelle 21: Training der verschiedenen Modelle auf Daten des Jahres 2015 und 2016. Die Validierung wurde auf den Daten des Sommers von 2017 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.140 | 3.878 | 6.548 | 3.851 | 4.203 | 7.713 |
| KNN | 0.0 | 0.0 | 0.0 | 6.605 | 6.981 | 9.737 |
| RF | 2.218 | 3.897 | 6.377 | 4.102 | 4.873 | 8.290 |
| SVR | 2.197 | 3.942 | 6.627 | **3.831** | **4.062** | 7.721 |
| CNN3 | 2.222 | 3.963 | 6.599 | 3.861 | 4.108 | **7.701** |

Tabelle 22: Training der verschiedenen Modelle auf Daten des Jahres 2015 und 2016. Die Validierung wurde auf den Daten des Herbstes von 2017 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.140 | 3.878 | 6.548 | **3.529** | 6.014 | 10.681 |
| KNN | 0.0 | 0.0 | 0.0 | 12.847 | 13.290 | 15.777 |
| RF | 2.232 | 3.903 | 6.359 | 9.122 | 11.138 | 12.814 |
| SVR | 2.197 | 3.942 | 6.627 | 3.740 | **6.001** | 10.708 |
| CNN3 | 2.283 | 3.947 | 6.583 | 3.629 | 6.016 | **10.64** |

Tabelle 23: Training der verschiedenen Modelle auf Daten des Jahres 2015 und 2016. Die Validierung wurde auf den Daten des Winters von 2017 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.140 | 3.878 | 6.548 | 3.796 | 6.672 | 11.883 |

Tabelle 24: Training einer Linearen Regression auf Daten des Jahres 2015 und 2016. Die Validierung wurde auf den Daten des Januars und Februars von 2017 durchgeführt.

### 4.2.4 Training BTC; Validierung BTC

In Zusammenarbeit mit der EWE Trading GmbH hat die BTC AG ebenfalls ein Prognosemodell für die Preisentwicklung am Intraday-Markt entwickelt. Die Ergebnisse dieses Modells wurden mir freundlicherweise zur Verfügung gestellt, sodass ein Vergleich zwischen den in dieser Arbeit entwickelten Modellen und dem BTC-Modell gezogen werden kann. Die Validierungsergebnisse des BTC-Modells beschränken sich dabei auf den Zeitraum vom 01.04.2016 bis 31.07.2016. Das Training wurde auf den Daten vom 01.01.2015 bis 31.03.2016 durchgeführt. Um eine Vergleichbarkeit der Modelle zu gewährleisten, werden auch die eigens entwickelten Modelle auf diesen Zeiträumen trainiert bzw. validiert.

Die Ergebnisse sind in Tabelle 25 dargestellt. Es zeigt sich, dass alle in dieser Arbeit entwickelten Modelle, mit Ausnahme der k-Nearest Neighbor Regression, bessere Ergebnisse als das BTC-Modell erreichen.

| Modell | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| LR | 2.309 | 4.274 | 7.082 | **1.484** | 2.454 | 4.561 |
| KNN | 0.0 | 0.0 | 0.0 | 3.852 | 4.248 | 5.742 |
| RF | 2.390 | 4.217 | 6.822 | 1.590 | 2.483 | 4.565 |
| SVR | 2.364 | 4.350 | 7.181 | 1.606 | **2.412** | 4.485 |
| CNN3 | 2.347 | 4.289 | 7.106 | 1.536 | **2.412** | **4.500** |
| BTC | 2.580 | 4.530 | 7.470 | 1.650 | 2.820 | 5.580 |

Tabelle 25: Training der verschiedenen Modelle auf Daten vom 01.01.2015 bis 31.03.2016. Die Validierung wurde auf den Daten vom auf den Daten vom 01.04.2016 bis 31.07.2016 durchgeführt. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

## 4.3 Weitergehende Experimente

Im Folgenden werden noch kleine Versuche durchgeführt, die einen Ausblick darauf geben sollen, wie die Modelle auf

- ein größeres *look back window*,
- eine höhere *forecast resolution*,
- oder Viertelstundenprodukte

reagieren. Da die Lineare Regression in der vorherigen Evaluation immer sehr gute Ergebnisse im Vergleich zu den anderen Verfahren erzielen konnte, werden diese Experimente nur für die Linare Regression durchgeführt. Trainiert wird hierbei auf den Daten des Jahres 2015 und die Validierung findet auf den Daten von 2016 statt.

### 4.3.1 Größeres look back window

In Tabelle 26 sind die Ergebnisse der Linearen Regression für ein *look back window* von 90 Minuten dargestellt. Im Vergleich dazu stehen noch einmal die Ergebnisse der Linearen Regression aus Tabelle 17. Es zeigt sich, dass die Prognosegüte für die Early- und Mid-Targets abnimmt, während sich das Late-Target verbessert.

### 4.3.2 Höhere forecast resolution

In Tabelle 27 sind die Ergebnisse der Linearen Regression für eine *forecast resolution* von 10 Minuten dargestellt. Im Vergleich dazu stehen noch einmal die Ergebnisse der Linearen Regression aus Tabelle 17. Dafür wurden die drei Targets jeweils in zwei Hälften unterteilt. Die Prognosequalität nimmt durch die feinere Unterteilung offenbar ab. Lediglich bei dem Late-Target kann für die erste Hälfte ein besseres Ergebnis erzielt werden. Bildet man jedoch den Durchschnitt der zwei 10-minütigen

| look back window | Training RMSE | | | Validation RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| 90min | 2.319 | 4.354 | 7.128 | 1.962 | 3.399 | **5.714** |
| 60min | 2.333 | 4.369 | 7.147 | **1.901** | **3.235** | 5.805 |

Tabelle 26: Training einer Linearen Regression auf Daten des Jahres 2015. Die Validierung wurde auf den Daten von 2016 durchgeführt. Dabei wurde das *look back window* im Vergleich zu den bisherigen Versuchen vergrößert. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

Intervalle für das Late-Target, so ist das Ergebnis für das 20-minütige Late-Target wiederum besser.

| forecast resolution | Validation RMSE | | | | | |
|---|---|---|---|---|---|---|
| | Early | | Mid | | Late | |
| | 1 | 2 | 1 | 2 | 1 | 2 |
| 10min | 2.251 | 2.238 | 3.516 | 3.709 | **4.862** | 7.479 |
| 20min | **1.901** | | **3.235** | | 5.805 | |

Tabelle 27: Training einer Linearen Regression auf Daten des Jahres 2015. Die Validierung wurde auf den Daten von 2016 durchgeführt. Dabei wurde die *forecast resolution* im Vergleich zu den bisherigen Versuchen erhöht. In diesem Fall wird aus Platzgründen auf die Angabe des Training RMSE verzichtet. Die besten Ergebnisse für die verschiedenen Targets sind hervorgehoben.

### 4.3.3 Viertelstundenprodukte

In Tabelle 28 sind die Ergebnisse der Linearen Regression für Viertelstundenprodukte dargestellt. Im Vergleich dazu stehen noch einmal die Ergebnisse der Linearen Regression aus Tabelle 17. Die Ergebnisse zeigen offensichtich, dass die Prognose der Preisentwicklung für Viertelstundenprodukte ein deutlich komplexeres Problem als für Stundenprodukte darstellt.

| Produkttyp | Training RMSE | | | Validation RMSE | | |
|---|---|---|---|---|---|---|
| | Early | Mid | Late | Early | Mid | Late |
| Viertelstunde | 7.318 | 10.368 | 13.676 | 7.018 | 8.060 | 10.219 |
| Stunde | 2.333 | 4.369 | 7.147 | 1.901 | 3.235 | 5.805 |

Tabelle 28: Training einer Linearen Regression auf Daten des Jahres 2015. Die Validierung wurde auf den Daten von 2016 durchgeführt. Im Gegensatz zu den bisherigen Versuchen wurden nun die Daten der Viertelstundenprodukte verwendet.

## 4.4 Bewertung der Ergebnisse

Die Evaluation aus Kapitel 4.2 hat gezeigt, wie sich die entwickelten Prognosemodelle in verschiedenen Situationen verhalten. Die Güte der Modelle wurde dabei immer durch den RMSE angegeben, also die Wurzel des Durchschnitts der quadratischen Fehler. Das bedeutet, dass für jedes Beispiel aus der Validierungsmenge die Differenz zwischen erwartetem Wert und dem Prognosewert gebildet und anschließend quadriert wird. Dann wird der Durchschnitt für alle Validierungsbeispiele gebildet, um daraus anschließend wieder die Wurzel zu ziehen. Er gibt somit eine durchschnittliche Abweichung von dem Erwartungswert an, wobei durch das Wurzelziehen erst nach der Durchschnittsbildung der Quadrate große Ausreißer stärker bestraft werden. Wenn nun der RMSE für das Early-Target 2,934 beträgt (wie in Kapitel 4.2.2), dann kann für neue Prognosen davon ausgegangen werden, dass der tatsächliche Preis durchschnittlich um 2,934€ von dem prognostizierten Wert abweicht. Demnach können also Prognosen erstellt werden, die deutlich näher an dem wirklichen Wert liegen. Umgekehrt werden aber auch einige Prognosen noch größere Abweichunen von dem tatsächlichen Wert haben.

Da ein solches Prognosemodell einen Akteuer am Intraday-Markt bei der Entscheidungsfindung unterstützen soll, ist natürlich eine bestmögliche Prognosegüte wünschenswert. Die durchgeführte Evaluation hat gezeigt, dass die in dieser Arbeit entwickelten Modelle definitiv konkurrenzfähig im Vergleich zu einer bestehenden Lösung ist (sieh Kapitel 4.2.4). Auf der anderen Seite konnte aber auch gezeigt werden, dass die Prognosegüte im Jahr 2017 deutlich im Vergleich zum Vorjahr abgenommen hat. Gerade die Unsicherheit für die späteren Zeitpunkte (Mid, Late) hat stark zugenommen. Inwiefern eine solche Lösung also wirtschaftlich rentabel eingesetzt werden kann, müsste genauer analysiert oder getestet werden. Dies geht aber über den Rahmen dieser Arbeit hinaus. Durch die Analysen im Bezug auf die Jahreszeiten (siehe Kapitel 4.2.3) konnte aber gezeigt werden, dass solche Prognosemodelle im Frühling und Sommer deutlich verlässlicher arbeiten, als im Herbst und Winter.

# 5 Zusammenfassung und Ausblick

## 5.1 Zusammenfassung der Arbeit

Angetrieben durch den Klimaschutz und eine absehbare Verknappung von natürlichen Ressourcen, wird viel in den kontinuierlichen Zubau von Erzeugungsanlagen auf Basis von regenerativen Energiequellen investiert. Diese Veränderung in der Erzeugungsstruktur von elektrischem Strom macht einen Wandel in vielen Aspekten der Energiewirtschaft notwendig. Der notwendige Ausbau des deutschen Stromnetzes zur deutschlandweiten Verteilung des in küstennähe produzierten Windstroms wurde in den letzten Jahren immer wieder breit diskutiert. Die Abhängigkeit der Stromerzeugung einer Erneuerbare-Energie-Anlage von den Umweltbedingungen erlaubt nur relativ kurzfristige Prognosen der zu einem Zeitpunkt zur Verfügung stehenden Leistung. Das ist ein drastischer Unterschied zu der langfristigen Planbarkeit von Großkraftwerken, die bis zu mehrere Jahre in die Zukunft möglich ist. Da es das langfristige Ziel ist, diese Großkraftwerke durch die erneuerbaren Energien zu ersetzen, findet eine Verschiebung an den Strommärkten statt. Energieversorgungsunternehmen und andere Strombezieher müssen immer kurzfristiger ihren Strombedarf abdecken. Zu diesem Zweck wurden Strombörsen geschaffen, die einen sehr kurzfristigen Handel von Strom erlauben. An dem Intraday-Markt der EPEX SPOT kann Strom noch bis kurz vor der physikalischen Lieferung gehandelt werden. Der Preis für eine gehandelte Strommenge ist eine flexible Größe, die sich aus den Geboten von Käufer und Verkäufer bestimmen.

In dieser Arbeit wurde durch den Einsatz von Maschinellem Lernen ein Prognosemodell entwickelt, das die Preisentwicklung am Intraday-Markt vorhersagen soll. Hierfür wurden verschiedene Datenquellen herangezogen, deren Rohdaten in einem ersten Schritt zunächst aufbereitet wurden. Die so erhaltenen Daten wurden als Eingabe für verschiedene Machine Learning Modelle eingesetzt. Dabei kamen Lineare Regressionen, k-Nearest Neighbor Regressionen, Random Forest Regressionen, Support Vector Regressionen und verschiedenste neuronale Netze zum Einsatz. Durch eine Parameteroptimierung wurden diese Modelle auf das Problem spezieller angepasst, sodass dann eine Evaluation der Prognosequalität vorgenommen werden konnte. In der Evaluation konnte gezeigt werden, dass die entwickelten Modelle zunächst einmal konkurrenzfähig im Bezug auf das durch die BTC AG entwickelte Prognosemodell sind. Weitergehende Analysen ergaben, dass der Einsatz solcher Prognoseverfahren immer in einem Kontext betrachtet werden sollte. Einerseits ist eine wachsende Komplexität des Prognoseproblems beobachtbar, andererseits hängt die Qualität der Vorhersagen auch von den Zeitpunkten ab, für die eine Prognose erstellt werden soll.

## 5.2 Ausblick

Im Folgenden sollen einige Überlegungen vorgestellt werden, die im Rahmen dieser Arbeit nicht umgesetzt werden konnten.

### 5.2.1 Kombination verschiedener Verfahren

In der Evaluation konnte beobachtet werden, das oftmals verschiedene Verfahren für eines der Targets die besten Ergebnisse geliefert hat. Es wäre also möglich, dass man verschiedene Modelle für die unterschiedlichen Targets trainiert, und diese dann logisch zu einem neuen Modell zusammenfasst. Soll dieses Modell eine Prognose erstellen, so wird dann das Eingabedatum an die einzelnen Teilmodelle weitergegeben, die dann für die unterschiedlichen Targets die Prognosen erstellen. Diese werden dann zusammengefasst und als Vorhersage für die verschiedenen Targets zurückgegeben. Für den Anwender ist von außen gar nicht erkennbar, dass intern verschiedene Verfahren für die Prognose eingesetzt wurden.

Eine zweite Möglichkeit wäre es, dass für jedes Verfahren ein Modell trainiert wird, die jeweils für alle Targets eine eigene Prognose liefern. Aus den Einzelvorhersagen kann dann mit einer Durchschnittsberechnung (ggf. auch gewichtet) eine kombinierte Vorhersage getroffen werden. So könnten sich gewisse Schwächen der verschiedenen Modelle ggf. ausgleichen.

### 5.2.2 Korrelation zwischen Stunden- und Viertelstundenprodukten

In dieser Arbeit wurde bei dem Training und der Validierung der Prognosemodelle immer eine strikte Trennung zwischen den Stunden- und Viertelstundenprodukten vorgenommen. Wie in Kapitel A.1.2 untersucht wurde, bestehen die Transaktionslogs zu ca. 47,97% aus Transaktionen für Stundenprodukte und zu ca. 51,83% aus Transaktionen für Viertelstundenprodukte. Wird sich also nur auf eines dieser Produkte konzentriert, so gehen knapp 50% des Informationsgehalts der Transaktionslogs verloren. Es ist denkbar, dass Transaktionen für Stundenprodukte und die dazugehörigen Viertelstundenprodukten korrelieren. Ist das tatsächlich der Fall, so könnte die Prognose für Stundenprodukte durch die Transaktionsdaten der Viertelstundenprodukte profitieren. Auch andersherum könnten die Prognosen für Viertelstundenprodukte durch die Transaktionsdaten der Stundenprodukte verbessert werden.

Um diese These zu untersuchen, könnte zunächst eine Korrelationsanalyse zwischen der Preisentwicklung des einen Produkttyps und den Merkmalen des anderen Produkttyps durchgeführt werden. Sollte sich eine solche Korrelation bestätigen, würden die entsprechenden Merkmale mit in die Eingabedaten des Modells aufgenommen werden. Eine Evaluation kann dann analog wie in dieser Arbeit durchgeführt werden, um die Vergleichbarkeit der Ergebnisse zu gewährleisten.

### 5.2.3 Sliding Window

In dieser Ausarbeitung wurden die nicht-äquidistanten Zeitreihen in eine bestimmte Anzahl an Bins eingeteilt, in denen dann die Features berechnet wurden. Bei der Klassifikation von menschlichen Bewegungen anhand von Accelerometer- und Gyroskopdaten ist es ein gängiges Verfahren, dass keine solche statische Einteilung vorgenommen wird, sondern ein *Sliding Window* eingesetzt wird [BI04]. Dies ist vom Prinzip her sehr ähnlich zu einer statischen Bin-Einteilung, jedoch erlaubt dieses Verfahren auch Überlappungen der Bins. Dadurch soll vermieden werden, dass sich gewisse Signalausprägungen innerhalb eines Bins durch die Aggregation herausmitteln.

### 5.2.4 Direkte Analyse nicht-äquidistanter Zeitreihen

In dieser Arbeit wurden die nicht-äquidistanten Zeitreihen zu äquidistanten Zeitreihen transformiert, um viele gängige Verfahren einsetzen zu können. In den letzten Jahren wurde auch vermehrt die direkte Analyse nicht-äqudistanter Zeitreihen erforscht. Ein Beispiel hierfür sind die Phased LSTM, die eine Erweiterung des konventionellen Long short-term memory sind [Dan16]. Hierbei wird ein neues Gate eingeführt, das als Eingabe einen Zeitstempel für die aktuellen Eingabewerte erhält. Dieses Zeit-Gate fließt dann mit in die Berechnungen ein, sodass das Phased LSTM die nicht-äquidistanten Abstände besser verstehen kann. Daher wäre der Einsatz einer solchen Methode in dieser Problemstellung denkbar.

### 5.2.5 Veränderungen des Marktes

In Kapitel 4.2.1 wurde analysiert, wie sich das Prognosemodell verhält, wenn zwischen dem Zeitraum, für den neue Prognosen erstellt werden sollen, und dem Zeitraum der Trainingsdaten eine gewisse Distanz liegt. Die Ergebnisse haben gezeigt, dass die Prognosegüte des Modells abnimmt. Im Umkehrschluss bedeutet das, dass sich der Markt verändert haben muss, denn das Modell kann das Marktverhalten offensichtlich nicht mehr so gut beschreiben.

Die EPEX SPOT nimmt immer wieder Veränderungen an den Marktregelungen vor, um die Flexibilität zu erhöhen. Solche Änderungen, wie bspw. die Verlängerung der Handelszeit bis 5 Minuten vor Stromlieferung, fallen auch in den Zeitbereich der für diese Arbeit verwendeten Datengrundlage. Durch Mitarbeiter der EWE Trading GmbH wurde aber bestätigt, dass diese Verlängerung bislang keine großen Auswirkungen auf den Markt gehabt hat. Es könnten aber neue Marktveränderungen eintreten, die größere Auswirkungen haben können. Für das zweite Quartal 2018 ist im Rahmen des *Cross-Border Intraday Market Projects* (XBID) die Verbindung der Intraday-Märkte der verschiedenen europäischen Strombörsen zu einem einzelnen grenzübergreifenden Intraday-Markt geplant. So eine Veränderung könnte sich spürbar am Markt auswirken, sodass auch das Prognosemodell dadurch beeinflusst

wird. Doch auch durch den Ausbau der erneuerbaren Energien verändert sich der Markt konstant, da die Erzeugungsleistung der neuen Anlagen vermarktet werden muss.

Daher sollte bei einem praktischen Einsatz eines solchen Prognosemodells der Markt stets beobachtet werden. Darüber hinaus sollte das Modell immer mit den neuesten Daten trainiert werden, sodass es sich leichten Änderungen anpassen kann.

## Literatur


[17] *Gesetz für den Ausbau erneuerbarer Energien - Erneuerbare-Energien-Gesetz (EEG) §8 Anschluss*. 2017.

[50H] 50Hertz Transmission, Amprion, TenneT, TransnetBW. *Regelleistung.net. Internetplatform zur Vergabe von Regelleistung*. URL: https://www.regelleistung.net (besucht am 10. 05. 2018).

[50H13] 50Hertz Transmission, Amprion, TenneT, TransnetBW. *Netzentwicklungsplan Strom 2013*. 2013.

[Alt] Prof. Dr.-Ing. Helmut Alt. *Wie funktioniert die deutsche Strombörse?*

[BI04] Ling Bao und Stephen S. Intille. "Activity Recognition from User-Annotated Acceleration Data". In: (2004).

[Bre15] Jörg Bremer. "Constraint-Handling mit Supportvektor-Dekodern in der verteilten Optimierung". Diss. Carl von Ossietzky Universität Oldenburg, 2015.

[Buna] Bundesregierung. *Bundesregierung beschließt Ausstieg aus der Kernkraft bis 2022*. URL: https://www.bundesregierung.de/Content/DE/StatischeSeiten/Breg/Energiekonzept/05-kernenergie.html (besucht am 10. 05. 2018).

[Bunb] Bundesregierung. *Mobilität der Zukunft – Neue Kraftstoffe und Antriebe*. URL: http://www.bundesregierung.de/Webs/Breg/DE/Themen/Energiewende/Mobilitaet/mobilitaet_zukunft/_node.html (besucht am 10. 05. 2018).

[Bun11] Bundesregierung. *Kernkraftwerke kommen auf den Prüfstand*. 2011. URL: https://web.archive.org/web/20110319060816/http://www.bundesregierung.de/Content/DE/Artikel/2011/03/2011-03-15-bund-laender-kkw-pruefungen.html (besucht am 10. 05. 2018).

[Bun12] Bundesnetzagentur. *Modell zur Berechnung des regelzonenübergreifenden einheitlichen Bilanz-ausgleichsenergiepreises (reBAP) unter Beachtung des Beschlusses BK6-12-024 der Bundesnetzagentur vom 25.10.2012*. 2012.

[CIA] Central Intelligence Agency (CIA). *The World Factbook*. URL: https://www.cia.gov/library/publications/the-world-factbook/rankorder/2233rank.html (besucht am 03. 05. 2018).

[Dan16] Shih-Chii Liu Daniel Neil Michael Pfeiffer. "Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences". In: (2016).

[den14] Deutsche Energie Agentur GmbH (dena). *dena-Studie Systemdienstleistungen 2030. Sicherheit und Zuverlässigkeit einer Stromversorgung mit hohem Anteil erneuerbarer Energien*. 2014.

[Deu08] Wissenschaftliche Dienste des Deutschen Bundestages. *Fragen zur Preisbildung an der Leipziger Strombörse (EEX)*. 2008.

[Eck14] Andreas Eckner. "A Framework for the Analysis of Unevenly Spaced Time Series Data". In: (2014).

[EEX18a] European Energy Exchange (EEX). *Börsenordnung der EEX*. 2018.

[EEX18b] European Energy Exchange (EEX). *EEX: Deutliche Volumensteigerung in Strom- und Emissionsmärkten. Handelsergebnisse im Februar 2018*. 2018.

[EEX18c] European Energy Exchange (EEX). *Kontraktspezifikationen*. 2018.

[EPE16] EPEX SPOT, Nord Pool, OMIE, OPCOM, GME, OTE, TGE. *EUPHEMIA Public Description. PCR Market Coupling Algorithm*. 2016.

[EPE17] EPEX SPOT. *EPEX SPOT website and ftp server files specifications*. 2017.

[EPE18] EPEX SPOT. *XBID: Cross-Border Intraday Market Project*. 2018. URL: https://www.epexspot.com/en/market-coupling/xbid_cross_border_intraday_market_project (besucht am 05. 10. 2018).

[GBC16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[Hay09] S.S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 2009. URL: https://books.google.de/books?id=K7P36lKzI%5C_QC.

[HTF09] Trevor Hastie, Robert Tibshirani und Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[Kar02] Dietmar Grichnik und Karin Vortmeyer. *Ökonomische Analyse des Energiehandels am Beispiel der European Energy Exchange*. 2002.

[KB14] Diederik P. Kingma und Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (22. Dez. 2014).

[Kraa] Next Kraftwerke. *Regelenergie*. URL: https://www.next-kraftwerke.de/unternehmen/presse/grafiken (besucht am 03. 05. 2018).

[Krab] Next Kraftwerke. *Übertragungsnetzbetrieb*. URL: https://www.next-kraftwerke.de/unternehmen/presse/grafiken (besucht am 03. 05. 2018).

[KV16] Aymen Salah-Abou-El-Enien Karsten Neuhoff Nolan Ritter und Philippe Vassilopoulos. *Intraday Markets for Power: Discretizing the Continuous Trading*. University of Cambridge, 2016.

[Net07] Verband der Netzbetreiber. *TransmissionCode 2007. Netz- und Systemregeln der deutschen Übertragungsnetzbetreiber*. 2007.

[Pan08] Rina Panigrahy. *An Improved Algorithm Finding Nearest Neighbor Using Kd-trees*. Microsoft Research, 2008.

[RB93] Martin Riedmiller und Heinrich Braun. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm". In: *IEEE International Conference on Neural Networks*. 1993, S. 586–591.

[SC08] Malcolm Slaney und Michael Casey. "Locality-Sensitive Hashing for Finding Nearest Neighbors". In: *IEEE Signal Processing Magazine* (2008).

[SPOa] European Power Exchange (EPEX SPOT). *Kontinuierlicher Intraday-Handel*. URL: https://www.epexspot.com/de/produkte/intradaycontinuous/intraday_vorlaufzeit (besucht am 03. 05. 2018).

[SPOb] European Power Exchange (EPEX SPOT). *PCR: Price Coupling of Regions*. URL: http://www.epexspot.com/de/Marktkopplung/PCR_Price_Coupling_of_Regions (besucht am 03. 05. 2018).

[SPO17a] European Power Exchange (EPEX SPOT). *EPEX SPOT Operational Rules*. 2017.

[SPO17b] European Power Exchange (EPEX SPOT). *Exchange Council approves the introduction of 15-minute contracts on the Belgian and Dutch market*. 2017.

[SS98] J. Smola und B. Schölkopf. *A Tutorial on Support Vector Regression*. 1998.

[Ste09] Dan Steinberg. "CART: Classification and Regression Trees". In: *The Top Ten Algorithms in Data Mining* (2009).

[Umw12] Umweltbundesamt. *Herkunftsnachweisregister (HKNR). Häufig gestellte Fragen*. 2012.

[Umw15] Umweltbundesamt. *Marktdaten: Bereich Mobilität. Motorisierter Individualverkehr*. 2015. URL: http://www.umweltbundesamt.de/daten/private-haushalte-konsum/gruene-produkte-marktzahlen/marktdaten-bereich-mobilitaet (besucht am 10. 05. 2018).

[Umw17a] Umweltbundesamt. *Kraftstoffverbrauch im Straßenverkehr in Millionen Liter*. 2017. URL: https://www.umweltbundesamt.de/sites/default/files/medien/384/bilder/dateien/3_tab_kraftstoffverbrauch-strv-sektor_2017-04-06.pdf (besucht am 10. 05. 2018).

[Umw17b] Umweltbundesamt. *Kraftwerke und Photovoltaikleistung in Deutschland*. 2017. URL: https://www.umweltbundesamt.de/bild/kraftwerke-photovoltaikleistung-in-deutschland (besucht am 03. 05. 2018).

[Umw17c] Umweltbundesamt. *Kraftwerke und Windleistung in Deutschland*. 2017. URL: https://www.umweltbundesamt.de/bild/kraftwerke-windleistung-in-deutschland (besucht am 03. 05. 2018).

[Umw18] Umweltbundesamt. *Stromerzeugung erneuerbar und konventionell*. 2018. URL: https://www.umweltbundesamt.de/daten/energie/stromerzeugung-erneuerbar-konventionell (besucht am 03. 05. 2018).

[Vap13] Vladimir Vapnik. *Nature of Statistical Learning Theory*. Springer New York, 29. Juni 2013. URL: http://www.ebook.de/de/product/25445838/vladimir_vapnik_nature_of_statistical_learning_theory.html.

[VL63] V. Vapnik und A. Lerner. "Pattern recognition using generalized portrait method". In: *Automation and Remote Control* 24 (1963), S. 774–780.

[Win] Bundesverband WindEnergie. *Netze*. URL: https://www.wind-energie.de/themen/netze (besucht am 03. 05. 2018).

[Win15] Bundesverband WindEnergie. *Weiterentwicklung des Einspeise-Managements. Bewertung von Ansätzen*. Ecofys GmbH, 2015.

[Wis15] Carsten Wissing. "Marktbasiertes Redispatch mit Flexibilitäten von Netznutzern für das Verteilnetz". Diss. Carl von Ossietzky Universität Oldenburg, 2015.

[Zei13] Frankfurter Allgemeine Zeitung. *Ein Windpark, der Strom verbraucht und Diesel frisst*. 2013. URL: http://www.faz.net/aktuell/wirtschaft/wirtschaftspolitik/erneuerbare-energien-ein-windpark-der-strom-verbraucht-und-diesel-frisst-12475399.html (besucht am 03. 05. 2018).

[Zim17] Henning Zimmer. "Regeldynamik konventioneller Kraftwerke im Kontext veränderter Erzeugungsstrukturen". Diss. Technischen Universität Darmstadt, 2017.

# A Anhang

## A.1 Untersuchungen der Transaktionslogs

### A.1.1 Grenzübergreifender Handel

Tabelle A.1: Anzahl der Transaktionen, sowie deren Anteil an allen ausgeführten Transaktionen, unterteilt nach den Marktgebieten, zwischen denen diese Transaktionen stattgefunden haben.

| Market_Area_Buy | Market_Area_Sell | count | fraction | fraction_accumulated |
|---|---|---|---|---|
| DE | DE | 18952269 | 0.876824 | 0.876824 |
| DE | AT | 817693 | 0.037830 | 0.914655 |
| DE | CH | 331526 | 0.015338 | 0.929993 |
| DE | FR | 308993 | 0.014296 | 0.944288 |
| AT | DE | 279658 | 0.012938 | 0.957227 |
| CH | DE | 255199 | 0.011807 | 0.969033 |
| FR | DE | 245116 | 0.011340 | 0.980374 |
| AT | AT | 239645 | 0.011087 | 0.991461 |
| DE | BE | 37446 | 0.001732 | 0.993193 |
| BE | DE | 36064 | 0.001668 | 0.994862 |
| DE | NL | 32540 | 0.001505 | 0.996367 |
| NL | DE | 32500 | 0.001504 | 0.997871 |
| FR | AT | 14448 | 0.000668 | 0.998539 |
| CH | AT | 10886 | 0.000504 | 0.999043 |
| AT | FR | 9276 | 0.000429 | 0.999472 |
| AT | CH | 5923 | 0.000274 | 0.999746 |
| NL | AT | 1852 | 0.000086 | 0.999832 |
| BE | AT | 1806 | 0.000084 | 0.999915 |
| AT | BE | 943 | 0.000044 | 0.999959 |
| AT | NL | 886 | 0.000041 | 1.000000 |

### A.1.2 Anteil der Stunden- und Viertelstundenprodukte

Tabelle A.2: Anzahl der Transaktionen, sowie deren Anteil an allen ausgeführten Transaktionen, unterteilt nach den verschiedenen Stunden- und Viertelstundenprodukten. Block- und Halbstundenprodukte werden nicht dargestellt. Ihr Anteil an allen ausgeführten Transaktionen berechnet sich zu $1 - 0.998039 = 0.001961$.

| Hour_from | Hour_to | count | fraction | fraction_accumulated |
|---|---|---|---|---|
| 14 | 14 | 575281 | 0.026615 | 0.026615 |
| 15 | 15 | 570685 | 0.026403 | 0.053018 |
| 13 | 13 | 564960 | 0.026138 | 0.079156 |

| | | | | |
|---|---|---|---|---|
| 16 | 16 | 564608 | 0.026122 | 0.105277 |
| 17 | 17 | 546341 | 0.025276 | 0.130554 |
| 12 | 12 | 531951 | 0.024611 | 0.155164 |
| 18 | 18 | 530690 | 0.024552 | 0.179717 |
| 19 | 19 | 529683 | 0.024506 | 0.204222 |
| 20 | 20 | 518440 | 0.023986 | 0.228208 |
| 11 | 11 | 485755 | 0.022473 | 0.250681 |
| 21 | 21 | 479274 | 0.022174 | 0.272855 |
| 22 | 22 | 442223 | 0.020459 | 0.293314 |
| 23 | 23 | 429689 | 0.019880 | 0.313194 |
| 10 | 10 | 426684 | 0.019740 | 0.332934 |
| 24 | 24 | 416413 | 0.019265 | 0.352200 |
| 9 | 9 | 381248 | 0.017638 | 0.369838 |
| 8 | 8 | 341488 | 0.015799 | 0.385637 |
| 5 | 5 | 306599 | 0.014185 | 0.399822 |
| 7 | 7 | 305912 | 0.014153 | 0.413975 |
| 4 | 4 | 303603 | 0.014046 | 0.428021 |
| 6 | 6 | 296296 | 0.013708 | 0.441729 |
| 3 | 3 | 291353 | 0.013479 | 0.455208 |
| 2 | 2 | 270431 | 0.012511 | 0.467720 |
| 1 | 1 | 259001 | 0.011983 | 0.479702 |
| 18qh4 | 18qh4 | 174255 | 0.008062 | 0.487764 |
| 17qh4 | 17qh4 | 172341 | 0.007973 | 0.495738 |
| 10qh4 | 10qh4 | 171367 | 0.007928 | 0.503666 |
| 11qh4 | 11qh4 | 170751 | 0.007900 | 0.511566 |
| 14qh4 | 14qh4 | 167936 | 0.007770 | 0.519335 |
| 12qh4 | 12qh4 | 167337 | 0.007742 | 0.527077 |
| 16qh4 | 16qh4 | 165779 | 0.007670 | 0.534747 |
| 15qh4 | 15qh4 | 161386 | 0.007467 | 0.542213 |
| 13qh4 | 13qh4 | 161132 | 0.007455 | 0.549668 |
| 9qh4 | 9qh4 | 160186 | 0.007411 | 0.557079 |
| 18qh1 | 18qh1 | 158407 | 0.007329 | 0.564408 |
| 19qh4 | 19qh4 | 154401 | 0.007143 | 0.571551 |
| 17qh1 | 17qh1 | 150456 | 0.006961 | 0.578512 |
| 16qh1 | 16qh1 | 143606 | 0.006644 | 0.585156 |
| 19qh1 | 19qh1 | 141577 | 0.006550 | 0.591706 |
| 12qh3 | 12qh3 | 140626 | 0.006506 | 0.598212 |
| 18qh3 | 18qh3 | 140395 | 0.006495 | 0.604707 |
| 17qh3 | 17qh3 | 140285 | 0.006490 | 0.611197 |

| | | | | |
|---|---|---|---|---|
| 11qh3 | 11qh3 | 139737 | 0.006465 | 0.617662 |
| 16qh3 | 16qh3 | 139681 | 0.006462 | 0.624125 |
| 14qh3 | 14qh3 | 139242 | 0.006442 | 0.630567 |
| 8qh4 | 8qh4 | 137670 | 0.006369 | 0.636936 |
| 15qh3 | 15qh3 | 137252 | 0.006350 | 0.643286 |
| 13qh3 | 13qh3 | 136533 | 0.006317 | 0.649603 |
| 20qh4 | 20qh4 | 135777 | 0.006282 | 0.655884 |
| 10qh3 | 10qh3 | 135267 | 0.006258 | 0.662142 |
| 7qh4 | 7qh4 | 134946 | 0.006243 | 0.668386 |
| 14qh1 | 14qh1 | 134338 | 0.006215 | 0.674601 |
| 15qh1 | 15qh1 | 133255 | 0.006165 | 0.680766 |
| 23qh4 | 23qh4 | 132284 | 0.006120 | 0.686886 |
| 21qh4 | 21qh4 | 132113 | 0.006112 | 0.692998 |
| 18qh2 | 18qh2 | 131665 | 0.006091 | 0.699090 |
| 17qh2 | 17qh2 | 130706 | 0.006047 | 0.705137 |
| 16qh2 | 16qh2 | 130317 | 0.006029 | 0.711166 |
| 10qh1 | 10qh1 | 129545 | 0.005993 | 0.717159 |
| 11qh1 | 11qh1 | 128907 | 0.005964 | 0.723123 |
| 12qh1 | 12qh1 | 128073 | 0.005925 | 0.729048 |
| 14qh2 | 14qh2 | 127120 | 0.005881 | 0.734930 |
| 24qh4 | 24qh4 | 126907 | 0.005871 | 0.740801 |
| 19qh3 | 19qh3 | 126667 | 0.005860 | 0.746661 |
| 15qh2 | 15qh2 | 125128 | 0.005789 | 0.752450 |
| 13qh1 | 13qh1 | 124817 | 0.005775 | 0.758225 |
| 9qh3 | 9qh3 | 123051 | 0.005693 | 0.763918 |
| 22qh4 | 22qh4 | 123035 | 0.005692 | 0.769610 |
| 9qh1 | 9qh1 | 122839 | 0.005683 | 0.775293 |
| 11qh2 | 11qh2 | 121862 | 0.005638 | 0.780931 |
| 12qh2 | 12qh2 | 121862 | 0.005638 | 0.786569 |
| 20qh1 | 20qh1 | 121535 | 0.005623 | 0.792192 |
| 19qh2 | 19qh2 | 120057 | 0.005554 | 0.797746 |
| 7qh1 | 7qh1 | 120055 | 0.005554 | 0.803300 |
| 13qh2 | 13qh2 | 119422 | 0.005525 | 0.808826 |
| 10qh2 | 10qh2 | 118100 | 0.005464 | 0.814289 |
| 21qh1 | 21qh1 | 116845 | 0.005406 | 0.819695 |
| 8qh1 | 8qh1 | 113919 | 0.005270 | 0.824966 |
| 20qh3 | 20qh3 | 113723 | 0.005261 | 0.830227 |
| 24qh1 | 24qh1 | 112175 | 0.005190 | 0.835417 |
| 23qh1 | 23qh1 | 109764 | 0.005078 | 0.840495 |

| | | | | |
|---|---|---|---|---|
| 6qh4 | 6qh4 | 109468 | 0.005065 | 0.845560 |
| 21qh3 | 21qh3 | 109372 | 0.005060 | 0.850620 |
| 9qh2 | 9qh2 | 109220 | 0.005053 | 0.855673 |
| 20qh2 | 20qh2 | 106292 | 0.004918 | 0.860590 |
| 22qh1 | 22qh1 | 106267 | 0.004916 | 0.865507 |
| 8qh3 | 8qh3 | 105859 | 0.004898 | 0.870404 |
| 23qh3 | 23qh3 | 104658 | 0.004842 | 0.875246 |
| 7qh3 | 7qh3 | 104162 | 0.004819 | 0.880065 |
| 24qh3 | 24qh3 | 101593 | 0.004700 | 0.884765 |
| 21qh2 | 21qh2 | 101334 | 0.004688 | 0.889454 |
| 22qh3 | 22qh3 | 100738 | 0.004661 | 0.894114 |
| 23qh2 | 23qh2 | 98600 | 0.004562 | 0.898676 |
| 8qh2 | 8qh2 | 98517 | 0.004558 | 0.903234 |
| 7qh2 | 7qh2 | 97666 | 0.004519 | 0.907752 |
| 1qh4 | 1qh4 | 94729 | 0.004383 | 0.912135 |
| 22qh2 | 22qh2 | 94409 | 0.004368 | 0.916503 |
| 24qh2 | 24qh2 | 94300 | 0.004363 | 0.920866 |
| 6qh1 | 6qh1 | 92359 | 0.004273 | 0.925139 |
| 2qh4 | 2qh4 | 89686 | 0.004149 | 0.929288 |
| 5qh4 | 5qh4 | 86936 | 0.004022 | 0.933310 |
| 6qh3 | 6qh3 | 85112 | 0.003938 | 0.937248 |
| 1qh1 | 1qh1 | 83321 | 0.003855 | 0.941103 |
| 6qh2 | 6qh2 | 83122 | 0.003846 | 0.944948 |
| 4qh4 | 4qh4 | 80150 | 0.003708 | 0.948656 |
| 3qh4 | 3qh4 | 80058 | 0.003704 | 0.952360 |
| 1qh3 | 1qh3 | 78481 | 0.003631 | 0.955991 |
| 2qh3 | 2qh3 | 75749 | 0.003505 | 0.959496 |
| 2qh1 | 2qh1 | 74171 | 0.003432 | 0.962927 |
| 5qh1 | 5qh1 | 73378 | 0.003395 | 0.966322 |
| 1qh2 | 1qh2 | 72772 | 0.003367 | 0.969689 |
| 5qh3 | 5qh3 | 72331 | 0.003346 | 0.973035 |
| 3qh3 | 3qh3 | 69508 | 0.003216 | 0.976251 |
| 4qh3 | 4qh3 | 68864 | 0.003186 | 0.979437 |
| 2qh2 | 2qh2 | 68855 | 0.003186 | 0.982622 |
| 5qh2 | 5qh2 | 68643 | 0.003176 | 0.985798 |
| 3qh1 | 3qh1 | 67761 | 0.003135 | 0.988933 |
| 4qh1 | 4qh1 | 66711 | 0.003086 | 0.992020 |
| 4qh2 | 4qh2 | 65889 | 0.003048 | 0.995068 |
| 3qh2 | 3qh2 | 64213 | 0.002971 | 0.998039 |

## A.2 Topologien der neuronalen Netze

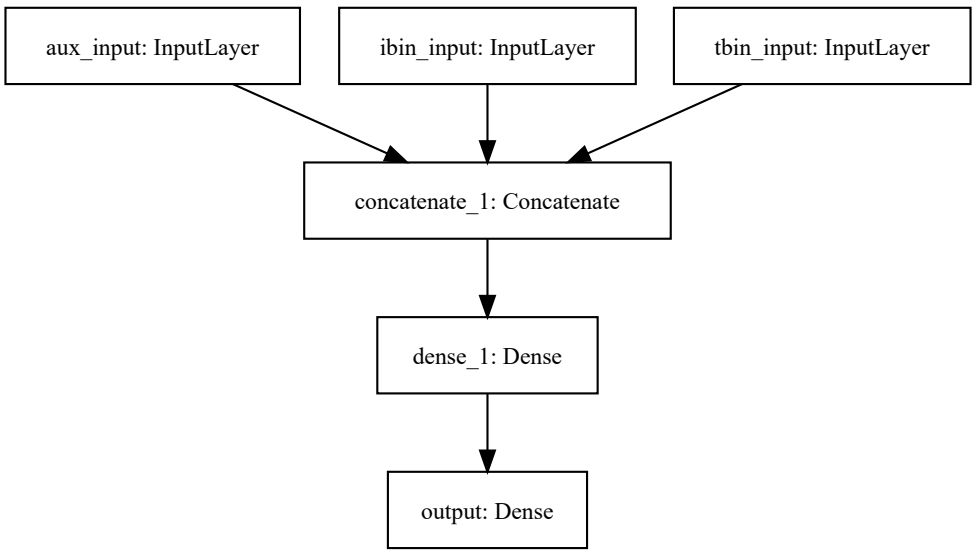



Abbildung A.1: MLP1: Feedforward Network

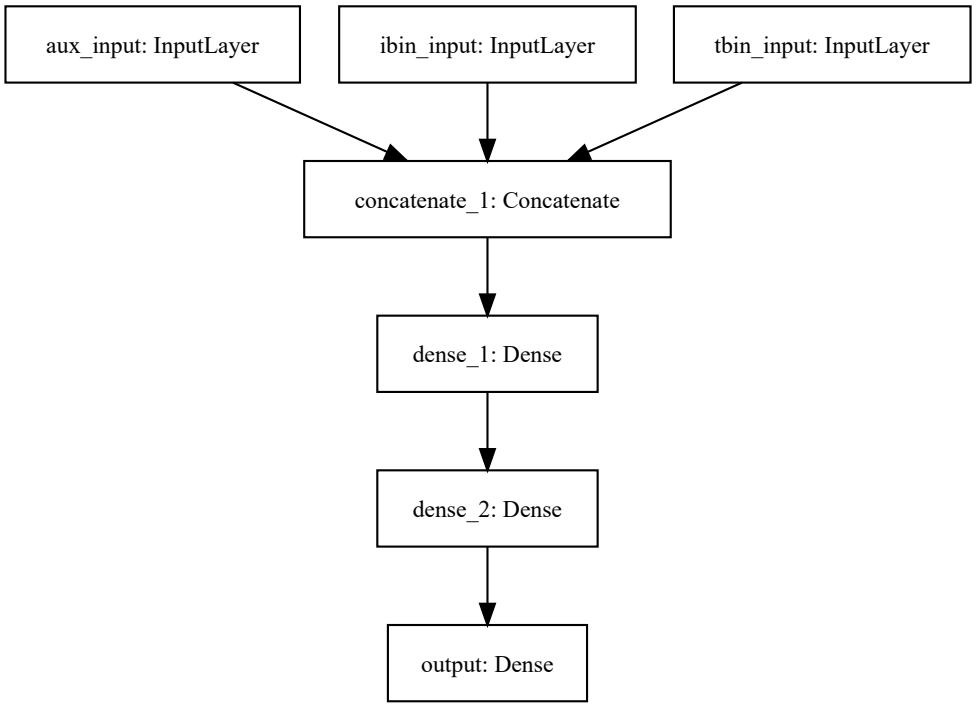



Abbildung A.2: MLP2: Feedforward Network

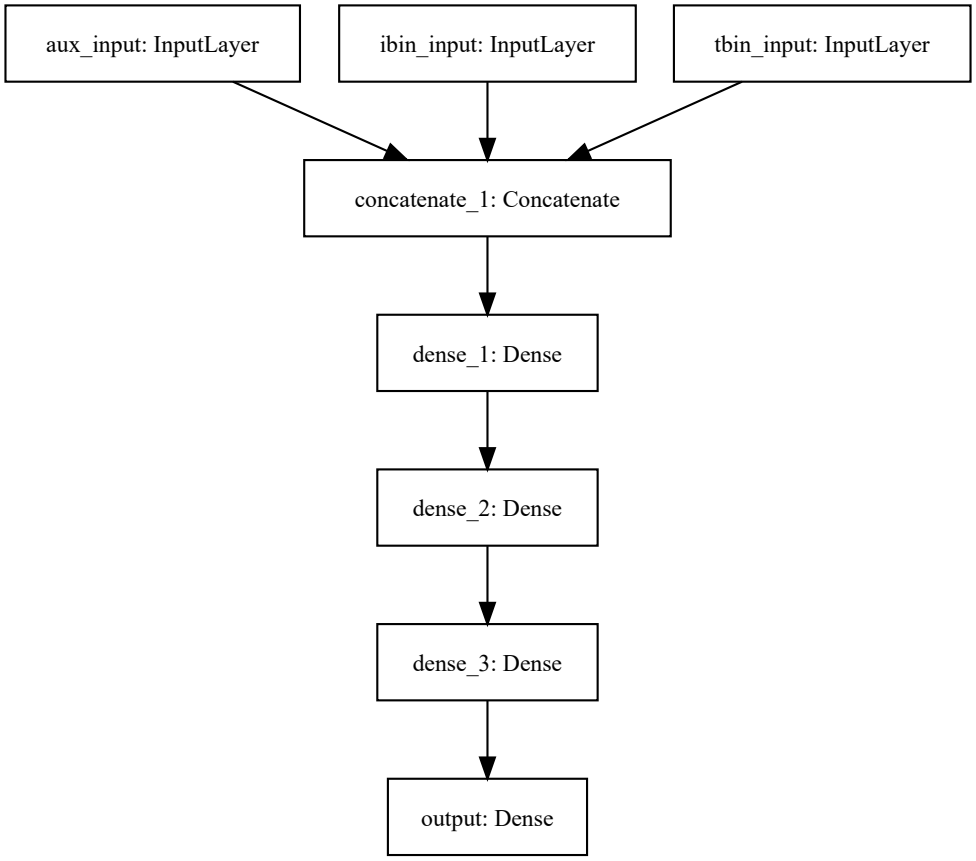



Abbildung A.3: MLP3: Feedforward Network

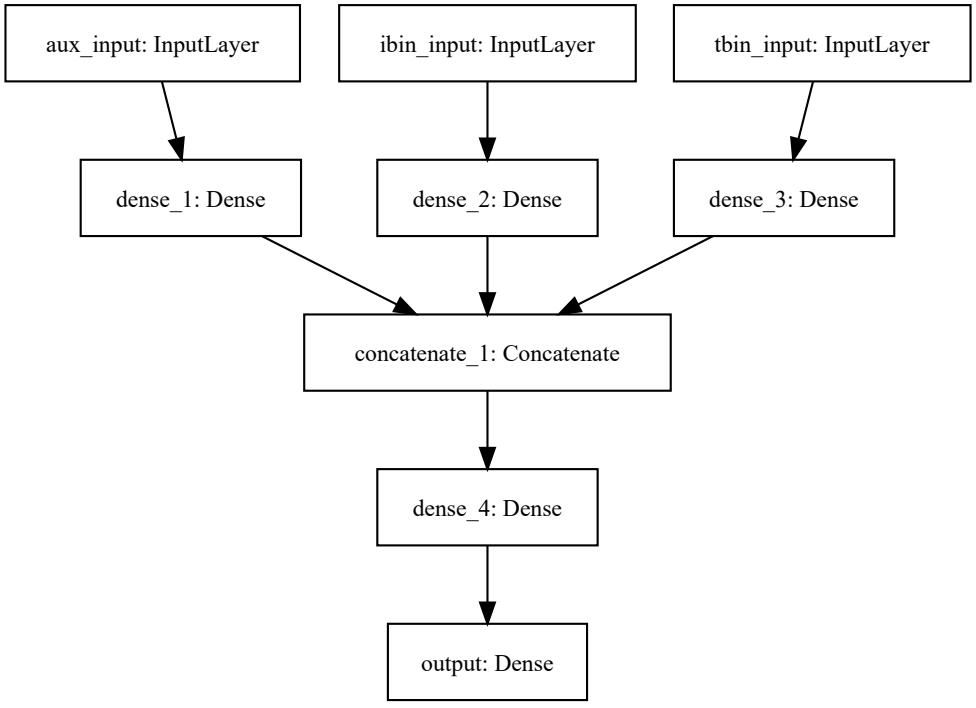



Abbildung A.4: MLP4: Feedforward Network

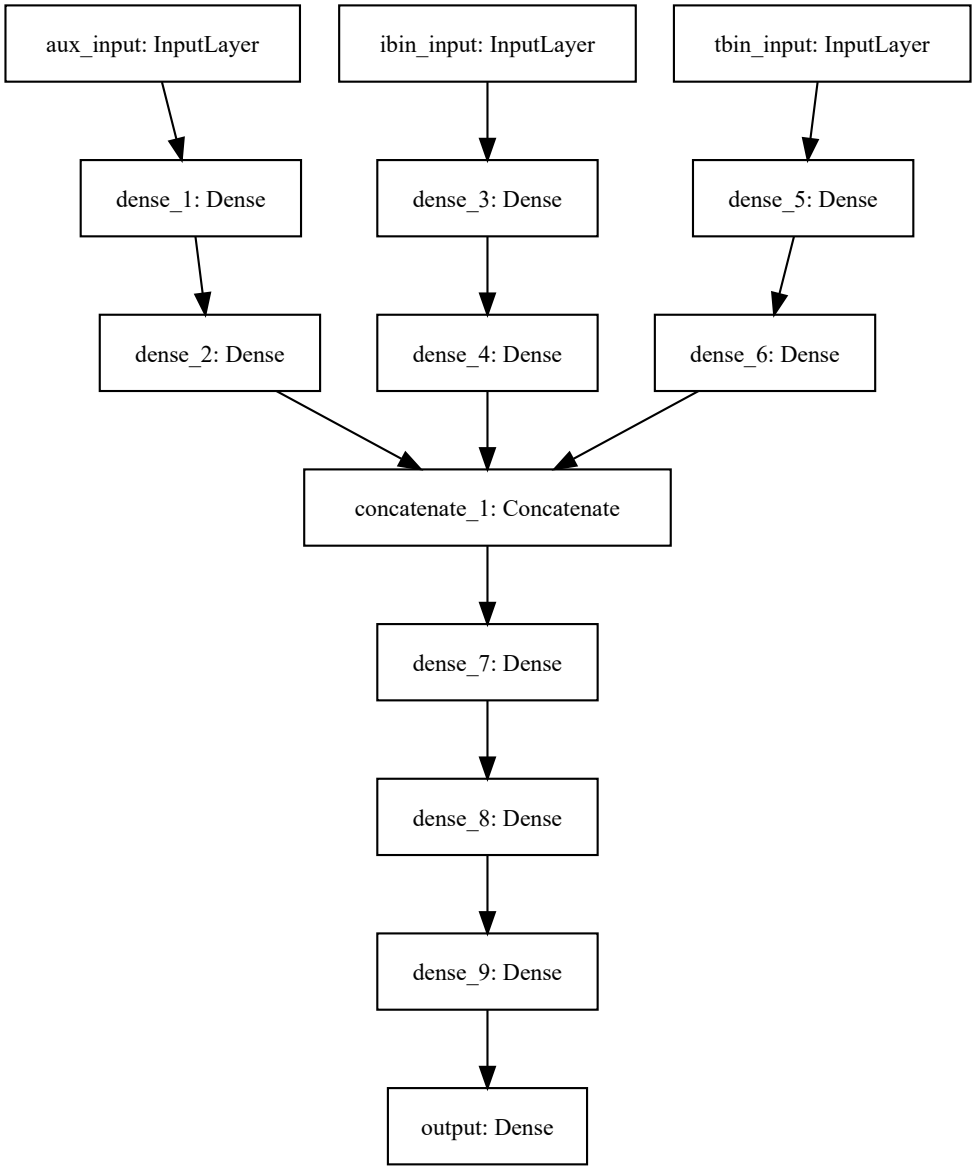



Abbildung A.5: MLP5: Feedforward Network

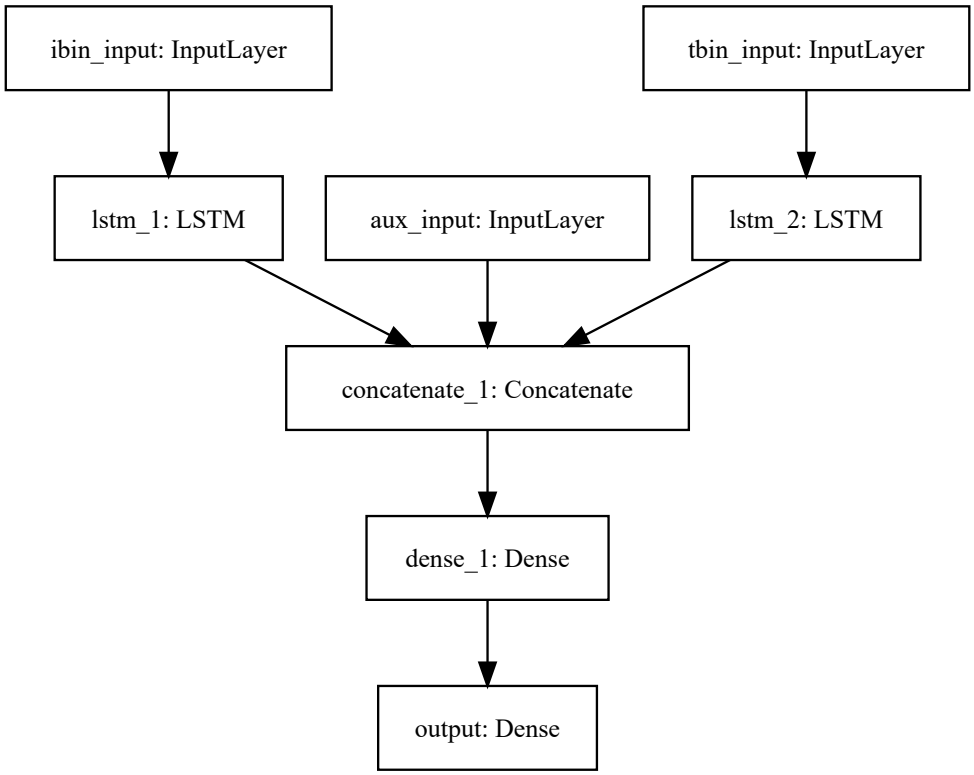



Abbildung A.6: LSTM1: Long sort-term memory-Network

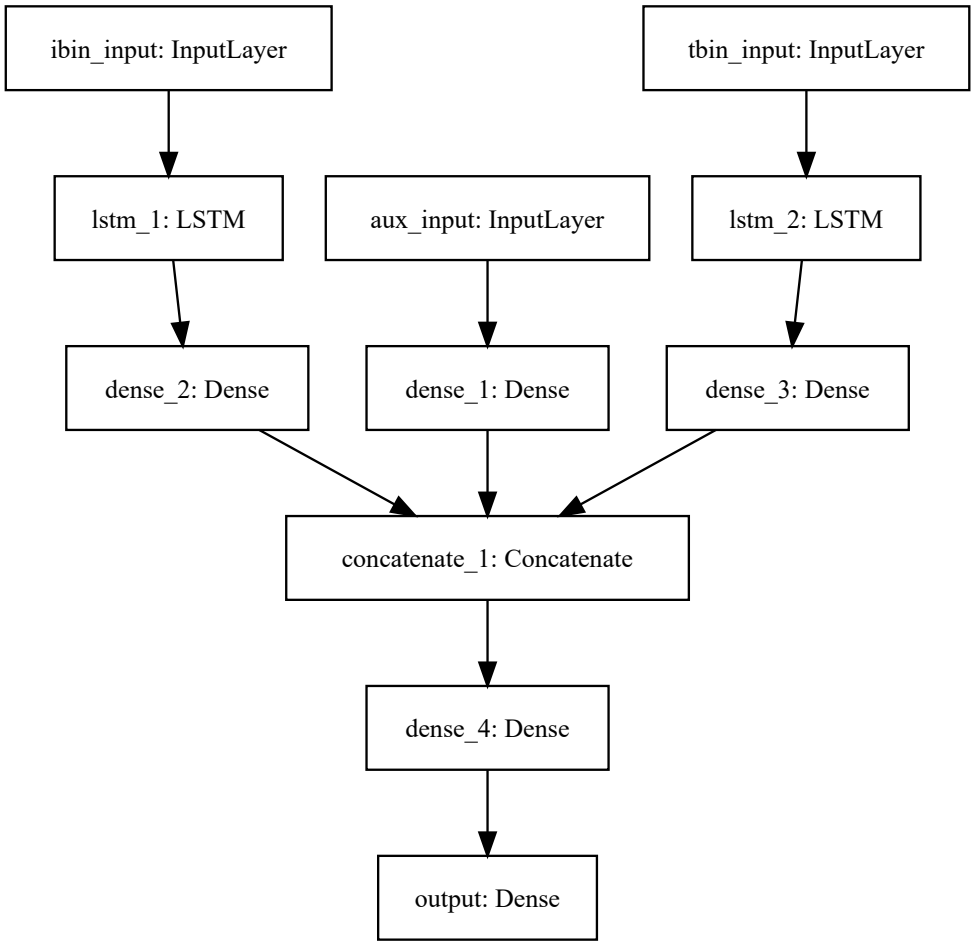



Abbildung A.7: LSTM2: Long sort-term memory-Network

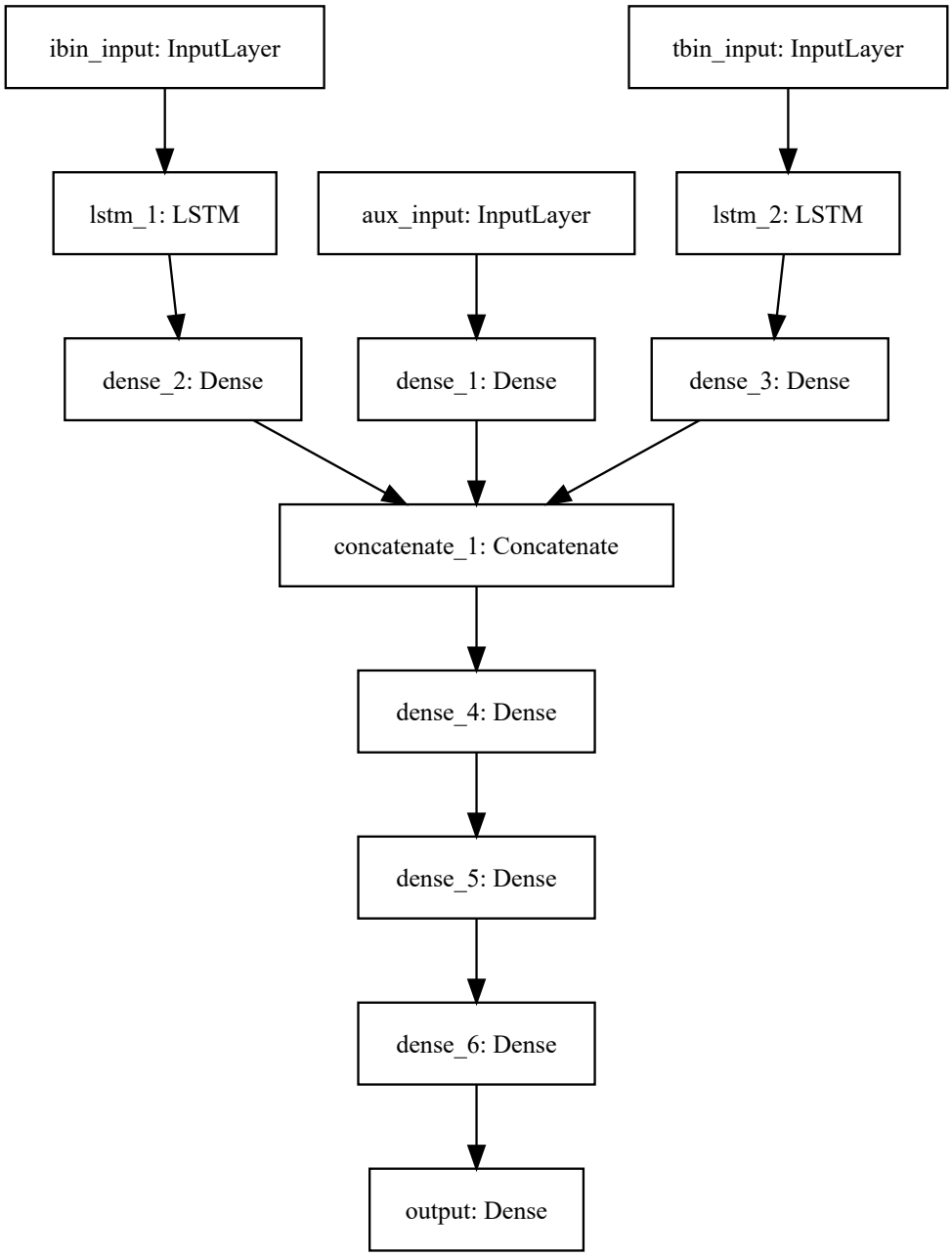



Abbildung A.8: LSTM3: Long sort-term memory-Network

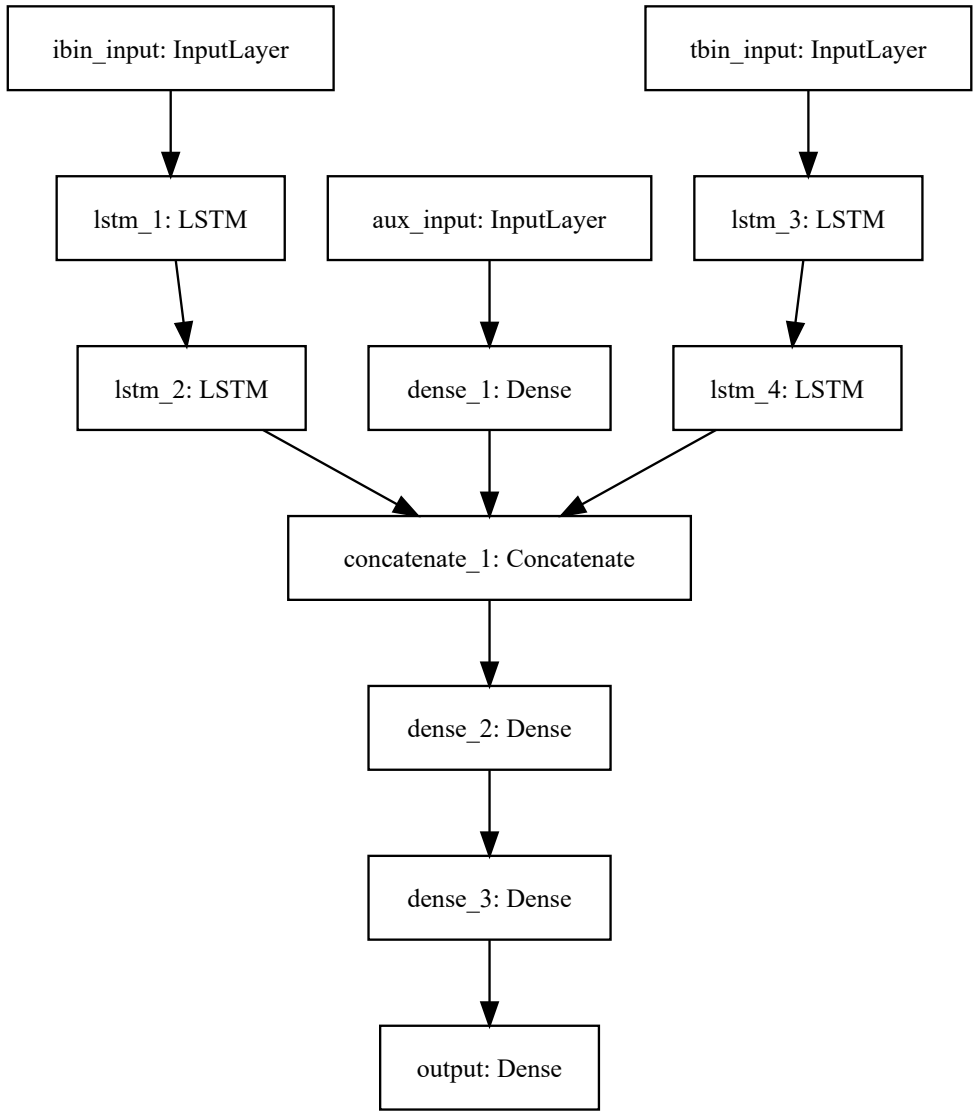



Abbildung A.9: LSTM4: Long sort-term memory-Network

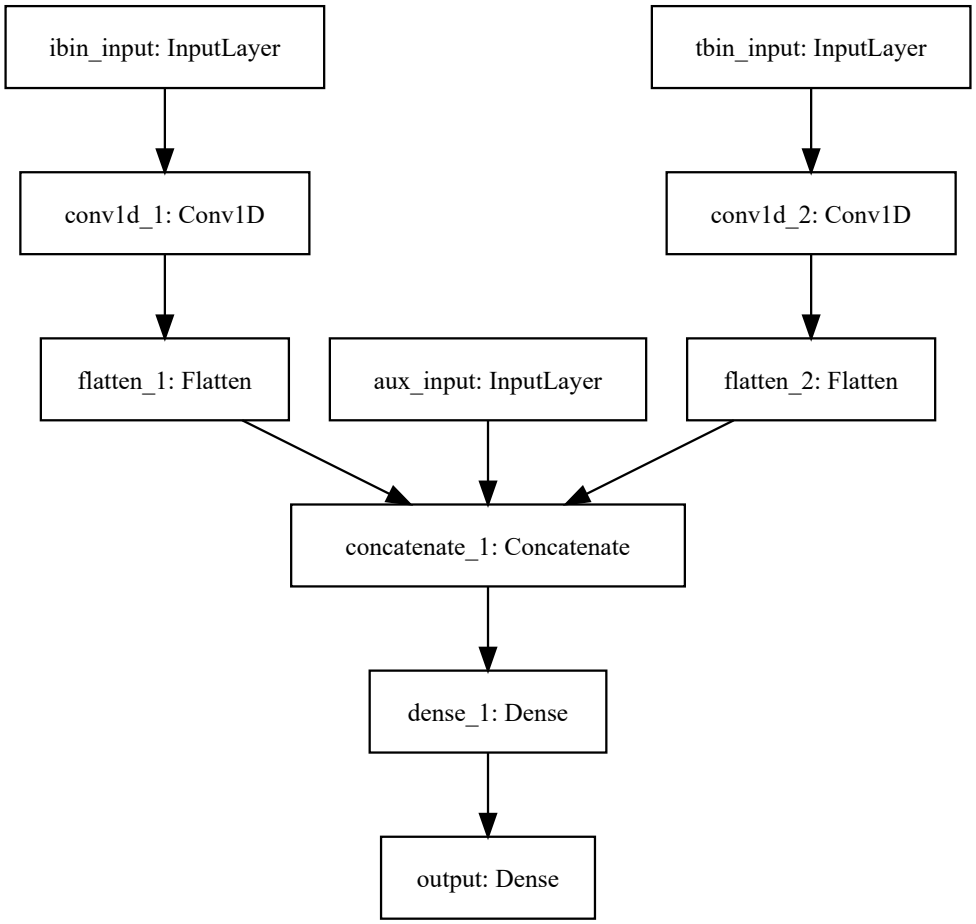



Abbildung A.10: CNN1: Convolutional Neural Network

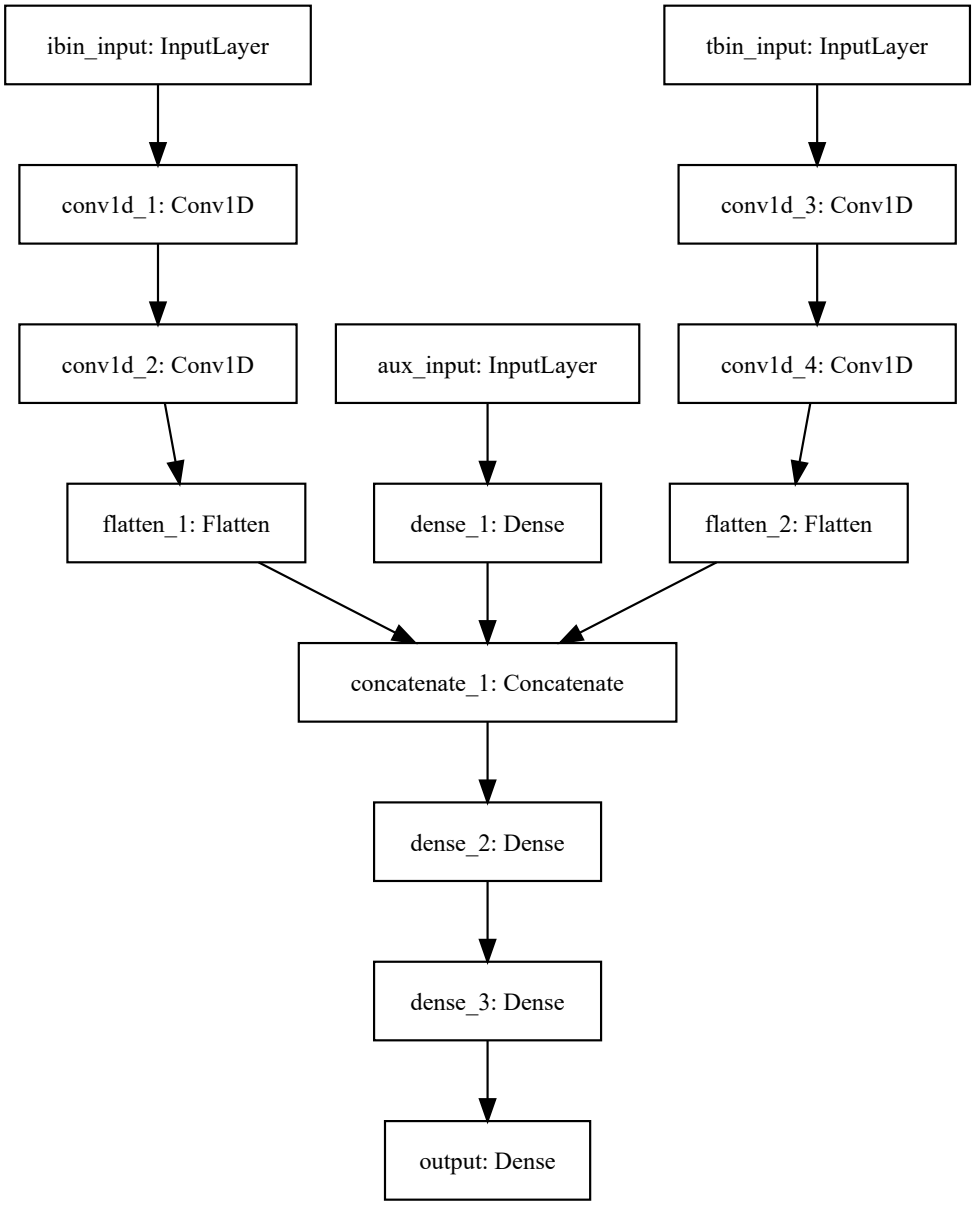



Abbildung A.11: CNN2: Convolutional Neural Network

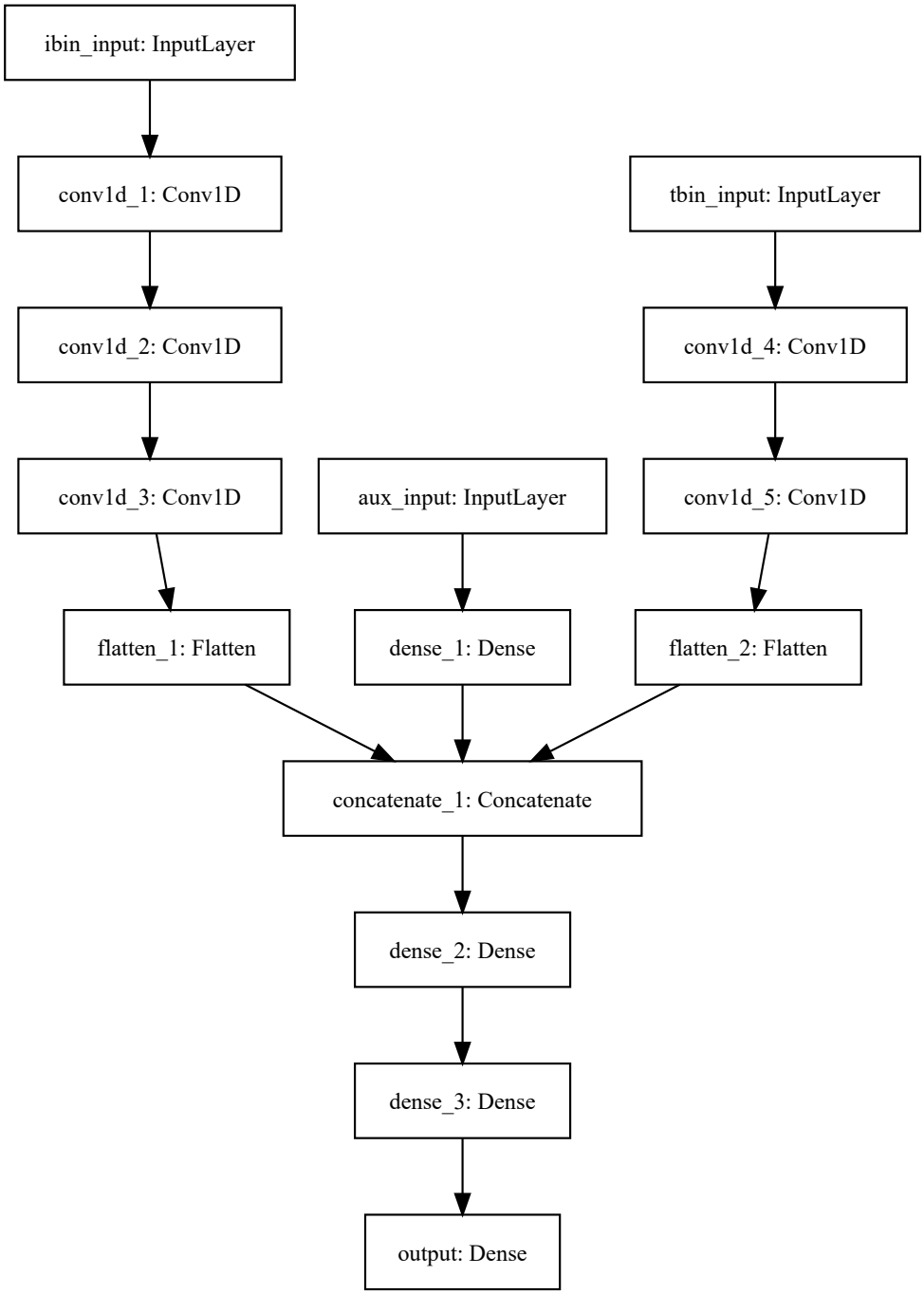



Abbildung A.12: CNN3: Convolutional Neural Network

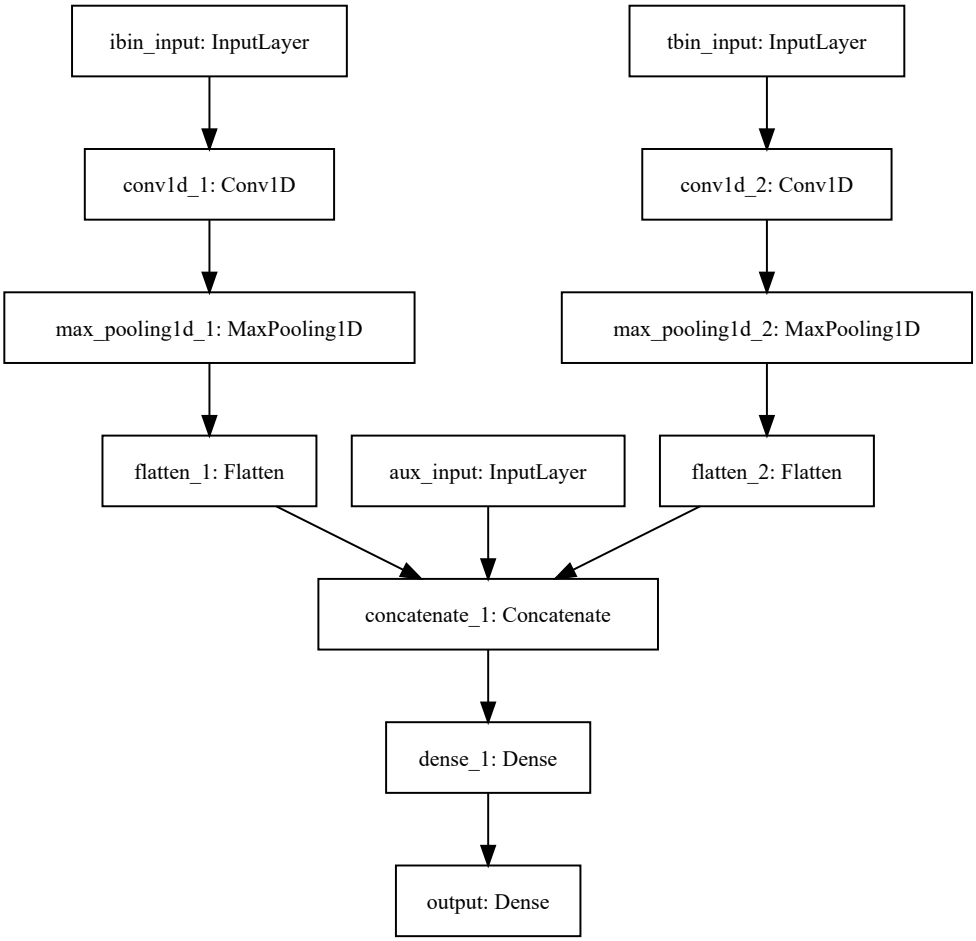



Abbildung A.13: CNN4: Convolutional Neural Network

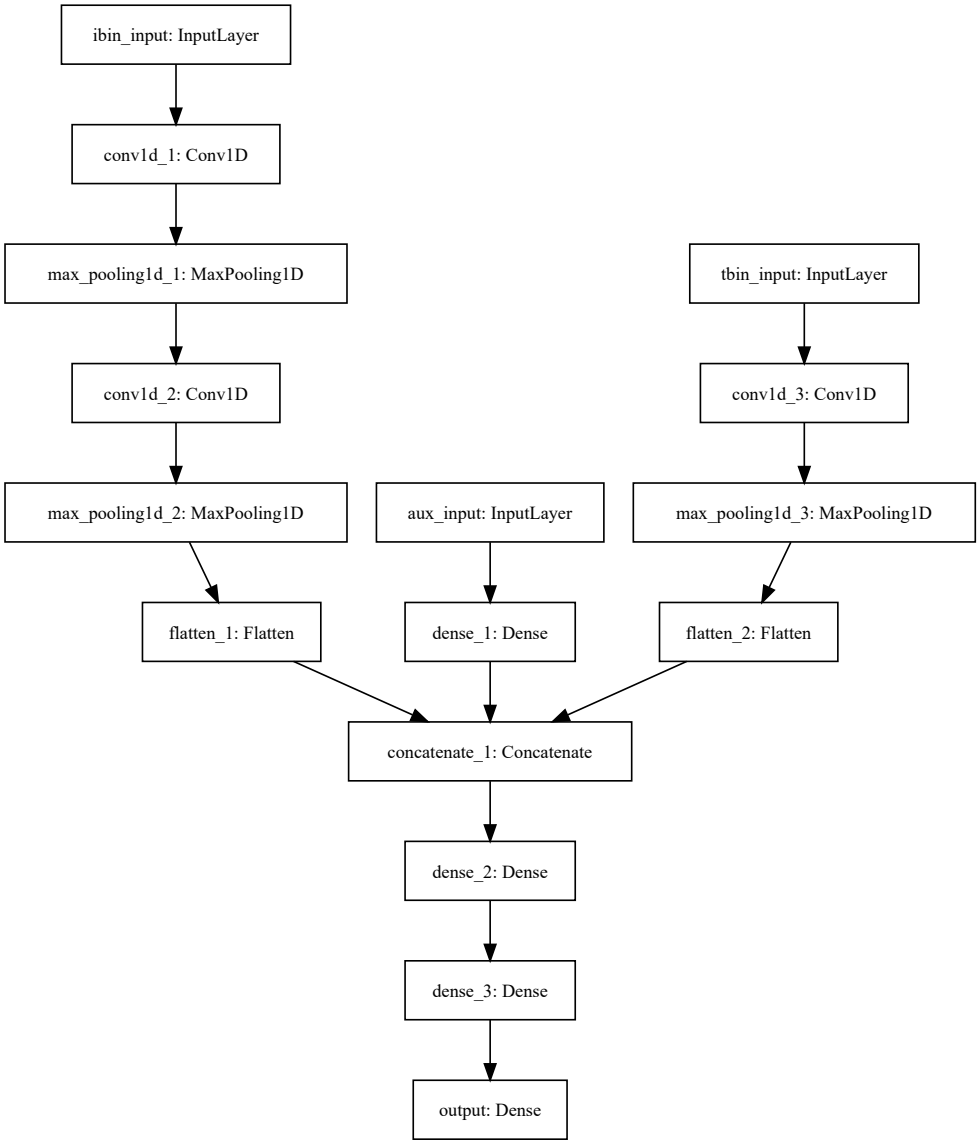



Abbildung A.14: CNN5: Convolutional Neural Network

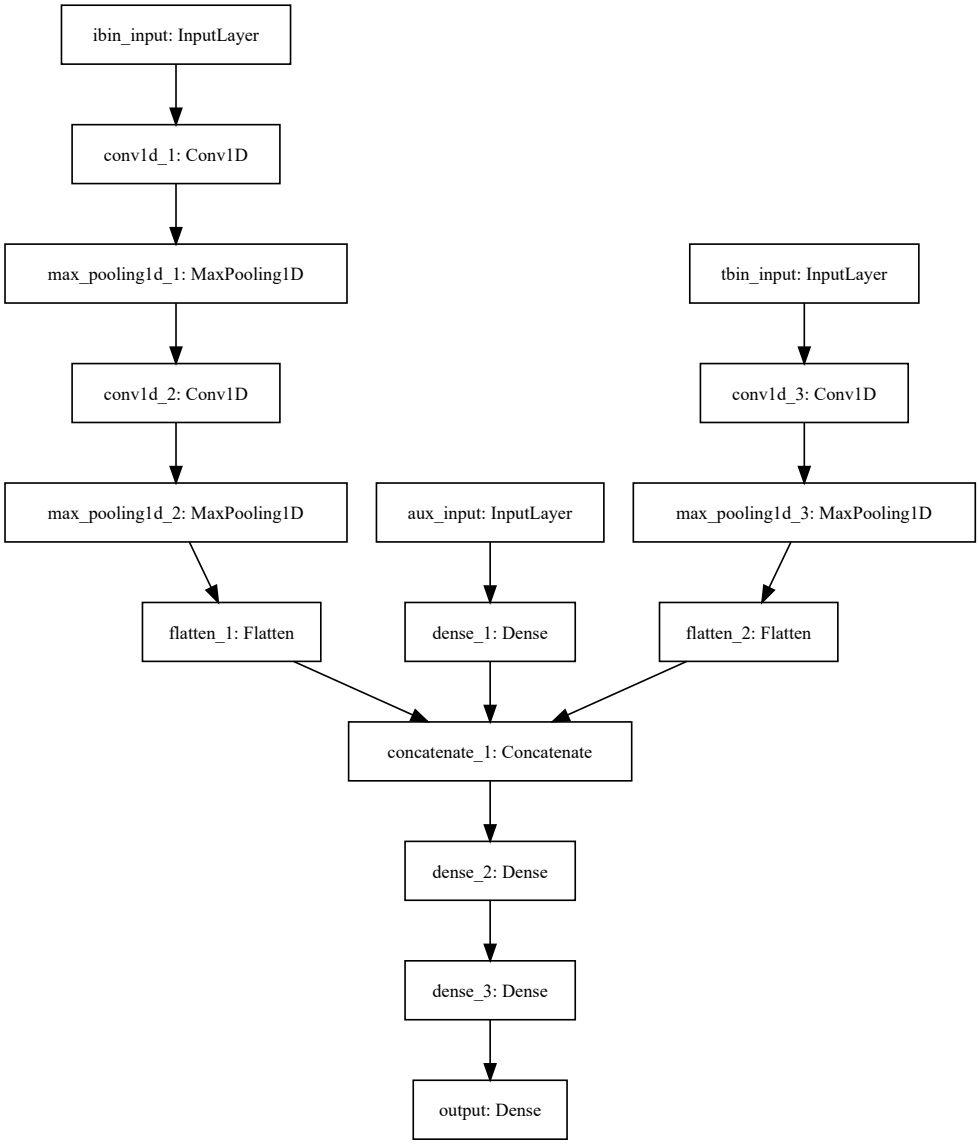



Abbildung A.15: CNN6: Convolutional Neural Network

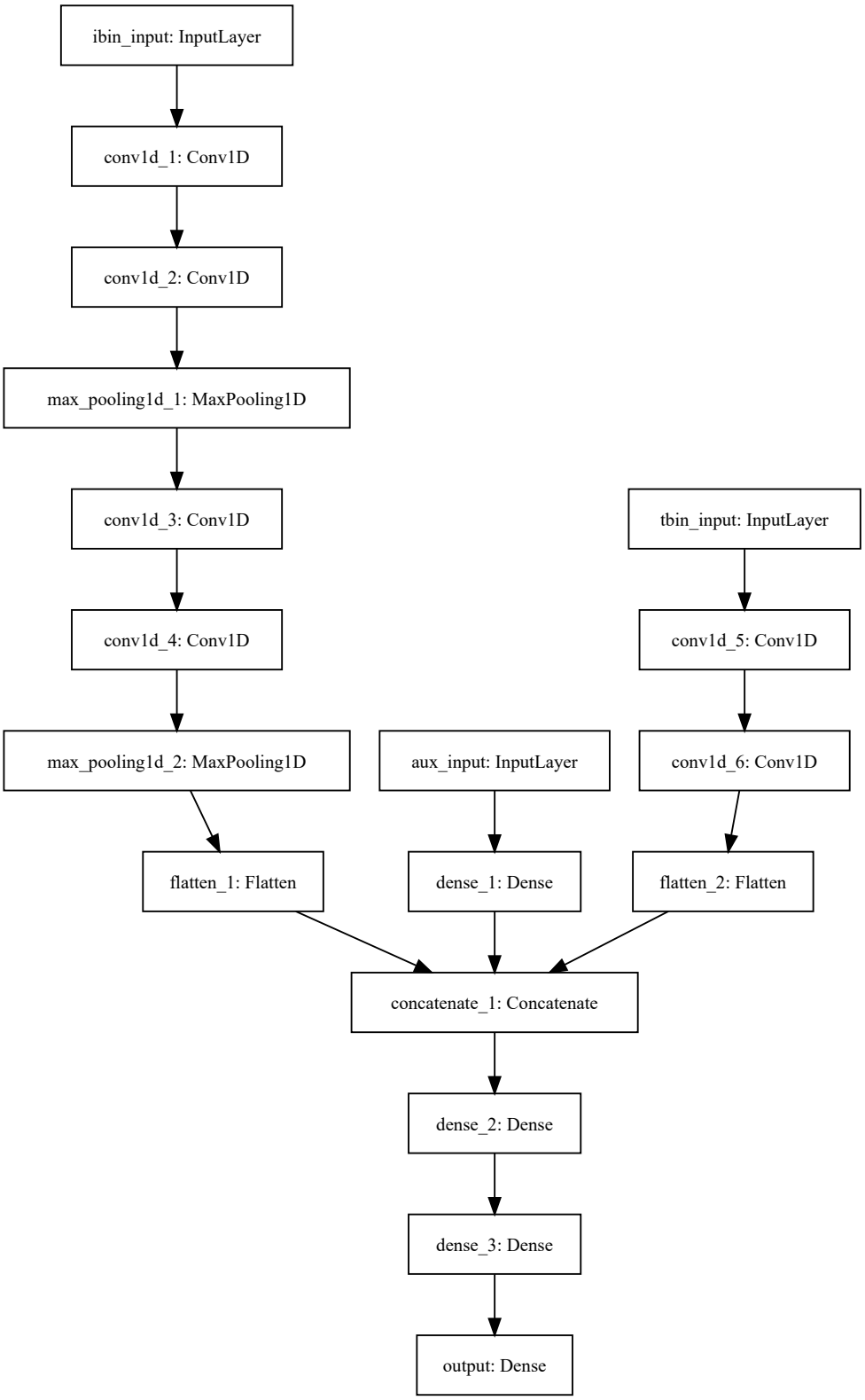



Abbildung A.16: CNN7: Convolutional Neural Network

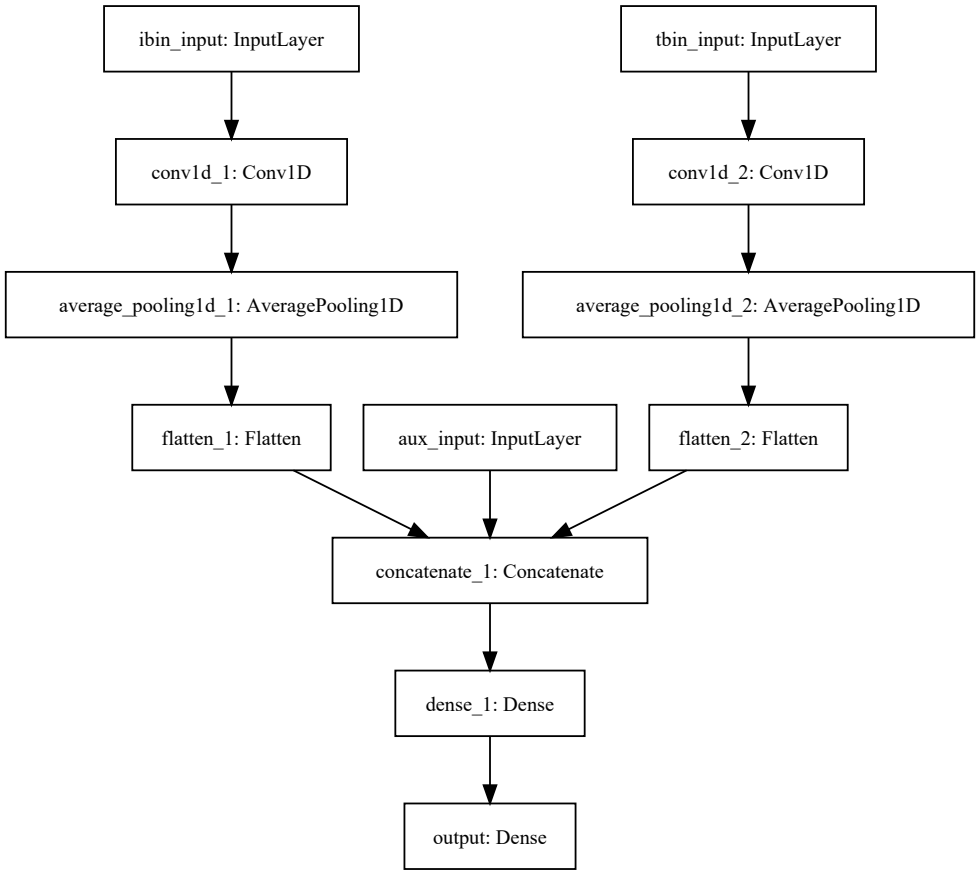



Abbildung A.17: CNN8: Convolutional Neural Network

## A.3 Ergebnisse der Modellparameteroptimierung

### A.3.1 Lineare Regression

Tabelle A.3: Alle 4 Ergebnisse der Modellparameteroptimierung für die Lineare Regression.

| look back resolution | $b = 0$ | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|
| | | Early | Mid | Late | Early | Mid | Late |
| 20min | Ja | 2.347 | 4.384 | 7.172 | 1.89 | 3.175 | 5.881 |
| 20min | Nein | 2.347 | 4.384 | 7.172 | 1.89 | 3.175 | 5.881 |
| 5min | Nein | 2.293 | 4.276 | 7.007 | 2.319 | 3.455 | 6.159 |
| 5min | Ja | 2.297 | 4.764 | 7.418 | 1.022e+12 | 1.447e+13 | 1,678e+14 |

### A.3.2 K-Nearest Neighbor Regression

Tabelle A.4: Alle 36 Ergebnisse der Modellparameteroptimierung für die k-Nearest Neighbor Regression.

| look back resolution | $k$ | weight | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|
| | | | Early | Mid | Late | Early | Mid | Late |
| 5min | 20 | distance | 0.0 | 0.0 | 0.0 | 5.419 | 5.984 | 7.633 |
| 5min | 20 | uniform | 5.575 | 6.787 | 8.786 | 5.443 | 6.005 | 7.648 |
| 5min | 10 | distance | 0.0 | 0.0 | 0.0 | 5.375 | 5.987 | 7.738 |
| 5min | 10 | uniform | 5.161 | 6.341 | 8.298 | 5.393 | 6.003 | 7.75 |
| 5min | 30 | distance | 0.0 | 0.0 | 0.0 | 5.55 | 6.09 | 7.667 |
| 5min | 30 | uniform | 5.874 | 7.087 | 9.056 | 5.581 | 6.117 | 7.687 |
| 5min | 5 | distance | 0.0 | 0.0 | 0.0 | 5.524 | 6.196 | 8.123 |
| 5min | 5 | uniform | 4.704 | 5.782 | 7.684 | 5.538 | 6.208 | 8.133 |
| 5min | 50 | distance | 0.0 | 0.0 | 0.0 | 5.834 | 6.336 | 7.88 |
| 5min | 50 | uniform | 6.256 | 7.459 | 9.404 | 5.878 | 6.375 | 7.912 |
| 5min | 75 | distance | 0.0 | 0.0 | 0.0 | 6.073 | 6.556 | 8.051 |
| 5min | 75 | uniform | 6.571 | 7.751 | 9.658 | 6.121 | 6.601 | 8.089 |
| 5min | 3 | distance | 0.0 | 0.0 | 0.0 | 5.88 | 6.589 | 8.631 |
| 5min | 3 | uniform | 4.344 | 5.272 | 6.941 | 5.889 | 6.597 | 8.636 |
| 5min | 100 | distance | 0.0 | 0.0 | 0.0 | 6.269 | 6.744 | 8.22 |
| 5min | 100 | uniform | 6.799 | 7.962 | 9.853 | 6.321 | 6.793 | 8.263 |
| 20min | 10 | distance | 0.0 | 0.0 | 0.0 | 7.013 | 7.525 | 8.99 |
| 20min | 10 | uniform | 5.902 | 6.952 | 8.786 | 7.046 | 7.552 | 9.014 |
| 20min | 20 | distance | 0.0 | 0.0 | 0.0 | 7.165 | 7.636 | 9.019 |
| 20min | 20 | uniform | 6.574 | 7.635 | 9.445 | 7.21 | 7.675 | 9.053 |
| 20min | 5 | distance | 0.0 | 0.0 | 0.0 | 7.082 | 7.66 | 9.235 |

Tabelle A.4: Alle 36 Ergebnisse der Modellparameteroptimierung für die k-Nearest Neighbor Regression.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 20min | 5 | uniform | 5.242 | 6.252 | 8.03 | 7.103 | 7.674 | 9.251 |
| 20min | 30 | distance | 0.0 | 0.0 | 0.0 | 7.318 | 7.767 | 9.107 |
| 20min | 30 | uniform | 6.977 | 8.023 | 9.814 | 7.371 | 7.816 | 9.149 |
| 20min | 50 | distance | 0.0 | 0.0 | 0.0 | 7.578 | 8.002 | 9.292 |
| 20min | 50 | uniform | 7.519 | 8.551 | 10.308 | 7.641 | 8.06 | 9.342 |
| 20min | 3 | distance | 0.0 | 0.0 | 0.0 | 7.383 | 8.025 | 9.674 |
| 20min | 3 | uniform | 4.625 | 5.511 | 7.149 | 7.403 | 8.04 | 9.692 |
| 20min | 75 | distance | 0.0 | 0.0 | 0.0 | 7.815 | 8.217 | 9.476 |
| 20min | 75 | uniform | 7.945 | 8.969 | 10.683 | 7.886 | 8.283 | 9.535 |
| 20min | 100 | distance | 0.0 | 0.0 | 0.0 | 8.01 | 8.401 | 9.64 |
| 20min | 100 | uniform | 8.243 | 9.258 | 10.948 | 8.088 | 8.475 | 9.707 |
| 5min | 1 | uniform | 0.0 | 0.0 | 0.0 | 7.713 | 8.565 | 11.171 |
| 5min | 1 | distance | 0.0 | 0.0 | 0.0 | 7.713 | 8.565 | 11.171 |
| 20min | 1 | uniform | 0.0 | 0.0 | 0.0 | 8.86 | 9.629 | 11.8 |
| 20min | 1 | distance | 0.0 | 0.0 | 0.0 | 8.86 | 9.629 | 11.8 |

### A.3.3 Random Forest Regression

Tabelle A.5: Die 50 besten Ergebnisse der Modellparameteroptimierung für die Random Forest Regression.

| look back resolution | max depth | min. samples per leaf | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|
| | | | Early | Mid | Late | Early | Mid | Late |
| 5min | 5 | 3 | 2.394 | 4.273 | 6.834 | 2.237 | 3.363 | 5.836 |
| 5min | 5 | 7 | 2.465 | 4.327 | 6.926 | 2.233 | 3.351 | 5.856 |
| 5min | 5 | 5 | 2.428 | 4.285 | 6.864 | 2.236 | 3.345 | 5.868 |
| 20min | 5 | 3 | 2.4 | 4.304 | 6.846 | 2.171 | 3.466 | 5.823 |
| 5min | 10 | 7 | 1.885 | 3.355 | 5.524 | 2.164 | 3.358 | 5.95 |
| 5min | 10 | 3 | 1.667 | 3.099 | 5.111 | 2.156 | 3.387 | 5.948 |
| 5min | 5 | 10 | 2.61 | 4.465 | 7.037 | 2.284 | 3.316 | 5.904 |
| 20min | 5 | 1 | 2.391 | 4.268 | 6.815 | 2.171 | 3.496 | 5.837 |
| 5min | 10 | 5 | 1.777 | 3.205 | 5.294 | 2.18 | 3.356 | 5.973 |
| 20min | 10 | 7 | 1.921 | 3.447 | 5.639 | 2.113 | 3.453 | 5.951 |
| 20min | 5 | 7 | 2.486 | 4.372 | 6.944 | 2.19 | 3.44 | 5.897 |
| 20min | 5 | 5 | 2.441 | 4.323 | 6.893 | 2.182 | 3.448 | 5.901 |
| 20min | 10 | 3 | 1.653 | 3.133 | 5.211 | 2.084 | 3.525 | 5.925 |
| 20min | 10 | 5 | 1.795 | 3.281 | 5.434 | 2.101 | 3.471 | 5.962 |
| 5min | 5 | 1 | 2.388 | 4.243 | 6.796 | 2.257 | 3.401 | 5.876 |

Tabelle A.5: Die 50 besten Ergebnisse der Modellparameteroptimierung für die Random Forest Regression.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5min | None | 7 | 1.703 | 2.989 | 4.823 | 2.17 | 3.372 | 5.995 |
| 5min | 20 | 7 | 1.703 | 2.99 | 4.827 | 2.17 | 3.373 | 5.996 |
| 20min | 10 | 10 | 2.11 | 3.68 | 5.901 | 2.12 | 3.478 | 5.949 |
| 20min | 5 | 10 | 2.572 | 4.452 | 7.02 | 2.185 | 3.478 | 5.885 |
| 5min | 10 | 1 | 1.621 | 2.999 | 5.041 | 2.163 | 3.442 | 5.955 |
| 20min | 20 | 7 | 1.776 | 3.13 | 4.991 | 2.119 | 3.465 | 5.978 |
| 20min | None | 7 | 1.776 | 3.129 | 4.99 | 2.119 | 3.466 | 5.977 |
| 5min | 10 | 10 | 2.137 | 3.663 | 5.836 | 2.242 | 3.33 | 5.996 |
| 5min | 20 | 5 | 1.503 | 2.676 | 4.331 | 2.191 | 3.391 | 6.006 |
| 20min | None | 10 | 2.036 | 3.502 | 5.513 | 2.125 | 3.488 | 5.977 |
| 20min | 20 | 10 | 2.036 | 3.502 | 5.514 | 2.125 | 3.488 | 5.978 |
| 5min | None | 5 | 1.505 | 2.672 | 4.324 | 2.203 | 3.396 | 5.998 |
| 20min | 20 | 5 | 1.562 | 2.807 | 4.508 | 2.106 | 3.493 | 6.006 |
| 20min | None | 5 | 1.562 | 2.805 | 4.5 | 2.109 | 3.493 | 6.003 |
| 5min | None | 10 | 2.039 | 3.456 | 5.395 | 2.254 | 3.335 | 6.031 |
| 5min | 20 | 10 | 2.039 | 3.456 | 5.395 | 2.254 | 3.339 | 6.03 |
| 5min | 20 | 3 | 1.263 | 2.356 | 3.767 | 2.188 | 3.438 | 6.033 |
| 5min | None | 3 | 1.263 | 2.35 | 3.734 | 2.192 | 3.455 | 6.018 |
| 20min | 10 | 1 | 1.595 | 3.006 | 5.061 | 2.115 | 3.545 | 6.012 |
| 20min | 20 | 3 | 1.285 | 2.412 | 3.831 | 2.104 | 3.558 | 6.014 |
| 20min | None | 3 | 1.282 | 2.406 | 3.803 | 2.101 | 3.567 | 6.013 |
| 5min | None | 1 | 1.069 | 1.962 | 3.131 | 2.203 | 3.475 | 6.072 |
| 20min | None | 1 | 1.041 | 1.935 | 3.04 | 2.129 | 3.606 | 6.061 |
| 5min | 20 | 1 | 1.082 | 1.988 | 3.211 | 2.209 | 3.527 | 6.086 |
| 20min | 20 | 1 | 1.045 | 1.958 | 3.138 | 2.118 | 3.614 | 6.119 |
| 5min | 3 | 10 | 3.722 | 5.279 | 7.698 | 3.321 | 3.977 | 6.201 |
| 5min | 3 | 1 | 3.7 | 5.277 | 7.687 | 3.348 | 4.007 | 6.204 |
| 5min | 3 | 3 | 3.7 | 5.277 | 7.687 | 3.348 | 4.007 | 6.204 |
| 5min | 3 | 5 | 3.7 | 5.277 | 7.687 | 3.348 | 4.007 | 6.204 |
| 5min | 3 | 7 | 3.7 | 5.277 | 7.687 | 3.348 | 4.007 | 6.204 |
| 20min | 3 | 10 | 3.749 | 5.419 | 7.715 | 3.402 | 4.346 | 6.326 |
| 20min | 3 | 7 | 3.741 | 5.418 | 7.715 | 3.428 | 4.342 | 6.326 |
| 20min | 3 | 5 | 3.741 | 5.415 | 7.715 | 3.428 | 4.349 | 6.326 |
| 20min | 3 | 1 | 3.741 | 5.415 | 7.708 | 3.428 | 4.349 | 6.332 |
| 20min | 3 | 3 | 3.741 | 5.415 | 7.708 | 3.428 | 4.349 | 6.332 |

### A.3.4 Support Vector Regression

Tabelle A.6: Die 50 besten Ergebnisse der Modellparameteroptimierung für die Support Vector Regression.

| look back resolution | $C$ | $\epsilon$ | Kern | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | 0.6 | 0.3 | linear | 2.38 | 4.447 | 7.243 | 1.9 | 3.125 | 5.636 |
| 5min | 0.5 | 0.01 | linear | 2.388 | 4.453 | 7.256 | 1.903 | 3.122 | 5.637 |
| 5min | 0.5 | 0.3 | linear | 2.387 | 4.454 | 7.251 | 1.905 | 3.12 | 5.637 |
| 5min | 0.6 | 0.01 | linear | 2.382 | 4.446 | 7.247 | 1.898 | 3.13 | 5.634 |
| 5min | 0.6 | 0.2 | linear | 2.38 | 4.447 | 7.244 | 1.899 | 3.129 | 5.635 |
| 5min | 0.5 | 0.05 | linear | 2.389 | 4.454 | 7.256 | 1.903 | 3.125 | 5.638 |
| 5min | 0.5 | 0.2 | linear | 2.387 | 4.453 | 7.254 | 1.904 | 3.125 | 5.638 |
| 5min | 0.6 | 0.05 | linear | 2.382 | 4.447 | 7.248 | 1.899 | 3.134 | 5.634 |
| 5min | 0.5 | 0.1 | linear | 2.389 | 4.454 | 7.255 | 1.904 | 3.127 | 5.638 |
| 5min | 0.8 | 0.3 | linear | 2.372 | 4.437 | 7.234 | 1.892 | 3.137 | 5.643 |
| 5min | 0.6 | 0.1 | linear | 2.382 | 4.447 | 7.247 | 1.901 | 3.138 | 5.635 |
| 5min | 1.0 | 0.3 | linear | 2.367 | 4.432 | 7.228 | 1.887 | 3.147 | 5.654 |
| 5min | 0.8 | 0.1 | linear | 2.373 | 4.439 | 7.236 | 1.894 | 3.156 | 5.64 |
| 5min | 0.8 | 0.2 | linear | 2.373 | 4.437 | 7.235 | 1.892 | 3.159 | 5.639 |
| 5min | 1.0 | 0.2 | linear | 2.368 | 4.431 | 7.229 | 1.891 | 3.163 | 5.644 |
| 5min | 0.3 | 0.2 | linear | 2.416 | 4.48 | 7.288 | 1.927 | 3.12 | 5.653 |
| 5min | 0.8 | 0.01 | linear | 2.374 | 4.44 | 7.236 | 1.895 | 3.167 | 5.639 |
| 5min | 0.3 | 0.1 | linear | 2.415 | 4.481 | 7.29 | 1.929 | 3.12 | 5.655 |
| 5min | 0.3 | 0.3 | linear | 2.415 | 4.481 | 7.287 | 1.93 | 3.122 | 5.652 |
| 5min | 0.8 | 0.05 | linear | 2.373 | 4.439 | 7.236 | 1.895 | 3.17 | 5.641 |
| 5min | 0.3 | 0.05 | linear | 2.414 | 4.481 | 7.29 | 1.93 | 3.121 | 5.656 |
| 5min | 0.3 | 0.01 | linear | 2.413 | 4.482 | 7.29 | 1.929 | 3.121 | 5.658 |
| 5min | 1.0 | 0.01 | linear | 2.369 | 4.433 | 7.23 | 1.89 | 3.182 | 5.642 |
| 5min | 1.0 | 0.05 | linear | 2.368 | 4.433 | 7.23 | 1.892 | 3.182 | 5.648 |
| 5min | 1.0 | 0.1 | linear | 2.368 | 4.432 | 7.23 | 1.892 | 3.184 | 5.646 |
| 20min | 1.0 | 0.05 | linear | 2.398 | 4.476 | 7.309 | 1.965 | 3.157 | 5.681 |
| 20min | 1.0 | 0.2 | linear | 2.399 | 4.477 | 7.311 | 1.966 | 3.151 | 5.687 |
| 20min | 1.0 | 0.1 | linear | 2.398 | 4.476 | 7.31 | 1.965 | 3.154 | 5.686 |
| 20min | 1.0 | 0.01 | linear | 2.398 | 4.476 | 7.308 | 1.965 | 3.159 | 5.682 |
| 20min | 1.0 | 0.3 | linear | 2.399 | 4.478 | 7.31 | 1.972 | 3.15 | 5.686 |
| 20min | 0.8 | 0.01 | linear | 2.407 | 4.488 | 7.318 | 1.984 | 3.176 | 5.71 |
| 20min | 0.8 | 0.1 | linear | 2.409 | 4.488 | 7.319 | 1.993 | 3.173 | 5.712 |
| 20min | 0.8 | 0.2 | linear | 2.408 | 4.489 | 7.322 | 1.993 | 3.174 | 5.711 |

Tabelle A.6: Die 50 besten Ergebnisse der Modellparameteroptimierung für die Support Vector Regression.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 20min | 0.8 | 0.05 | linear | 2.408 | 4.488 | 7.319 | 1.992 | 3.176 | 5.711 |
| 20min | 0.8 | 0.3 | linear | 2.409 | 4.49 | 7.325 | 1.997 | 3.176 | 5.713 |
| 20min | 0.6 | 0.2 | linear | 2.426 | 4.509 | 7.34 | 2.035 | 3.204 | 5.752 |
| 20min | 0.6 | 0.01 | linear | 2.426 | 4.509 | 7.34 | 2.034 | 3.208 | 5.755 |
| 20min | 0.6 | 0.05 | linear | 2.426 | 4.51 | 7.341 | 2.034 | 3.207 | 5.757 |
| 20min | 0.6 | 0.3 | linear | 2.425 | 4.509 | 7.342 | 2.04 | 3.211 | 5.749 |
| 20min | 0.6 | 0.1 | linear | 2.426 | 4.51 | 7.341 | 2.035 | 3.209 | 5.76 |
| 20min | 0.5 | 0.2 | linear | 2.439 | 4.524 | 7.359 | 2.064 | 3.236 | 5.78 |
| 20min | 0.5 | 0.1 | linear | 2.441 | 4.523 | 7.361 | 2.065 | 3.23 | 5.786 |
| 20min | 0.5 | 0.3 | linear | 2.439 | 4.524 | 7.358 | 2.072 | 3.237 | 5.78 |
| 20min | 0.5 | 0.05 | linear | 2.442 | 4.525 | 7.36 | 2.069 | 3.234 | 5.791 |
| 20min | 0.5 | 0.01 | linear | 2.442 | 4.526 | 7.361 | 2.07 | 3.236 | 5.791 |
| 20min | 0.3 | 0.01 | linear | 2.493 | 4.583 | 7.419 | 2.172 | 3.338 | 5.887 |
| 20min | 0.3 | 0.05 | linear | 2.495 | 4.583 | 7.418 | 2.176 | 3.34 | 5.885 |
| 20min | 0.3 | 0.1 | linear | 2.494 | 4.583 | 7.419 | 2.181 | 3.344 | 5.883 |
| 20min | 0.3 | 0.3 | linear | 2.491 | 4.581 | 7.424 | 2.186 | 3.342 | 5.883 |
| 20min | 0.3 | 0.2 | linear | 2.493 | 4.582 | 7.421 | 2.192 | 3.34 | 5.88 |
| 20m | 0.3 | 0.2 | linear | 2.493 | 4.582 | 7.421 | 2.192 | 3.34 | 5.88 |

### A.3.5 Neuronale Netze

#### Anzahl Neuronen, Aktivierungsfunktion

Tabelle A.7: Die 50 besten Ergebnisse der Modellparameteroptimierung (Anzahl Neuronen, Aktivierungsfunktion) für die neuronalen Netze.

| look back resolution | Netz | $\varphi(x)$ | Anzahl Neuronen | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN3 | linear | 128 | 2.399 | 4.411 | 7.165 | 1.904 | 3.062 | 5.591 |
| 5min | CNN1 | linear | 16 | 2.367 | 4.388 | 7.155 | 1.888 | 3.117 | 5.624 |
| 5min | CNN2 | linear | 32 | 2.374 | 4.38 | 7.146 | 1.912 | 3.113 | 5.606 |
| 5min | CNN1 | linear | 256 | 2.407 | 4.412 | 7.173 | 1.932 | 3.11 | 5.615 |
| 5min | CNN2 | linear | 64 | 2.37 | 4.391 | 7.147 | 1.938 | 3.098 | 5.623 |
| 5min | CNN3 | linear | 32 | 2.368 | 4.375 | 7.12 | 1.92 | 3.109 | 5.631 |
| 5min | CNN3 | linear | 64 | 2.376 | 4.385 | 7.155 | 1.929 | 3.115 | 5.628 |
| 5min | CNN1 | linear | 128 | 2.376 | 4.385 | 7.151 | 1.926 | 3.122 | 5.625 |
| 5min | CNN2 | linear | 16 | 2.366 | 4.389 | 7.158 | 1.952 | 3.109 | 5.616 |
| 5min | MLP5 | linear | 128 | 2.412 | 4.415 | 7.2 | 1.969 | 3.113 | 5.601 |
| 5min | MLP3 | linear | 256 | 2.41 | 4.429 | 7.222 | 1.959 | 3.109 | 5.617 |

Tabelle A.7: Die 50 besten Ergebnisse der Modellparameteroptimierung (Anzahl Neuronen, Aktivierungsfunktion) für die neuronalen Netze.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5min | CNN3 | linear | 16 | 2.394 | 4.39 | 7.192 | 1.954 | 3.109 | 5.623 |
| 5min | MLP5 | linear | 256 | 2.398 | 4.426 | 7.204 | 1.967 | 3.098 | 5.625 |
| 5min | CNN2 | linear | 128 | 2.369 | 4.391 | 7.129 | 1.943 | 3.131 | 5.616 |
| 5min | MLP4 | linear | 256 | 2.396 | 4.416 | 7.188 | 1.953 | 3.11 | 5.628 |
| 5min | MLP2 | linear | 64 | 2.359 | 4.4 | 7.162 | 1.917 | 3.143 | 5.635 |
| 5min | MLP3 | linear | 32 | 2.391 | 4.404 | 7.191 | 1.96 | 3.112 | 5.623 |
| 5min | MLP5 | linear | 16 | 2.37 | 4.405 | 7.171 | 1.929 | 3.136 | 5.632 |
| 5min | MLP5 | linear | 64 | 2.371 | 4.412 | 7.178 | 1.946 | 3.128 | 5.623 |
| 5min | CNN1 | linear | 32 | 2.361 | 4.367 | 7.128 | 1.912 | 3.152 | 5.637 |
| 5min | MLP1 | linear | 64 | 2.382 | 4.397 | 7.182 | 1.935 | 3.123 | 5.644 |
| 5min | MLP3 | linear | 64 | 2.378 | 4.396 | 7.182 | 1.925 | 3.131 | 5.646 |
| 5min | MLP4 | linear | 64 | 2.397 | 4.417 | 7.204 | 1.95 | 3.133 | 5.619 |
| 5min | MLP3 | linear | 128 | 2.405 | 4.415 | 7.195 | 1.951 | 3.121 | 5.632 |
| 20min | CNN2 | linear | 256 | 2.391 | 4.438 | 7.232 | 1.951 | 3.112 | 5.644 |
| 5min | CNN2 | linear | 256 | 2.371 | 4.382 | 7.162 | 1.925 | 3.143 | 5.641 |
| 5min | MLP4 | linear | 32 | 2.409 | 4.423 | 7.184 | 1.97 | 3.121 | 5.619 |
| 5min | LSTM2 | linear | 16 | 2.439 | 4.43 | 7.204 | 2.005 | 3.151 | 5.555 |
| 5min | MLP1 | linear | 16 | 2.365 | 4.387 | 7.171 | 1.93 | 3.134 | 5.647 |
| 5min | MLP2 | linear | 32 | 2.368 | 4.383 | 7.17 | 1.938 | 3.131 | 5.643 |
| 5min | MLP1 | linear | 32 | 2.365 | 4.387 | 7.17 | 1.928 | 3.149 | 5.636 |
| 5min | MLP1 | linear | 256 | 2.372 | 4.39 | 7.189 | 1.948 | 3.131 | 5.635 |
| 5min | MLP4 | linear | 128 | 2.421 | 4.422 | 7.196 | 1.966 | 3.116 | 5.632 |
| 20min | CNN2 | linear | 128 | 2.376 | 4.428 | 7.261 | 1.937 | 3.125 | 5.653 |
| 5min | MLP3 | linear | 16 | 2.39 | 4.412 | 7.223 | 1.941 | 3.122 | 5.655 |
| 5min | MLP5 | linear | 32 | 2.386 | 4.395 | 7.201 | 1.97 | 3.127 | 5.621 |
| 20min | CNN1 | linear | 16 | 2.387 | 4.437 | 7.236 | 1.937 | 3.141 | 5.641 |
| 5min | MLP4 | linear | 16 | 2.396 | 4.406 | 7.209 | 1.967 | 3.122 | 5.63 |
| 5min | MLP2 | linear | 16 | 2.363 | 4.396 | 7.174 | 1.919 | 3.152 | 5.653 |
| 5min | CNN1 | linear | 64 | 2.38 | 4.382 | 7.148 | 1.94 | 3.136 | 5.649 |
| 20min | CNN2 | linear | 16 | 2.391 | 4.432 | 7.216 | 1.967 | 3.114 | 5.65 |
| 5min | MLP2 | linear | 256 | 2.387 | 4.392 | 7.187 | 1.946 | 3.139 | 5.646 |
| 5min | CNN3 | linear | 256 | 2.41 | 4.425 | 7.2 | 1.978 | 3.129 | 5.625 |
| 20min | CNN1 | linear | 256 | 2.376 | 4.419 | 7.231 | 1.945 | 3.144 | 5.646 |
| 5min | MLP2 | linear | 128 | 2.362 | 4.392 | 7.169 | 1.941 | 3.146 | 5.648 |
| 20min | CNN2 | linear | 64 | 2.382 | 4.43 | 7.253 | 1.951 | 3.134 | 5.655 |
| 5min | MLP1 | linear | 128 | 2.379 | 4.38 | 7.169 | 1.951 | 3.151 | 5.645 |
| 20min | CNN1 | linear | 128 | 2.375 | 4.426 | 7.227 | 1.953 | 3.157 | 5.64 |

Tabelle A.7: Die 50 besten Ergebnisse der Modellparameteroptimierung (Anzahl Neuronen, Aktivierungsfunktion) für die neuronalen Netze.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 20min | CNN1 | linear | 32 | 2.38 | 4.424 | 7.236 | 1.943 | 3.156 | 5.654 |
| 20min | CNN1 | linear | 64 | 2.373 | 4.417 | 7.225 | 1.948 | 3.162 | 5.645 |
| 20min | MLP5 | linear | 32 | 2.405 | 4.424 | 7.235 | 1.956 | 3.151 | 5.651 |

**Optimierungsalgorithmus, Learning Rate, Batch Size**

Tabelle A.8: Die 50 besten Ergebnisse der Modellparameteroptimierung (Optimierungsalgorithmus, Learning Rate, Batch Size) für die neuronalen Netze.

| look back resolution | Netz | Optimierer | Learning Rate | Batch Size | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN3 | rmsprop | 0.001 | 16 | 2.422 | 4.421 | 7.185 | 1.961 | 3.093 | 5.582 |
| 5min | CNN1 | rmsprop | 0.001 | 16 | 2.381 | 4.4 | 7.187 | 1.931 | 3.101 | 5.613 |
| 5min | CNN3 | adam | 0.001 | 16 | 2.382 | 4.371 | 7.121 | 1.924 | 3.115 | 5.622 |
| 5min | MLP1 | rmsprop | 0.01 | 32 | 2.389 | 4.407 | 7.188 | 1.949 | 3.088 | 5.63 |
| 20min | CNN2 | adam | 0.01 | 32 | 2.456 | 4.496 | 7.232 | 1.962 | 3.094 | 5.612 |
| 5min | CNN1 | adam | 0.001 | 64 | 2.376 | 4.402 | 7.203 | 1.935 | 3.114 | 5.619 |
| 5min | CNN2 | adam | 0.01 | 64 | 2.393 | 4.398 | 7.161 | 1.961 | 3.099 | 5.61 |
| 5min | CNN1 | rmsprop | 0.01 | 64 | 2.379 | 4.383 | 7.166 | 1.937 | 3.105 | 5.631 |
| 5min | CNN2 | adam | 0.001 | 128 | 2.409 | 4.424 | 7.205 | 1.962 | 3.103 | 5.612 |
| 5min | MLP2 | adam | 0.01 | 32 | 2.413 | 4.414 | 7.215 | 1.966 | 3.086 | 5.627 |
| 5min | CNN1 | adam | 0.001 | 128 | 2.389 | 4.404 | 7.188 | 1.938 | 3.122 | 5.623 |
| 20min | CNN2 | rmsprop | 0.01 | 32 | 2.404 | 4.429 | 7.222 | 1.948 | 3.104 | 5.633 |
| 5min | MLP3 | rmsprop | 0.001 | 16 | 2.376 | 4.387 | 7.181 | 1.931 | 3.141 | 5.614 |
| 5min | CNN1 | rmsprop | 0.001 | 32 | 2.374 | 4.409 | 7.169 | 1.931 | 3.128 | 5.627 |
| 5min | MLP1 | rmsprop | 0.001 | 32 | 2.365 | 4.382 | 7.173 | 1.925 | 3.134 | 5.629 |
| 5min | CNN3 | rmsprop | 0.001 | 128 | 2.4 | 4.396 | 7.16 | 1.942 | 3.111 | 5.638 |
| 5min | CNN3 | rmsprop | 0.001 | 64 | 2.412 | 4.395 | 7.157 | 1.936 | 3.125 | 5.633 |
| 5min | MLP5 | rmsprop | 0.001 | 32 | 2.365 | 4.385 | 7.191 | 1.927 | 3.125 | 5.644 |
| 5min | MLP5 | adam | 0.001 | 64 | 2.393 | 4.387 | 7.171 | 1.967 | 3.113 | 5.616 |
| 5min | MLP1 | rmsprop | 0.001 | 16 | 2.355 | 4.389 | 7.174 | 1.918 | 3.136 | 5.642 |
| 5min | MLP2 | rmsprop | 0.01 | 64 | 2.397 | 4.406 | 7.201 | 1.969 | 3.115 | 5.613 |
| 5min | CNN2 | rmsprop | 0.001 | 16 | 2.377 | 4.373 | 7.144 | 1.941 | 3.122 | 5.637 |
| 5min | CNN2 | adam | 0.001 | 16 | 2.384 | 4.394 | 7.175 | 1.96 | 3.116 | 5.626 |
| 20min | CNN1 | rmsprop | 0.01 | 32 | 2.423 | 4.418 | 7.239 | 1.969 | 3.123 | 5.61 |
| 5min | CNN3 | adam | 0.001 | 32 | 2.378 | 4.398 | 7.14 | 1.949 | 3.142 | 5.612 |
| 20min | CNN2 | adam | 0.01 | 64 | 2.401 | 4.44 | 7.236 | 1.969 | 3.11 | 5.624 |
| 5min | MLP2 | rmsprop | 0.001 | 32 | 2.415 | 4.416 | 7.215 | 1.966 | 3.117 | 5.621 |
| 5min | CNN1 | rmsprop | 0.01 | 32 | 2.418 | 4.417 | 7.23 | 1.978 | 3.096 | 5.63 |

Tabelle A.8: Die 50 besten Ergebnisse der Modellparameteroptimierung (Optimierungsalgorithmus, Learning Rate, Batch Size) für die neuronalen Netze.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 20min | CNN2 | rmsprop | 0.01 | 16 | 2.436 | 4.467 | 7.304 | 1.993 | 3.081 | 5.63 |
| 20min | CNN1 | rmsprop | 0.01 | 16 | 2.383 | 4.418 | 7.253 | 1.928 | 3.147 | 5.63 |
| 5min | CNN1 | adam | 0.001 | 32 | 2.386 | 4.403 | 7.188 | 1.948 | 3.122 | 5.635 |
| 5min | CNN3 | adam | 0.001 | 64 | 2.389 | 4.381 | 7.161 | 1.947 | 3.137 | 5.621 |
| 5min | MLP5 | adam | 0.001 | 16 | 2.432 | 4.429 | 7.225 | 2.014 | 3.083 | 5.609 |
| 5min | CNN2 | rmsprop | 0.001 | 32 | 2.36 | 4.418 | 7.155 | 1.94 | 3.148 | 5.62 |
| 20min | CNN1 | rmsprop | 0.001 | 32 | 2.368 | 4.415 | 7.207 | 1.916 | 3.145 | 5.648 |
| 5min | CNN1 | rmsprop | 0.001 | 128 | 2.363 | 4.383 | 7.156 | 1.932 | 3.146 | 5.632 |
| 5min | MLP4 | adam | 0.001 | 16 | 2.395 | 4.403 | 7.203 | 1.963 | 3.113 | 5.634 |
| 5min | MLP1 | adam | 0.01 | 32 | 2.37 | 4.419 | 7.202 | 1.947 | 3.127 | 5.636 |
| 5min | CNN3 | rmsprop | 0.001 | 32 | 2.417 | 4.426 | 7.166 | 1.951 | 3.136 | 5.624 |
| 5min | MLP5 | rmsprop | 0.001 | 16 | 2.401 | 4.421 | 7.223 | 1.96 | 3.134 | 5.62 |
| 5min | MLP3 | rmsprop | 0.001 | 64 | 2.406 | 4.399 | 7.189 | 1.964 | 3.124 | 5.627 |
| 5min | CNN3 | adam | 0.001 | 128 | 2.375 | 4.39 | 7.143 | 1.934 | 3.131 | 5.651 |
| 20min | CNN1 | adam | 0.01 | 64 | 2.379 | 4.422 | 7.221 | 1.942 | 3.128 | 5.646 |
| 5min | MLP3 | adam | 0.001 | 16 | 2.381 | 4.377 | 7.156 | 1.955 | 3.138 | 5.625 |
| 20min | CNN2 | adam | 0.001 | 32 | 2.369 | 4.417 | 7.208 | 1.926 | 3.141 | 5.651 |
| 5min | MLP4 | rmsprop | 0.001 | 32 | 2.4 | 4.412 | 7.204 | 1.97 | 3.117 | 5.631 |
| 20min | CNN2 | rmsprop | 0.001 | 16 | 2.383 | 4.417 | 7.212 | 1.958 | 3.129 | 5.632 |
| 20min | CNN2 | adam | 0.001 | 16 | 2.371 | 4.411 | 7.212 | 1.933 | 3.142 | 5.646 |
| 20min | CNN1 | adam | 0.01 | 128 | 2.366 | 4.416 | 7.213 | 1.921 | 3.136 | 5.665 |
| 5min | MLP2 | adam | 0.01 | 64 | 2.409 | 4.392 | 7.187 | 1.987 | 3.102 | 5.635 |
| 20min | CNN1 | rmsprop | 0.001 | 16 | 2.367 | 4.42 | 7.212 | 1.928 | 3.151 | 5.647 |

**Dropout, Max-norm Regularisierung**

Tabelle A.9: Die 50 besten Ergebnisse der Modellparameteroptimierung (Dropout Rate, max-norm Regularisierung) für die neuronalen Netze.

| look back resolution | Netz | Dropout Rate | max-norm | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN3 | 0.0 | None | 2.397 | 4.409 | 7.167 | 1.93 | 3.093 | 5.595 |
| 5min | CNN1 | 0.0 | 2 | 2.399 | 4.416 | 7.214 | 1.925 | 3.106 | 5.607 |
| 5min | CNN1 | 0.0 | 3 | 2.369 | 4.384 | 7.179 | 1.925 | 3.114 | 5.622 |
| 5min | CNN1 | 0.0 | None | 2.377 | 4.391 | 7.188 | 1.927 | 3.114 | 5.623 |
| 5min | CNN3 | 0.0 | 2 | 2.384 | 4.416 | 7.216 | 1.919 | 3.104 | 5.643 |
| 5min | MLP5 | 0.0 | None | 2.377 | 4.403 | 7.203 | 1.943 | 3.11 | 5.618 |
| 5min | CNN3 | 0.0 | 4 | 2.368 | 4.391 | 7.141 | 1.889 | 3.154 | 5.636 |

Tabelle A.9: Die 50 besten Ergebnisse der Modellparameteroptimierung (Dropout Rate, max-norm Regularisierung) für die neuronalen Netze.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5min | CNN2 | 0.0 | None | 2.396 | 4.389 | 7.174 | 1.954 | 3.113 | 5.617 |
| 5min | CNN3 | 0.0 | 3 | 2.37 | 4.403 | 7.175 | 1.912 | 3.096 | 5.682 |
| 5min | MLP4 | 0.0 | 2 | 2.43 | 4.442 | 7.246 | 1.962 | 3.102 | 5.628 |
| 5min | MLP5 | 0.0 | 4 | 2.403 | 4.44 | 7.215 | 1.971 | 3.1 | 5.625 |
| 5min | MLP3 | 0.0 | 3 | 2.415 | 4.431 | 7.209 | 1.951 | 3.125 | 5.623 |
| 5min | MLP3 | 0.0 | 4 | 2.417 | 4.42 | 7.201 | 1.981 | 3.1 | 5.62 |
| 5min | MLP4 | 0.2 | 2 | 2.414 | 4.431 | 7.221 | 1.988 | 3.089 | 5.624 |
| 5min | MLP3 | 0.0 | None | 2.415 | 4.425 | 7.204 | 1.959 | 3.126 | 5.617 |
| 5min | MLP4 | 0.0 | 4 | 2.376 | 4.376 | 7.155 | 1.932 | 3.13 | 5.641 |
| 20min | CNN1 | 0.0 | 4 | 2.364 | 4.409 | 7.202 | 1.91 | 3.138 | 5.655 |
| 20min | CNN1 | 0.0 | None | 2.371 | 4.416 | 7.215 | 1.914 | 3.151 | 5.639 |
| 5min | MLP5 | 0.0 | 3 | 2.431 | 4.43 | 7.225 | 1.972 | 3.108 | 5.626 |
| 20min | CNN1 | 0.0 | 3 | 2.369 | 4.417 | 7.213 | 1.927 | 3.145 | 5.639 |
| 5min | MLP4 | 0.0 | 3 | 2.384 | 4.386 | 7.168 | 1.949 | 3.144 | 5.62 |
| 20min | CNN2 | 0.0 | None | 2.419 | 4.429 | 7.233 | 1.977 | 3.12 | 5.622 |
| 5min | CNN2 | 0.0 | 3 | 2.404 | 4.41 | 7.195 | 1.962 | 3.146 | 5.616 |
| 5min | CNN2 | 0.0 | 2 | 2.454 | 4.418 | 7.213 | 2.008 | 3.114 | 5.605 |
| 5min | CNN1 | 0.0 | 4 | 2.354 | 4.383 | 7.144 | 1.937 | 3.151 | 5.64 |
| 5min | MLP4 | 0.2 | 3 | 2.411 | 4.409 | 7.181 | 1.98 | 3.12 | 5.635 |
| 20min | MLP5 | 0.0 | None | 2.38 | 4.435 | 7.239 | 1.945 | 3.16 | 5.634 |
| 5min | MLP1 | 0.0 | None | 2.398 | 4.412 | 7.21 | 1.984 | 3.114 | 5.643 |
| 20min | CNN1 | 0.0 | 2 | 2.381 | 4.42 | 7.221 | 1.939 | 3.154 | 5.651 |
| 5min | MLP2 | 0.0 | None | 2.374 | 4.385 | 7.155 | 1.942 | 3.146 | 5.662 |
| 5min | CNN2 | 0.0 | 4 | 2.432 | 4.411 | 7.181 | 2.003 | 3.117 | 5.632 |
| 5min | MLP4 | 0.0 | None | 2.427 | 4.42 | 7.241 | 1.983 | 3.129 | 5.643 |
| 5min | LSTM3 | 0.0 | None | 2.451 | 4.418 | 7.247 | 2.063 | 3.122 | 5.574 |
| 5min | MLP4 | 0.2 | 4 | 2.405 | 4.433 | 7.211 | 1.993 | 3.143 | 5.629 |
| 5min | MLP3 | 0.0 | 2 | 2.459 | 4.425 | 7.218 | 2.005 | 3.118 | 5.65 |
| 5min | MLP3 | 0.5 | 2 | 2.464 | 4.448 | 7.249 | 2.035 | 3.113 | 5.626 |
| 5min | MLP4 | 0.5 | 2 | 2.428 | 4.432 | 7.2 | 2.022 | 3.133 | 5.62 |
| 20min | MLP3 | 0.0 | None | 2.371 | 4.418 | 7.228 | 1.949 | 3.174 | 5.653 |
| 5min | MLP1 | 0.2 | None | 2.426 | 4.418 | 7.264 | 2.018 | 3.126 | 5.636 |
| 5min | CNN1 | 0.2 | 4 | 2.554 | 4.516 | 7.313 | 2.073 | 3.103 | 5.605 |
| 20min | MLP3 | 0.0 | 3 | 2.398 | 4.438 | 7.283 | 1.95 | 3.157 | 5.677 |
| 5min | CNN1 | 0.2 | 3 | 2.547 | 4.53 | 7.328 | 2.058 | 3.106 | 5.62 |
| 5min | MLP3 | 0.7 | None | 2.476 | 4.433 | 7.252 | 2.069 | 3.107 | 5.611 |

Tabelle A.9: Die 50 besten Ergebnisse der Modellparameteroptimierung (Dropout Rate, max-norm Regularisierung) für die neuronalen Netze.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 20min | CNN4 | 0.0 | None | 2.447 | 4.448 | 7.244 | 1.967 | 3.155 | 5.676 |
| 5min | MLP4 | 0.2 | None | 2.417 | 4.419 | 7.202 | 2.006 | 3.15 | 5.645 |
| 5min | MLP3 | 0.7 | 2 | 2.492 | 4.438 | 7.255 | 2.091 | 3.112 | 5.599 |
| 5min | MLP5 | 0.0 | 2 | 2.522 | 4.48 | 7.259 | 2.055 | 3.118 | 5.632 |
| 5min | MLP3 | 0.2 | 3 | 2.418 | 4.483 | 7.203 | 1.969 | 3.189 | 5.649 |
| 20min | CNN4 | 0.0 | 4 | 2.405 | 4.375 | 7.179 | 1.969 | 3.161 | 5.683 |
| 5min | MLP3 | 0.5 | None | 2.453 | 4.451 | 7.31 | 2.011 | 3.126 | 5.678 |

## A.4 Ergebnisse der Datensatzoptimierung für die neuronalen Netze

Tabelle A.10: Alle Ergebnisse der Datensatzoptimierung für die neuronalen Netze.

| look back resolution | Netz | Training RMSE | | | Validierung RMSE | | |
|---|---|---|---|---|---|---|---|
| | | Early | Mid | Late | Early | Mid | Late |
| 5min | CNN1 | 2.386 | 4.393 | 7.177 | 1.932 | 3.097 | 5.614 |
| 15min | CNN1 | 2.387 | 4.41 | 7.205 | 1.913 | 3.12 | 5.642 |
| 5min | CNN3 | 2.372 | 4.403 | 7.186 | 1.925 | 3.12 | 5.639 |
| 10min | CNN3 | 2.372 | 4.4 | 7.173 | 1.926 | 3.131 | 5.629 |
| 10min | CNN1 | 2.363 | 4.388 | 7.168 | 1.924 | 3.131 | 5.635 |
| 5min | MLP5 | 2.369 | 4.388 | 7.173 | 1.928 | 3.133 | 5.633 |
| 5min | MLP3 | 2.433 | 4.43 | 7.215 | 2.001 | 3.089 | 5.606 |
| 15min | MLP2 | 2.399 | 4.441 | 7.232 | 1.965 | 3.095 | 5.638 |
| 20min | CNN2 | 2.379 | 4.458 | 7.227 | 1.947 | 3.141 | 5.62 |
| 5min | MLP4 | 2.426 | 4.424 | 7.196 | 1.96 | 3.113 | 5.639 |
| 20min | CNN1 | 2.388 | 4.424 | 7.228 | 1.934 | 3.142 | 5.641 |
| 5min | CNN2 | 2.39 | 4.39 | 7.17 | 1.954 | 3.129 | 5.634 |
| 15min | CNN3 | 2.373 | 4.423 | 7.202 | 1.952 | 3.139 | 5.627 |
| 15min | CNN2 | 2.401 | 4.424 | 7.237 | 1.952 | 3.142 | 5.626 |
| 5min | MLP1 | 2.363 | 4.389 | 7.178 | 1.937 | 3.139 | 5.647 |
| 15min | MLP1 | 2.374 | 4.463 | 7.226 | 1.949 | 3.152 | 5.624 |
| 15min | MLP5 | 2.365 | 4.403 | 7.21 | 1.932 | 3.156 | 5.641 |
| 10min | CNN2 | 2.408 | 4.439 | 7.243 | 1.973 | 3.133 | 5.627 |
| 10min | MLP3 | 2.395 | 4.417 | 7.206 | 1.971 | 3.133 | 5.63 |
| 15min | MLP4 | 2.385 | 4.423 | 7.205 | 1.967 | 3.137 | 5.636 |
| 10min | MLP2 | 2.384 | 4.412 | 7.217 | 1.982 | 3.136 | 5.641 |
| 15min | MLP3 | 2.397 | 4.419 | 7.241 | 1.975 | 3.149 | 5.637 |
| 5min | MLP2 | 2.39 | 4.468 | 7.193 | 1.956 | 3.18 | 5.639 |

Tabelle A.10: Alle Ergebnisse der Datensatzoptimierung für die neuronalen Netze.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10min | MLP4 | 2.39 | 4.411 | 7.204 | 1.975 | 3.171 | 5.641 |
| 10min | MLP5 | 2.37 | 4.403 | 7.199 | 1.981 | 3.169 | 5.639 |
| 10min | MLP1 | 2.369 | 4.396 | 7.18 | 1.958 | 3.184 | 5.651 |
| 20min | MLP3 | 2.375 | 4.437 | 7.224 | 1.967 | 3.165 | 5.673 |
| 20min | MLP5 | 2.403 | 4.447 | 7.246 | 1.994 | 3.161 | 5.657 |
| 20min | MLP1 | 2.391 | 4.494 | 7.269 | 1.97 | 3.196 | 5.661 |
| 20min | CNN4 | 2.417 | 4.397 | 7.173 | 1.973 | 3.17 | 5.695 |
| 20min | MLP4 | 2.383 | 4.438 | 7.238 | 2.005 | 3.207 | 5.666 |
| 1min | LSTM2 | 2.44 | 4.169 | 6.944 | 2.141 | 3.211 | 5.552 |
| 20min | MLP2 | 2.414 | 4.444 | 7.244 | 2.051 | 3.184 | 5.671 |
| 10min | LSTM1 | 2.383 | 4.277 | 7.074 | 2.07 | 3.204 | 5.645 |
| 5min | LSTM1 | 2.48 | 4.377 | 7.182 | 2.115 | 3.204 | 5.608 |
| 2min | LSTM3 | 2.492 | 4.423 | 7.233 | 2.11 | 3.211 | 5.629 |
| 10min | LSTM3 | 2.462 | 4.413 | 7.32 | 2.057 | 3.173 | 5.73 |
| 2min | CNN7 | 2.479 | 4.462 | 7.25 | 2.093 | 3.207 | 5.668 |
| 20min | LSTM3 | 2.395 | 4.291 | 7.116 | 2.093 | 3.223 | 5.669 |
| 5min | LSTM4 | 2.51 | 4.37 | 7.155 | 2.187 | 3.208 | 5.597 |
| 2min | LSTM1 | 2.483 | 4.352 | 7.119 | 2.115 | 3.206 | 5.675 |
| 15min | LSTM3 | 2.507 | 4.378 | 7.208 | 2.14 | 3.248 | 5.616 |
| 15min | LSTM2 | 2.401 | 4.282 | 7.092 | 2.085 | 3.271 | 5.673 |
| 2min | MLP4 | 2.532 | 4.503 | 7.269 | 2.172 | 3.193 | 5.666 |
| 5min | LSTM2 | 2.51 | 4.354 | 7.151 | 2.201 | 3.238 | 5.598 |
| 10min | LSTM2 | 2.367 | 4.202 | 6.992 | 2.048 | 3.289 | 5.717 |
| 2min | MLP3 | 2.48 | 4.468 | 7.236 | 2.136 | 3.221 | 5.702 |
| 2min | MLP5 | 2.463 | 4.456 | 7.23 | 2.108 | 3.217 | 5.738 |
| 2min | MLP2 | 2.406 | 4.405 | 7.197 | 2.116 | 3.22 | 5.731 |
| 20min | LSTM2 | 2.404 | 4.278 | 7.119 | 2.137 | 3.241 | 5.696 |
| 1min | LSTM4 | 2.616 | 4.46 | 7.245 | 2.211 | 3.279 | 5.596 |
| 5min | LSTM3 | 2.558 | 4.474 | 7.29 | 2.236 | 3.227 | 5.629 |
| 2min | CNN1 | 2.393 | 4.394 | 7.165 | 2.13 | 3.233 | 5.73 |
| 2min | LSTM4 | 2.568 | 4.439 | 7.276 | 2.197 | 3.22 | 5.679 |
| 2min | LSTM2 | 2.55 | 4.371 | 7.125 | 2.162 | 3.254 | 5.682 |
| 1min | MLP5 | 2.44 | 4.403 | 7.157 | 2.086 | 3.269 | 5.748 |
| 1min | MLP4 | 2.534 | 4.455 | 7.22 | 2.164 | 3.246 | 5.694 |
| 1min | MLP1 | 2.547 | 4.478 | 7.216 | 2.113 | 3.275 | 5.72 |
| 2min | MLP1 | 2.412 | 4.424 | 7.186 | 2.157 | 3.258 | 5.695 |
| 1min | MLP2 | 2.459 | 4.386 | 7.136 | 2.159 | 3.238 | 5.74 |
| 2min | CNN3 | 2.391 | 4.344 | 7.094 | 2.092 | 3.281 | 5.765 |

Tabelle A.10: Alle Ergebnisse der Datensatzoptimierung für die neuronalen Netze.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1min | LSTM3 | 3.512 | 5.262 | 7.938 | 2.232 | 3.237 | 5.671 |
| 1min | MLP3 | 2.593 | 4.519 | 7.301 | 2.205 | 3.23 | 5.731 |
| 10min | LSTM4 | 2.571 | 4.278 | 7.105 | 2.304 | 3.192 | 5.678 |
| 15min | LSTM1 | 2.404 | 4.231 | 6.93 | 2.108 | 3.326 | 5.767 |
| 20min | LSTM1 | 2.432 | 4.277 | 7.097 | 2.194 | 3.296 | 5.712 |
| 20min | CNN8 | 2.787 | 4.698 | 7.423 | 2.2 | 3.302 | 5.739 |
| 2min | CNN2 | 2.613 | 4.516 | 7.346 | 2.195 | 3.241 | 5.809 |
| 1min | CNN8 | 2.623 | 4.535 | 7.347 | 2.246 | 3.274 | 5.807 |
| 1min | CNN5 | 2.809 | 4.6 | 7.376 | 2.329 | 3.274 | 5.724 |
| 20min | LSTM4 | 2.477 | 4.186 | 6.972 | 2.266 | 3.398 | 5.705 |
| 1min | CNN1 | 2.458 | 4.424 | 7.211 | 2.223 | 3.302 | 5.846 |
| 1min | CNN2 | 2.567 | 4.412 | 7.131 | 2.242 | 3.324 | 5.832 |
| 5min | CNN8 | 2.83 | 4.659 | 7.378 | 2.33 | 3.323 | 5.775 |
| 2min | CNN8 | 2.523 | 4.455 | 7.216 | 2.247 | 3.332 | 5.851 |
| 15min | LSTM4 | 2.497 | 4.349 | 7.15 | 2.32 | 3.377 | 5.741 |
| 1min | CNN4 | 2.63 | 4.546 | 7.211 | 2.27 | 3.383 | 5.827 |
| 5min | CNN4 | 2.747 | 4.531 | 7.257 | 2.294 | 3.359 | 5.83 |
| 2min | CNN4 | 2.582 | 4.475 | 7.19 | 2.236 | 3.368 | 5.89 |
| 5min | CNN5 | 2.817 | 4.545 | 7.281 | 2.408 | 3.356 | 5.8 |
| 1min | CNN7 | 2.883 | 4.736 | 7.432 | 2.39 | 3.414 | 5.79 |
| 1min | CNN3 | 2.537 | 4.412 | 7.12 | 2.416 | 3.437 | 5.939 |
| 2min | CNN6 | 2.775 | 4.638 | 7.302 | 2.421 | 3.544 | 5.882 |
| 5min | CNN7 | 3.128 | 4.802 | 7.459 | 2.62 | 3.477 | 5.824 |
| 2min | CNN5 | 3.02 | 4.683 | 7.43 | 2.633 | 3.418 | 5.87 |
| 10min | CNN8 | 3.249 | 4.927 | 7.563 | 2.575 | 3.522 | 5.906 |
| 10min | CNN4 | 3.25 | 4.903 | 7.503 | 2.631 | 3.492 | 5.888 |
| 1min | LSTM1 | 2.853 | 4.748 | 7.506 | 2.546 | 3.535 | 5.934 |
| 1min | CNN6 | 2.758 | 4.432 | 7.044 | 2.708 | 3.596 | 5.994 |
| 15min | CNN4 | 3.411 | 5.187 | 7.793 | 2.855 | 3.711 | 6.06 |
| 5min | CNN6 | 3.394 | 5.125 | 7.727 | 2.901 | 3.739 | 6.09 |
| 15min | CNN8 | 3.664 | 5.39 | 7.927 | 2.982 | 3.766 | 6.093 |

## Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

| | |
|---|---|
| Ort, Datum | Thies de Graaff |