Maschinelles Lernen zur Schätzung von Ankunftszeiten in der Schifffahrt

Masterarbeit

von

Benjamin Hosenfeld

Matrikelnummer: 1697174

An der Fakultät für Wirtschaftswissenschaften

Digital Service Innovation (DSI)

Karlsruhe Service Research Institute (KSRI) &
Institute of Information Systems and Marketing (IISM)

Erstgutachter:	Prof. Dr. Gerhard Satzger
Zweigutachter:	Prof. Dr. Hansjörg Fromm
Tag der Abgabe:	30. Juni 2021

Declaration of Academic Integrity

I hereby confirm that the present thesis is solely my own work and that if any text passages or diagrams from books, papers, the Web or other sources have been copied or in any other way used, all references—including those found in electronic media—have been acknowledged and fully cited.

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die der/dem Aufgabensteller(in) bereits bekannte Hilfe selbst angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

Karlsruhe, 30. Juni 2021 *Benjamin Hosenfeld*

Zusammenfassung

Ungenaue Schätzungen der Ankunftszeit in der Schifffahrt sorgen für verspätet oder verfrüht eintreffende Schiffe. Die Kapazitäten zum Ent- und Beladen am Hafen stehen nicht jederzeit auf Abruf zur Verfügung, sondern sind fest eingeplant. Abweichend anlegende Schiffe sorgen folglich zu Überlastungen am Hafen. Anschließende Logistik wird unplanbar und letztlich können Engpässe in Lagern zu Knappheit lebensnotwendiger Güter führen. Verwandte Arbeiten zeigen, dass sich maschinelle Lernmodelle zur Behandlung ähnlicher Probleme eignen. In dieser Arbeit wird maschinelles Lernmodell zur präzisen Ankunftszeitschätzung untersucht und löst das Problem unplanbarer Logistikabläufe. Die Grundlage bilden historische Daten des dänischen Automatic Identification System (AIS). Zunächst werden die ursprünglichen Daten des AIS zu Zeitreihen in Form von Anfahrten eines Schiffs zu einem Zielhafen aggregiert. Anschließend werden für 12 Häfen im ersten Schritt maschinelle Lernmodelle *(Basismodelle)* auf ihren eigenen Daten trainiert und evaluiert. Aufgrund geringer Datenmengen können für kleine Häfen keine eigenen Modelle trainiert werden. Durch künstliches Ausdünnen der Datenmenge werden kleine Häfen simuliert. Im zweiten Schritt findet ein Wissenstransfer der eingangs trainierten *Basismodelle* auf jeden anderen der 12 Häfen statt. Unter Einsatz sog. *Transferstrategien* werden verschiedene Vorgehensweisen zum Transfer des neuronalen Netzes exploriert. Das Anpassen der Gewichte verschiedener Layer des Netzes, Fine-Tuning sowie Ausdünnen der Datenbasis werden unterschiedlich kombiniert und angewendet. Die Güte der resultierenden transferierten Modelle wird mit Hilfe des Mean Absolute Error (MAE) bewertet. Die Performance der Basismodelle wird mit der Güte der transferierten Modelle ins Verhältnis gesetzt, um einen Anhaltspunkt für gut performende transferierte Modelle zu erhalten. Der Vergleich durchgeführter Transfers untereinander lässt schließlich Schlussfolgerungen zu, welche Transferstrategie die praktisch besten Resultate erzielt. So werden die besten Ergebnisse beim Transfer von Modellen gewonnen, wenn die Gewichte der letzten vier Layer des angewandten neuronalen Netzes zur Anpassung freigegeben werden und das Modell im Anschluss einem Fine-Tuning-Prozess unterzogen wird. In der Evaluation wird festgestellt, dass die Trainingsdaten zum erfolgreichen Transfer von maschinellen Lernmodellen stark ausgedünnt werden können. Im Rahmen dieser Arbeit schätzen Modelle, die mit 10% der verfügbaren Trainingsbeispiele transferiert wurden, durchschnittlich 4,5 Minuten schlechtere Ankunftszeiten als die entsprechenden Basismodelle.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Marine Lieferketten sind in der heutigen Zeit nicht mehr wegzudenken. Dem Report der United Nations Conference on Trade and Development UNCTAD (2021) zur Folge findet etwa 80% des internationalen Transports auf Seewegen statt. Mit stetig steigendem Trend seit 2009 wurden im Jahr 2019 weltweit 11,08 Mrd. metrische Tonnen Güter im Seehandel abgefertigt (vgl. UNCTAD (2020)). Die ökonomische Tragweite verspäteter Frachtschiffe ist im März diesen Jahres am Suezkanal deutlich geworden: Durch die Blockade entstanden pro Tag laut BBC News (2021) Kosten im Wert von 9,6 Mrd. US-Dollar auf. Sobald ein Frachtschiff ungeplant auf seine Weiterfahrt wartet, entstehen Kosten und Verzögerungen im gesamten Verlauf der anschießenden Versorgungskette. Selbst ohne Katastrophe globalen wirtschaftlichen Ausmaßes stauen sich in Hafenzufahrten Schiffe, die tagelang ungenutzt vor Anker liegen und auf ihre Einfahrtgenehmigung warten. Eine der Ursachen sind unzuverlässige Schätzungen der Ankunftszeit (Buzoianu, 2020). Umweltverschmutzung und ineffizient genutzte Verladungskapazitäten senken die Produktivität am Hafen. Eine präzisere Prognose der Ankunftszeiten in der kommerziellen Schifffahrt trägt somit wesentlich zur besseren Planbarkeit (teil-) mariner Lieferketten bei.

1.1 Motivation

Die kommerzielle Schifffahrt benötigt Planungssicherheit: Verspätete Schiffe müssen entladen werden, was am Hafen zu erheblichem Nachholbedarf und Überlastung führt. Anschließende Logistik verspätet sich und letztlich kann es zu Engpässen in Lagern einzelner Läden kommen. Im Extremfall werden lebensnotwendige Güter knapp. Die Komponenten einer Lieferkette sind von vorangehenden Gliedern abhängig und damit besonders von zuverlässigen Ankunftszeiten des ersten Kettengliedes: (Container)-Schiffen. Eine Lieferung auf ihrem Weg zu verfolgen, gehört heute zum normalen Bestellprozess dazu. Ankunftszeiten lassen sich jederzeit verfolgen (DHL, Amazon, ...), moderne Tracking-Methoden schätzen die Ankunftszeit mit hoher Genauigkeit. Allerdings bezieht sich der Anwendungsfall dieser Arbeit nicht auf terrestrische Routenplanung im Straßenverkehr, sondern auf Schiffe, die einen Hafen ansteuern. Ein Routenplaner wie Google Maps (Google, 2021b) zur Schätzung der Ankunftszeit eines Schiffs am Hafen existiert nicht.

Jedes Transportschiff ist Teil einer Versorgungskette, die als zentraler Bestandteil im *Supply Chain Management (SCM)* den Transport von Gütern umsetzt. SCM plant den Warentransport von ihrer Herkunft zu einem Zielort. Miteinander verbundene und voneinander abhängige Netzwerke modellieren den Warenfluss und kombinieren unterschiedliche Transportmittel (Berthold, 2019). Der Transportweg

formt ein Netzwerk von der Herkunft der Ware über mehrere Zwischenstationen zum gewünschten Zielort. In der Praxis wird ein solches Netzwerk als (gerichteter) Graph modelliert. Ein solcher Graph ermöglicht die Optimierung von Transportwegen anhand einer Zielfunktion. Eine Kante im Graph repräsentiert einen Transportweg zwischen zwei Knoten: Dem aktuellen Urpsprungsort und dem Zielort der Ware. Die gesamte Transportzeit ergibt sich aus der Summe aller Transportwege (Kanten) einer Lieferkette (Pfad), wobei ein Produkt von mehreren vorausgehenden Lieferketten abhängig sein kann (Hitchcock, 1941). Diese verwobenen Abhängigkeiten verdeutlichen die Relevanz zuverlässiger Planung im SCM. Sobald sich Warenankünfte in einem Teil des Netzwerks von Lieferketten verspäten, betrifft die Verspätung alle abhängigen Pfade.

1.2 Konsequenzen verspäteter Schiffe

Abweichungen von vertraglich vereinbarten Ent- und Beladezeiten eines Schiffs beeinflussen die Kapazitäten am Hafen. Wann die Fracht eines Schiffs umgeschlagen werden kann, hängt von den verfügbaren Kapazitäten am Hafen und der *tatsächlichen* Ankunftszeit des Schiffs ab. Legt ein Schiff planmäßig am Hafen an, sind die allozierten Entladekräfte in der Lage, im zeitlich zugeteilten Rahmen ein Schiff zu ent- und beladen. Folglich bestimmt sich die Liegezeit eines Schiffs aus der benötigten Zeit zum Ent- und Beladen eines Schiffs. Für überschrittene Liegezeiten, erhebt der Frachtanbieter eine Geldstrafe. Im Frachtverkehr wird zwischen zwei unterschiedlichen Arten von Geldstrafen bei der Lagerung von Containern unterschieden.

Demurrage ist eine Gebühr für volle Container, die bei einer Überschreitung der Karenzzeit bis zur Entladung eines Schiffs erhoben wird (Lexico.com, 2021b). Der Anwendungsbereich wird in der Literatur vom Entladen eines Schiffs auf den Abtransport vom Lager des Hafens erweitert (Trade Finance Global (2021a), Ocean Insights (2021b)).

Detention bezeichnet eine Gebühr für Container, die vom Importeur abgeholt, aber nicht zum vereinbarten Rückgabeort transportiert wurden (Ocean Insights, 2021b).

Die Abkürzung *D&D* wird in der Literatur zur Bezeichnung der beiden Arten von Strafzahlungen verwendet. Um dynamisch auf Abweichungen von Lieferungen reagieren zu können, lagern Häfen Fracht üblicherweise für drei bis fünf Tage kostenlos (Trade Finance Global, 2021b). Jeder Hafen hat individuelle Regelungen zur Lagerung von Fracht. Beispielsweise wird am Hamburger Hafen für den Export eines 20-Fuß ISO-Standard Container aus Deutschland gemäß Area Central Europe (2021) in den ersten 10 Tagen keine Gebühr erhoben, ab Tag 11 30 € bis hin zu 55 € ab dem 18. Tag. Beim Import der gleichen Fracht fallen in den ersten 7 Tagen keine Gebühren an, ab dem 8. Tag 35 € und 55 € nach 15 Tagen. Die Strafzahlungen

werden täglich erhoben und variieren je nach Maß und Art der Fracht, Import oder Export und finden für unterschiedliche Länder differenziert Anwendung.

Eine Ursache von Demurrage sind verspätete Ankunftszeiten von Schiffen. Die Ent- und Beladekapazitäten am Hafen sind begrenzt. Kommen zu viele Schiffe gleichzeitig am Hafen an, wird die Kapazität eines Hafens überschritten. Folglich kann nicht jedes Schiff zu jeder beliebigen Zeit abgefertigt werden. Die Abfertigung verspäteter Schiffe verzögert sich weiter, wenn die Kapazitäten am Hafen für andere Schiffe verplant sind. Anschließende Logistik verzögert sich dementsprechend oder muss neu geplant werden. Die folgende Rechnung veranschaulicht anfallende *D&D* - Gebühren, wenn die Ladung eines Schiffs nicht planmäßig abgefertigt wird: Das größte Containerschiff *MSC Gulsun* kann bis zu 23.756 Container transportieren (BBC, 2019). Für eine vollständig homogene Fracht aus 20-Fuß ISO-Standard Containern, sind beim Import ab dem 8. Tag tägliche Gebühren in Höhe von 831.460 € fällig. Solche Gebühren vervielfachen sich, wenn mehrere Schiffe nicht fristgerecht Ent- und Beladen werden können. Das konstruierte Beispiel zeigt die Relevanz zuverlässiger Planung und präziser Ankunftszeiten zur effizienten Abfertigung anfallender Fracht an Häfen.

1.2.1 Port Congestion

In Pressemeldungen wird *Port Congestion* als schwerwiegendes Problem in der kommerziellen Schifffahrt bezeichnet. Ocean Insights (2021b) führt Port Congestion als einen Haupttreiber bei der Erhebung von *D&D*-Strafzahlungen an. In der Literatur und den Medien findet der Begriff im engen Sinn Anwendung, wenn Schiffe auf die Einfahrt in einen Hafen warten und Staus bilden Shiani (2019). Der Begriff wird vereinzelt verwendet, um Verzögerungen beim Ent- und Beladen von Schiffen zu bezeichnen. Bremerhaven hat, Stand 2019, nach Shiani eine durchschnittliche Wartezeit von 55 Stunden, bis ein Schiff am Hafen anlegen kann. Wie Shiani anführt, ist der Trend steigend.

Staus vor Hafeneinfahrten wirken sich auf die nachfolgende Versorgungskette aus. Kosten akkumulieren sich aufgrund ungenutzter Transportkapazitäten, zusätzlich benötigtem Kraftstoff und *D&D*-Gebühren (Ocean Insights, 2021a). Die folgenden Beiträge zeigen auf, in welchem wirtschaftlichen Ausmaß sich Port Congestion auf Unternehmen auswirkt. Die Lagerbestände von Foot Locker sind im vierten Quartal 2020 aufgrund von Port Congestion im Vergleich zum vierten Quartal des Vorjahres um 23,6% ($ 923 Mio) gesunken. Leonard (2021), Foot Locker (2021)

Um Knappheiten in Lagern vorzubeugen, stocken von Port Congestion betroffene Unternehmen ihre Lagerkapazitäten auf. Beispielsweise hat Nike aufgrund von unzuverlässigem Transport und Port Congestion angekündigt, jährlich $ 200 Mio

zu investieren, um die verfügbaren Lagerbestände auf 7 bis 14 Tage zu erweitern. (National Customs Brokers & Forwarders Association of America , NCBFAA)

1.2.2 Umweltbelastung durch Port Congestion

Einer Studie des kalifornischen Hafens in Long Beach zur Folge haben Staus im ersten Quartal des Jahres 2015 die Luftqualität im Jahr 2016 negativ beeinflusst (Navingo, 2016). Demnach stieg die Konzentration von Stickoxiden und Treibhausgasen um 2% bzw. 7%. Zur Behebung des Problems wurde unter anderem Energie von der Küste beschafft und fortschrittliche Technologien eingesetzt.

Czermański u. a. (2021) untersuchten Ansätze zur Reduktion von Emissionen in der Container-Schifffahrt. Sie stellen fest, dass Öl- und Chemikalientanker 17% bzw. 14% ihres gesamten Energieverbrauchs beim Warten auf freie Ankerplätze verbrauchen gefolgt von Liquid Natural Gas (LNG) Tanker und Stückgut Frachtern mit 11% bzw. 9%. Eine der möglichen Strategien zur Reduktion der Emissionen sieht Optimierungen der Route hinsichtlich Geschwindigkeit und *anderen* Routen-Parametern vor (Czermański u. a., 2021, S. 2). Auf Basis präzise geschätzter Ankunftszeiten ist vor Ankunft am Hafen ersichtlich, wann die Kapazitäten eines Hafens überschritten werden. Entsprechend können Schiffe vor Ankunft im Hafen ihre Geschwindigkeit anpassen, um sich effizient abfertigen zu lassen und Kraftstoff zu sparen.

1.3 Daten-basierte Erschließung von Wissen

Zur Steigerung der Produktivität und Effektivität von Häfen wurden in den letzten Jahrzehnten Investitionen in Digitalisierung und Automatisierung getätigt. Heilig u. a. (2019) stellen evolutionär die Implementierung neuer Technologien seit den 1980ern vor. Besonderer Fokus liegt auf der zweistufigen Betrachtung einerseits technisch realisierbarer Innovationen mit Blick auf operationale Umsetzbarkeit. Hohes Potential erwarten sowohl (Heilig u. a., 2019, P. 6) als auch Hafen-/Schiffbetreiber im Bereich Datenanalyse. Doch besteht eine Diskrepanz zwischen bereits existierenden Datensammlungen und deren Aufbereitung in BI-Systemen, um tatsächlich operativ Nutzen erzielen zu können. Heilig u. a. (2019) beschreiben existierende Daten als *under-processed* oder *under-analyzed*, um tatsächlich Mehrwert zu schaffen. Kannan u. a. (2018) führen an, dass der Wert von Daten nicht a-priori messbar ist. Sie müssen erst in einer Anwendung verarbeitet und das Ergebnis bewertet werden. Sie identifizieren Facetten zur Bestimmung des Werts von Daten wie beispielsweise das Layout des Datensatzes, Alter, Menge und Format der Daten (Kannan u. a., 2018, S. 6-8). Je besser zugänglich, aktueller, gut formatiert und größer das Datensatz, desto höher der (potentielle) Wert der Daten. Das hauptsächliche Problem

besteht dennoch weiter: Der Großteil gesammelter Daten bleibt ungenutzt. Lucidworks schätzt, dass täglich 7,5 Septillionen Gigabytes an Daten weltweit erhoben werden, wovon lediglich 10% analysiert werden (Barker, 2016). Die restlichen 90% bezeichnet Barker als *Dark Data*, die potentielle Sicherheitsrisiken darstellen können oder ungenutzte Chancen zur Profitsteigerungen verbergen. Hauptgrund dafür ist laut Lucidworks meist die beschränkte Kapazität der Analysen und Werkzeuge, die von der Menge an Daten überwältigt sind. Unstrukturierte Datensätze oder inkompatible Datenformate stellen eingesetzte Tools vor weitere Schwierigkeiten im Umgang mit gesammelten Daten. Priceonomics (2019) zeigen eine Diskrepanz ungenutzter Dark Data zwischen Lucidworks' Erkenntnissen und der Schätzung von Unternehmern auf: Demnach schätzen 1300 Geschäftsführer 55% der gesammelten Daten ihrer Unternehmen als Dark Data ein, was einen Unterschied von 35% gegenüber der Schätzung von Lucidworks aufweist.

Wie angeführt, ist der Anteil tatsächlich produktiv analysierter Daten gering. Die von Unternehmern wahrgenommene Effektivität von Daten-Analyse Verfahren unterscheidet sich signifikant vom tatsächlich analysierten Anteil der Daten. Maschinelle Lernalgorithmen können einerseits explorativ zur „Erkundung" eines Datensatzes *(unsupervised, vgl. Kap. 3.1.2)* angewandt werden (*bspw. Clustering*-Verfahren). Die heutzutage hohe Dimensionalität von Daten behandelt der Bereich des *Representation Learnings* (Schölkopf u. a., 2021) *(Preprocessing)*. Gleichwohl finden anwendungsspezifische Vorverarbeitungen Anwendung, um *rohe* Prozessdaten in brauchbare Datensammlungen zu transformieren. Klassifikatoren oder Schätzer können auf Basis dieser vorverarbeiteten Daten enthaltenes Wissen extrahieren und Analysen durchführen. Analog zur menschlichen Fähigkeit, auf Basis von Beobachtungen zu lernen, muss sich ein (Lern-) Algorithmus Zusammenhänge, Strukturen und letztlich Wissen anhand von Beispielen (Datenbasis) erschließen. Der Bereich des *maschinellen Lernens* nutzt das beschriebene Paradigma: Lernen durch Beobachtung von Trainingsdaten zur Erschließung von Wissen (Heilig u. a., 2019, S. 2). Während traditionelle, vorwiegend statistische, *Data Mining* Methoden mit der überwältigenden Menge an Daten nicht produktiv anwendbar sind, treten maschinelle Lernalgorithmen wie beispielsweise Deep Learning Techniken mit *Convolutional Neural Networks (CNN)* und *Recurrent Neural Networks (RNN)* in den Vordergrund (Wang u. a., 2019). Besonders mit steigender Rechenkapazität von Grafik-Prozessoren und herausragenden Ergebnissen gewinnt Deep Learning an Beliebtheit im Umgang mit großen Datenmengen (Qui u. a., 2016, S. 3 f.).

1.4 Anwendungsbereiche neuronaler Netze

Maschinelle Lernmodelle erschließen Wissen durch Training auf einer Datenbasis. Je nach Anwendungsbereich variiert die Art der Datenbasis, mit deren Hilfe ein Modell trainiert und eingesetzt werden soll. Bilderkennung ist ein typischer Anwendungsbereich des maschinellen Lernens und neuronaler Netze. Das Ziel eines Netzes ist die Erkennung von Motiven auf Bildern. Folglich setzt sich der Datenbestand aus unterschiedlichen Bildern (gleicher Auflösung) zusammen, deren Inhalt von einem Modell klassifiziert werden soll. Die Aufgabe eines Netzes kann beispielsweise die Einordnung von Bildern gemäß einer Tierart wie Hund, Katze oder Hase sein *(Klassifikation)*. Ein solches Netz zur Klassifikation von Bildern gibt üblicherweise an, wie wahrscheinlich ein Bild einer Klasse zugeordnet wird. Eine mögliche Ausgabe könnte $P(Hund) = 0.95$, $P(Katze) = 0.045$ und $P(Hase) = 0.005$. Das Netz schätzt folglich mit einer Wahrscheinlichkeit von 95%, dass sich auf dem analysierten Bild ein Hund befindet.

Der Anwendungsbereich von neuronalen Netzen beschränkt sich nicht auf die Klassifikation von Bildern. Automatisch generierte Prozessdaten sind oftmals Sensordaten, die den aktuellen Zustand einer Maschine beschreiben. Einzelne Datenpunkte können Ausprägungen beliebiger Art enthalten und eine hohe Dimensionalität aufweisen. Sensoren liefern einen kontinuierlichen Datenstrom, der den Zustand einer Anlage über die Zeit abbildet. Die so entstehende Zeitreihe kann, ähnlich der Eingabe von Bildern, von einem neuronalen Netz verarbeitet werden. Populäre Anwendungen sind die Detektion von Anomalien (Chalapathy u. Chawla (2019), Pang u. a. (2021)), (Wetter-) Forecasting (Liu u. a. (2014), Lim u. Zohren (2021)) oder Spracherkennung (Roger u. a., 2020). Gamboa (2017) bietet eine Übersicht weiterer Modelle im praktischen Einsatz.

1.5 Maschinelles Lernen zur Schätzung von Ankunftszeiten auf historischen Daten

Aufgrund manueller Eingabe sind die im dänischen Automatic Identification System (AIS) (Danish Maritime Authority, 2021) Datensatz enthaltenen Ankunftszeiten ungenau. Die Ankunftszeit muss manuell aktualisiert werden, um Änderungen im Verlauf der Route korrekt abzubilden. Basierend auf den aktuellen Gegebenheiten eines Schiffs könnte ein maschinelles Lernmodell zur automatischen Schätzung der Ankunftszeit eine signifikante Verbesserung gegenüber manueller Schätzung erzielen. Dieses Modell wird in dieser Arbeit als *ETA Sensor* bezeichnet. Weiter könnte der ETA Sensor auf Basis aktueller AIS Daten kontinuierlich die Schätzung der Ankunftszeit, ohne manuelle Interaktion der Crew anpassen. Ein solcher ETA Sen-

sor löst das Problem nicht aktueller Schätzungen der Ankunftszeit, die auf Basis veralteter Daten getroffen wurden. Indem live Daten zur Bestimmung der voraussichtlichen Ankunftszeit verwendet werden, sind beispielsweise Veränderungen des Kurses und der Geschwindigkeit implizit abgebildet und fließen in die Schätzung der Ankunftszeit ein. Weiter können fehlerhafte Eingaben bei manuellem Setzen der Ankunftszeit ausgeschlossen werden. Die im Datensatz des AIS enthaltene Schätzung der Ankunftszeit ändert sich innerhalb der Anfahrt eines Schiffs meist nicht. In manchen Fällen ist die geschätzte Ankunftszeit nicht angegeben, fehlerhaft, liegt weit in der Zukunft oder gar in der Vergangenheit - vermutlich aus Eintragungen für vorherige Routen.

Der in dieser Arbeit verwendete Datensatz des AIS besteht aus Statusmeldungen von Schiffen auf dem Weg zu unterschiedlichen Häfen. Ein Schiff sendet in zeitlichen Abständen aktuelle Informationen an das *Automatic Identification System (AIS)*, die chronologisch persistiert werden. Ein maschineller Lernalgorithmus muss folglich, basierend auf historischen Zeitreihen, eine Schätzung zur Ankunftszeit eines Schiffs am Zielhafen treffen können. Lernen eines Sachverhalts auf historischen Daten ist mit unterschiedlichen maschinellen Lernalgorithmen umsetzbar. Die vorliegende Problemstellung zur Prädiktion von Ankunftszeiten könnte als Regressionsproblem mit einem neuronalen Netz umgesetzt werden. Demnach ist die Problemstellung dieser Arbeit nicht wie im vorigen Beispiel die Kategorisierung von Eingaben anhand vordefinierter Klassen sondern die Vorhersage einer Dauer bis zur Ankunft eines Schiffs am Zielhafen.

Forschungsfrage 1: Ist Deep Learning auf historischen Daten des dänischen Automatic Identification System (AIS) ein geeigneter Ansatz zur Schätzung von Ankunftszeiten in der Schifffahrt?

1.6 Domänen-bedingte Daten-Knappheit

Die Datenbasis bildet das Rückgrat eines jeden maschinellen Lernalgorithmus. Um einen Sachverhalt in ausreichender Genauigkeit repräsentieren zu können, sind nicht immer ausreichend Daten vorhanden. Gründe dafür können hohe Kosten bei der Erhebung von Daten oder eine natürliche Knappheit von Beispielen in der Umgebung der Problemstellung sein. Besonders im Bereich des *Supervised Learning (vgl. 3.1.2, 1. Abs.)* entstehen hohe Kosten, wenn Daten in einem Vorverarbeitungsschritt manuell annotiert werden müssen. Incze (2019) weist die Kosten eines maschinellen Lernprojekts drei Kategorien zu. Kategorie bezieht sich auf Kosten für Daten. Die angeführte Beispielrechnung räumt je nach Komplexität der Annotationen Kosten in Höhe von $10.500 - $85.000 zur Generierung von 100.000 gelabelten Trainingsdaten

ein. In dieser Rechnung existieren bereits *ungelabelte* Daten. Falls aber die Generierung oder Akquise von Daten aus natürlichen Gründen begrenzt ist, muss zum Training eines maschinellen Lernalgorithmus eine Methode angewendet werden, die trotz geringer Anzahl Trainingsbeispiele ein aussagekräftiges Modell erzeugt. Wie viele Daten zur Generierung eines maschinellen Lernmodells benötigt werden, ist nicht pauschal für jeden Anwendungsfall äquivalent. Mitsa (2019) stellt in ihrem Artikel für unterschiedliche Arten des maschinellen Lernens Daumenregeln zu Mindestgrößen der notwendigen Trainingsdaten zusammen. Die Bestimmung der Mindestzahl an Trainignsdaten hängt stets von der Architektur des neuronalen Netzes ab. Die Anzahl Parameter und Größe von Eingaben wird einbezogen. Empirische Versuche beziffern die Anzahl notwendiger Trainingsbeispiele auf das 10-fache der Anzahl Modell-Parameter, was in der Literatur als *rule of 10* oder *One in ten rule* bezeichnet wird (Haldar (2015), Peduzzi u. a. (1996))

Die Domäne dieser Arbeit ist das Schätzen von Ankunftszeiten für Schiffe, die einen Hafen ansteuern. Als Datengrundlage dient der AIS-Datensatz der *Danish Maritime Authority (2021)*. Alle Daten von Schiffen mit selbem Zielhafen sollen als Trainingsdaten für einen Hafen genutzt werden. Je mehr Schiffe einen Hafen anfahren, desto mehr Daten stehen zum Training des Modells zur Verfügung. An häufig angefahrenen Häfen legen mehr Schiffe an, wodurch mehr Daten generiert werden. Das Training eines Modells für große Häfen stellt kein Problem hinsichtlich des Datenvolumens dar. Allerdings sollen nicht ausschließlich Modelle zur Schätzung von Ankunftszeiten für große Häfen trainiert werden. Möglichst viele Häfen unterschiedlicher Größe, die im Datensatz der DMA angefahren werden, sollen einen individuellen ETA Sensor erhalten. Problematisch ist das Training solcher Häfen, die selten angefahren werden und folglich wenige Daten zum Training eines eigenen Modells besitzen. Ein weiteres Problem tritt auf, wenn ein hoher Anteil der verfügbaren Daten eines Hafens vom selben Schiff auf der selben Anfahrt stammen. Ein Modell, das lediglich auf einen Teil möglicher Anfahrtswege (und Schiffstypen) spezialisiert ist, kann keine zuverlässigen Schätzungen für *fremdartige* Schiffe abgeben, die sich auf *unbekannten* Anfahrtswegen befinden.

1.7 Training von Modellen auf geringen Datenmengen

Häfen mit zu geringen Datenmengen, um ein eigenständiges maschinelles Lernmodell zu generieren, sollen dennoch ETA Sensor zur Schätzung von eintreffenden Schiffen trainieren können. Folglich muss ein maschineller Lernalgorithmus dazu in der Lage sein, auf Basis *mangelhafter* Repräsentation einer fremdartigen Domäne, ähnliche Merkmale und Strukturen zu extrahieren, wie ein Algorithmus auf ausreichender Datenbasis. Das folgende Beispiel verdeutlicht das potentielle Problem, gelerntes

Wissen einer bekannten Domäne auf einen anderen Anwendungsfall zu übertragen. Eine Person lernt die Strategie zur Lösung einer *neuartigen* Aufgabe. Sie nutzt alle zugehörigen Ressourcen, um die Aufgabe zu lösen. Das gewonnene Wissen stützt sich allein auf die betrachteten Ressourcen. Die Person ist nun in der Lage, unter Einsatz des gewonnen Wissens weitere Aufgaben der gleichen Art zu lösen. Bekommt die Person eine ähnliche Aufgabe, die zur Lösung jedoch eine andere *Strategie* (anderes Wissen) benötigt, kann die Aufgabe ggf. nicht korrekt bewältigt werden. Lernt ein Mensch beispielsweise das Vorgehen, einen Apfel mit einem Sparschäler zu schälen, kann er Äpfel und Lebensmittel mit *ähnlicher* Schäl-Technik schälen. Soll er aber eine Banane schälen, ist das gewonnene Wissen des Schälens mit einem Sparschäler die falsche Methode zum Schälen einer Banane. Kennt ein Mensch die unterschiedlichen Methoden zum Schälen von Obst und Gemüse, kann er die Strategie je nach Anwendungsfall anpassen und die entsprechende Vorgehensweise anwenden. Selbst bis dato unbekannte Früchte und Gemüse, können unter Anwendung des breiten Wissens über diverse Schälmethoden von ihrer Schale befreit werden. Ein maschinelles Lernmodell zur Schätzung von Ankunftszeiten verhält sich analog: Ist das beim Training gelernte Wissen anwendbar, kann das Modell zuverlässige Prognosen ermitteln. Entsprechend *unpassend* ist die Anwendung gelernten Wissens auf Daten einer *unbekannten* Domäne. Um dennoch fremdartige Daten zuverlässig verarbeiten zu können, muss ein maschinelles Lernmodell über allgemeingültiges Wissen verfügen, das an einen spezifischen Anwendungsfall angepasst wird, analog zum Anwenden einer Schälmethode auf unbekannte Frucht- oder Gemüsesorten. Zur Generierung eines Modells für einen Hafen mit wenigen Daten kann folglich allgemeines Wissen zur Schätzung von Ankunftszeiten in der Schifffahrt hilfreich sein. Ein Schätzer, der auf Basis eines ausreichend großen Datensatzes generiert wurde, hat Wissen aus vielen unterschiedlichen Anfahrten und Schiffstypen extrahiert und ist an den zugehörigen Zielhafen angepasst. Demnach könnte ausschließlich das spezifische Wissen über den zugehörigen Zielhafen angepasst werden, um einen Schätzer für einen Hafen mit wenigen Daten zu erstellen.

Empirischen Studien zur Folge extrahieren Convolutional Neural Networks (CNN) mit zunehmender Tiefe allgemeinere Merkmale (*high level Features*). Khandelwal (2020) visualisiert die Arbeitsweise eines CNNs am Beispiel der Klassifikation von Bildern. Zunächst werden Kanten und abstrakte Formen wie Ecken erkannt. Diese allgemeinen Merkmale existieren analog im vorliegenden Fall beim Training auf historischen Daten des AIS Datensatzes (Zeitreihen). Vorstellbar sind hafenübergreifende Merkmale wie das Fahrverhalten von Schiffen gleichen Typs, die dieses allgemeine Wissen repräsentieren, könnten beim Training eines Modells für einen anderen Hafen übernommen werden. Die letzten Schichten eines neuronalen Netzes bilden anwendungsspezifisches Wissen ab. Ein Beispiel für hafenspezifische Merkma-

le ist dessen Zufahrtsweg. Der Hamburger Hafen ist über die Elbe mit der Nordsee verbunden, was seine Zufahrt im letzten Streckenabschnitt verengt. Im Gegensatz dazu steht der Hafen von Skagen (Dänemark). Er liegt direkt an der Nordsee und kann aus unterschiedlichen Richtungen angefahren werden. Der Transfer eines bereits trainierten Modells auf einen anderen Hafen könnte die genannten Probleme im Zusammenhang mit geringen Datenmengen lösen. Dazu muss die Genauigkeit der Schätzungen des transferierten Modells in einem angemessenem Verhältnis stehen.

Forschungsfrage 2: Inwiefern kann Transfer-Lernen die Schätzung von Ankunftszeiten von Schiffen unterstützen und wie verhalten sich transferierte im Vergleich zu klassischen Modellen?

Forschungsfrage 3: Kann der Prozess des Transfers eines trainierten Modells auf einen anderen Kontext übertragen werden?

1.8 Forschungsdesign

Zur Begründung, Strukturierung und Evaluation wendet diese Arbeit die Methodik des *Design Science Research (DSR)* nach Hevner u. Chatterjee (2010) an. Demnach soll Forschung (1) wissenschaftlichen Qualitätsansprüchen genügen, (2) reale Problemstellungen adressieren und (3) praktisches Design-Wissen generieren (Hevner u. Chatterjee, 2010, S. 211). Ziel ist die Erstellung eines *Artefakts*, das den Kriterien der Relevanz *(Relevance)* und Rigorosität *(Rigor)* genügt (Hevner u. Chatterjee, 2010, S. 12). Abbildung 1 veranschaulicht, wie sich der DSR-Forschungsprozess in einer *Umgebung* begründet *(Relevanz-Zyklus)* und Forschungserkenntnisse aus dem Forschungsprozess *(Design-Zyklus)* mit Grundlagen der zugehörigen *Wissensbasis (Rigor-Zyklus)* weiterentwickelt.

Definition 1: Artefakt: *Das Hauptanliegen von Forschung gemäß des Design Science Research Paradigma ist die Generierung, Evaluation und Optimierung von Artefakten. Ein Artefakt ist ein soziotechnisches (Zwischen-) Resultat, dessen Instanzen zur Lösung der Problemstellung praktisch eingesetzt werden können (Relevance) und dessen neue Evaluations-Erkenntnisse einen Beitrag zur Wissensbasis liefern können (Rigor). Gregor u. Hevner (2013) (S. 340 f.)*

Forschungsaktivitäten im Sinne des DSR entspringen einer Problemstellung aus der Realität. Die zugehörige *Umgebung* bzw. *Anwendungsdomäne* definiert die Rahmenbedingunen und bietet Chancen, die durch neue Forschungserkenntnisse erschlossen werden können. Der höchste Anteil von Forschungsaktivitäten baut auf vorhandenes Wissen auf. Die zum Zeitpunkt der Forschung aktuelle *Wissensbasis* dient als

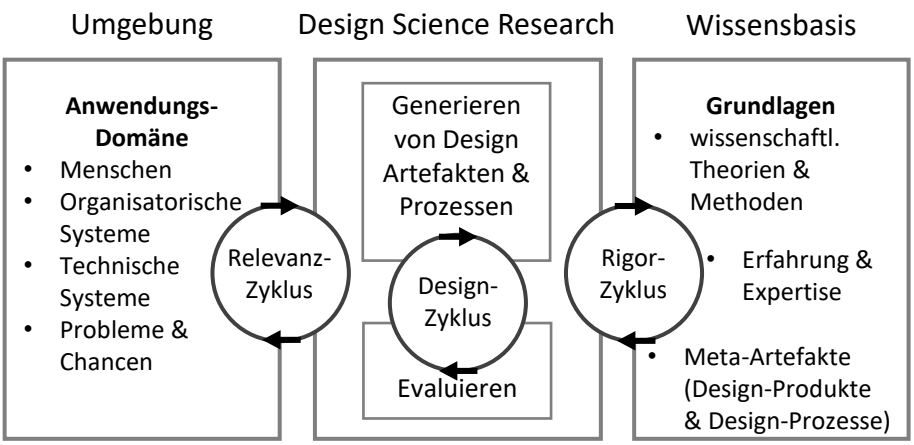

Abbildung 1: Design Science Research (DSR) Framework zur Einordnung von Forschungsaktivitäten. Der Forschungsprozess findet im im Design-Zyklus der Design Science Research Box statt. Er entspringt und interagiert mit seiner Umgebung mittels des Relevanz-Zyklus. Im Forschungsprozess angewandte Grundlagen werden im Rigor-Zyklus identifiziert und Erkenntnisse der Forschungsergebnisse der Wissensbasis hinzugefügt. (Hevner, 2007)

Grundlage zur Weiterentwicklung von Technologien und Methoden, sammelt Erfahrung oder bringt neuartige Design-Produkte hervor. (Hevner, 2007)

Zur Einordnung eines Forschungsprozesses in seine Umgebung dient der Relevanz-Zyklus. Er überprüft und verdeutlicht den Nutzen des Artefakts in seiner Forschungsumgebung. Effektiver DSR soll Probleme in Anwendungen aus der realen Welt adressieren und Beiträge zur Verbesserung in der realen Anwendungsumgebung hervorbringen (Gregor u. Hevner, 2013, S. 342). Abseits des Anregens zur Forschung liefert der Relevanz-Zyklus Kriterien zur Bewertung von Evaluationsergebnissen des Artefakts im Forschungsbereich. Faktisch fließt das Endergebnis des DSR (Artefakt) in die Umgebung zurück, aus der es stammt, um es in realen Szenarien zu untersuchen. Die Ergebnisse einer Evaluation bestimmen, ob weitere Iterationen des Relevanz-Zyklus erforderlich sind. Gründe hierfür können Ergebnisse sein, die die gewünschten Anforderungen gemäß erkannter Chancen aus der Umgebung nicht erfüllen oder die Problemstellung nicht hinreichend lösen. (Hevner u. Chatterjee, 2010, S. 17)

Forschung baut stets auf existierendem Wissen auf. Der Rigor-Zyklus ermittelt, welches Wissen als Basis für das Design des Artefakts herangezogen wird (Hevner u. Chatterjee, 2010, S. 17). Beim Bestimmen der Grundlagen und verwandten Theorien

des Forschungspojekts stellt der Rigor-Zyklus sicher, dass die Problemstellung *innovativen* Charakter aufweist und nicht als *Routine-Design* ausschließlich bekannte Theorien oder Prozesse anwendet. (Hevner u. Chatterjee, 2010, S. 18)

Die eigentliche Forschung eines DSR-Projekts wird im Design-Zyklus umgesetzt. Dieser Prozess generiert und evaluiert Artefakte, die auf Basis der Evaluationsergebnisse verbessert werden. Hevner und Chatterjee betonen den Unterschied zwischen der Abhängigkeit des Design-Zyklus von den beiden anderen Zyklen und dem unabhängigen Forschungsprozess selbst innerhalb des Design-Zyklus. Folglich sind mehrere interne Iterationen des Design-Zyklus zur Entwicklung eines einsetzbaren Artefakts notwendig, bevor Ergebnisse in den Relevanz- und Rigor-Zyklus einfließen können (Hevner u. Chatterjee, 2010, S. 19). Der bereits verwendete Begriff eines Artefakts im Sinne des DSR ist das resultierende Produkt aus dem iterativen Design-Prozess (-Zyklus), wobei *Design* im DSR als *„[...] building software artifacts which solve a human problem.“* bezeichnet wird (Hevner u. Chatterjee, 2010, S. 2).

Die Anwendung des DSR ermöglicht die Zuordnung der Inhalte dieser Arbeit zu den vorgestellten Prozessschritten. (Hevner u. Chatterjee, 2010, S. 20) bieten eine Checkliste mit acht Fragen, die den drei Prozessen (Zyklen) aus Abbildung 1 zugeordnet sind und deren Beantwortung den Forschenden unterstützen soll. Tabelle A9 zeigt die Zugehörigkeit der Fragen zu einem Zyklus und den Abschnitt dieser Arbeit, der die Frage beantwortet. Erkenntnisse aus der Evaluation der transferierten Artefakte tragen zur Erweiterung der Wissensbasis bei.

1.9 Aufbau der Arbeit

Das Problem unzuverlässiger Schätzungen von Ankunftszeiten in der Schifffahrt soll mit einen maschinellen Lernansatz gelöst oder verbessert werden. Zunächst sollen anhand der Datensätze des dänischen Automatic Identification System (AIS) geeignete Beispiele zum Training eines Modells identifiziert werden. Die Architektur des Modells (vgl. Kapitel 5.4) trainiert auf auf den generierten Trainingsbeispielen und gibt eine Schätzung der Ankunftszeit ab. Die trainierten Modelle werden anhand ihrer mittleren absoluten Abweichung der Schätzung evaluiert und gegenübergestellt. Zum Transfer eines Modells auf einen anderen Hafen werden alle Häfen mit einem Basis-Modell permutiert. Alle Paare aus Basis- und Transfer-Modell werden erneut evaluiert und zyklisch Anpassungen vorgenommen. Unterschiedlicher Konfigurationen von Transfers beschreiben die zyklischen Anpassungen und dienen der Nachvollziehbarkeit und Kommunikation. Jedes auftretende Modell ist ein Artefakt im Sinne von Hevner u. Chatterjee (2010), Gregor u. Hevner (2013).

Im Folgenden führt Kapitel 2 mit verwandten Forschungsarbeiten ähnliche Ideen ein und zeigt den aktuellen Stand der Forschung auf. Kapitel 3 legt daraufhin Grundstei-

ne und stellt Techniken vor, worauf diese Arbeit aufbaut. Das Forschungsparadigma des Design Science Research (DSR), woran sich der Forschungsprozess dieser Arbeit orientiert, wird in Kapitel 4 erläutert, ordnet die Thematik in seine praktische Umgebung ein und zeigt den möglichen Beitrag zur Forschung auf. Der Forschungsprozess selbst wird in Kapitel 5 beschrieben. Die vollständige Pipeline von Datenvorverarbeitung, über das Training von Modellen bis hin zur Optimierung und dem Transfer von Modellen werden vorgestellt. Kapitel 6 stellt die angewandte Methode zur Evaluierung vor und diskutiert die Resultate auf Basis von Plots. Abschließend leitet Kapitel 7 die unmittelbaren Folgerungen der Erkenntnisse dieser Arbeit ab und stellt mögliche anschließende Problemstellungen vor.

2 Verwandte Literatur

Existierende Forschung hat gezeigt, dass mittels maschineller Lernmodelle zuverlässige Schätzer für Ankunftszeiten umsetzbar sind. Die Problemstellung des Schätzens von Ankunftszeiten ist nicht auf die Schifffahrt begrenzt, sondern lässt sich auf unterschiedliche Transportnetzwerke wie den Flugverkehr (Balster u. a., 2020) anwenden. Aktuelle Forschungsansätze behandeln die Problemstellung überwiegend mit Routen-basierten Ansätzen (vgl. Kap. 2.1), wobei die Zugehörigkeit eines Transportmittels zu einer Route bestimmt und die Dauer zur Bewältigung der restlichen Route zur Schätzung der Ankunftszeit genutzt wird. Andere Ansätze kategorisieren die Abweichung eines Transportmittels zur geplanten Ankunftszeit (vgl. Kap. 2.2). Hierbei wird die Ankunftszeitschätzung als Unsicherheitsproblem interpretiert.

2.1 Routen-basierte Schätzung von Ankunftszeiten

Eine intuitiv aussagekräftige Information zur Schätzung der Ankunftszeit eines Schiffs auf dem Weg zu einem Hafen ist die Zuordnung einer Route. Der Ansatz von Park u. a. (2021) basiert auf der Identifikation möglicher Trajektorien eines Schiffs. Als Datengrundlage dienen AIS Datensätze, worauf im ersten Schritt mit klassischen Data Mining Methoden und *Reinforcement Learning* (Sutton u. Barto, 2020) die Zugehörigkeit einer Route ermittelt wird. Ist die Route bekannt, kann die Entfernung des Schiffs zum Hafen in Abhängigkeit der aktuellen Position berechnet werden. Zur Schätzung der Ankunftszeit wird die voraussichtliche Geschwindigkeit *(Speed Over Ground, SOG)* unter Einsatz der Markov-Kette (Lexico.com (2021a); Kuntz (2020)) und Bayes-Statisik für den restlichen Abschnitt der Route geschätzt.

Alessandrini u. a. (2019) behandeln eine ähnliche Ausgangslage zur Schätzung von Ankunftszeiten in der Schifffahrt : Aus historischen (Tracking-) Daten des dänischen Automatic Identification System (AIS) in Kombination mit LRIT *(Long Range Identification and Tracking)* (EMSA, 2014) sollen Reise- und Ankunftszeiten geschätzt werden. Die zur Lösung der Problemstellung angewandte Strategie und Technologie unterscheidet sich aber signifikant von Park u. a. (2021). Zunächst bestimmt ein vorgelagerter Pfadsuche-Algorithmus in einer dynamischen Umgebung (Alessandrini u. a., 2019, S. 8f.) die optimale Route zwischen zwei geografischen Punkten. Dessen Output repräsentiert den restlichen Weg, den das Schiff bis zur Ankunft am Zielhafen zurücklegen muss. Die Ankunftszeit ergibt sich durch die Geschwindigkeit des Schiffs auf der restlichen Route. Für ein Schiff auf dem Weg zwischen zwei Häfen wird die Verteilung des *Speed Over Ground (SOG)* bestimmt. Mittels eines *Geschwindigkeitsprofils* wird anschließend für die befahrene Route die restliche Fahrtzeit ermittelt. Die mittlere absolute Abweichung, 48 Stunden vor Ankunft im

Zielhafen, beträgt 6 Stunden (Alessandrini u. a., 2019, S. 13).

Araujo u. Sancricca (2021) reduzieren das Problem, wie die vorigen Autoren, auf die Fahrt eines Schiffs von einem Hafen zu einem anderen Hafen. Vesseltracker (vesseltracker.com, 2021) dient als Datengrundlage und weißt ähnliche Merkmale wie der Datensatz des AIS auf. Das von Araujo u. Sancricca (2021) eingesetzte Random-Forest-Regressionsmodell erzielt einen R2-Score von 0.9473.

2.2 Unsicherheiten in Ankunftszeiten

Ein verwandter Ansatz zur Präzisierung geschätzter Ankunftszeiten versucht, die Abweichung einer *ursprünglichen* ETA zu schätzen. Pani u. a. (2013) setzen drei unterschiedliche maschinelle Lernmodelle um: Naive Bayes, Decision Trees und *Random Forests* (Breiman, 2001). Schiffe mit einer Abweichung von ±15 Minuten werden als pünktlich gehandhabt. Das Verfahren klassifiziert Schiffe auf Basis von fünf Variablen. Dazu wurden über einen Zeitraum von 30 Monaten folgende Daten erhoben: *Total number of mother ships delayed*, *Proportion of delayed vessels*, *Total length of delayed vessles*, *Total number of delayed containers* und *Total delay hours* (Pani u. a., 2013, S. 6, Tabelle 2). Die Modelle klassifizieren die Daten in drei Gruppen: Niedrige, mittlere und hohe Abweichung. Die besten Ergebnisse erzielte das Modell unter Einsatz des Random Forest Algorithmus mit einer fälschlichen Klassifikationsrate von 29% (Pani u. a., 2013, S. 6, Tabelle 2; S. 12).

Pani (2014) setzt den Fokus des Problems unzuverlässiger Ankunftszeiten auf die letzten 24 Stunden vor Ankunft eines Schiffs im Hafen. Änderungen der Ankunftszeit in diesem Zeitraum stellen das Hafenmanagement und anschließende Logistik vor besonders große Herausforderungen (Pani, 2014, S. 3f.). Pani untersucht spezifische Merkmale der Häfen von Cagliari und Antwerpen, andere Häfen werden nicht betrachtet. Analog zur vorigen Arbeit Pani u. a. (2013), wird eine Ankunft mit einer Abweichung von ±15 Minuten als pünktlich klassifiziert. Die trainierten Modelle setzen Algorithmen mit logistischer Regression, Klassifikation, Regressionsbaum und Random Forest um. Die beste Performance erzielen Random Forest Algorithmen.

Unter Einsatz der Technologien Logistic Regression, CART (Breiman u. a., 1984), und Random Forsest, ähnlich zu Pani (2014), untersucht Pani u. a. (2015) den Einfluss spezifischer Variablen zweier Häfen (Cagliari und Antwerpen) auf unsichere Schätzungen von Ankunftszeiten. Die Variablen beschreiben unter anderem die physikalische Struktur eines Schiffs, dessen Besitzer, Schiffsposition und Wetterdaten. Zur detaillierten Betrachtung der eingeführten Variablen unterteilt Pani die Häfen in ihre einzelnen Terminals. Für jeden Terminal eines Hafens wird eine Verspätungs-Verteilung auf Basis historischer Daten ermittelt. Erneut liefern die Schätzungen des Random Forest Algorithmus die besten Ergebnisse.

2.3 Kombination aus Hafen- und ETA-Prädiktion

Die Arbeit von Bodunov u. a. (2018) entstand im Rahmen der DEBS Grand Challenge 2018 und setzt ein zweistufiges Verfahren zur (1) Klassifikation eines Zielhafens und (2) Schätzung der Ankunftszeit eines Schiffs um. Die Schätzung eines Zielhafens wird im Folgenden nicht behandelt, weil dieses Problem für die Problemstellung dieser Arbeit nicht relevant ist. Bodunov u. a. (2018) fassen die Schätzung der Ankunftszeit als Regressionsproblem auf. Ähnlich der Datengrundlage dieser Arbeit, sind im Datensatz der DEBS Challenge keine Ankunftszeiten enthalten. Sie werden aus dem Endzeitpunkt einer Anfahrt berechnet. Bodunov u. a. (2018) stehen zum Training des Modells drei Features und die berechnete Ankunftszeit zur Verfügung. Ein neuronales feed-forward Netz wird verwendet, um Features in einem *hidden Layer* zu extrahieren. Darauf folgt ein *Dense Net* zur Schätzung der Ankunftszeit (Bodunov u. a., 2018, Kap. 3.2.1). Das Modell erzielt eine Genauigkeit von 90% für geschätzte Ankunftszeiten im Verhältnis zu tatsächlichen Ankunftszeit.

2.4 Deep Learning zur Schätzung von Ankunftszeiten

Der Thematik zur Schätzung von Ankunfts- oder Abfahrtszeiten mit maschinellen Lernmodellen widmen sich Arbeiten in unterschiedlichen Anwendungsgebieten. Shao u. a. (2021) behandeln die Problemstellung zur ETA-Schätzung in der Luftfahrt unter Einsatz eines Deep Learning Modells (TranjCNN). Der Ansatz erzielt eine mittlere Abweichung von 18 Minuten bei der Schätzung der Abflugzeit am Los Angeles International Airport.

3 Grundlagen

Im folgenden Kapitel werden grundlegende Techniken erläutert und Begrifflichkeiten eingeführt. Kapitel 3.1 führt in die Thematik des Maschinellen Lernens mit Fokus auf die Verarbeitung von Zeitreihen ein. Die Funktionsweise tiefer Faltungsnetze mit konkreter Anwendung maschinellen Lernens wird in Kapitel 3.2 behandelt. Das letzte Kapitel 3.3 zeigt aktuelle Methoden und Anwendungsfälle im Bereich des Transfer-Lernens auf.

3.1 Maschinelles Lernen

„Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience.“ — Tom M. Mitchell (1997)

Der Begriff *machine learning* wurde vom Amerikaner Arthus Samuel (IBM) im Jahr 1959 geprägt. Er beschreibt, dass eine Maschine im *game of checkers* auf Basis der Regeln des Spiels, vorgegebenen Parametern sowie unbekannten und unspezifizierten Gewichten das Spiel lernen kann. Dieses Prinzip des maschinellen Lernens kann nach Samuel (1959) auf andere Kontexte übertragen werden. Maschinelles Lernen zeichnet sich im Sinne von Samuel durch das Zusammenspiel von Daten und Algorithmen zur Nachbildung menschlicher Lernprozesse aus. Dieser Aspekt trifft verstärkt auf den Teilbereich der neuronalen Netze zu. Ein neuronales Netz besteht, analog zum menschlichen Gehirn, aus miteinander vernetzten Neuronen (Rosenblatt, 1958).

3.1.1 Klassifikation versus Regression

Im Bereich des maschinellen Lernens werden zwei unterschiedliche Problemklassen unterschieden. Verfahren, die Daten vordefinierten Klassen zuweisen behandeln *Klassifikationsprobleme*. Die Erkennung von Tiermotiven auf Bildern ist ein klassischen Beispiel. Andere Verfahren, die beispielsweise eine *kontinuierlich* des Wetter schätzen, behandeln *Regressionsprobleme*. (Brownlee, 2019) In dieser Arbeit werden keine Klassen zur Ankunftszeitschätzung definiert. Das Verfahren schätzt auf Basis historischer Zeitreihen eine Dauer bis zur Ankunft eines Schiffs am Zielhafen. Folglich handelt es sich bei der Problemstellung dieser Arbeit um ein Regressionsproblem.

3.1.2 Einordnung maschineller Lernansätze

Maschinelles Lernen unterteilt sich in drei grundlegenden Kategorien: Überwachtes, unüberwachtes und semi-überwachtes Lernen.

Überwachtes (supervised) Lernen ist definiert als Lernen auf gelabelten Daten. Das Label eines Datums im Datensatz ist der tatsächliche Zielwert, der vom Algorithmus angenähert werden soll. Ein Datensatz mit Bildern von Tieren ist gelabelt, wenn für jedes Bild das abgebildete Tier a-priori bekannt ist. Der Anwendungsbereich beschränkt sich nicht auf Bild-Datensätze, sondern kann auf jede Art von Daten erweitert werden. In dieser Arbeit werden Ankunftszeiten basierend auf Zeitreihen-Daten prognostiziert. Das Label eines Datums repräsentiert dessen Dauer bis zur Ankunft am Zielhafen. Die Problemstellung dieser Arbeit ist schlussfolgernd überwachtem Lernen zugeordnet.

Die zweite Kategorie des unüberwachten (unsupervised) Lernens agiert auf ungelabelten Datensätzen, um Zusammenhänge in den Daten zu erkennen. Diese Kategorie eignet sich zur explorativen Herangehensweise an einen Datensatz, um beispielsweise Muster in Kundendaten zu erkennen. Ein bekannter Clustering-Algorithmus zur Gruppierung von Daten ist *k-means* (vgl. Jin u. Han (2010)).

Der Ansatz des semi-überwachten (semi-supervised) Lernens ist eine Mischung aus überwachtem und unüberwachtem Lernen. Er kommt zum Einsatz, wenn nicht ausreichend Daten zum überwachten Lernen vorhanden sind. Auf einem kleinen gelabelten Datensatz werden Features extrahiert und im Anschluss auf einen größeren, ungelabelten Datensatz angewendet.

3.1.3 Anwendung eines maschinellen Lernalgorithmus

Die grundlegende Funktion eines maschinellen Lernalgorithmus ist die Schätzung der gewünschten Ausgabe *(Output)* auf Basis von Eingabedaten *(Input)*. Die Genauigkeit der Schätzung wird mittels einer Fehlerfunktion bewertet. Im Fall (semi-) überwachten Lernens ist die Genauigkeit der Ausgabe am tatsächlichen Label messbar. Zuletzt erfolgt, analog zum menschlichen Lernprozess, die Verhaltensanpassung des Algorithmus, um die Prognose für den Datensatz zu verbessern. Iterativ passt der Optimierungsprozess interne Gewichte des Algorithmus an, um die Abweichung der Prognose zum Label zu reduzieren, bis die Genauigkeit der Schätzung gegen einen Grenzwert konvergiert.

3.1.4 Trainings- und Testdaten

Zur Bestimmung der Genauigkeit eines Modells wird bei supervised Learning der Output mit dem tatsächlichen Label des Datums verglichen. Die zum Testen verwendeten Daten stammen nicht aus den Daten zum Training des Modells, sondern werden als Testdaten zurückgehalten. Vor dem Training eines maschinellen Lernalgorithmus hat sich die Aufteilung aller verfügbaren Daten in Test- und Trainingsda-

tensatz als übliches Vorgehen etabliert (Stone, 1974). So kann gezeigt werden, dass ein trainiertes Modell generalisiert und nicht ausschließlich an die Trainingsdaten angepasst ist. Häufig werden 80% als Trainingsdaten verwendet und die restlichen 20% dienen der Validierung.

3.2 Deep Learning und Convolutional Neural Networks

Mitchells Zitat am Anfang von Kapitel 3.1 beschreibt die grundlegende Arbeitsweise eines maschinellen Lernalgorithmus zutreffend mit *„[Algorithms, that] improve through experience“*. Lernen durch Erfahrung setzt ein maschineller Lernalgorithmus durch Erschließung von Wissen auf Basis von Daten um (Neupert u. a., 2021, S. 3). Das menschliche Gehirn verhält sich analog: Wissen wird auf Basis täglicher Erfahrungen generiert und Neuronen verknüpft. Diese Funktionsweise wenden Convolutional Neural Networks (CNNs) an (O'Shea u. Nash, 2015). Sie sind im maschinellen Lernen dem Bereich des Deep Learning (oder hierarchical Learning) zugeordnet (Deng u. Yu, 2014, S. 198). *Deep* bezieht sich auf beliebig vielschichtige Architekturen aus unterschiedlichen Layern innerhalb eines Netzwerks. Grundsätzlich extrahiert das Netz mit zunehmender *Tiefe* zunehmend anwendungsspezifische Features (Deng u. Yu, 2014, S. 199-201). Veranschaulicht anhand der Analyse von Bildern, werden in frühen Schichten abstrakte geometrische Formen wie Kanten und Ellipsen erkannt. Spätere Layer kombinieren abstrakte Formen zu (Teil-) Bildern, bis schließlich die gesamte Eingabe als Ziffer, Tier oder Gesicht klassifiziert wird.

Im Fall von Convolutional Neural Networks (CNNs) werden Gewichte interner Verbindungen auf Basis gelabelter Trainingsdaten so lange angepasst, bis sich die Schätzung der Ausgabe nicht mehr verbessert. Sie ahmen die Struktur des menschlichen Gehirns nach: Miteinander verbundene Knoten (Neuronen) bilden ein Netzwerk. Ein CNN besteht aus sog. Layern (Schichten). Die erste Schicht wird Input-Layer genannt und nimmt die Trainingsdaten entgegen. Die letzte Schicht des Netzes nennt sich Output-Layer. Sie gibt den gewünschten Zielwert aus, der beliebig viele Dimensionen aufweisen kann. Klassifikatoren bilden beispielsweise auf den Wertebereich ihrer möglichen Klassen ab. Da das Netz dieser Arbeit Schätzungen für Ankunftszeiten abgeben soll, ist die Ausgabe 1-dimensional. Das heißt, der Output-Layer nimmt die Anzahl Verknüpfungen der vorigen Schicht entgegen und bildet auf einen Zielwert ab: Die Dauer bis zur Ankunft in Minuten. Alle Schichten zwischen Input- und Output Layer werden als *Hidden Layer* bezeichnet.

In einem neuronalen *feed-forward Netz* (Abb. 2) sind alle Pfade vom Input- bis zum Output-Layer frei von Zyklen, gegensätzlich zum sog. *Recurrent Neural Network*. Neuronen sind in feed-forward Netzen stets *vorwärts* von Input in Richtung Output verbunden. Ausgehend vom Input-Layer, verknüpft ein solches Netz eine beliebige

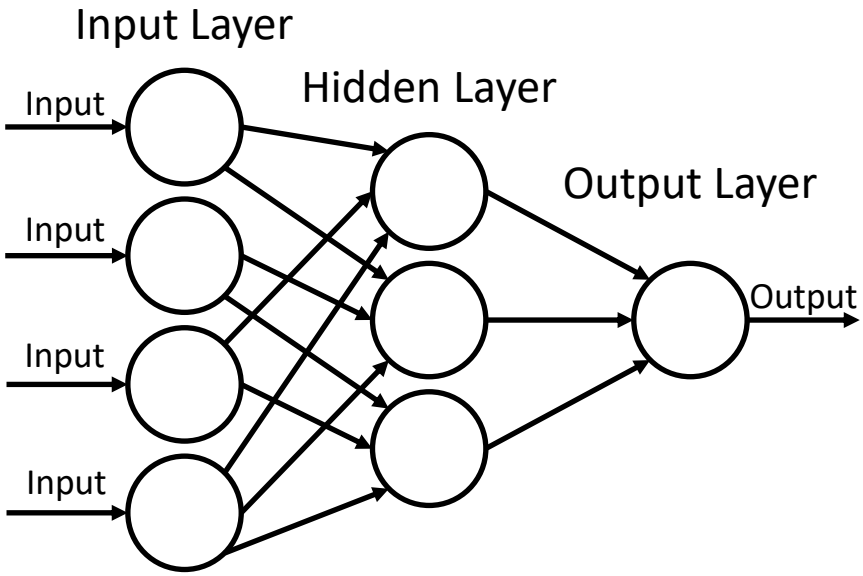

Abbildung 2: Architektur eines neuronalen feed-forward Netzes. Die Neuronen des Input-Layer nehmen die Eingaben entgegen. Hidden-Layer repräsentieren das Innere des neuronalen Netzes. Die Abbildung zeigt einen Hidden-Layer. Neuronale Netze können allerdings beliebig viele Hidden-Layern beinhalten. Schließlich nimmt der Output-Layer die Ausgaben der Hidden-Layer entgegen und produziert die gewünschte Ausgabe.

Anzahl Layer unterschiedlicher Zusammensetzung und mündet in der Ausgabe des gewünschten Zielwerts (*Output*) (vgl. Abb. 2). Es verarbeitet eine Eingabe beliebiger Dimensionalität, wobei in verschiedenen Bereichen der Eingabe eines Layer unterschiedliche Merkmale codiert sind. Der Beitrag von Dertat (2017) bietet eine umfangreiche Übersicht zur Funktionsweise eines CNNs. Eine *Feature Map* (Abbildung 3) zeigt die *aktivierten* Neuronen einer Schicht eines neuronalen Netzes. Abbildung 3 zeigt die erkannten Merkmale eines Katzenbilds. Bilderkennung (Klassifikation) ist eine klassische und häufige Anwendung von CNNs. Als Eingabe dienen einzelne Bilder, die Größe des Input-Layer entspricht der Größe der Eingabebilder. Jeder Pixel eines Bilds wird mit einem Neuron des Input Layers verbunden. Anschließend extrahieren Hidden Layer Merkmale der Eingabe und schließlich klassifiziert der Output Layer die gewünschte Klasse des Bilds. Der Anwendungsbereich von CNNs ist nicht auf die Klassifikation von Bildern beschränkt. Beliebige Arten von Daten, die auf einen 3-dimensionalen Input abbildbar sind, können als Eingabe eines CNNs verwendet werden. Zur vereinfachten Erklärung sei die Eingabedimension $1 \times 37 \times T$, wobei T die Anzahl betrachteter Zeitpunkte repräsentiert. Im Rahmen dieser Arbeit werden sog. Batches verarbeitet, die der Eingabe eine dritte Dimension verleihen. Die Funktionsweise der Schichten eines CNNs für Eingaben in Batch-Form ist äquivalent zu eindimensionalen Eingaben. Im Folgenden wird die Funktionsweise

besonders relevanter Schichten für das neuronale Netz dieser Arbeit erläutert.

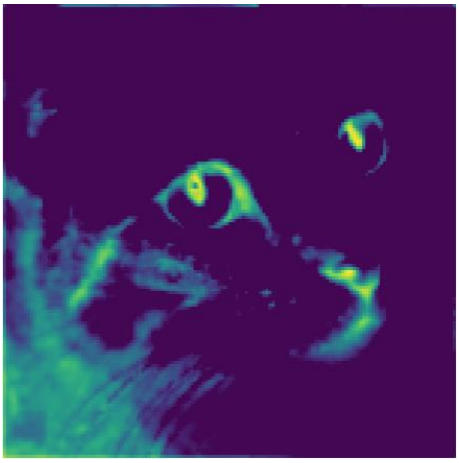

Abbildung 3: Feature Map der Ausgabe des ersten Layers im Netz von Dertat (2017), dessen Problemstellung die Klassifikation von Bildern behandelt. Die hellen gelben und grünen Bereiche repräsentieren besonders hohe Gewichte von Neuronen, die zur Erkennung einer Katze wichtig sind. Augen, der Bereich um die Nase der Katze, einzelne Schnurrhaare und *vermutlich* der Hals sind in der Feature Map erkennbar.

3.2.1 Convolutional Layer

Ein relevantes Konzept von neuronalen Netzen sind Faltungen (Convolutions). Eine Faltung ist ein Layer innerhalb des Netzes, der eine Funktion auf die aktuelle Eingabe des Layers anwendet, um lokale Ausprägungen einer Eingabe gemäß einer Funktion zu transformieren. Die angewandte Funktion wird *Kernel* genannt und codiert die auszuführende Operation. Abbildung 4 zeigt die Anwendung einer Faltung. Der dargestellte Prozess betrachtet *stückweise* die Eingabe mittels des Kernels und projiziert eine Ausgabe.

Zur einfachen Erklärung wird die dritte Dimension auf 1 gesetzt. Folgtlich besteht die Eingabe aus einer $1 \times 5 \times 7$ Matrix, worauf ein $1 \times 3 \times 3$ Kernel folgt. Der Kernel wird schrittweise auf den gesamten Input angewendet. Im Fall der Abbildung beträgt die Schrittlänge $= 1$, was den Kernel entlang der x-Dimension über jede Spalte und entlang der y-Dimension über jede Zeile des Inputs schiebt. Die Zellen der Kernel-Matrix aggregieren die jeweils betrachteten Einträge des Inputs (Neuronen) gemäß einer Funktion $\bigoplus$ und bilden das Ergebnis ab. Im Fall von Abbildung 4 bleibt die Dimension nach Anwendung des Kernels erhalten. Kernels können, je nach Konfiguration, die Dimension einer Eingabe reduzieren, erhöhen oder erhalten.

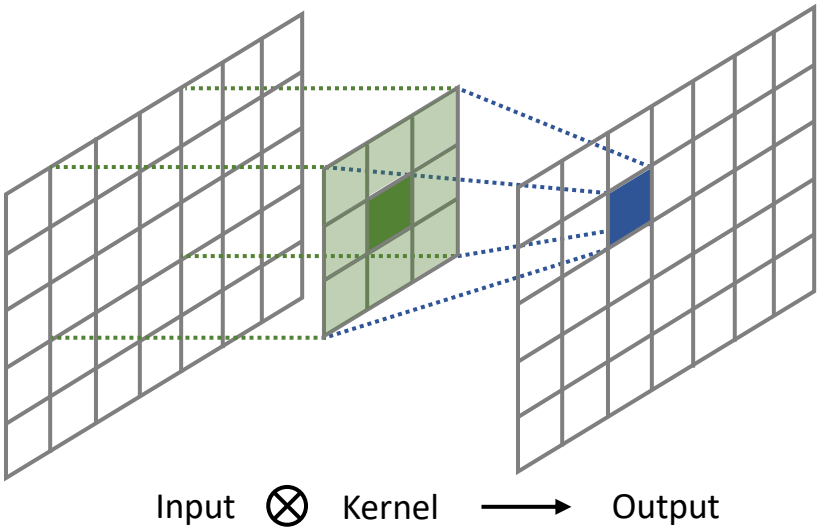

Abbildung 4: Anwendung einer Faltung mit 3×3-Kernel auf einen Input der Größe 7×5. Dargestellt ist die Faltung der dritten Spalte und zweiten Zeile. Die unterschiedlichen Grüntöne des Kernels verdeutlichen die Gewichtung betrachteter Neuronen des Inputs. Die äußeren hellgrünen Bereiche des Kernels beziehen betreffende Neuronen des Inputs schwächer in die Berechnung ein als die Neuronen im dunkelgrünen Bereich. Die hellgrünen Bereiche des Kernels befinden sich bei Betrachtung der äußeren Neuronen des Inputs *außerhalb* der Input-Dimension und werden nicht in die Berechnung mit einbezogen. Das Resultat der Faltung ist eine 1×1-große Ausgabe (blau). Die Ausgaben über alle Spalten und Zeilen des Inputs bilden den Output.

3.2.2 Fully Connected Layer

Der Name eines Fully-Connected-Layer (FC-Layer) in einem neuronalen Netz erklärt die Funktion: Jedes Neuron des vorigen Layers ist mit allen Neuronen des FC-Layers verbunden. Üblicherweise werden FC Layer im Anschluss an Feature-Extraktionen eingesetzt, wie die folgende Auswahl zeigt. AlexNext (Krizhevsky u. a., 2017) beinhaltet fünf Convolutional Layer, gefolgt von drei FC-Layern. Analog schließen drei FC Layer die Pipeline zur Feature Extraktion des VGG-16-Modells (Simonyan u. Zisserman, 2015) ab. Dieses Modul aufeinander folgender FC-Layer wird als *Dense Net* bezeichnet. Nach Abschluss der abstrakten Feature-Extraktion in Convolutional Layern (vgl. Kap. 3.2.1), extrahiert ein Dense Net anwendungsspezifische Merkmale und gibt die gewünschte Schätzung aus (Output). Im Fall eines Klassifikationsproblems wird die Ausgabe der Hidden Layer auf die möglichen Klassen abgebildet. Merkmale, die sich im Bereich der Hidden Layer (vgl. Abb. 2) ausprägen, werden als *Hidden Features* bezeichnet. Behandelt das Netz ein Regressionsproblem, kann der

Output des Dense Nets die identifizierten Features beispielsweise auf eine Zeitreihe zur Wettervorhersage abbilden (Liu u. a., 2014).

3.2.3 Residual Connectors

Der in den vorigen Kapiteln beschriebene Aufbau eines neuronalen Netzes verbindet stets die Neuronen zweier Layer, die direkt aufeinander folgen. Basierend auf dem Output des vorigen Layers wird die Operation des nächsten Layers ausgeführt. Ein Merkmal der Daten, das von einem Layer nicht beachtet wird, ist in dessen Output nicht repräsentiert. Folglich kann Wissen, das sich erst in späteren Schichten aus der Kombination von aktuellen Features und *älteren* (vergessenen) Merkmalen ergibt, nicht gelernt werden. Um den Fluss von Wissen über mehrere Layer hinweg zu ermöglichen, werden *Residual Connectors* eingesetzt. Residual Connectors sind Direktverbindungen zweier Layer, die im Netz nicht direkt aufeinander folgen. Sie ermöglichen die Betrachtung von Outputs früherer Schichten im Netz (oder der ursprünglichen Eingabe), um Zusammenhänge von Merkmalen zu erkennen, die sich nicht in aufeinanderfolgenden Layern ausprägen. Netze mit residualen Verbindungen werden häufig als *ResNet* oder *Highway Net* (Srivastava u. a., 2015) bezeichnet.

3.2.4 Algorithmus zur Optimierung eines neuronalen Netzes: Gradientenabstieg

Gradientenabstieg *(Gradient Descent, GD)* ist eine populäre Methode zur Optimierung eines maschinellen Lernalgorithmus (Ruder, 2017). In Abhängigkeit zur Ableitung erster Ordnung der Loss-Funktion setzt Gradientenabstieg die Anpassung der Gewichte eines neuronalen Netzes um (Doshi, 2019). Ziel der GD-Funktion ist die Minimierung des Funktionswerts resp. des Loss'. Je nach Anwendungsbereich kommen verschiedene Arten von Gradientenabstieg zum Einsatz. *Stochastic Gradient Descent (SGD)* passt die Gewichte eines Modells bei der Verarbeitung jedes Trainingsbeispiels im Datensatz an, weshalb die Parameter eines Modells in kurzer Zeit konvergieren. Allerdings weisen die Parameter eine hohe Varianz auf, können trotz Erreichen des globalen Minimums weiter driften oder in lokalen Minima terminieren (Bottou, 1998). *Mini-Batch Gradient Descend* behandelt das Problem fluktuierender Parameter bei SGD, indem Gewichte nach Bearbeitung eines *Batches* angepasst werden. Ein Batch ist ein Subset aus n Trainingsbeispielen des Trainingsdatensatzes. Keskar u. a. (2017) explorieren, wie sich unterschiedliche Größen von Batches auf die Performance und Generalisierung eines Modells auswirken. So führen große Batches, wie beispielsweise ein Batch über den vollständigen Datensatz, zu Qualitätseinbußen des Modells hinsichtlich Generalisierung. Unter Beachtung der Nachteile klassischer Optimierungsalgorithmen wie SGD oder Mini-Batch GD, findet in modernen neuro-

nalen Netzen häufig *Adaptive Moment Estimation (Adam)* Anwendung (Kingma u. Ba, 2017). Adam vermeidet große Anpassungen der Gewichte, um das gesuchte Minimum nicht zu überspringen. Weiter beziehen zwei Kennziffern *(moving averages)* das *Momentum* der Gradienten in die Anpassung der Gewichte ein: $M(t)$ repräsentiert ein exponentiell sinkendes Mittel der vorigen Gradienten, $V(t)$ bildet die nicht-zentrierte Varianz der Gradienten ab. Die Berechnung einer nicht-zentrierten Varianz verläuft analog zur üblichen Varianz, ohne Subtraktion des Mittelwerts.

3.2.5 Effizienter Gradientenabstieg: Backpropagation

Ein neuronales Netzes verknüpft, wie in Kapitel 3.1.3 beschrieben, Neuronen miteinander. Im Fall eines feed-forward neuronalen Netzes (vgl. Abb. 2) sind alle Pfade vom Input des Netzes bis zum Output frei von Zyklen. Der Pfad vom letzten Layer des Netzes bis zum Input-Layer ist folglich eindeutig. Zur effizienten Bestimmung der Gradienten kann der *Backpropagation* Algorithmus angewendet werden. Backpropagation bezeichnet unabhängig vom anschließenden Verwendungszweck ausschließlich die Berechnung von Gradienten. Der Algorithmus erlaubt einen *Rückwärts-Fluss* bei der Berechnung der Gradienten durch ein Netzwerk in Abhängigkeit der finalen Gewichte, um redundante Berechnungen zu vermeiden. (Goodfellow u. a., 2016, S. 200)

3.2.6 Erzeugung eines optimalen maschinellen Lernmodells

Die grundlegende Vorgehensweise des Algorithmus zum Generieren eines neuronalen Netzes optimiert ein Modell über mehrere *Epochen*. Der Algorithmus versucht, das Modell in jeder Epoche über alle Trainingsdaten hinweg zu verbessern. Um festzustellen, ob die aktuelle Epoche ein neues optimales Modell erzeugt hat, werden im Anschluss die Testdaten verarbeitet. In dieser Phase findet keine Anpassung der Parameter statt, das Modell gibt ausschließlich Prädiktionen für die Validierungsdaten ab. Auf Basis dieses Outputs berechnet eine Loss-Funktion die Abweichung der Schätzungen von ihren korrespondierenden Labeln. Typische Loss-Funktionen sind der durchschnittliche absolute- *(MAE)* oder quadratische *(MSE)* Fehler (vgl. Kap. 5.5). Je nach Implementierung ist das Modell mit minimalen Loss der eingesetzten Funktion das optimale Modell des Lernalgorithmus.

3.2.7 Angewandtes Konzept zur Feature Extraktion: Inception Time

Die Architektur jedes neuronalen Netzes richtet sich nach der Aufgabe, die es zu erfüllen hat. Im Fall dieser Arbeit dienen Zeitreihen als Datengrundlage zur Schätzung von Ankunftszeiten in der Schifffahrt. Die Aufgabe des neuronalen Netzes besteht darin, aus einem Subset einer Zeitreihe deren Ankunftszeit zu schätzen. Dabei bil-

den sich Merkmale in den Daten in zeitlich unterschiedlich langen Perioden ab. Die aktuelle Position eines Schiffs ist beispielsweise innerhalb eines Datenpunkts genau bestimmt. Zusammenhänge wie das Fahrverhaltens eines Schiffs bilden sich hingegen über längere Abschnitte im Verlauf der Zeitreihe über mehrere Datenpunkte ab. Eine Architektur zur Klassifikation von Zeitreihen-Daten wird im Artikel von Ismail Fawaz u. a. (2020) vorgestellt. Der signifikante Unterschied bei der Analyse von Zeitreihen gegenüber einer Klassifikation von Bildern liegt in der temporal codierten Information (Bagnall u. a., 2016). Die Architektur eines solchen Netzes erlaubt Eingaben der Form $1 \times N \times T$ mit N = Anzahl Features und T = Länge der Zeitreihe. Die benötigte Länge eines Subsets der Zeitreihe, die zur Extraktion eines zeitlichen Zusammenhangs benötigt wird, ist a-priori unbekannt und kann variieren. Das sogenannte Receptive Field (RF) beschreibt das Verhalten eines neuronalen Netzes auf unterschiedlichen Regionen einer Eingabe. Das RF bezeichnet den Teil einer Eingabe, den das neuronale Netz im aktuellen Verarbeitungsschritt betrachtet (vgl. Abb. 4 Im Anwendungsfall von Zeitreihen erzielt nach Luo u. a. (2017) ein Netz mit größerem RF bessere Ergebnisse in der Detektion längerer zeitlicher Zusammenhänge. Ismail Fawaz u. a. (2020) setzen die Erkennung von Merkmalen, die sich über unterschiedlich lange Perioden ausprägen, durch Faltungen mit Kernels verschiedener Größen um. Simultane Aggregation der ursprünglichen Eingabe und Zusammenführung der einzelnen Outputs bilden mit den Kernels unterschiedlicher Größe einen *Inception-Block* (vgl. Abb. 5). Ein vollständiges Modell kann mehrere Inception-Blöcke verketten und beliebige weitere Layer an deren Ausgabe koppeln. Abbildung 5 zeigt die Architektur eines Inception-Blocks nach Ismail Fawaz u. a. (2020).

3.3 Transfer Learning

Die bisher eingeführte Vorgehensweise des maschinellen Lernens der vorigen Kapitel sieht vor, ein Modell auf *einem* Datensatz zu trainieren. In der praktischen Anwendung eines trainierten Modells sind Eingabedaten ähnlich ausgeprägt, wie die zum Training verwendeten Daten. Sie stammen aus der selben Domäne. Eingangs im Zitat bezeichnet Mitchell (vgl. Kapitel 3.1) maschinelles Lernen als *„[...] verbessern durch Erfahrung.“*. Erfahrung kann nur gesammelt werden, wenn der Akteur (resp. das maschinelle Lernmodell) in einer bekannten Domäne agiert (lernt). Gelerntes Wissen aus einer fremden Domäne kann meist nicht ohne Anpassung angewandt werden. Ein Modell ist folglich immer an die Daten der Domäne angepasst, womit es trainiert wurde. Die verfügbare Datenmenge variiert in unterschiedlichen Domänen. Gründe können einerseits hohe Kosten bei der Erhebung oder natürliche Knappheit bei der Sammlung Domänen-spezifischer Daten sein. Im medizinischen Bereich können mittels MRT-Scans Krankheiten mit hoher Genauigkeit festgestellt werden.

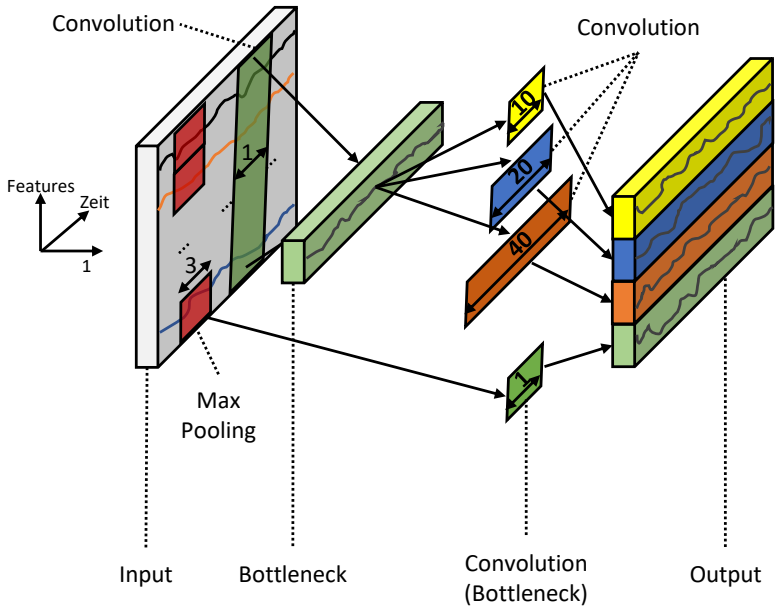

Abbildung 5: Architektur eines Inception Time Blocks nach Ismail Fawaz u. a. (2020). Eine n-dimensionale Eingabe (Zeitreihe) wird mittels Convolution auf eine m-dimensionale Zeitreihe reduziert. Zur Übersicht besitzt der Bottleneck eine y-Dimension von m=1. Drei Convolutions mit Kernels der Größen 10, 20 und 40 werden auf den Bottleneck angewendet und deren Ergebnis konkateniert. Das Ergebnis einer parallelen Max-Pooling Operation, gefolgt von Convolution zu Reduktion der Dimensionalität (Bottleneck) wird ebenfalls mit den Ausgaben der drei *anderen* Convolutions konkateniert. Die Ausgabe eines Inception-Blocks ist eine mehrdimensionale Zeitreihe.

Doch im Fall seltener Krankheiten ist die Datenmenge zum Training eines *eigenen* Klassifikators unzureichend. Um dennoch präzise maschinelle Lernmodelle trainieren zu können, werden generelle Features eines Modells zunächst auf nicht medizinischen Beispielbildern vortrainiert und im Anschluss in einem mehrstufigen Prozess an die gewünschte Domäne angepasst. Nach aktuellen Forschungsergebnissen erzielt laut Aswiga u. a. (2021) ein Transfer mit weniger als 10% der notwendigen Daten zum Training eines eigenen Modells eine Genauigkeit von bis zu 90%. Definition 2 basiert auf der Arbeit von Zhuang u. a. (2019).

In der Praxis werden die vortrainierten Modelle der Visual Graphics Group (VGG) aus Oxford VGG-16 (Simonyan u. Zisserman, 2015), InceptionV3 von Google (Szegedy u. a., 2015) und ResNet50 von Microsoft (He u. a., 2015) verwendet, um auf einem anderen Kontext verwandte Aufgaben zu lösen. Die Modelle werden mit vortrainierten Gewichten geladen und anwendungsspezifische Erweiterungen an deren Ausgaben gekoppelt. Innerhalb weniger Epochen erzielt ein transferiertes Modell mit geringen Datenmengen ein hohes Maß an Genauigkeit. Die genannten vortrainieren

Modelle agieren in der Domäne der Bilderkennung. Als Eingabe dienen Datensätze aus Bildern, deren Motive erkannt werden sollen. Die Aufgabe könnte beispielsweise die Erkennung verschiedener Tierarten wie Hunde oder Katzen sein. Wenn ein Eingabebild das Motiv eines Hundes enthält, soll der Klassifikator das Bild mit möglichst hoher Genauigkeit als *Hund* klassifizieren, analog für beliebige weitere Tierarten.

Definition 2: Transfer Learning: *Transfer Learning überträgt in einer Domäne gewonnenes Wissen eines maschinellen Lernmodells auf eine andere, aber ähnliche Domäne.*

4 Forschungsansatz

Diese Arbeit wendet die Methodik des Design Science Research (DSR) an. DSR ist ein Forschungsparadigma im Bereich des Information System Research (IS Research). Es dient dem Zweck der systematischen Begründung von Forschung, Einordnung in bestehende Wissensbasen und strukturiert den Forschungsprozess (Gregor u. Hevner, 2013). Kapitel 4.1 stellt die Weichen zur Anwendung von DSR: Abhängig von der zutreffenden Art des Forschungsansatzes dieser Arbeit (*Design Research* oder *Action Research*), kategorisiert Kapitel 4.3 die Forschung anhand des *DSR Knowledge Contribution Frameworks*.

4.1 Design Research versus Action Research

Sowohl Gregor u. Hevner (2013) als auch Hevner u. Chatterjee (2010) (S. 180-183) unterscheiden grundlegend zwischen *Design Research)* und *Action Research*. Design Research zeichnet sich durch die Generierung von Artefakten und zugehöriger Theorien aus, die unternehmerische Problemstellungen aus der Praxis lösen sollen. Forschung als Design Research untersucht folglich künstliche Phänomene und ist dem Forschungsgebiet *(The) Science of (the) Artificial* nach Simon (1996) zugeordnet. Das Ziel von Design Science und Design Research ist die Generierung neuer und innovativer Artefakte. Das Artefakt identifiziert sich nicht ausschließlich durch seine Nützlichkeit in der Praxis, sondern liefert gleichwohl theoretisches Wissen zu seiner Erzeugung. (Hevner u. Chatterjee, 2010, S. 180 f.)

Action Research setzt den Fokus der Forschung auf die Beobachtung von Änderungen komplexer sozialer Prozesse durch induzieren von Änderungen genau dieser sozialen Systeme (Hevner u. Chatterjee (2010), S. 182 i.V.m. Baskerville (2001)).

Die Problemstellung dieser Arbeit entspringt realen unzuverlässigen Schätzungen der Ankunftszeit von Schiffen am Zielhafen und versucht, mittels eines digital (künstlichen) Prozesses die Situation zu verbessern. Auf Basis existierender Techniken und Konzepte des maschinellen Lernens soll sich das Artefakt iterativ verfeinern und zur Lösung der realen Problemstellung praktisch einsetzbar sein. Folglich ist der Forschungsprozess dieser Arbeit Design Research zugeordnet.

4.2 Arten von Wissensbasen

Nachdem im vorigen Kapitel 4.1 der Forschungsprozess eingeordnet wurde, stellt dieses Kapitel unterschiedliche Typen von Wissensbasen vor und ordnet den Beitrag der Forschung dieser Arbeit entsprechend ein. Forschung nutzt bestehende Theorien, Methoden und Prozesse, um weiterführende Ideen zu explorieren. Den existierenden

Wissensstand bezeichnen Gregor u. Hevner (2013) und Hevner u. Chatterjee (2010) als Wissensbasis, die aus *„Useful Knowledge“* besteht. Gregor u. Hevner (2013) unterscheiden zwei Arten nützlichen Wissens: *Deskriptives Wissen* (Ω) und *Präskriptives Wissen* (Λ). Tabelle 1 zeigt die Zuordnung von Artefakten nach Gregor u. Hevner (2013) (S. 8) gemäß dem Typ generierten Wissens.

Ω - Deskriptives Wissen

- **Phänomene** (natürlich, künstlich, menschlich)
  - Beobachtungen
  - Klassifikation
  - Messung
  - Katalogisierung
- **Begründen** *Sense-making*
  - Naturgesetze
  - Gesetzmäßigkeiten
  - Prinzipien
  - Muster
  - Theorien

Λ - Präskriptives Wissen

- **Konstrukte**
  - Konzepte
  - Symbole
- **Modelle**
  - Repräsentation
  - Semantik/Syntax
- **Methoden**
  - Algorithmen
  - Techniken
- **Instanziierung**
  - Systeme
  - Produkte/Prozesse
- **Design Theorie**

Tabelle 1: Die zwei Typen von Wissen der Wissensbasis im Design Science Research (DSR) nach Gregor u. Hevner (2013) (S. 344). Die Kategorien und konkreten Ausprägungen unterstützen die Identifikation von generiertem Wissen im Forschungsprozess des DSR (vgl. Abb. 6) und ordnen die entsprechende Wissensart (Ω oder Λ) zu.

Das Artefakt dieser Arbeit ist Resultat des Design Zyklus, der iterativ die Performance eines maschinellen Lernmodells evaluiert und optimiert. Das Artefakt selbst ist die im aktuellen Generierungsprozess beste Instanz, die im Verlauf des Algorithmus erzeugt und letztlich persitiert wird.

4.3 Arten von Forschungsbeiträgen: *Reifegrad* von Wissen und Anwendungsdomäne

Ein fundamentales Problem der Forschung ist, dass nichts wirklich *neu* ist, weil jedes Forschungsprojekt auf bestehendem Wissen aufbaut (Gregor u. Hevner, 2013, S. 344). Um dennoch den Grad der Neuheit eines Forschungsprojekts identifizieren zu können, stellen Gregor u. Hevner (2013) zur Kategorisierung unterschiedlicher Level eines Forschungsbeitrags vor. Sie schlagen die Einordnung von Forschungsergebnissen anhand der Dimensionen *Solution Matunity* und *Application Domain Matunity* vor. Abbildung 6 bezeichnet die resultierenden vier *Neuheits*-Level. Zur Einordnung der Forschung wird untersucht, welchen *Reifegrad* (von en. *mature*), von hoch nach niedrig, die Anwendungsumgebung des Artefakts (x-Achse) und die Lösung selbst (y-Achse) aufweisen.

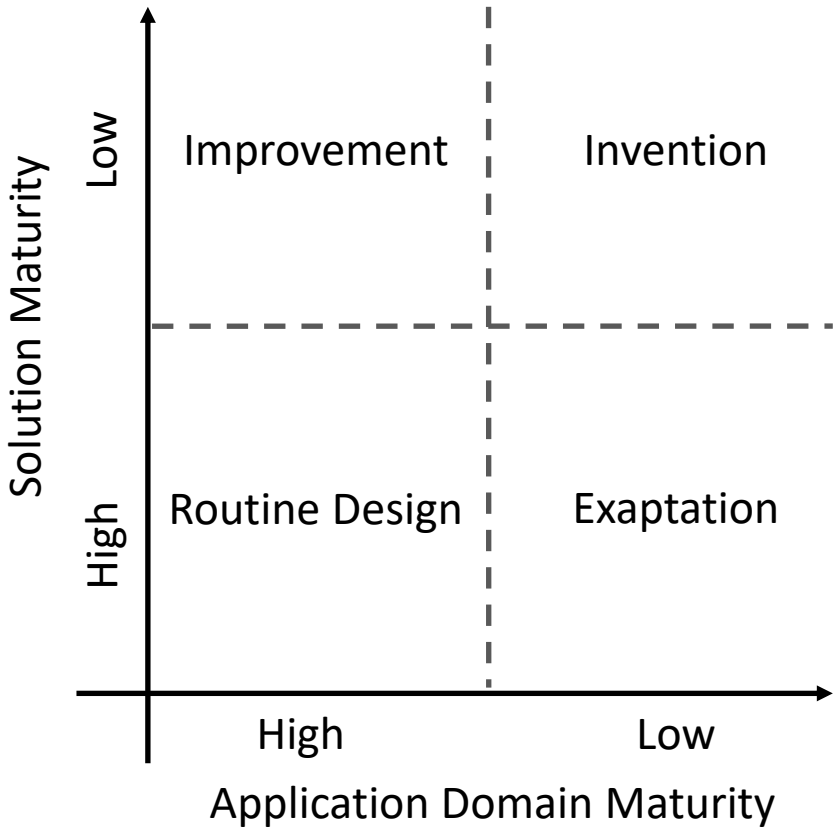

Abbildung 6: DSR Knowledge Contribution Framework nach Gregor u. Hevner (2013) (S. 345). Anhand der abgebildeten Dimensionen wird ein Artefakt des Forschungsprozesses (vgl. Abb. 6) einem der vier Quadranten zugeordnet zugeordnet. Die Zuordnung unterstützt den Forschenden bei der Identifikation seines Forschunsbeitrags.

„Invent new solutions for new problems“—
(Gregor u. Hevner, 2013, S. 345, Invention)

Forschungsergebnisse als *Invention* (folgend: Erfindung) zu identifizieren, erfordert die explorative Suche nach Lösungen in einer unerforschten Umgebung. Die Ungewissheit der zugehörigen Umgebung fordert eine Evaluation des Artefakts erfinderischer Forschungsaktivitäten im realen Kontext und dessen Wissenbeitrag zur Ω und/oder Λ Wissensbasis, um als DSR betrachtet zu werden. Gregor und Hevner diskutieren, dass echte Erfindungen nicht vollständig sauber in den Forschungsprozess gemäß DSR eingeordnet werden können. (Gregor u. Hevner, 2013, S. 346) Simon (1996) führt an, dass die Konzeptionalisierung des Problems und dessen Lösungsansätze selbst, getrieben von reinem Interesse des Forschenden, ein grundlegender Teil des Wissensbeitrags ist. Gregor und Hevner stellen fest, dass Erfindungen stets als Artefakt oder Instanz auftreten. Keine Theorie, basierend auf einer Ansammlung existierenden Wissens und Artefakten, kann als Erfindung deklariert werden, weil die Anwendungsdomäne aufgrund zahlreicher Publikationen hohe Reife aufweist. (Gregor u. Hevner, 2013, S. 346)

„Develop new solution for known problems“—
(Gregor u. Hevner, 2013, S. 345, Improvement)

In bekannten Anwendungsumgebungen effektivere und effizientere Produkte, Prozesse, Services, Technologien oder Ideen zu entwickeln ist das Ziel von *Improvement* (folgend: Verbesserung). Innovative und nützliche Artefakte lösen ein Problem in einer solide verstandenen Umgebung und treiben den Fortschritt voran. Gregor u. Hevner (2013) verweisen beispielhaft auf die Notwendigkeit eines besseren Werkzeugs nach McLaren u. a. (2011) (S. 2) in einer wohlbekannten Umgebung. Drei kumulative Stufen ordnen den Wissensbeitrag einer Verbesserung den Wissensbasen zu: In Stufe 1 tragen verbesserte Instanzen von Artefakten Wissen zur Λ-Wissensbasis bei. Repräsentiert der Prozess zur Erstellung eines Artefakts (Methode, Modell, Design-Prinzip) neuartiges Wissen, identifizieren Gregor und Hevner solche *allgemeinere* Artefakte als Beitrag der Stufe 2. Einen Wissenbeitrag der Stufe 3 liefert ein Artefakt, das *„[...] mid-range design theory [...]“(Gregor u. Hevner, 2013, S. 346)* in Form verbesserten Verständnisses seiner Problemstellung und Lösungsmethode erzielt. Ein Beitrag von Ω-Wissen ist möglich, wenn die Auswertung des Artefakts zum besseren Verständnis zugrunde liegender *Kernel Theorien* (Hevner u. Chatterjee (2010) (S. 39) i.V.m. Walls u. a. (1992)) beiträgt. (Gregor u. Hevner, 2013)

„Extend known solutions to new problems“—
(Gregor u. Hevner, 2013, S. 345, Exaptation)

Ideen zur Forschung im Sinne von DSR fußen auf der Notwendigkeit zur Lösung einer realen Problemstellung. Demnach erzeugen Forscher Artefakte in Abhängigkeit des zugehörigen Anwendungsfalls. Die Strategie eines Artefakts zur Lösung einer Problemstellung können in einer fremden Umgebung suboptimal oder nicht anwendbar sein. Folglich müssen Artefakte zur Lösung eines ähnlichen Problem in fremden Umgebungen *angepasst* werden. Gregor und Hevner verwenden die Terminologie *exapt (exaptation)* zur Präzisierung der Vorgehensweise, die ein Forscher anwendet, um bekannte Artefakte für neue Umgebungen zu transformieren. Sie verweisen auf die Bedeutung des Begriffs im Sinne biologischer Evolution: Nach Gould u. Vrba (1982) beschreibt *Exapatation* die Anpassung einer ursprünglichen (existenten) Eigenschaft oder Funktion zu einen anderen Zweck. Die Übertragung bekannten Wissens auf andere Domänen kann Λ-Wissen in den gleichen drei Stufen wie Verbesserungen beitragen. Beiträge zur Ω-Wissensbasis sind im Fall verbesserten Verständnisses neuer Artefakte möglich. (Gregor u. Hevner, 2013)

„Apply known solutions to known problems“—
(Gregor u. Hevner, 2013, S. 345, Routine Design)

Routine Design weist den höchsten Reifegrad in beiden Dimensionen auf. Bekanntes Wissen löst bekannte Probleme. Gregor und Hevner führen an, dass nach Stokes (2011) in der Routine-mäßigen Anwendung bekannten Wissens neue Erkenntnisse entdeckt werden können. In solchen Fällen finden die anderen Quadranten aus Abbildung 6 entsprechend Anwendung. Weiter betonen Gregor und Hevner, dass zwischen qualitativ hochwertigem, professionellem und/oder kommerziellem Einsatz existierender Artefakte und Forschung im Sinne des DSR differenziert werden müssen. Schlüssel dazu ist die Identifikation von Beiträgen zur Ω- und Λ-Wissensbasis. (Gregor u. Hevner, 2013, S. 347)

Die folgenden beiden Abschnitte analysieren den Forschungsprozess dieser Arbeit anhand der vorgestellten Dimensionen „Solution Maturity“ und „Application Domain Maturity“.

4.3.1 Untersuchung *Solution Maturity*

Die Problemstellung dieser Arbeit sieht vor, ein maschinelles Lernmodell zur Schätzung von Ankunftszeiten in der Schifffahrt zu erzeugen. Nach explorieren möglicher Vorgehensweisen wurde der Bereich des Deep Learning mit neuronalen Netzen als mögliches Verfahren identifiziert. Die Architektur des neuronalen Netzes dieser Arbeit (vgl. Abb. 12) verwendeten etablierte Techniken *(Layer)*. Im Design-Prozess zur iterativen Optimierung des Artefakts kommen etablierte Methoden und Algorithmen aus dem Bereich des maschinellen Lernens zum Einsatz.

Im zweiten Design-Schritt dieser Arbeit werden Modelle (Instanzen des Artefakts) auf andere Zielhäfen transferiert, um individuelle Instanzen der Zielhäfen zu erhalten. Die angewandte Methode zur Identifikation eines optimalen Modells folgt eigens definierten Konfigurationen, die es bestimmten Teile des *ursprünglichen* neuronalen Netzes erlaubt, sich an einen neuen Hafen anzupassen. Diese Technik findet in verwandten Domänen bereits Anwendung, wobei je nach Anwendungsfall verschiedene Konfigurationen *neue* optimale Artefakt-Instanzen hervorbringen. Folglich weist der Transfer-Prozess hohe Reife auf, was die Lösung dem dritten oder vierten Quadranten zuordnet *(High Solution Maturity)*.

4.3.2 Untersuchung *Application Domain Maturity*

Einerseits finden automatische Systeme zur Schätzung der Ankunftszeit in der Praxis keine Anwendung, andererseits schlagen Forscher in der Literatur Methoden und Prinzipien vor, wie Modelle zur Schätzung von Ankunftszeiten in unterschiedlichen Umgebungen umgesetzt werden können. Angeführte Evaluationsergebnisse zeigen empirisch, dass im Rahmen der vorgestellten Forschungsprozesse (vgl. Kap. 2) vielversprechende Resultate erzielt werden. Um den Reifegrad der Anwendungsumgebung dieser Arbeit feststellen zu können, muss zunächst die Umgebung selbst identifiziert werden. Kapitel 1 stellt die Umgebung fest, woraus die vorliegende Problemstellung unzuverlässiger Schätzungen von Ankunftszeiten in der Schifffahrt entspringt. Die eingesetzte Technik gehört dem weitreichenden Gebiet des maschinellen Lernens an, was implizit die Frage nach der verwendeten Datenbasis aufwirft. Zum Design der maschinellen Lernmodelle dienen historische Daten in Form *unformatierter* Zeitreihen. „Unformatiert“ heißt, dass Form und Inhalt der Daten keinen Aufschluss über *Meta-Informationen* wie die Zugehörigkeit eines Datenpunkts zu einer Route enthalten (zur Sichtung enthaltener Felder, vgl. Tabelle A10). Folglich befindet sich die Anwendugnsdomäne dieser Arbeit in der Schätzung von Ankunftszeiten mittels eines maschinellen Lernmodells auf historischen AIS Datensätzen. Die Mehrzahl verwandter Problemstellungen in der Literatur wendet Methoden zur Klassifikation der Abweichung von ungenauen geplanten Ankunftszeiten an. Die auftretende Unsicherheit in Schätzungen wird besipielsweise mit Verteilungen approximiert: Die resultierenden Artefakte klassifizieren verspätet oder verfrüht ankommende Schiffe. Die Literatur wendet häufig einen *Random Forest Algorithmus* zur Klassifikation an. Weitere vertretene Algorithmen sind *Markov-Ketten*, *Bayes-Statistik*, *Entscheidungsbäume*, allgemeine *Cluster-Verfahren* sowie *Reinforcement Learning* (vgl. Kap. 2.1 & 2.2). Wenige Arbeiten verfolgen die Strategie, mittels *Deep Learning* eine präzise Schätzung von Ankunftszeiten umzusetzen. Bodunov u. a. (2018) verwenden einen mehrstufigen Ansatz, der zuerst den Zielhafen, gefolgt von der Ankunftszeit eines Schiffs prädiziert. Die *abweichende* Betrachtung der Problemstellung setzt diese

Arbeit von der Literatur ab. Meist werden Routen als Ankerpunkt für Ankunftszeitschätzungen verwendet und Klassifikatoren schätzen die Abweichung von einer geplanten Ankunftszeit. Diese Arbeit verfolgt eine in der Literatur bislang selten vertretene Lösungsstrategie. Unter Einsatz eines neuronalen Netzes soll auf Basis historischer Daten des AIS die Dauer bis zur Ankunft eines Schiffs geschätzt werden, woraus sich die leicht die Ankunftszeit ermitteln lässt. Geplante Ankunftszeiten oder Modellierungen von Unsicherheiten auf historischen Daten finden keine Anwendung. Außerdem werden a-priori keine Routen ermittelt oder explizit bestimmt, auf welcher Route sich ein Schiff befindet. Eine (Teil-) Route ergibt sich implizit aus konsekutiven historischen Datenpunkten.

Zusammenfassend existiert eine Vielzahl von Ansätzen, die andere Techniken des maschinellen Lernens anwenden und/oder andere Vorgehensweisen zur Lösung der Problemstellung (nicht ausschließlich in der Schifffahrt) verfolgen. Deep Learning zur Schätzung von Ankunftszeiten ist selten vertreten, die meisten Verfahren wenden Klassifikationsverfahren an, basieren auf Routen oder modellieren Unsicherheiten ursprünglich geplanter Ankunftszeiten. Dem zur Folge lässt sich der Forschungsprozess anhand der Dimension „Application Domain Maturity“ als niedrig einstufen. Mit der Folgerung des vorigen Abschnitts 4.3.1, der die „Solution Maturity“ hoch einstuft, gehört der Forschungsprozess dieser Arbeit insgesamt der Exaptation-Kategorie (3. Quadrant) an.

Irgendwo anders hin: Problem, wenn keine geplante Ankunftszeitschätzung existiert

4.4 Angewandter Design Science Research

Die folgenden Unterkapitel wenden das theoretische Paradigma des Design Science Research auf die vorliegende Problemstellung der Prädiktion von Ankunftszeiten in der Schifffahrt mit anschließendem Wissenstransfer auf einen anderen Hafen an. Abbildung 7 zeigt die Interaktion des DSR-Forschungsprozesses mit der vorliegenden Umgebung und Wissensbasis mittels des Relevanz-Zyklus resp. Rigor-Zyklus.

4.4.1 Umgebung: Identifikation und Relevanz

Die Umgebung eines Forschungsansatzes nach DSR umfasst jene Akteure, Organisationen und Technologie, die von der Forschung betroffen sind. DSR versucht reale Problemstellungen aus einer Umgebung zu lösen. Die Problemstellung dieser Arbeit fasst Fuß in unzuverlässigen Ankunftszeiten in der Schifffahrt, weshalb anschließendes *Supply Chain Management (SCM)* (vgl. Harland (1996)) und Logistik letztlich unplanbar werden. Außerdem ergibt sich ein technisches Problem aus der

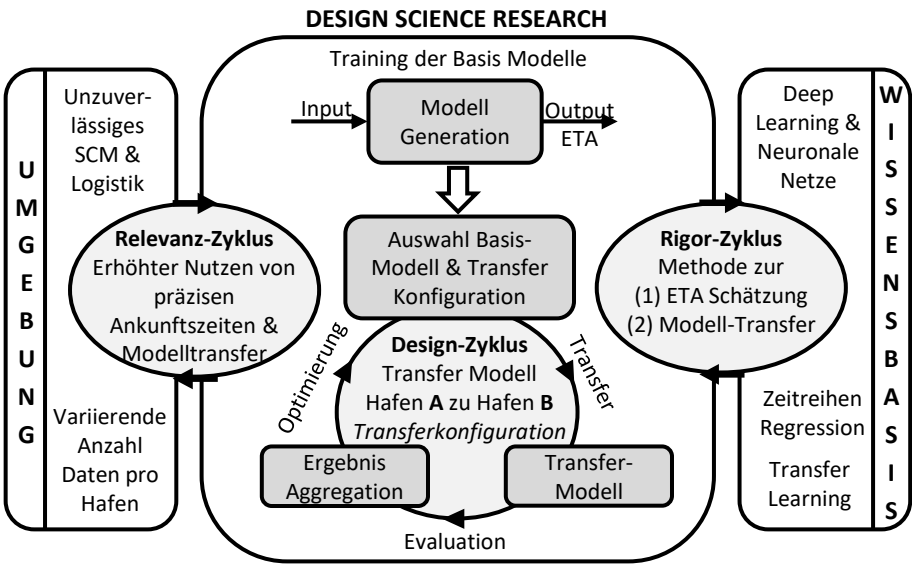

Abbildung 7: Anwendung des DSR Knowledge Contribution Framework, adaptiert von Gregor u. Hevner (2013) (S. 345). Der in dieser Arbeit angewanddte Forschungsprozess ist im zentralen Element der Abbildung (Design Science Research) abgebildet. Im Design-Zyklus findet die iterative Generierung und Verbesserung von Artefakten statt. Der Relevanz-Zyklus begründet das praktische Anwendungsgebiet. Der Rigor-Zyklus betrachtet das generierte Artefakt hinsichtlich seines Beitrags zur existierenden Wissensbasis.

variierenden Anzahl Daten pro Hafen. Maschinelle Lernmodelle, insbesondere im Bereich Deep Learning mit neuronalen Netzen, benötigen große Datenmengen (vgl. Kap. 1.6). Daraus eröffnet sich die Chance, eine Methode zur Generierung eines Modells für einen Hafen, dessen Datengrundlage *eigentlich* zu gering zum Training eines eigenen Modells ist.

Zum fundierten Forschen stellt der Relevanz-Zyklus stets die Frage nach dem Nutzen des Artefakts in der ursprünglichen Umgebung. Iterativ wird überprüft, ob die aktuelle *Version* des Artefakts die Problemstellung(en) adressiert und eine gewünschte Verbesserung erzielt. Hevner u. Chatterjee (2010) (S. 209 f.) führen ein häufig auftretendes, in Journals wie *MIS Quartely* und *Communications of the Association for Information System* diskutierte *Krise* an: Eine *„[...]crisis in the IS field“*, die Probleme von Forschungsprozessen hinsichtlich *utilization* und *relevance* aufweist. Zu viel Forschung sei *method/theory driven* und nicht *problem driven*. Die Fragestellung dieser Arbeit begründet sich in einem Problem aus der Praxis, was implizit das Dilemma der Utilization und Relevance löst (Hevner u. Chatterjee, 2010, S. 210).

4.4.2 Wissensbasis: Grounding, Beiträge und Chancen

Das vorige Kapitel 4.4.1 stellt fest, welcher Umgebung der Forschungsprozess entspringt und welche praxisrelevanten Problemstellungen bestehen. Das theoretische Handwerkszeug zur iterativen Generierung eines Artefakts liefert das initiale Grounding des Rigor-Zyklus hinsichtlich des Forschungsthemas. Existierendes Wissen aus der zugehörigen Wissensbasis wird benannt und zur Lösung der Problemstellung angewendet. Mit dem Ziel der Verbesserung von Ankunftszeitschätzungen und dem Wissenstransfer maschineller Lernmodelle auf *andere* Häfen, bilden Techniken aus dem Bereich des Deep Learning & neuronaler Netze die Grundlage zum Design der Artefakte. Weiter finden Techniken zur Bearbeitung von Zeitreihen und die Modellierung als Regressionsproblem Anwendung. Wissen über CNNs, unterschiedliche Arten von Layern und der Prozess zum Training maschineller Lernmodelle (vgl. Kap. 3.1) wird aus der Wissensbasis geschöpft. Wissen aus dem verwandten Themengebiet der Klassifikation von Bildern, hat Einfluss auf die Generierung der Artefakte. Besonderer Fokus liegt hierbei auf Techniken, die bereits trainierte *Basis*-Modelle auf Nischen-Anwendungsfälle mit, natürlich bedingten, geringen Datenmengen anwenden (vgl. Kap. 3.3).

Die Art des Beitrags dieser Arbeit zur Wissensbasis wurde in Kapitel 4.2 als *Exaptation* identifiziert. Er soll Aufschluss darüber geben, ob bekannte Techniken (1) zu einer Verbesserung der Ankunftszeitschätzung in der Schifffahrt liefern können, und inwiefern (2) der Transfer von Wissen von einem Hafen auf einen anderen Hafen mit geringen Datenmengen in ausreichender Qualität möglich ist.

4.4.3 Der Forschungsprozess im Sinne von DSR

Forschung unter Anwendung des DSR ist nach Gregor u. Hevner (2013) ein sozialer Prozess, der seiner Umgebung (vgl. Kap. 4.4.1) entspringt und versucht, einen neuartigen Beitrag zur Wissensbasis (vgl. Kap. 4.4.2) zu liefern. Der tatsächliche Forschungsprozess findet im Design-Zyklus des Design Science Reasearch Framework (vgl. Abb. 1) statt. Der Design-Zyklus iteriert häufiger als die anderen beiden Zyklen, wobei jede Iteration die

- Generierung eines Artefakts,
- dessen Evaluation und
- anschließendes Feedback zur Optimierung des Artefakts

durchführt (Hevner, 2007, S. 90). Dabei muss das Artefakt stets dem Kriterium der Relevanz und Rigor gerecht werden.

4.4.4 Umsetzung des Design-Zyklus

Im Folgenden wird abstrakt das Forschungsdesign dieser Arbeit, sprich die Umsetzung des Design-Zyklus, vorgestellt. Die drei vorgestellten Aspekte einer Iteration des Design-Zyklus bieten die Rahmenbedingungen des angewandten Forschungsprozesses.

Zum Anstoßen des Design-Zyklus werden *Basis*-Modelle von Häfen benötigt, die auf andere Häfen transferiert werden können. Initial werden folglich Modelle auf eigenen Daten trainiert und auf Basis der Evaluationsergebnisse optimiert. Die Evaluation stützt sich allein auf die Daten des eigenen Hafens. Das Resultat sind optimale Artefakte (Modelle) mehrerer Häfen mit unterschiedlicher Präzision bei der Schätzung von Ankunftszeiten. Diese Artefakte sind im Design Science Research-Bereich der Abbildung 7 unter „Training der Modelle“ abgebildet, die als Basis-Modell in den Design-Zyklus einfließen. In Verbindung mit der Methode zur Umsetzung eines Transfers (in Abbildung 7 „Transfer Konfiguration“) wird ein neues Artefakt (Transfer-Modell) erstellt. Es folgt dessen Evaluation, die Ergebnisse und das Artefakt werden zur Aggregierung gespeichert. Auf Basis der Ergebnisse werden Kombinationen aus Basis-Modell und Transfer-Konfiguration anhand der Evaluationsergebnisse bewertet und letztlich optimiert. Die ganzheitliche Betrachtung der Artefakte aller Iterationen ermöglicht es, Abhängigkeiten von Basis-Häfen und Transfer-Konfigurationen zur optimalen Performance transferierter Modelle zu identifizieren.

Definition 3: ETA Sensor: *Die im Design Prozess des DSR auftretenden Instanzen der Artefakte werden in dieser Arbeit als ETA Sensor bezeichnet. Ein ETA Sensor ist ein maschinelles Lernmodell zur Schätzung von Ankunftszeiten in der Schifffahrt für einen Zielhafen.*

5 Neuronales Netz zur Ankunftszeitschätzung: ETA Sensor

Im vorigen Kapitel 4 wird der Forschungsansatz anhand des Frameworks des Design Science Research erläutert. Die praktische Umsetzung und Erstellung eines Artefakts zur Schätzung von Ankunftszeiten in der Schifffahrt wird in diesem Kapitel erläutert.

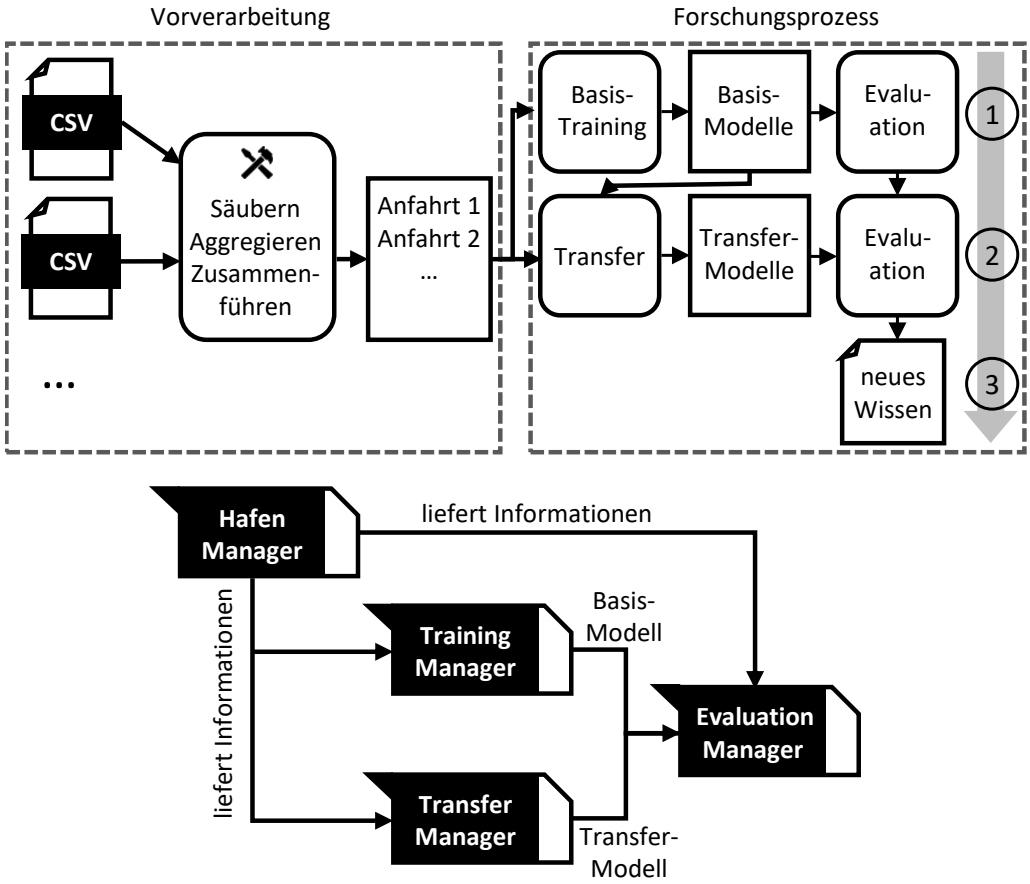

Abbildung 8: Abstrakte Verarbeitungs-Pipeline unter Einsatz des Design Science Research Forschugnsparadigma und organisatorische Komponenten. An die Vorverarbeitung des dänischen AIS Datensatzes schließt der Forschungsprozess. Er setzt in zwei Schritten untergeordnete Prozesse zur Generation und Verbesserung von Artefakten. Schließlich resultiert die Evaluation der Artefakte in generiertem Wissen. Der untere Bereich zeigt umgesetzte Module zur Organisation, Handhabung und ihrer Interaktion.

Der Aufbau der in Abbildung 8 dargestellten Verarbeitungs-Pipeline orientiert sich an den klassischen Schritten zur Umsetzung eines maschinellen Lernprojekts. Die Datzensätze des dänischen AIS müssen eine vorgelagerte Verarbeitung durchlaufen, um zum Training eines maschinellen Lernalgorithmus geeignet zu sein (Abb. 8 Vor-

verarbeitung). Im Anschluss folgt der tatsächliche Forschungsprozess, wo zunächst Basis-Modelle erzeugt und evaluiert werden. Darauf folgt der Transfer von Basis-Modellen auf andere Zielhäfen, deren Modelle wiederum evaluiert werden und in Verbindung mit den Evaluationsergebnissen der Basis-Modelle (und der Konfiguration des Transfers) neues Wissen bezüglich der Transfer-Methode liefern.

Der untere Teil von Abbildung 8 zeigt Abhängigkeiten eingeführter Module zur Handhabung und Organisation der Pipeline. Sie dienen der Haltung von Zuständen und Evaluationsergebnissen, verknüpfen Schritte der Pipeline und stellen zum Training und Transfer notwendige Informationen zur Verfügung. Die folgenden Kapitel führen die Module ein, sobald sie zur Durchführung eines Verarbeitungsschritts in der Pipeline benötigt werden.

Die folgenden Kapitel erläutern die Funktionsweise einzelner Komponenten und deren Interaktion aus Abbildung 8. Die abgebildete inhaltliche Zugehörigkeit lässt sich technologisch beim Training und der Evaluation von Basis- und Transfer-Modellen nicht trennen. Aus diesem Grund werden früh Terminologien eingeführt, die in folgenden Kapiteln Anwendung finden. 5.1 behandelt die notwendigen Schritte zur Formatierung der ursprünglichen AIS Daten. Kapitel 5.2 beschreibt die Vorgehensweise zur automatischen Identifikation und *Labeln* von Trainingsbeispielen. Die Vorverarbeitung schließt mit der Bereitstellung der Daten in Kapitel 5.2 ab. Im Bereich des Forschungsprozesses leitet Kapitel 5.4 mit der Architektur des eingesetzten neuronalen Netzes ein, gefolgt von der Vorstellung des Trainingsprozesses in Kapitel 5.5. Die Idee und Umsetzung eines Transfers behandelt Kapitel 5.6.

5.1 Beschaffenheit und Vorverarbeitung der AIS Daten

Als Grundlage zum Training der maschinellen Lernmodelle zur Schätzung von Ankunftszeiten dienen historische Daten des dänischen Automatic Identification Systems (AIS). Die Danish Maritime Authority (DMA) stellt die Daten in Form von .csv-Dateien kostenlos zur Verfügung. Eine .csv-Datei enthält die Daten eines Tages. Im Rahmen dieser Arbeit wurden alle 365 Dateien des Jahres 2020 verarbeitet. Die folgende Auflistung listet jene Felder auf, die zum Training eines maschinellen Lernmodells als geeignet eingestuft wurden.

Definition 4: Timestamp: *Der Zeitpunkt, zu dem ein Datenpunkt an der dänischen AIS Basis-Station ankommt, wird als Zeitstempel im Format DD/MM/YYYY hh:mm:ss dem Datenpunkt zugeordnet und gespeichert.*

Definition 5: MMSI: *Das Global Maritime Distress and Safetly System (GMDSS) weist jedem Schiff eine eindeutige Rufnummer einer Seefunkstelle zu, die Maritime Mobile Service Identity (MMSI).*

Definition 6: Latitude: *Der Breitengrad der Position, an dem sich ein Schiff zum Zeitpunkt des Sendens des Datenpunkts befindet, wird in Grad im Feld Latitude gespeichert.*

Definition 7: Longitude: *Der Längengrad der Position, an dem sich ein Schiff zum Zeitpunkt des Sendens des Datenpunkts befindet, wird in Grad im Feld Longitude gespeichert.*

Definition 8: Navigational Status: *Der aktueller Navigationsstatus des Schiffs beschreibt die aktuelle vom Schiff ausgeführte Aufgabe.*

Definition 9: SOG: *SOG (Speed Over Ground) hält die aktuelle Geschwindigkeit des Schiffs, relativ zur Oberfläche der Erde fest.*

Definition 10: COG: *COG (Course Over Ground) hält die aktuelle Fahrtrichtung des Schiffs, relativ zur Oberfläche der Erde fest.*

Definition 11: Ship Type: *Typ des Schiffs*

Definition 12: Width: *Breite des Schiffs in Metern.*

Definition 13: Length: *Länge des Schiffs in Metern.*

Definition 14: Draught: *Tiefgang des Schiffs bis zur Wasserlinie in Metern.*

Definition 15: Destination: *Zielhafen des Schiffs.*

Die vorgelagerte Formatierung von Rohdaten sorgt dafür, dass der maschinelle Lernalgorithmus mit den Ausprägungen der Felder umgehen kann. Die Funktionsweise aller Lernalgorithmen erfordert numerische Typen, um die Eingaben verarbeiten zu können. Folglich sind undefinierte, leere und kategorische Datenpunkte nicht interpretierbar und müssen behandelt werden. Datenpunkte mit leeren oder undefinierten Feldern werden direkt nach dem initialen Laden der Rohdaten herausgefiltert. Das folgende Kapitel 5.1.1 zeigt, wie die kategorischen Felder *Navigational Status* und *Ship Type* auf numerische Werte abgebildet werden.

5.1.1 Behandlung kategorischer Datentypen

Neuronale Netze erwarten numerische Eingabetypen. Zur Handhabung nicht numerischer, kategorischer Eingaben wird die Technik des One-Hot-Encoding auf die Felder *Navigational Status* (Def. 8) und *Ship type* (Def. 11) angewendet. Das Grundprinzip des One-Hot-Encoding erweitert ein 1-dimensionales kategorisches Feld auf einen Vektor der Größe $1 \times n$, wobei n der Anzahl möglicher Ausprägungen des Feldes entspricht. Der resultierende Vektor für ein kategorisches Feld X besteht aus

Einträgen x_i mit $i \in [0, n-1]$, wobei

$$x_i = \begin{cases} 1, & \text{wenn Feld ist in i ausgeprägt} \\ 0, & \text{sonst} \end{cases} \tag{5.1}$$

Folglich ist genau ein Eintrag $x_i = 1$, alle anderen Einträge sind 0. Die beiden Felder *Navigational Status* und *Ship Type* können nicht definierte (NaN, undefined) oder fehlerhafte Werte enthalten. Jeder nicht definierte oder fehlerhafte Eintrag wurde auf eine eigenen Kategorie abgebildet. Die Aussagekraft solcher Kategorien ist intuitiv gering und kann mit sich selbst im Widerspruch stehen, wenn sich beispielsweise der Typ eines Schiffs innerhalb einer Anfahrt ändert. Um solchen unerwünschten Nebeneffekten entgegen zu wirken, wurden eigene *Container*-Klassen eingeführt, die nicht quantifizierbare Ausprägungen der Kategorischen Felder abbilden. Nach der Codierung jedes Eintrags in einen Vektor der Größe $1 \times n$, werden die einzelnen möglichen n Ausprägungen als jeweils eigenes Feature an die Eingabedaten für das neuronale Netz angehängt. Die Codierung eines kategorischen Feldes mit n Ausprägungen führt folglich zur Vergrößerung der Eingabe um $n - 1$ Features.

Die offizielle AIS Dokumentation listet für das Feld *Ship Type* 100 mögliche Ausprägungen (VT Explorer, 2021b). Die One-Hot-Codierung würde demnach jeden Eintrag auf einen Vektor der Größe 1×100 abbilden, was die Input-Dimension extrem aufbläht. Um die Eingabedimension nicht zu überladen, sind 11 relevante Klassen für das Feld *Ship Type* identifiziert worden, die sich aus Zusammenfassungen *ähnlicher* Klassen ergeben (vgl. Tab. A11). Beispielsweise sind unter der verwendeten Ausprägung *Cargo* sechs ursprüngliche Klassen zusammengefasst, die alle Cargo-Sub-Klassen sind. Das Argument, mehrere Klassen zu vereinen begründet sich in der Natur One-Hot-Encodeter Eingaben: Für ein Netz ist die Ähnlichkeit von Cargo-Sub-Klassen nicht ersichtlich. Der codierte Vektor enthält stets einen Eintrag = 1 ohne Referenz auf die anderen fünf Cargo-Sub-Klassen; diese anderen *ähnlichen* Klassen sind, genau wie alle anderen Klassen von *Ship Type* mit 0 codiert. Das Netz kann folglich keinen Zusammenhang zu anderen Eingabefeldern von *Ship Type* herstellen und muss spezifisch für jede Sub-Klasse eigene Merkmale in den Daten feststellen. Das zweite kategorische Feld *Navigational Status* hat nach der offiziellen Dokumentation 16 mögliche Ausprägungen VT Explorer (2021a). Auch hier wurden die relevantesten Klassen identifiziert und die Dimension des One-Hot Vektors auf 1×14 reduziert (vgl. Tab. A12).

5.1.2 Die Problematik von *Zeit* als Trainingsdatum & Output

Zeiten nehmen in dieser Arbeit eine zentrale Rolle ein: Zum einen ist das ultimative Ziel die Schätzung einer Ankunftszeit und zum anderen beinhalten die Trainingsdaten einen Zeitstempel (vgl. Def. 4). Das Problem ergibt sich aus der *Beschaffenheit*

von Zeit selbst, sie schreitet stets voran. Folglich sind die Zeitstempel aller Trainingsdaten niedriger als solche, die ein trainiertes Modell im praktischen Einsatz erhält. Dieser Fakt lässt sich auf den Output eines Modells übertragen. Geschätzte Ankunftszeiten würden immer weiter in der Zukunft liegen, je länger ein Modell im Praktischen Einsatz ist. Alle Prognose lägen außerhalb des bekannten Wertebereichs des Datensatzes. Es folgt ein (intuitiver) Widerspruch der grundlegenden Definition *„[...] improve through experience.“* (vgl. Kap. 3.1), da ein Netz keine Erfahrung von etwas *unbekannten* sammeln kann. Das Problem lässt sich lösen, in dem sowohl der Zeitstempel in den Eingabedaten als auch die Ausgabe der Modelle als *Dauer* innerhalb des Jahres modelliert werden, in dem sie sich befinden. Der referenzierte *Startpunkt* jedes Jahres ist am 1. Januar um 00:00:00, von dessen Basis aus die vergangene Zeit innerhalb des Folgejahres abgebildet wird.

Definition 16: Time within Year: *Time within year* t_i^Y *ist die vergangene Zeit seit dem 1. Januar um 00:00:00 eines Jahres in Sekunden. Es gilt für alle* $t_i^Y \in [0, 31622400]$ *mit i=Index des Datenpunktes.*

Die Referenz zum zugehörigen aktuellen Jahr des Datenpunkts wird extern in der Definition eines Datensets (vgl. Kap. 5.2) festgehalten.

5.1.3 Skalieren der Eingaben: *Feature Scaling*

Im aktuellen Stand der Datenvorverarbeitung sind relevante Felder identifiziert und kategorische Datentypen numerisch repräsentiert worden. Die Eingaben weisen bislang unterschiedliche Wertebereiche, entsprechend ihres Typs auf (vgl. Tab. A10). Minimal- und Maximalwerte unterscheiden sich je nach Feature deutlich: Geografische Positionen können negative Werte beinhalten, die restlichen Features nicht. *Time* und *Label* haben einen vielfach höheren Wertebereich als alle andere Features. *Width* hat, abgesehen von den kategorischen Eingaben, den kleinsten Wertebereich mit $[0, 80]$. Um beispielsweise die Relevanz des *kleinen* Features *Width* im Vergleich zu den beiden Zeiten nicht zu untergraben, können die Eingaben eines neuronalen Netzes auf einheitliche Wertebereiche skaliert werden. Diese Vorgehensweise in der Vorverarbeitung wird *Feature Scaling* genannt.

„If we want to get the best-mixed juice, we need to mix all fruit not by their size but based on their right proportion.“—
(Roy, 2020, In Depth Analysis für Feature Scaling)

Roy (2020) zeigt detailliert und beispielhaft, was Feature Scaling für ein neuronales Netz bewirkt. Wie am Anfang dieses Kapitels angeführt, besitzt jedes Feld des AIS Datensets einen spezifischen Wertebereich. Die Pipeline normalisiert nach der

Codierung kategorischer Felder (siehe 5.1.1) den Wertebereich jedes Felds auf das Intervall $[0, 1]$. Die Beschaffenheit der Daten des AIS Datensatzes erlaubt es im Fall dieser Arbeit nicht, die übliche Vorgehensweise zur Bestimmung der minimalen und maximalen Ausprägungen aus den Daten zu ermitteln. Der Datensatz müsste jede mögliche Ausprägung aller Felder enthalten, was insbesondere für geografische Positionsdaten *(Latitude, Longitude)* nicht angenommen werden kann. Das gleiche Problem besteht für sämtliche Eigenschaften von Schiffen *(Length, Width, Draught, SOG, Course)*, deren vollständige Definition des Wertebereichs nicht auf Basis der enthaltenen Ausprägung im Datensatzes angenommen werden kann. Aus diesem Grund wurden für alle Felder fest Minima und Maxima definiert, die zum Skalieren der Eingaben angewendet werden. Die unteren und oberen Intervallgrenzen sind in Tabelle A10 aufgeführt. Folglich sind die Wertebereiche jeder Ausprägungen über alle Modelle äquivalent, was im späteren Verlauf der Arbeit den Transfer von Modellen unterstützt.

5.1.4 Identifikation relevanter Zielhäfen

Stichprobenartigen Analysen von Daten einzelner Tage zur Folge, ermittelt das dänische AIS Daten für mehr als 1800 verschiedene Zielhäfen. Der überwältigenden Mehrheit der Häfen stehen zu geringe Datenmengen zum Training eines eigenen Modells zur Verfügung. Folglich wurden 12 vielversprechende Häfen aus dem Jahr 2020 aufgrund der *potentiell* hohen Anzahl Trainingsdatenpunkte ausgewählt. Tabelle 2 zeigt die Anzahl *Anfahrten* der ausgewählten Zielhäfen.

Zielhafen	Anzahl Anfahrten
Esbjerg	2337
Rostock	2192
Kiel	1498
Skagen	1032
Thyboron	784
Trelleborg	751
Hirtshals	645
Hvidesande	662
Aalborg	629
Copenhagen	621
Goteborg	620
Grenaa	509

Tabelle 2: 12 ausgewählte Zielhäfen des dänischen AIS Datensatzes. Die Anzahl Anfahrten sind der Grund zur Auswahl der gelisteten Häfen. Die Anfahrten wurden im Rahmen der Vorverarbeitung des vorliegenden Kapitels identifiziert, was alle 366 Datensätze des Jahres 2020 verarbeitet.

Alle Datenpunkte, die keinem der relevanten Zielhäfen aus Tabelle 2 zugeordnet sind, werden in der folgenden Pipeline nicht verarbeitet. Dieses Filtern erleichtert Handhabung der Datenbasis erheblich. Die ursprünglichen Daten des AIS im Jahr 2020 besitzen eine Größe von 655 GB in Form von .csv-Dateien. Das folgende Kapitel 5.2 beschreibt die weitere Verarbeitung und Aufarbeitung zusammengehöriger Dateien.

5.2 Vom *rohen* Datenpunkt zu Trainingsbeispielen

Das vorige Kapitel 5.1 hat im ersten Kontakt mit den *rohen* Daten des dänischen AIS die Datenbasis hinsichtlich fehlender und fälschlicher Ausprägungen bereinigt und relevante Datenpunkte mit zugehörigem Hafen identifiziert (vgl. Tab. 2). Das vorliegende Kapitel reichert die Daten in zwei Schritten an:

1. Zusammenführen von *Anfahrten*, die über mehrere Dateien hinweg spannen
2. Automatische Identifikation des Labels einer Anfahrt

In der kommerziellen Schifffahrt sind Handelsrouten über lange Strecken keine Seltenheit. Containerschiffe kreuzen Ozeane und umfahren Kontinente. Selbst die letzten Streckenabschnitte können sich über mehrere Tage oder Wochen erstrecken. Das dänische AIS speichert die ankommenden Daten täglich in separaten .cvs-Dateien.

Folglich sind Daten eines Schiffs über mehrere Dateien hinweg verteilt und die Pipeline benötigt ein System zur Datei-übergreifenden Verknüpfung von Datenpunkten. Ein *Anfahrt* ist demnach wie folgt definiert:

Definition 17: Anfahrt (1): *Eine Anfahrt besteht aus allen zusammenhängenden Datenpunkten, die dem selben Schiff (MMSI, vgl. Def. 5) und selben Zielhafen zugeordnet sind. Die enthaltenen Datenpunkte sind aufsteigend nach ihrer Time within Year (vgl. Def. 16) sortiert.*

5.2.1 Bestimmung der Ankunftszeit einer Anfahrt

Im bisherigen Verlauf von Kapitel 5 wird bereits deutlich, dass ursprüngliche Daten des AIS zum Training eines neuronalen Netzes formatiert werden müssen. Die definierten Merkmale (Def. 4 bis 15) beziffern das relevante Subset ursprünglicher Felder des AIS. Die gewünschte *Zielvariable*, nämlich *tatsächliche* Ankunftszeit des Schiffs am Zielhafen, ist bisher nicht enthalten. Auf Basis der existierenden Merkmale und der Gruppierung von Datenpunkten in Anfahrten, kann die gewünschte Zielvariable ermittelt werden. Aus der Kombination einer Anfahrt und zugehöriger Zielvariable entsteht eine Menge aus *Trainingsdatenpunkten*.

Definition 18: Trainingsdatenpunkt: *Ein Trainingsdatenpunkt X_t ist ein Datenpunkt einer Zeitreihe T zu einem Zeitpunkt t, der aus 36 Features (vgl. Tabelle A10) und zugehörigem Label besteht. Jeder Trainingsdatenpunkt ist eindeutig der MMSI eines Schiffs, einem Zielhafen und einer Anfahrt zugeordnet.*

Das Label eines Trainingsdatenpunkts beziffert die Dauer bis zur Ankunft des Schiffs am Zielhafen, wozu der Datensatz a-priori keine Angabe enthält. Ein *ursprünglicher* Datenpunkt kann gelabelt werden, wenn er in der Menge von Datenpunkten einer Anfahrt enthalten ist.

Definition 17 spezifiziert den Begriff *Anfahrt* als eine Zeitreihe, bestehend aus Datenpunkten. Eine Anfahrt beinhaltet folglich kein Label. Im Folgenden wird der Begriff *Anfahrt* dennoch zur Bezeichnung einer Zeitreihe aus *Trainingsdatenpunkten* verwendet. Bevor das Verfahren zur automatischen Ermittlung der Labels einer Anfahrt vorgestellt werden kann, setzt diese Arbeit ein Modul zur Organisation und Handhabung von (Ziel-)Häfen um. Die Implementierung des **Hafen-Managers** (vgl. Abb. 8) trägt die folgenden Funktionalitäten zur bisherigen Pipeline bei:

- Registrieren und Identifizieren von Häfen
- Bereitstellen Hafen-spezifischer Informationen
  - Alias Bezeichnungen

  - Geografische Position
  - Definition eines *Hafenbereichs*

- Berechnen der Entfernung einer GPS Position zum Hafen
- Speichern und Laden von Trainingsdurchläufen

Zum Labeln der Datenpunkte einer Anfahrt muss die Ankunftszeit des Schiffs am Zielhafen bestimmt werden. Eine Methode zur Identifikation solcher Datenpunkte wird benötigt, deren Position die Ankunft eines Schiffs am Zielhafen repräsentiert. Folglich müssen alle Datenpunkte klassifiziert werden, die sich innerhalb des Gebiets der Anlegestellen eines Zielhafens befinden.

Definition 19: Hafenbereich: *Ein Hafenbereich ist die Menge aller geografischen Positionen (in Höhen- und Breitengrad), die sich innerhalb einer Fläche um einen Hafen befinden. Er definiert sich aus dem geografischen Mittelpunkt eines Hafens und einem Radius r in km. Der Radius definiert einen Kreis um den Hafen-Mittelpunkt auf der Oberfläche der Erde. Die Fläche im Inneren des Kreises beinhaltet alle Anlegestellen eines Hafens.*

Abbildung 9: Hafenbereich von Esbjerg (DK), definiert durch seinen geografischen Mittelpunkt und einen Radius r. Die Pfeile an den gestrichelten Linien zeigen die Übergangspunkte zum Hafenbereich. Der erste Datenpunkt einer Anfahrt innerhalb des Hafenbereichs wird zur Generierung der Labels herangezogen. (Google, 2021a)

Aufgrund der zeitlich aufsteigenden Reihenfolge der Datenpunkte innerhalb einer Anfahrt befinden sich ab einem Zeitpunkt t_i, dessen geografische Position innerhalb des Hafenbereichs liegt, alle folgenden Datenpunkte mit den Zeitpunkten $t_{i+1}, ..., t_n$

ebenfalls im Hafenbereich ($i \in [1, n]$ und n = Anzahl Datenpunkte einer Anfahrt). Um die Zugehörigkeit eines Datenpunkts zum Hafenbereich festzustellen, darf der Punkt nicht weiter als Radius r vom Hafenmittelpunkt entfernt sein. Die genaue Entfernung ergibt sich mit der Haversine Funktion (Korn u. Korn, 2013) zur Berechnung der Abstände zweier Punkte auf einer Kreisoberfläche. Aus der Menge aller Datenpunkte innerhalb des Hafenbereichs, wird der Datenpunkt mit minimalem Zeitpunkt t_{min} (Time wihtin Year, vgl. Def. 16) zur Berechnung des Labels der Anfahrt verwendet. Alle Datenpunkte mit $t_i > t_{min}$ sind kein Teil der betrachteten Anfahrt und werden entfernt. Zur Modellierung der Ankunftszeit vom aktuellen Zeitpunkt t_i bis zum Zielhafen, erhält jeder Datenpunkt ein individuelles Label l_i mit

$$l_i = t_{min} - t_i$$

und t_i = Time wihtin Year des i-ten Datenpunkts. Das Label beziffert die Dauer bis zur Ankunft des Schiffs zum Zielhafen vom betrachteten Datenpunkt. Für den Wertebereich der Labels l_i einer Anfahrt gilt folglich: $l_i \in [0, t_{min} - t_1]$ Da gemäß Definition 16 alle Time within Year $t_i \in [0, 31622400]$, beschränkt sich die Obergrenze des Intervalls auf 31622400, wenn der erste Datenpunkt einer Anfahrt den Zeitpunkt $t_1 = 0$ (1. Januar) und der letzte Datenpunkt den Zeitpunkt $t_{min} = 31622400$ (31. Dezember) aufweist. Folglich liegen alle möglichen Prädiktionen des Modells im Intervall $[0, 31622400]$. Die Modellierung des Labels als Dauer bietet den Vorteil, dass der Wertebereich aller möglichen Schätzungen bekannt ist. Konkrete Zeitpunkte in der Zukunft zu schätzen hieße, im praktischen Einsatz für jeden Input einen Output zu schätzen, der außerhalb des bekannten Wertebereich der Trainingsdaten liegt.

Sobald ein Schiff einen Hafen verlässt, ändert sich der Eintrag des Zielhafens in den Datenpunkten. Folglich werden bei der Identifikation von Labels keine Datenpunkte gelöscht, die zu einer neuen Anfahrt gehören.

5.2.2 Struktur der ermittelten Anfahrten

Das Ergebnis des vorigen Kapitels 5.2.1 sind Zeitreihen aus Trainingsdatenpunkten (vgl. Def. 18). Tabelle 3 zeigt beispielhaft die Struktur einer Anfahrt, wobei zur vereinfachten Darstellung die Anzahl Features verringert wurde.

Aus der Zuordnung jeder Anfahrt zu einem Zielhafen und Schiff ergibt sich für jeden Zielhafen die folgende Struktur. Ein Schiff, das einen Hafen innerhalb des betrachteten Jahres 2020 mehrfach anfährt, wird in mehrfachen Anfahrten repräsentiert (vgl. Abb. 10).

	Time within Year (Sekunden)	Latitude (Grad)	Longitude (Grad)	...	Label (Sekunden)
0	1.000	55.4691	8.3701	...	10.000
1	2.000	55.4727	8.3865	...	9.000
...	...	...	...	...	...
8	9.000	55.4753	8.4019	...	1.000
9	10.000	55.4775	8.4110	...	0

Tabelle 3: Vereinfachte Struktur einer Anfahrt mit zugehörigem Label. Time within Year steigt mit zunehmendem Fortschritt der Anfahrt, die geografische Position nähert sich dem Zielort und die Zeit bis zur Ankunft am Zielhafen (Label) verringert sich, bis zum Eintritt in den Hafenbereich (vgl. Def. 19) des Zielhafens. Zur Vereinfachung zeigt diese Tabelle nicht-skalierte Ausprägungen. Die *tatsächlichen* Eingaben für das neuronale Netz sind auf $[0, 1]$ normiert.

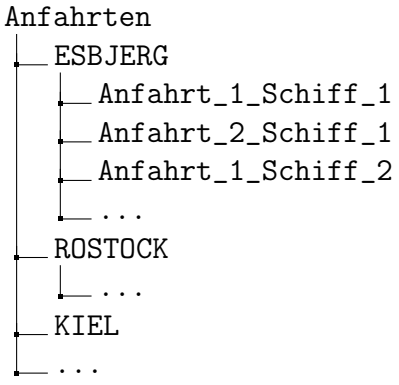

Abbildung 10: Resultat des Vorverarbeitungspipeline. Alle Anfahrten sind pro Hafen und Schiff zur einfachen Zuordnung strukturiert.

5.3 Identifikation von Trainingsbeispielen

Ein Deep Learning Algorithmus zum Training eines neuronalen Netzes optimieren iterativ die Gewichte eines maschinellen Lernmodells über sog. *Epochen*, bis keine Verbesserung mehr festzustellen ist (vgl. Kap. 3.2). Eine Epoche repräsentiert den Index einer Iteration. In jeder Epoche wird der gesamte Trainingsdatensatz verarbeitet (vgl. Kapitel 3.1.3). Um die Trainingszeit zu minimieren, ist effizientes Laden der Trainingsdaten unerlässlich. Im aktuellen Status der Pipeline sind für jeden Hafen die Anfahrten mit Trainingsdatenpunkten (vgl. Def. 18) in Form von Zeitreihen gruppiert und in einzelnen Dateien gespeichert. Im Folgenden wird erläutert, wie *Trainingsbeispiele* unter Anwendung der *Sliding Window Technik* und *Batch Loading* identifiziert werden.

5.3.1 Sliding Window als Eingabe des neuronalen Netzes

Die Aussagekraft eines einzelnen Trainingsdatenpunkts (vgl. Def. 18) zur Vorhersage der Ankunftszeit ist begrenzt aussagekräftig. Innerhalb eines einzigen Datenpunkts kann kein zeitlicher Verlauf der Features abgebildet werden. Wenn sich beispielsweise Teile unterschiedlicher Anfahrten überschneiden sind *wenige* Datenpunkte nicht geeignet, um eine individuelle Anfahrt auf Basis von Höhe-und Breitengrad zu unterscheiden. Außerdem reduziert die Betrachtung mehrerer Datenpunkte die Auswirkung fehlerhafter Ausprägungen, die zu unerwünschten Updates der Modell-Gewichte führen. Ein Ausreißer innerhalb eines Sliding Window wird durch die *umliegenden* Datenpunkte abgeschwächt. Ein weiterer Vorteil von Sliding Windows als Eingabe des neuronalen Netzes liefert die Natur einer Zeitreihe selbst. Der zeitliche Verlauft enthaltener Datenpunkte codiert Informationen, die nicht von einzelnen Datenpunkten erfasst werden können. Beispielsweise ist ein Anstieg der Geschwindigkeit eines Schiffs in keinem der einzelnen Features abgebildet. Das *ähnlichste* Feature, woraus eine Geschwindigkeitsänderung abgeleitet werden könnte, ist Speed over Ground (SOG, vgl. Def. 9). Mehrere Datenpunkte können abbilden, dass ein Schiff beschleunigt, wenn sich sein SOG im zeitlichen Verlauf erhöht. Analog können sich weitere Merkmale abbilden, die in einzelnen Datenpunkten nicht ausgeprägt sind. Abbildung 11 verdeutlicht die Extraktion eines Sliding Windows der Dimension $n \times w$, wobei n=Anzahl Features enthaltener Datenpunkte der Zeitreihe und w=Anzahl Datenpunkte, die das Sliding Window beinhaltet (im Folgenden als *Breite* w des Sliding Window bezeichnet).

Um mehrere Sliding Windows aus einer Zeitreihe der Länge t zu erhalten, wird der Bereich zur Extraktion entlang der zeitlichen Dimension über die gesamte Länge der Zeitreihe *geschoben*. Beginnend beim ersten Datenpunkt der Zeitreihe, extrahiert die *Sliding Window Technik* für jede mögliche Startposition ein Sliding Window der Breite w solange, bis das Ende des Sliding Window am Ende der Zeitreihe angekommen ist.

Definition 20: Sliding Window: *Die Sliding Window Technik ermöglicht die konsekutive Selektion von Trainingsbeispielen aus einer Anfahrt. Pro Anfahrt der Länge t, Breite w des Sliding Window und $t \geq w$, ergeben sich $t - w + 1$ Trainingsbeispiele (wird im Folgenden Definiert).*

Wie Abbildung 11 bereits andeutet, dient das Resultat der Anwendung eines Sliding Window unmittelbar als Eingabe *(Input)* eines neuronalen Netzes. Eingaben eines neuronalen Netzes (oder allgemein: eines maschinellen Lernalgorithmus) werden als *Trainingsbeispiel* bezeichnet. Ein Trainingsbeispiel im Sinne dieser Arbeit ist folglich ein Ausschnitt einer Anfahrt, wobei das Label des zeitlich letzten Datenpunkts die

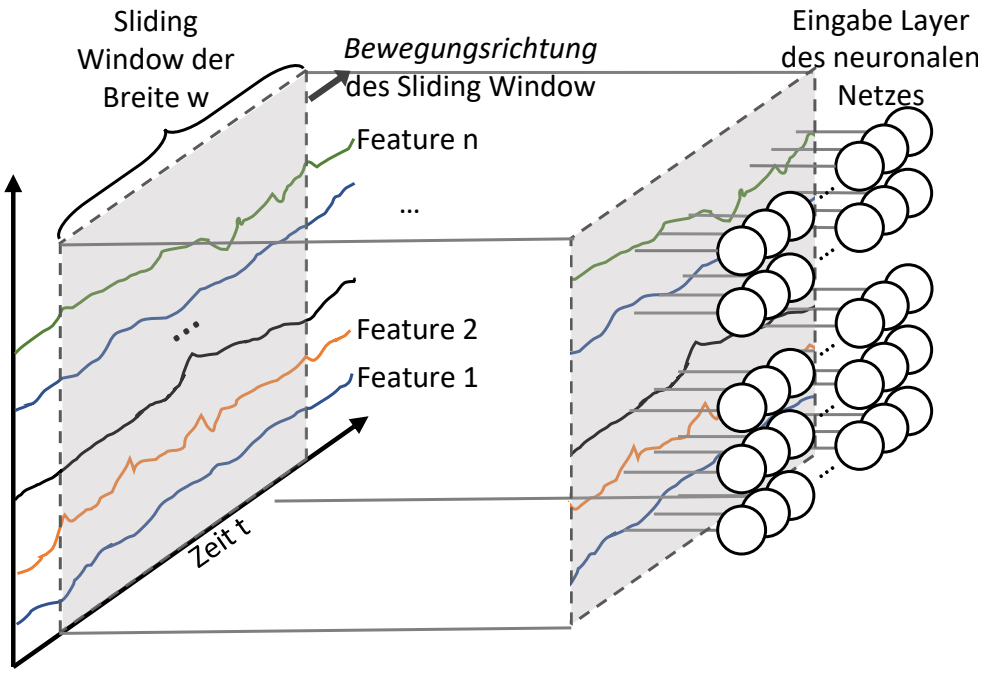

Abbildung 11: Extrahieren eines Sliding Window der Dimension $n \times w$ aus einer Zeitreihe. n=Anzahl Features und w=Anzahl Datenpunkte, die aus der Zeitreihe entnommen werden (Breite des Fensters). Weitere Sliding Windows (Bewegungsrichtung) werden entlang der zeitlichen Dimension der Zeitreihe extrahiert. Das Sliding Window dient als Eingabe des neuronalen Netzes.

Ankunftszeit des gesamten Ausschnitts repräsentiert.

Definition 21: Trainingsbeispiel: *Ein Trainingsbeispiel im Sinne dieser Arbeit besteht aus $n > 1$ Trainingsdatenpunkten (vgl. Def. 18). Das Label des letzten Trainingsdatenpunkts l_{n-1} repräsentiert das Label des Trainingsbeispiels.*

5.3.2 Effizientes Laden der Trainingsbeispiele

Der Trainingsprozess eines Modells für einen Hafen sieht vor, über mehrere Epochen Gewichte des neuronalen Netzes zu optimieren. Dazu wird in jeder Epoche über aber Trainingsbeispiele iteriert (vgl. Kapitel 3.1). Alle notwendigen Daten zu Beginn des Trainings in den Arbeitsspeicher zu laden ist die effizienteste Vorgehensweise zur Bereitstellung der Trainingsbeispiele. Zum Laden eines Datensets wird Tensorflows Bibliothek TimeseriesGenerator verwendet (Tensorflow, 2021c). Jede Anfahrt wird geladen und gemäß definierter Window Width w und Batch Size b = *Länge der Anfahrt* eine Zeitreihe generiert, bestehend aus Trainingsbeispielen gemäß Def. 21. Jede Anfahrt, die genug Trainingsdatenpunkte zur Extraktion von mindestens einem Sliding Window beinhaltet, wird zum Aufbau des Trainingsdatensatzes verwendet.

5.4 Architektur der neuronalen Netzes

Das Herzstück eines maschinellen Lernprojekts ist das neuronale Netz. Dessen Architektur setzt sich dem Anwendungsfall entsprechend aus individuell verbundenen Layern zusammen. Jedes neuronale Netz erwartet eine Eingabe bestimmter Dimensionalität und verarbeitet im Verlauf der Layer die Eingabe zur gewünschten Ausgabe. Das in dieser Arbeit angewandte neuronale Netz soll als Eingabe Ausschnitte von historischen Zeitreihen-Daten erwarten und eine Dauer bis zur Ankunftszeit des Schiffs schätzen. Folglich sind zeitlich codierte Informationen, die sich nicht auf Basis alleiniger Datenpunkte ausprägen von zentraler Bedeutung (vgl. Kap. 5.3.2). Das Netz muss dazu im Stande sein, über unterschiedlich (und unbekannt) lange Zeitperioden Merkmale zu extrahieren. Die theoretische Grundlage zur Erkennung solcher lokal auftretender Merkmale sind Kernels in Convolutions, die in verschiedenen Größen auf Teile einer Eingabe angewendet werden. Demnach sind mehrere verschieden große Kernels notwendig, um zeitlich unterschiedlich ausgeprägte Merkmale aus den Daten zu extrahieren. Gleichzeitig muss beachtet werden, dass sich Merkmale erst im Verlauf der Hidden Features eines neuronalen Netzes ausprägen.

Das implementierte neuronale Netz dieser Arbeit erwartet Trainingsbeispiele der Dimension $n \times w$, wobei $n = 36$ (Anzahl Features, vgl. Tab. A10) und $w = 50$ die Breite des Sliding Windows repräsentiert. Drei InceptionTime Blöcke (vgl. Kap. 3.2.7 u. Abb. 5) extrahieren Features, der letzte Layer führt alle Neuronen zu einem Output zusammen. Abbildung 12 zeigt die Architektur des eingesetzten neuronalen Netzes.

Im Folgenden werden die Hintergründe erläutert, die zur Wahl der Architektur gemäß Abbildung 12 führen. Die Anforderung ergeben sich aus der Problemstellung: Die Verarbeitung historischer Zeitreihen verlangt ein Modell, das Eingaben unabhängig von ihrer zeitlichen Dimension verarbeiten kann. Die eingesetzte Technik zur Extraktion eines Sliding Window resultiert im Parameter w (Breite des Sliding Window), der im Verlauf der Umsetzung dieser Arbeit exploriert und letztlich festgesetzt wurde. Folglich muss das neuronale Netz dynamisch auf Änderungen der zeitlichen Dimension von Eingaben reagieren. Das Konzept von Inception-Time ist die Handhabung zeitlich beliebig großer Eingaben mittels drei Convolutions mit unterschiedlich großen Kernels. Weiter soll das Modell möglichst robust gegenüber geringfügig falsch erkannten Merkmalen innerhalb der drei Convolutions sein. Paralleles Max-Pooling der ursprünglichen Eingaben eines Inception-Blocks erhält einen Teil des ursprünglichen Wissens. Der zweite und dritte Inception-Block beziehen folglich eine Mischung aus *abgeschwächten ursprünglichen* Merkmalen und extrahierten Features der vorigen Inception-Blöcke in die Feature Extraktion ein. Zuletzt biete ein Resudial Connector die Möglichkeit in tiefen Schichten des neuronalen Net-

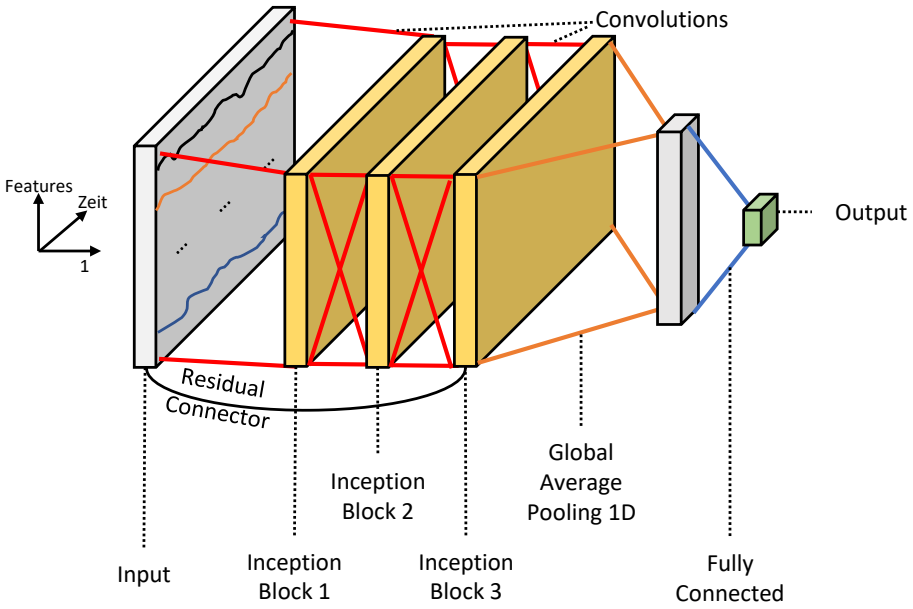

Abbildung 12: Architektur des eingesetzten neuronalen Netzes. Die Eingabe der Größe $n \times w$ mit n = Anzahl Features und w = Größe des Sliding Window (Zeit-Achse) wird in drei Inception-Blöcken (vgl. Abb. 5) verarbeitet. Der dritte Inception-Block bezieht die Eingabe mittels eines Residual Connectors in die Merkmalsextraktion ein. Anschließendes Global Average Pooling reduziert die zeitliche Dimension auf 1 und ein Fully Connected Netz führt die Ergebnisse zur gewünschten Ausgabe als Dauer bis zur Ankunfts eines Schiffs im Zielhafen zusammen.

zes Merkmale aus ursprünglichen Eingaben zu betrachten. Die Eingabe wird mit der Ausgabe des dritten Inception-Blocks verbunden, worauf ein Global Average Pooling zur Reduktion der zeitlichen Dimension auf 1 folgt. Das Resultat sind n identifizierte Features, die mittels eines Fully Connected Layer die gewünschte Ausgabe erzeugen: Eine Schätzung der Dauer bis zur Ankunfts des betrachteten Schiffs am Zielhafen.

5.5 Trainingsprozess

Nachdem in Kapitel 5.1 und 5.2 die Daten aufbereitet wurden, Kapitel 5.3 für die Bereitstellung von Trainungsbeispielen mittels der Sliding Window Technik sorgt und das vorige Kapitel 5.4 das umgesetzte neuronale Netz aufzeigt, behandelt dieses Kapitel die Vorgehensweise zum Training eines maschinellen Lernmodells. Dabei finden die Grundlagen aus Kapitel 3.1.3 Anwendung. Die relevanten Häfen wurden bereits in Tabelle 2 aufgrund ihrer Datenmengen identifiziert. Genau an dieser Stelle setzt der Prozess des Trainings der Modelle dieser Arbeit an. Die 12 identifizierten Häfen sind die Ansatzpunkte zum Initiieren des Trainings. Mit Hilfe des Port-Manager Moduls (eingeführt in Kapitel 5.2.1) kann die Pipeline alle existierenden Häfen aus

Tabelle 2 identifizieren und die zugehörigen Trainingsdaten laden.

Der Trainingsprozess umfasst das Training von maschinellen Lernmodellen für alle Häfen aus Tabelle 2. Der folgend beschriebene Prozess wird für jeden Hafen individuell durchgeführt.

Im ersten Schritt des Trainingsprozesses werden alle verfügbaren Trainingsdaten des *aktuellen* Hafens geladen. Gemäß Kapitel 3.1.4 werden die Trainingsdaten in zwei disjunkte Datensätze aufgeteilt: 80% der Trainingsdaten werden zur Anpassung der Gewichte des neuronalen Netzes verwendet (im Folgenden *Trainingsbeispiele* genannt). Die restlichen 20% der Trainingsdaten dienen zum Test der Gewichtsanpassungen (im Folgenden *Testbeispiele* genannt).

Nach dem Laden der Trainingsdaten muss das theoretisch definierte maschinelle Lernmodell initialisiert werden. Vor dessen Training werden die Parameter des Trainingsalgorithmus, Modellparameter und die aktuelle Zeit zur Identifikation des Trainings initialisiert. Die Parameter des Trainingsalgorithmus sind *Lernrate*, Anzahl *Epochen*, *Loss-Funktion* und *Optimierungsalgorithmus*. Das Modell erhält die Dimension der Trainingsbeispiele, um die Größe des Eingabe-Layer entsprechend zu initialisieren. Die folgende Tabelle 4 listet die Parameter und deren verwendete Ausprägung auf.

Parameter	Ausprägung
Parameter des Trainingsalgorithmus	
Lernrate	0,01
Epochen	50
Loss-Funktionen	*MAE & MSE*
Optimierungsalgorithmus	Adaptive Moment Estimation (Adam, vgl. Kingma u. Ba (2017))
Modellparameter	
Anzahl Features	37
Window Width	50

Tabelle 4: Parameter und deren initiale Belegung im Trainingsprozess eines Hafens

Die beiden Loss-Funktionen sind wie folgt definiert. *Mean Squared Error (MSE) (MathVault, 2021)*:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2$$

mit n = Anzahl Datenpunkte, X_i = Label des i-ten Trainingsdatums und Y_i = Schätzung des neuronalen Netzes für das i-te Trainingsdatum.

Mean Absolute Error (MAE) (Willmott u. Matsuura, 2005):

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |X_i - Y_i|$$

mit n = Anzahl Datenpunkte, X_i = Label des i-ten Trainingsdatums und Y_i = Schätzung des neuronalen Netzes für das i-te Trainingsdatum.

Der Trainingsalgorithmus versucht, jede Epoche die Abweichungen von der Schätzungen des *aktuell besten* Modells zum Label der Trainingsdaten zu verbessern. Der Abweichung der Schätzungen über alle Trainingsbeispiele wird mittels der Loss-Funktion ermittelt und als *Loss* bezeichnet. Im Folgenden verdeutlichen die Begriffe (neuronales) Netz und Modell den Fortschritt des Trainingsprozesses. Formal ist ein Modell ist eine Instanz eines neuronalen Netzes und das Resultat des Trainingsprozesses. Übertragen auf das DSR Paradigma, repräsentieren die generierten Modelle im Trainingsprozess Artefakte im Design Prozess. Im Folgenden wird der Modell-Begriff zur Benennung auftretender Artefakte verwendet. Der Trainingsalgorithmus teilt sich in die folgenden zwei Schritte auf.

Der erste Schritt optimiert die Gewichte des neuronalen Netzes auf Basis der Trainingsbeispiele. Er iteriert über alle Trainingsbeispiele und versucht, die Gewichte der einzelnen Layer zu verbessern. Zur Bestimmung neuer Gewichte wird *Gradientenabstieg* (vgl. Kap. 3.2.4) und der *Backpropagration* Algorithmus (vgl. Kap. 3.2.5) angewendet. Der Loss des MSE bewertet in diesem Schritt den Fehler des Trainingsalgorithmus beim Update der Gewichte. Je höher die Schätzung des Netzes vom Label des Trainingsbeispiels abweicht, desto hoher die *Bestrafung*. Ein Loss auf Basis des MSE bestraft folglich hohe Abweichungen mehr als geringfügige Fehler.

Im zweiten Schritt iteriert der Algorithmus über alle Testbeispiele, um die Güte der angepassten Gewichte auf bislang unbekannten Beispielen zu testen. Da sich die Testbeispiele von den Trainingsbeispielen unterscheiden, kann mit dieser Vorgehensweise festgestellt werden, ob das Modell des ersten Schritts *generalisiert* - sprich auf nicht-Trainingsbeispielen gute Schätzungen abgibt. Damit die Testbeispiele auch in weiteren Epochen zum Test weitere Modelle eingesetzt werden können, nimmt dieser Schritt keine Gewichtsanpassungen vor. Der zweite Schritt dient dem Trainingsalgorithmus zur Identifikation des aktuell besten Modells über alle bislang betrachteten Epochen. Das Modell mit minimalem *Validation-Loss* (Loss der Testbeispiele), ist das aktuell optimale Modell. Außerdem ist dieser Schritt analog zum produktiven Einsatz des Netzes: Es wird überprüft, welche Schätzungen das Netz auf Basis von Daten abgibt, die nicht zum Training der Gewichte eingesetzt wurden. Als Maß zur Bestimmung der Güte abgegebener Schätzungen kommt MAE zum Einsatz. In diesem Schritt ist keine Bestrafung für fehlerhafte Updates der Gewichte des Lernalgorithmus notwendig, weshalb der Austausch der Loss-Funktion im zweiten Schritt

keine Auswirkung auf die letztliche Performance des Modells hat. Die einfache Interpretierbarkeit des MAE-Loss begründet seinen Einsatz: Durchschnittliche absolute Abweichungen der Schätzungen geben direkten Aufschluss über die Ungenauigkeit der voraussichtlichen Ankunftszeit.

Das Resultat des beschriebenen Trainingsprozess ist ein Instanz (Artefakt) des neuronalen Netzes aus der aktuellen Epoche. Modelle aus dem soeben beschriebenen Trainingsprozess werden im Folgenden als *Basismodell* bezeichnet.

Definition 22: Basismodell: *Ein Basismodell M_H^B ist ein maschinelles Lernmodell eines Hafens H, das ausschließlich und auf den Trainingsbeispielen aller verfügbaren Anfahrten des zugehörigen Hafens trainiert wurde.*

Im Trainingsprozess werden über maximal 50 Epochen die Gewichtsbelegungen iterativ verbessert und das Modell mit geringstem Validation-Loss als Endergebnis identifiziert. Sollte sich bereits in den ersten Epochen keine Verbesserung der Gewichtsanpassungen ergeben, wendet der Trainingsprozess eine Methode zum Reduktion der Lernrate an (Tensorflow, 2021b). Eine geringere Lernrate sorgt für feinere Anpassungen der Gewichte und kann kleine Verbesserungen der Gewichte im Trainingsprozess erzielen. Sollte sich nach nach Verringerung der Lernrate keine bessere Belegung der Gewichte ergeben, wird das Training vor Abschluss der 50 Epochen beendet (Tensorflow, 2021a).

Zur Laufzeit des Trainingsprozesses sind für jedes zwischenzeitliche Modell MSE- und MAE-Loss und andere Training-spezifische Daten ermittelt worden. Deren Aufzeichnung ermöglicht nachgelagerte Analysen, Evaluation und Vergleiche unterschiedlicher Modelle. Zur Organisation und Handhabung sämtlicher zum Training gehöriger Prozess- und Modell-Daten implementiert diese Arbeit ein **Training-Manager** Modul (vgl. Abb. 8). Der Training-Manager im *Hafen-Manager* (vgl. Kap. 5.2.1) integriert. Jeder Hafen erhält die Referenz auf ein individuelles Basismodell und seine zugehörigen Prozess-Daten wie den Verlauf des Loss jeder Epoche *(Loss-Historie)*. Außerdem kann mit Hilfe des Training-Manager der Trainingsprozess unterbrochen oder wieder aufgenommen werden.

Nach Abschluss des Trainingsprozesses für jeden Hafen aus Tabelle 2 wurde für jeden Hafen ein individuelles Basismodell trainiert. Die besten Modelle weisen, wie in den vorigen Absätzen beschrieben, den minimalen Validation-Loss ihres Trainingsprozesses auf. Zwar repräsentiert das Minimum der jeweiligen MAE-Loss-Historie die Information des besten Modells, zur einfachen Handhabung und unter Voraussicht der anschließenden modellübergreifenden Evaluation dient das eigenständige **Evaluation-Manager** Modul. Es dient dem einfachen Zugriff auf trainierte Modelle, zugehöriger Prozessdaten des Trainings und bietet Funktionalitäten zur Aus-

wertung von Modellen. Die Auswertung von Modellen und Aggregation mehrerer Ergebnisse wird in Kapitel 6 behandelt.

5.6 Transfer von Modellen auf andere Häfen

Eine praktische Problemstellung dieser Arbeit beschreibt die geringe Anzahl Trainingsbeispiele bei selten angefahrenen Häfen. Für solche Häfen kann kein eigenes generalisierendes Basismodell (vgl. Def. 22) zur Schätzung der Ankunftszeit anfahrender Schiffe trainiert werden. Um dennoch einen ETA Sensor auf Basis geringer Datenmengen eines Hafens zu generieren, beschreibt das vorliegende Kapitel die Vorgehensweise zum Transfer von Basismodellen (vgl. Kap. 5.5) auf andere Häfen. Kapitel 5.6.1 führt das organisatorische Level zur Handhabung von Transfers ein, 5.6.2 beschreibt den Prozess zur Durchführung eines Transfers und 5.6.3 führt Besonderheiten an, die beim Transfer maschineller Lernmodelle beachtet werden müssen.

5.6.1 Organisation und Definition verschiedener *Strategien* zum Modell-Transfer

Das vorherige Kapitel 5.5 beschreibt den angewandten Trainingsprozess und liefert für jeden Hafen aus Tabelle 2 einen Basismodell. Um explorieren zu können, wie sich der Transfer eines Basismodells auf einen anderen Hafen auswirkt, soll jedes Basismodell auf jeden *anderen* Hafen transferiert werden. Folglich ergeben sich durch die Permutation aller Häfen aus Tabelle 2 132 Hafen-Paare der Form (H_B, H_T), wobei H_B den Hafen des Basismodells repräsentiert und H_T den Hafen, auf den das Basismodell transferiert werden soll. Zur Organisation der Transfers implementiert diese Arbeit das letzte bislang undefinierte Modul der Pipeline in Abbildung 8. Das **Transfer-Manager** Modul setzt die folgenden Funktionalitäten um:

- Permutieren der Häfen gemäß Tabelle 2
- Generieren und Bereitstellen von *Transferstrategien*
- Durchführen des Prozesses zum Transfer von Basismodellen auf andere Häfen
- Speichern und Laden von Transferdurchläufen

Die Umsetzung eines Transfers setzt einerseits ein Basismodell eines *anderen* Hafens voraus, andererseits muss der Transferprozess wissen, *wie* die Übertragung von Wissen eines maschinellen Lernmodells durchgeführt werden soll. Die verwandten

Arbeiten aus Kapitel 2 und die Grundlagen aus Kapitel 3.3 zeigen auf, dass lediglich ausgewählte Gewichte *bestimmter* Layer eines bereits trainierten Modells auf das Wissen neuer und *unbekannter* Trainingsbeispiele des *unbekannten* Hafens angepasst werden sollten. Im Folgenden wird zur Bezeichnung der Gewichte eines Layers in einem neuronalen Netz lediglich der Begriff *Layer* verwendet, weil ohnehin alle Gewichte eines Layers angepasst werden. Die Bestimmung anzupassender Layer folgt der Intuition der genannten ersten Kapitel, wo vorwiegend die Klassifikation von Bildern als Beispiel behandelt wird. So werden lediglich die letzten Layer bereits trainierter Modelle auf einen anderen, aber ähnlichen Kontext angepasst. *Transferstrategien* dienen zur Definition und Handhabung unterschiedlicher Transfers und der nachträglichen Zuordnung.

Definition 23: Transferstrategie: *Eine Transferstrategie S gibt an, wie ein Basismodell (vgl. Def. 22) auf einen anderen Hafen transferiert werden soll. Zur Konfiguration der Strategie finden die Felder* ***nth_subset****,* ***train_layers*** *und* ***tune*** *Anwendung.*

Die drei genannten Optionen zur Konfiguration einer Transferstrategie sind wie folgt definiert:

Musst hier noch hin, dass ein Transfer Daten braucht? Ich glaube nicht

nth_subset gibt an, jedes wievieltes Trainingsbeispiel eines Hafens zur Umsetzung eines Transfers verwendet werden soll. Es gilt $nth\_subset \in [1, n]$ mit n = Anzahl Trainingsbeispiele. Eine Transferstrategie mit $nth\_subset = 1$ verwendet jedes verfügbare Trainingsbeispiel des Datensatzes, eine Strategie mit $nth\_subset = 10$ ($= 10\%$) verwendet zum Transfer jedes zehnte Trainingsbeispiel.

train_layers spezifiziert die Layer des Basismodells, deren Gewichte beim Transfer angepasst werden sollen. Die Gewichte aller übrigen Layer bleiben beim Transfer des Basismodell unangetastet.

tune bestimmt, ob nach Anpassung der Gewichte in den spezifizierten $train\_layers$ und Identifikation des besten Modells, die Gewichte aller Layer des Modells fein abgestimmt werden sollen. Dieser Vorgang wird *Fine Tuning* genannt.

Definition 24: Fine-Tuning: *Fine-Tuning versucht, die Gewichte aller Layer eines transferierten maschinellen Lernmodells auf die Domäne (Datenbasis des Transferhafens) fein abzustimmen. Fine-Tuning folgt dem Prozess zum üblichen Training der Basismodelle (vgl. Kap. 5.5) mit den Trainingsbeispielen des Transferhafens und sehr geringer Lernrate.*

Fine-Tuning passt folglich insbesondere die Gewichte derjenigen Layer an, die im vorherigen Transfer-Schritt unangetastet geblieben sind. Die geringe Lernrate sorgt dafür, bereits gelerntes Wissen des Basismodells nicht zu „vergessen“.

Um unterschiedliche Strategien zum Transfer eines Basismodells auf einen anderen Hafen handhabbar und rückwirkend nachvollziehbar umsetzen zu können, vereinigen *Transferdefinitionen* das eingeführte Konzept der Transferstrategie und die ermittelten Hafen-Paare.

Definition 25: Transferdefinition: *Eine Transferdefinition $TD(S_i)$ gibt an, wie das Basismodell M_A^B eines Hafens A auf einen anderen Hafen B unter Einsatz einer Transferstrategie S_i transferiert werden soll:*

$$TD(S_i) : M_A^B \xrightarrow{S_i} M_{A \to B}^T$$

Auf Basis des Wissens verwandter Arbeiten werden beim Transfer von maschinellen Lernmodellen die letzten Layer zum Transfer auf eine andere Domäne angepasst. Entsprechend exploriert diese Arbeit, wie sich transferierte Modelle verhalten, wenn verschiedene Zusammensetzungen der *letzten* Layer auf einen *anderen* Hafen angepasst werden. Tabelle 5 zeigt die angewandten Transferstrategien.

	nth_subset	tune	train_layers
0	1	False	dense
1	1	True	dense
2	1	True	dense
3	1	True	conv1d_15, dense
4	1	True	conv1d_14, conv1d_15, dense
5	1	True	conv1d_13, conv1d_14, conv1d_15, dense
6	1	True	conv1d_12, conv1d_13, conv1d_14, conv1d_15, dense
7	10	True	conv1d_13, conv1d_14, conv1d_15, dense
8	20	True	conv1d_13, conv1d_14, conv1d_15, dense

Tabelle 5: Definition der Transferstrategien. Spalte *nth_subset* sub-sampled die verwendeten Trainingsbeispiele. Die Flagge *tune* legt fest, ob Fine-Tuning durchgeführt werden soll und *train_layers* beziffert die Layer des neuronalen Netzes, deren Gewichte beim Transfer angepasst werden.

Zur Bezeichnung, ob die Gewichte in Layern eines Modells beim Trainingsprozess angepasst werden, finden in der Literatur die Begriffe *freeze* und *unfreeze* Anwendung. Sind Layer (oder das gesamte Modell) gefreezt, wendet der Trainingsalgorithmus keine Backpropagation (und kein Gradientenabstieg) zur Anpassung der Gewichte an. Entsprechend schaltet der Vorgang des *unfreezing* von Layern (oder des Modells) die Gewichte zur Anpassung im Trainingsprozess frei.

Zum Initialisieren des Transferprozesses generiert der Transfer-Manager eine Transferdefinition gemäß Definition 25) für jede Kombination aus den 132 identifizierten Hafen-Paaren und den 9 Transferstrategien. Folglich ergeben sich 1188 Transfers von Basismodellen auf andere Zielhäfen, die im Anschluss angewendet werden.

5.6.2 Angewandter Prozess zum Transfer von Basismodellen auf andere Häfen

Der Prozess des Transferieren eines Basismodells auf einen anderen Hafen und der Prozess zum Training von Basismodellen aus Kapitel 5.5 sind in großen Teilen ähnlich. Dieses Kapitel wendet die vorgestellten Konzepte aus Kapitel 5.6.1 an. Es wird angenommen, dass die Vorgehensweise zum Training eines Basismodells bekannt ist. Das Ergebnis des Transfers eines Basismodells auf einen anderen Hafen ist ein *neues* Modell des *anderen* Hafens, was im Folgenden als *Transfermodell* bezeichnet wird.

Definition 26: Transfermodell: *Ein Transfermodell $M^T_{A\rightarrow B}$ ist das maschinelle Lernmodell des Hafens B, das vom Basismodell M^B_A (vgl. Def. 22) des Hafens A auf den (anderen) Hafen B transferiert wurde.*

Die ersten beiden Funktionen des Transfer-Managers haben im vorherigen Kapitel bereits die theoretischen Voraussetzungen zum Transfer eines Basismodells auf einen anderen Hafen geschaffen. Er liefert 1188 Transferdefinitionen, die jeweils in einem *einzelnen* Transferprozess umgesetzt werden. Äquivalent zum Trainingsprozess der Basismodelle müssen zunächst die *entsprechenden* Trainingsbeispiele geladen werden. Zum Transfer werden die Trainingsbeispiele desjenigen Hafens benötigt, auf den ein Basismodell transferiert werden soll. Algorithmus 1 zeigt die naive Vorgehensweise zur Umsetzung jeder Transferdefinition.

Algorithmus 1 Naiver Prozess zum Transfer eines Basismodells auf einen *anderen* Hafen

1: **for each** Transferdefinition $TD(S_i)$ **do**
2: $daten_B$ = ladeTrainingsbeispiele(Hafen B)
3: transferiereBasismodell($M_A, daten_B, TD(S_i)$)
4: **return** Transfermodell $M^T_{A\rightarrow B}$
5: **end for**

Die naive Umsetzung des Transferprozesses gemäß Algorithmus 1 lädt für jede der 1188 Transferdefinitionen die Trainingsbeispiele des Hafens B. Der Transfer-Manager reduziert die Anzahl Ladevorgänge im Transferprozess auf die Anzahl betrachteter Häfen. Die Transferdefinitionen werden nach ihrem Transferhafen B gruppiert, sodass die Trainingsbeispiele jedes Transferhafens B ein mal geladen werden

müssen und alle entsprechenden Transferdefinitionen durchgeführt werden können. Folglich ergibt sich der Transferprozess gemäß Algorithmus 2.

Algorithmus 2 Optimierter Prozess zum Transfer eines Basismodells auf einen *anderen* Hafen

```
1: for each Transferhafen B do
2:     daten_B = ladeTrainingsbeispiele(B)
3:     for each Transferdefinition TD(S_I) mit Transferhafen = B do
4:         for each Basismodell M_A do
5:             for each Transferstrategie S_i do
6:                 transferiereBasismodell(M_A, daten_B, TD(S_i))
7:                 return Transfermodell M^T_{A→B}
8:             end for
9:         end for
10:    end for
11: end for
```

Die im Pseudocode enthaltene Funktion *transferiereBasismodell*$(M_i^B, trainBsp, S_i)$ führt den Transfer eines Basismodells M_i^B anhand einer Transferstrategie S_i auf einen anderen Hafen durch. Im Folgenden wird beispielhaft die Anwendung der Transferdefinition

$$TD(S_7) : M_A^B \xrightarrow{S_7} M_{A\to B}^T$$

durchgeführt. Welche Häfen A und B konkret repräsentieren, ist zur Erläuterung nicht relevant. Aus Tabelle 5 folgt, dass für $i = 7$ jedes zehnte Trainingsbeispiel verwendet wird ($nth\_subset = 10$), die Gewichte der letzten vier Layer angepasst ($train\_layers = ...$) und das resultierende Modell Fine-Tuning unterzogen wird ($tune = True$). Die folgenden Erläuterungen stützen sich Transferstrategie Nr. 7, die in Kapitel 6 als beste der 9 vorgestellten Strategien aus Tabelle 5 identifiziert wird.

Der Transferprozess ist analog zum in Kapitel 5.5 beschriebenen Trainingsprozess aufgebaut. Zuerst werden die zum Transfer notwendigen Daten von Hafen B (vgl. Alg. 2 Zeile 2) und das Basismodell von Hafen A geladen. Im Anschluss wird die Transferdefinition $TD(S_7)$ angewendet.

Der Parameter $nth\_subset = 10$ sub-sampelt die Trainingsdaten des Hafens B, sodass zum Transfer lediglich das erste, zehnte, zwanzigste, etc. Trainingsbeispiel zur Verfügung steht. Danach freezt der Transferprozess alle Layer des geladenen Basismodells, bis auf die in $train\_layers$ spezifizierten Layer. Im Fall von S_7 werden die letzten vier Layer des Modells nicht gefreezt. Folglich können ausschließlich an diesen letzten vier Layern Anpassungen der Gewichte vorgenommen werden.

Mit den augedünnten Trainingsdaten trainiert der Transferprozess analog zum Trainingsprozess das teilweise gefreezten Modell von Hafen A. Die Trainingsdaten werden zu 80% in Trainingsbeispiele und 20% Testbeispiele aufgeteilt. Tabelle 6 listet die Parameter und deren verwendete Ausprägung auf.

Parameter	Ausprägung
Parameter des Transferprozesses	
Lernrate	0,01
Epochen	25
Loss-Funktionen	MAE & MSE (vgl. Kap. 5.5)
Optimierungsalgorithmus	Adaptive Moment Estimation (Adam, vgl. Kingma u. Ba (2017))
Modellparameter (unverändert)	
Anzahl Features	37
Window Width	50
Parameter zum Fine-Tuning	
Lernrate	0,00005
Epochen	20

Tabelle 6: Parameter und deren initiale Belegung im Transferprozess eines Basismodells auf einen anderen Hafen.

Die Parameter des Transferprozesses unterscheiden sich teilweise von den verwendeten Parametern im Trainingsprozess. Die Anzahl Epochen ist geringer, weil die Gewichte des Basismodells bereits initialisiert sind und folglich Wissen nicht vollständig neu gelernt werden muss. Die Modellparameter bleiben unverändert, da der Transferprozess kein neues Modell initialisiert und folglich auf die gleiche Dimension der Eingaben wie im *ursprünglichen* Trainingsprozess angewiesen ist. Neue Parameter werden beim optionalen Fine-Tuning des Transferprozesses benötigt. Wie Definition 24 anführt, wird beim Fine-Tuning in wenigen Epochen mit geringer Lernrate das ganze Modell fein auf die neue Datenbasis von Hafen B abgestimmt.

Nachdem die Trainings- und Testbeispiele sowie das Basismodell geladen sind und die Transferdefinition angewendet wurde, veranschaulicht Algorithmus 3 die Vorgehensweise zum Transfer des Basismodells M_A^B von Hafens A auf den anderen Hafen B.

Algorithmus 3 Abstrakte Vorgehensweise zum Transfer eines Basismodells M_A^B des Hafens A zu einem Transfermodell $M_{A\to B}^T$ unter Anwendung der Transferdefinition $TD(S_7)$

1: $daten_B$ = ladeTrainingsdaten(B, $nth\_subset = 10$)
2: $M_{A\to B}^T$ = trainiereModell(M_A^B, $daten_B$)
3: $M_{A\to B}^T$ = feinTune($M_{A\to B}^T$)

5.6.3 Sonderfall Batch-Normalization Layer

Die Architektur des Modells (vgl. Kap. 5.4) beinhaltet Layer des Typs *Batch-Normalization* (Bjorck u. a., 2018). Ein Batch-Normalization Layer enthält zwei trainierbare Gewichte. Beim Trainingsprozess der Basismodelle werden die beiden Gewichte, die Durchschnitt und Varianz des Inputs abbilden, optimiert. Freezen eines Batch-Normalization Layers fixiert dessen Gewichte zur Abbildung von Durchschnitt und Varianz. Dieses Verhalten ist gegensätzlich zu allen anderen Layern, weshalb alle Batch-Normalization Layer beim Transfer und Fine-Tuning gefreezt bleiben. Würde ein solcher Layer seine gelernten Gewichte anpassen, verliert das Modell die gelernte Repräsentation aller vorherigen Stadien. (Chollet, 2020)

6 Evaluation und Diskussion

Vor dem Hintergrund der Schätzung von Ankunftszeiten in der Schifffahrt mittels maschinellen Lernmodellen wurden in Kapitel 5.1.4 zunächst die 12 Häfen mit den größten Datensätzen identifiziert. Kapitel 5.5 trainiert für jeden Hafen zunächst ein Basismodell (vgl. Def. 22). Anschließend erzeugt Kapitel 5.6 durch paarweise Permutation der 12 relevanten Häfen und Anwenden von 9 unterschiedlichen Transferstrategien (vgl. Def. 23) auf jedes Hafen-Paar insgesamt 1188 Transfermodelle (vgl. Def. 26). Folglich ist im Rahmen dieser Arbeit eine Methode zur Handhabung von 1200 Evaluationsergebnissen notwendig. Das bereits in Kapitel 5.5 erwähnte Evaluation-Manager Modul ist die zentrale Anlaufstelle für alle Arten von Evaluation im Rahmen dieser Arbeit. Es biete die folgenden Funktionalitäten:

- Halten aller MAE-Losse der generierten Basis- und Transfermodelle
- Aggregieren der gespeicherten Losse
- Plotten von (Evalutations-) Grafiken

Die Evaluation auftretender Artefakte in Form von Basis- und Transfermodellen ist in den Trainings- und Transferprozess integriert. Die in den Kapiteln 5.5 und 5.6.2 beschriebenen Prozesse verbessern die Gewichtsbelegungen neuronaler Netze über mehrere Epochen hinweg. Ein *neues* besseres Modell im Trainings- oder Transferprozess des *aktuellen* Hafens A weist den bislang minimalen MAE aller Modelle des Hafens A auf Basis der Testbeispiele auf. Das Evaluation-Manager Modul bietet eine Schnittstelle zum setzen der MAEs auftretender Modelle. Basismodelle werden eindeutig aufgrund des zugehörigen Hafens (A) und der Startzeit des Trainingsprozesses referenziert (Notation M_A^B). Zur Referenzierung der Transfermodelle ist sowohl die Angabe des Basishafens als auch des Transferhafens notwendig (Notation $M_{A\to B}^T$).

6.1 Performance der Basismodelle

Mit Blick auf die Problemstellung unpräziser Ankunftszeiten in der Schifffahrt sind zunächst Basismodelle der 12 identifizierten Häfen aus Tabelle 2 trainiert und evaluiert worden. Sie bilden nicht nur die Modell-Grundlage zum anschließenden Transfer auf andere Häfen, sondern dienen gleichzeitig als Referenz. Abbildung 13 zeigt für jeden betrachteten Hafen den ermittelten MAE nach ausführen der Basistrainings.

Wie Abbildung 13 zeigt, reichen die MAEs von 3 Minuten für den Hafen von Trelleborg bis hin zu 87 Minuten für den Kieler Hafen. Die Mehrzahl der Modelle weißt eine Abweichung unter 24 Minuten auf. Die Interpretation der Güte eines Modells

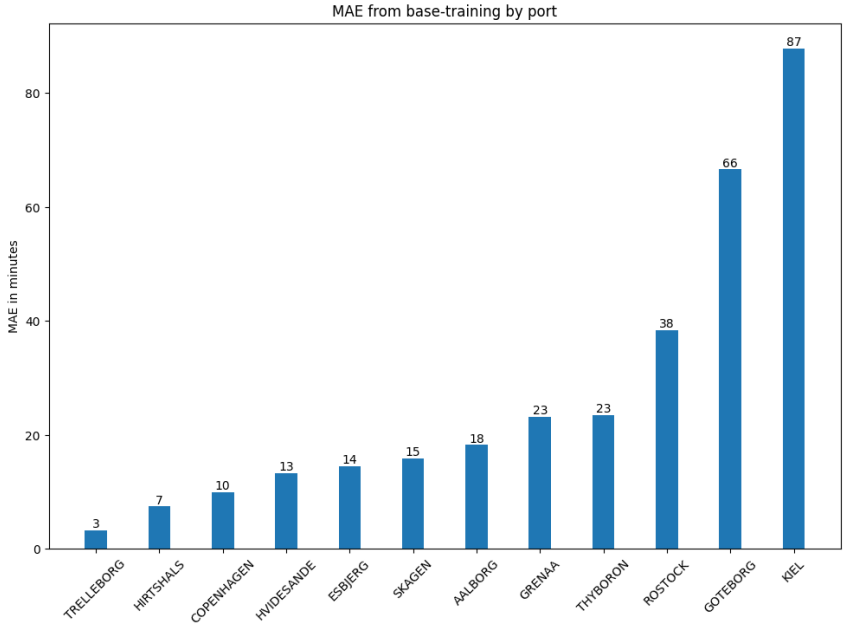

Abbildung 13: MAE bei der Schätzung von Ankunftszeiten der 12 betrachteten Basismodelle (vgl. Tab. 2) in zunehmender Reihenfolge. Die dargestellten MAEs stammen aus der Evaluation der Testbeispiele und wurden folglich nicht zur Anpassung der Gewichte eines Basismodells verwendet.

sollte jedoch nicht allein auf der Auswertung des MAEs basieren. Tabelle 7 zeigt eine Erweiterung der Tabelle 2 hinsichtlich *Diversität* von Anfahrten.

Definition 27: Diversität: *Die Diversität ist ein relatives Maß zur Bestimmung der Verschiedenheit von Anfahrten eines Hafens auf Basis des MMSI. Mit n = Anzahl Anfahrten eines Hafens und x_i = Anzahl Anfahrten des $i-$ten Schiffs (MMSI) berechnet sich die Diversität Div wie folgt:*

$$Div = 1 - \frac{-n + \sum_{i=1}^{n} x_i^2}{n^2}$$

Im Wertebereich von $[0.1, 1]$ repräsentiert das Maß der Diversität die Heterogenität der Anfahrten eines Hafens. Die Anfahrten eines Hafens weisen eine maximale Diversität von 1 auf, wenn jede Anfahrt einem anderen Schiff zugeordnet ist. Je niedriger die Diversität, desto mehr Anfahrten stammen von gleichen Schiffen. Tabelle 7 listet für jeden betrachteten Hafen die Diversität auf.

Unter Zuhilfenahme von Tabelle 7 ist festzustellen, dass die Mehrzahl an Häfen eine hohe Diversität von mehr als $0,95$ aufweisen. Lediglich Trelleborg liegt mit $Div = 0.627$ weit hinter allen anderen Häfen zurück. Dieser Fakt lässt sich mit Hilfe der geringen Anzahl unterschiedlicher Schiffe und der hohen Anzahl des maximal frequentierenden Schiffs ($= 326$) erklären. Im betrachteten Jahr 2020 sind 41 un-

Zielhafen	Anzahl Anfahrten	Diversität	# untersch. $MMSI$	$\#MMSI_{max}$
Esbjerg	2337	**0.985**	415	172
Rostock	2192	0.982	**570**	164
Kiel	1498	0.977	303	141
Skagen	1032	0.972	248	76
Thyboron	784	0.954	157	136
Trelleborg	751	**0.627**	**41**	**326**
Hirtshals	645	0.926	113	119
Hvidesande	662	0.959	107	66
Aalborg	629	0.931	134	99
Copenhagen	621	0.966	193	85
Goteborg	620	0.938	84	101
Grenaa	509	0.982	179	28

Tabelle 7: Ausgewählte Zielhäfen des dänischen AIS Datensatzes angereichert mit Diversitäten der Anfahrten. Die Spalte # untersch. $MMSI$ repräsentiert die Anzahl unterschiedlicher Schiffe, die den entsprechenden Hafen im Jahr 2020 angefahren haben. $\#MMSI_{max}$ beziffert die maximale Anzahl Anfahrten eines Schiffs im Jahr 2020 zum entsprechenden Hafen.

terschiedliche Schiffe identifiziert worden, die den Hafen von Trelleborg angefahren haben. Dabei hat das Schiff mit der höchsten Anzahl Anfahrten fast täglich (326 mal) am Hafen angelegt.

Die Evaluation der Basismodelle beim alleinigen Betrachten des MAE kann folglich zu Trugschlüssen führen. Nach Abbildung 13 weist das Modell des Hafens von Trelleborg den geringsten MAE von 3 Minuten auf, die zugrunde liegenden Daten weisen allerdings die geringste Diversität auf.

6.2 Durchschnittliche *(gruppierte)* Abweichungen der Ankunftszeitschätzungen

Die Intuition bezüglich der Genauigkeit von Schätzungen der Ankunftszeit eines Schiffs vermittelt: Je weiter ein Schiff von seinem Zielhafen entfernt ist, desto *ungewisser* ist die Ankunftszeit. Folglich müsste die Schätzung der Ankunftszeit umso *ungenauer* werden, je weiter ein Schiff vom Zielhafen entfernt ist. Zur Untersuchung dieses Phänomens kann auf Basis des Labels eines Trainingsbeispiels die tatsächliche Dauer bis zur Ankunft des Schiffs in beliebige Gruppen aufgeteilt werden. Für jede Gruppe kann im Anschluss ein individueller MAE berechnet werden, was Aufschluss über die Genauigkeit von Schätzungen eines Modells bei unterschiedlichen

tatsächlichen Ankunftszeiten gibt. Abbildung 14 zeigt den gruppierten MAE des Basismodells vom Hafen Esbjerg.

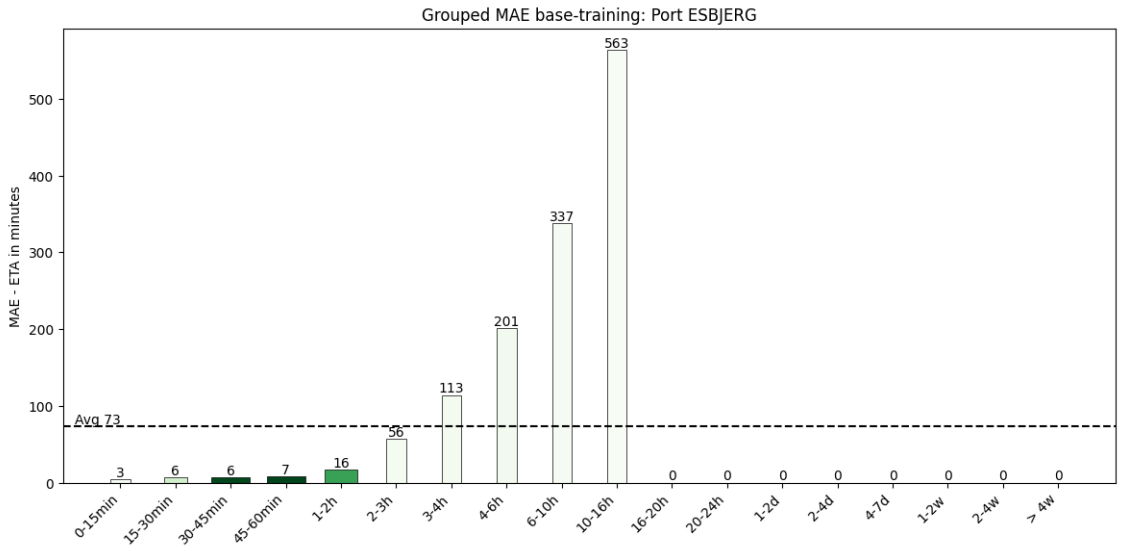

Abbildung 14: Gruppierter MAE des Basismodells vom Hafen Esbjerg. Die Höhe der Balken repräsentiert die mittlere absolute Abweichungen von Schätzungen der Ankunftszeit für die entsprechende Gruppe. Die Zuordnung in eine zutreffende Gruppe (x-Achse) richtet sich nach dem Label des Testbeispiels. Je näher die tatsächliche Ankunftszeit eines Schiffs, desto fein-granulärer die Gruppierungen. Die Intensität der Farbe einzelner Balken repräsentiert die Anzahl betrachteter Testbeispiele. So befindet sich die überwältigende Mehrheit in den Gruppen $30-45$ Minuten, $45-60$ Minuten und $1-2$ Stunden bis zur Ankunft im Hafen von Esbjerg. Die horizontale Linie repräsentiert den durchschnittlichen MAE (73 Minuten) über alle Gruppen.

Im Fall des Hafens von Esbjerg ist festzustellen, dass sich Ankunftszeiten mit zunehmender (zeitlicher) Nähe zum Zielhafen präziser schätzen lassen. Außerdem sind die Schätzungen für häufig vertretene Labels genauer, als die Schätzungen auf selten repräsentierten Testbeispielen. Die gruppierten Plots der andere 11 Häfen befinden sich im Anhang (vgl. Abb. 17 - 27). Der Trend des zunehmenden MAEs bei höheren Entfernungen vom Zielhafen und geringerer Datenmenge ist bei allen betrachteten Häfen festzustellen.

6.3 Auswirkungen von Transfers auf die Modell-Performance

Nachdem die vorherigen Kapitel 6.1 & 6.2 die Qualität von Ankunftszeitschätzungen der Basismodelle bestimmt und die angewandten Metriken zur Evaluation vorgestellt haben, setzt dieses Kapitel die Schätzungen der Basismodelle ins Verhältnis

zu den Schätzungen der Transfermodelle. Aufgrund der hohen Zahl von Transfermodellen (1188, vgl. Kap. 5.6.2), liefert dieses Kapitel eine Zusammenfassung anhand ausgewählter Transfers.

Der Transferprozess setzt für jedes mögliche Hafen-Paar alle definierten Transferstrategien (vgl. Tab. 23) um. Die Strategien Nr. 0 sowie Nr. 2 bis 4 finden in der Evaluation keine Anwendung, weil zur Abbildung des Verhaltens unterschiedlicher Transfers nicht notwendig sind. Abbildung 15 zeigt den durchschnittlichen MAE aller Transfermodelle, gruppiert nach dem Hafen, auf den transferiert wurde.

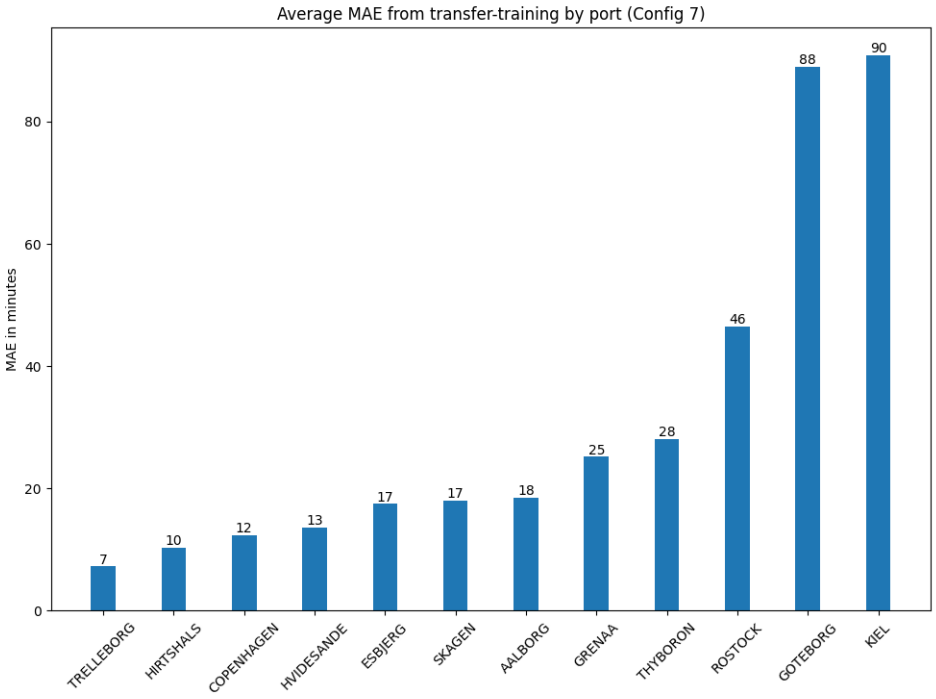

Abbildung 15: Durchschnitt aller MAEs der Transfermodelle pro Hafen unter Anwendung der Transferstrategie Nr. 7. Die x-Achse bezeichnet jeden der betrachteten Häfen aus Tabelle 2. Die Spitzen der einzelnen Balken des Diagramms beziffert den jeweiligen MAE in Minuten.

Im Vergleich zur Performance der Basismodelle aus Abbildung 13 zeichnet sich die gleiche Reihenfolge der Häfen auf der x-Achse ab. Tabelle 8 stellt die Resultate der Transfers gegenüber.

Wie Tabelle 8 zu entnehmen ist, führt der Transfer von Basismodellen auf die gelisteten Häfen zu Einbußen der Performance. Auffällig sind die beiden Häfen Hvidestande und Aalborg, deren Transfermodelle im Schnitt gleiche Abweichungen der Ankunftszeitschätzungen wie das Basismodell aufweisen. Weiterhin auffällig ist die durchschnittliche Verschlechterung der Transfermodelle vom Hafen in Goteborg von

Hafen	MAE Basismodell	Durchschnittl. MAE aller Transfermodelle	Delta
Trelleborg	3	7	+4
Hirtshals	7	10	+3
Copenhagen	10	12	+2
Hvidesande	13	13	**0**
Esbjerg	14	17	+3
Skagen	15	17	+2
Aalborg	18	18	**0**
Grenaa	23	25	+2
Thyboron	23	28	+5
Rostock	38	46	+8
Goteborg	66	88	+**22**
Kiel	87	90	+3
∅	26,4	30,9	**4,5**

Tabelle 8: Gegenüberstellung der durchschnittlichen Schätzung von Ankunftszeiten von Basismodellen und Transfermodellen unter Anwendung von Tranferstrategie Nr. 7 (vgl. Tab. 5). MAE dient als Performance-Maß eines Modells und gibt die absolute Abweichung der Schätzungen in Minuten an. Die Spalte „Delta“ indiziert einen Genauigkeitsverlust bei Abweichungen > 0 bzw. einen Genauigkeitsgewinn bei negativem Delta.

22 Minuten gegenüber seinem Basismodell. Der Transfer aller Häfen mittels Transferstrategie Nr. 7 büßt zusammenfassend 4,5 Minuten Genauigkeit bei der Schätzung von Ankunftszeiten ein.

Der Durchschnitt über mehrere Transfermodelle verschafft einen ersten Eindruck, ob ein Transfer praktisch sinnvoll sein kann. Zusammenhänge der Performance von Transfers unterschiedlicher Basishäfen können allerdings nicht hergestellt werden. Außerdem mittelt der Durchschnitt über alle angewandten Transferstrategien, weshalb etwaige ineffiziente Transfers die mittlere Performance (geringfügig) verschlechtern. Um die Güte von Transfers unterschiedlicher Transferstrategien und einzelner Häfen untersuchen zu können, wird im Folgenden eine differenzierte Betrachtungsweise vorgestellt.

Abbildung 16 aggregiert die Ergebnisse vom Transfer jedes Basismodells auf jeden anderen Hafen unter Einsatz der Transferstrategie Nr. 7 (vgl. Tab. 5). Eine solche Abbildung wird im Folgenden als *Transfer-Matrix* bezeichnet. Zur Erinnerung: Transferstrategie Nr. 7 verwendet zum Transfer eines Basismodells jedes zehnte Trainings- und Testbeispiel, passt die Gewichte der letzten vier Layer an und führt anschließendes Fine-Tuning durch.

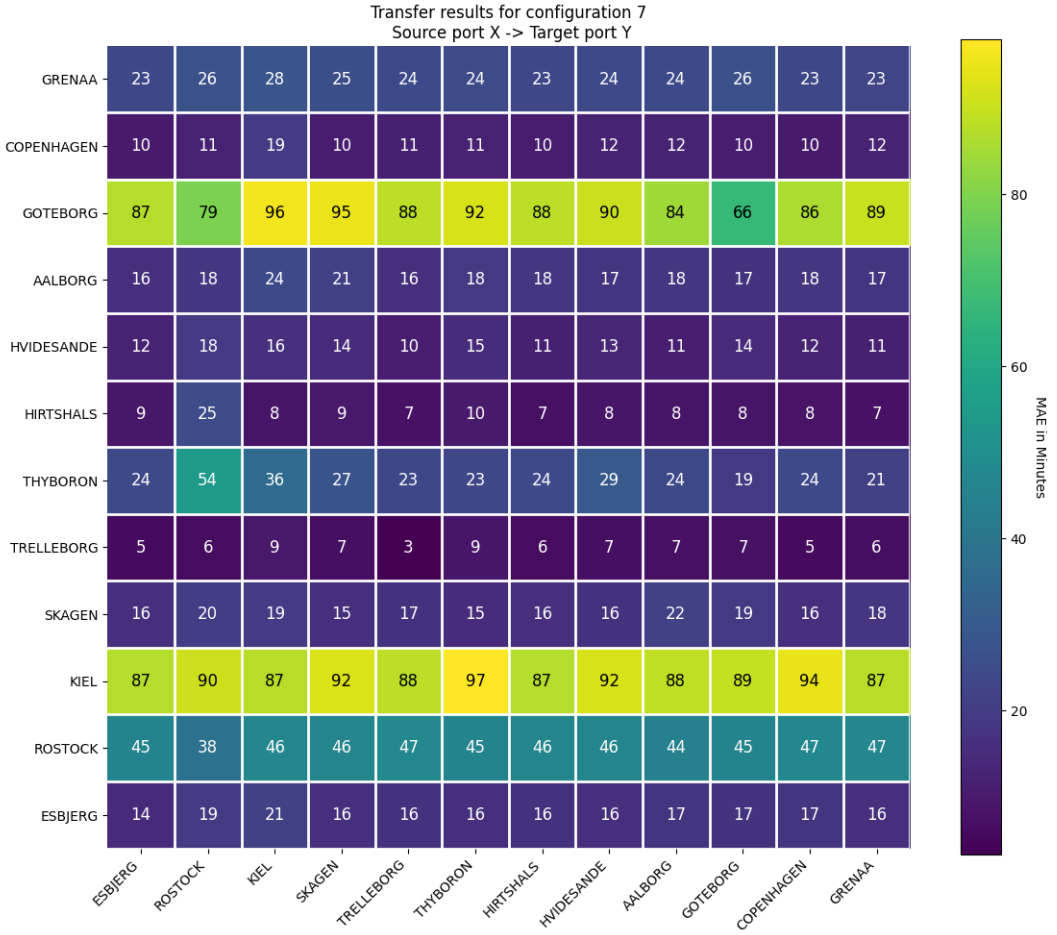

Abbildung 16: Die Transfer-Matrix zeigt die aus Transferstrategie Nr. 7 (vgl. Tab. 5) resultierenden MAEs aller Basis- und Transfermodelle Hafen-Paare (vgl. Kap. 5.6.1). Die x-Achse bezeichnet den Hafen, dessen Basismodell auf den jeweiligen Hafen der y-Achse transferiert wurde. Auf der *Anti-Diagonalen* (von links unten nach rechts oben) befinden sich die MAEs der Basismodelle, alle anderen Felder zeigen den MAE eines Transfermodells. Die Farbskala weist jedem Feld eine Farbe zur Darstellung der Qualität des Modells entsprechend der Höhe des MAE zu. Dunkles lila repräsentiert niedrige, blau mittlere und gelb hohe MAEs.

Die zeilenweise Betrachtung der Transfer-Matrix aus Abbildung 16 ermöglicht den Vergleich aller Modelle eines bestimmten Hafens. Es ist festzustellen, dass sich die Abweichungen von Schätzungen der Transfermodelle denen des Basismodells ähneln. Die ähnliche Farbgebung jeder Zeile visualisiert diese Erkenntnis. Leichte Abweichungen von dieser Schlussfolgerung zeigen sich in den Modellen von Goteborg, Hirtshals und Thyboron. In den folgenden Diskussionen werden die folgenden Abkürzungen für die betrachteten Häfen verwendet:

- Goteborg $\rightarrow GOT$
- Hirtshals $\rightarrow HIR$
- Thyboron $\rightarrow THY$
- Rostock $\rightarrow ROS$

Im Fall des Hafens von Goteborg weisen alle Transfermodelle einen höheren MAE als das eigene Basismodell auf. Während das Basismodell mit einer mittleren Abweichung von 66 Minuten die Ankunftszeit von Schiffen schätzt, kann das am besten performende Transfermodell $M^T_{ROS \rightarrow GOT}$ Ankunftszeiten durchschnittlich auf 79 Minuten genau prädizieren. Das am besten geeignete Basismodell zum Transfer auf den Hafen von Goteborg stammt demnach vom Rostocker Hafen.

Im zweiten, leicht abweichenden Fall des Hafens von Hirtshals fällt erneut der MAE des Transfermodells vom Rostocker Basishafen auf. Im Gegensatz zum Transfermodell $M^T_{ROS \rightarrow GOT}$, was die beste Performance aller Transfermodelle von Goteborg aufweist, ist $M^T_{ROS \rightarrow HIR}$ das am schlechtesten performende Transfermodell von Hirtshals. Analoges Verhalten bildet sich in der Performance des Transfermodells $M^T_{ROS \rightarrow THY}$ ab, was eine Verschlechterung von 29 Minuten vom Mittel (25 Minuten) aller Transfermodelle von Thyboron aufweist. Folglich ist das Wissen des Basismodells vom Rostocker Hafen nicht geeignet, um auf die Häfen Hirtshals und Thyboron übertragen zu werden. Das Rostocker Basismodell ist allerdings nicht zum Transfer auf alle übrigen Häfen ungeeignet. Die *restlichen* Transfermodelle mit Rostock als Basismodell weisen keine auffälligen Abweichungen des MAE auf.

7 Fazit und Ausblick

Im Rahmen dieser Arbeit ist auf Basis historischer Daten des Automatic Identification Systems (AIS) ein neuronales Netz trainiert, transferiert und jeweils evaluiert worden. Die Architektur und der Trainingsprozess des neuronalen Netzes orientieren sich an wissenschaftlichen Arbeiten, die ebenfalls ein Regressionsproblem auf einem Datensatz von Zeitreihen umsetzen. In enger Anlehnung an das InceptionTime-Konzept kombiniert das eingesetzte Netz Faltungen mit unterschiedlich großen Kernels, um in unterschiedlich langen zeitlichen Verläufen Merkmale extrahieren zu können. Letztlich reduziert Max-Pooling die zeitliche Dimension auf 1 und ein anschließendes Dense Net führt die Features zur Ausgabe einer geschätzten Dauer bis zur Ankunftszeit eines Schiffs am Zielhafen zusammen. Zunächst wurden 366 einzelne Dateien des Jahres 2020 von unzulässigen und fehlerhaften Einträgen befreit. Weiterhin sind die gesäuberten Daten Datei-übergreifend zu Zeitreihen formatiert worden, wobei jede Zeitreihe die Anfahrt eines Schiffs zu einem Zielhafen repräsentiert. Es wurden 12 Zielhäfen zur Betrachtung im Rahmen dieser Arbeit identifiziert. Für jeden dieser Häfen wurden Basismodelle trainiert, die als Grundlage zum Transfer auf *andere* Häfen dienen. Durch paarweise Permutation der betrachteten Häfen ergeben sich 132 mögliche Kombinationen zum Transfer der Basismodelle, wobei in dieser Arbeit eingeführte Transferstrategien unterschiedliche Vorgehensweisen zum Transfer eines Basismodells auf einen anderen Hafen umsetzen.

Die Auswertung der Basismodelle hat gezeigt, dass sich die historischen Daten des dänischen AIS zum Training repräsentativer Modelle eignen. Der anschließend angewandte Transfer von Basismodellen unter Zuhilfenahme von sog. *Transferstrategien* auf jeden anderen der 12 Häfen hat sich als produktiv und funktionstüchtig erwiesen. Mittels Transferstrategien konnten unterschiedliche Varianten von Transfers definiert und systematisch durchgeführt werden. Eine Transferstrategie spezifiziert beim Transfer anzupassende Gewichte in Layern des neuronalen Netzes. Außerdem dünnt sie die Datenbasis eines Hafens zur Simulation kleiner Häfen aus und stößt optional Fine-Tuning *(Feinabstimmung auf den neuen Hafen)* des transferierten Modells an. Im Vergleich der resultierenden transferierten Modelle hat die Evaluation im Anwendungsfall dieser Arbeit Transferstrategie Nr. 7 als beste Variante unter den angewandten Strategien (vgl. Tab. 5) identifiziert. Konkret wurden beim Transfer unter Einsatz der 7. Transferstrategie die Gewichte der letzten vier Layer des neuronalen Netzes zur Anpassung freigegeben. Die Datenbasis wurde künstlich auf lediglich 10% der verfügbaren Trainingsbeispiele reduziert und nachgelagertes Fine-Tuning angewendet. Höheres Ausdünnen der Daten führte zu höheren Einbußen der Performance transferierter Modelle. Die resultierenden Transfermodelle erzielten im Durchschnitt eine 4,5 Minuten schlechtere Schätzung von Ankunftszeiten als

ihre korrespondierenden Basismodelle. Folglich kann ein Basismodell mit geringen Einbußen der Performance auf Häfen mit kleiner Datenbasis transferiert werden.

In dieser Arbeit wurden die 12 Häfen unter der Prämisse ausgewählt, dass der Trainingsalgorithmus ausreichend Daten zur Generierung eines Basismodells vorfindet. Im praktischen Einsatz können auf Basis der Erkenntnisse dieser Arbeit Basismodelle auf Häfen mit *kleiner* Datenbasis transferiert werden. Je nach gewünschter Präzision der Ankunftszeitschätzungen, kann gemäß der Performance von Transfers (vgl. Abb. 15) in Verbindung mit der Anzahl verarbeiteter Anfahrten ermittelt werden, wie groß die Datenbasis eines Hafens zum Transfer sein muss. Wird beispielsweise die Performance des Transfermodells von Thyboron angestrebt (MAE = 28 Minuten), sind auf Basis der Erkenntnisse dieser Arbeit $784 \cdot 10\% \approx \mathbf{79}$ Anfahrten (vgl. Tab. 2) und eine Diversität von **0,954** notwendig, um ein Transfermodell zu generieren.

In dieser Arbeit wurden Trainingsbeispiele von Grund auf aus historischen Rohdaten des AIS generiert. Bis auf One-Hot-Encoding und Feature-Scaling im Intervall $[0, 1]$ sind die Daten unverändert zum Training und Transfer verwendet worden. Für das Netz ist die Zugehörigkeit eines Trainingsbeispiels zu einer Route nicht a-priori bekannt. Es bleibt offen zu untersuchen, ob weiteres Feature Engineering wie die Ermittlung von Routen, die Performance der Basis- und Transfermodelle verbessern kann. Präzisere Basismodelle und/oder zusätzliche (aussagekräftige) Features könnten die benötigte Anzahl Trainingsbeispiele zum Transfer weiter als die identifizierten 10% verringern.

A Anhang

Tabelle A9 beinhaltet eine Checkliste an Fragen des Design Science Research (DSR) nach Hevner u. Chatterjee (2010), die Verlauf eines Forschungsprojekts beantwortet werden. Jede Frage ist einem Prozess des DSR (vgl. Abbildung 1) und einem oder mehreren Abschnitten dieser Arbeit zugeordnet.

	Fragen	DSR-Zyklus	Kapitel
1	What is the research question (design requirements)?	Relevanz	1
2	What is the artifact? How is the artifact represented?	Design	5.5
3	What design processes (search heuristics) will be used to build the artifact?	Design	5.4
4	How are the artifact and the design processes grounded by the knowledge base? What, if any, theories support the artifact design and the design process?	Rigor	2 (& 3)
5	What evaluations are performed during the internal design cycles? What design improvements are identified during each design cycle?	Design	5.5
6	How is the artifact introducted into the application environment and how is it field tested? What metrics are used to demonstrate artifact utility and improvement over previous artifacts?	Relevanz	6
7	What new knowledge is added to the knowledge base and in what form (e.g., peer-reviewed literature, meta-artifacts, new theory, new method)?	Rigor	6
8	Has the research question been satisfactorily addressed?	Relevanz	7

Tabelle A9: Checkliste für Forschungsprojekte des DSR gemäß Hevner u. Chatterjee (2010)

Tabelle A10 zeigt die übernommenen Spalten des AIS Datensets. Die Index-Spalte repräsentiert den Index des Features in den Trainingsdaten. Der Wertebereich stammt aus der AIS Dokumentation[1] oder aus der natürlichen Begrenzung des Datums[2]. Jeder Wertebereich wurde zum Training auf das Intervall $[0, 1]$ normiert (vgl. Kapitel 5.1).

Index	Beschreibung	Wertebereich	Einheit
0	Latitude	$[-90, 90]$	Grad (WGS 84)
1	Longitude	$[-180, 180]$	Grad (WGS 84)
2	SOG	$[0, 110]$	Knoten
3	COG	$[0, 159.9]$	Grad (WGS 84)
4	Heading	$[0, 511]$	Grad (WGS 84)
5	Width	$[0, 80]$	Meter
6	Length	$[0, 500]$	Meter
7	Draught	$[0, 40]$	Meter
8-20	One-Hot codiertes Feature Ship Type, vgl. Tabelle A11		
21-35	One-Hot codiertes Feature Navigational Status, vgl. Tabelle A12		
36	Time	$[0, 31622400]$	Sekunden
37	Label	$[0, 31622400]$	Sekunden

Tabelle A10: Beschaffenheit des Trainingsdatensatzes

Tabelle A11 zeigt die verwendeten Ausprägungen des Feldes Ship Type (vgl. Def. 11) des AIS Datensets. Die Index-Spalte repräsentiert den Index des Features in den Trainingsdaten. Die ursprüngliche Ausprägung gemäß der AIS Dokumentation[3] beziffert die Typ-Code-Nummer in Klammern, gefolgt von ihrem Beschreibungstext.

[1] api.vtexplorer.com/docs/response-ais.html
[2] www.sostechnic.com/epirbs/ais/aisinformationenglish/index.php
[3] api.vtexplorer.com/docs/ref-aistypes.html

Index	Verwendete Ausprägung	Ursprüngliche Ausprägung
8	Cargo	(70-74, 79) Cargo
9	Dredging	(33) Dredging or underwater ops
10	HSC	(40-44, 49) High speed craft (HSC)
11	Fishing	(30) Fishing
12	Passenger	(60-64, 69) Passenger
13	Pleasure	(37) Pleasure Craft
14	Reserved	(1-19) Reserved for future use (25-29) Wing in ground (WIG) (38, 39) Reserved (45-48) High speed Craft (HSC) (65-68) Passenger, Reserved (75-78) Cargo, Reserved (85-88) Tanker, Reserved
15	Sailing	(36) Sailing
16	Tanker	(80-84, 89) Tanker
17	Towing	(31) Towing (32) Towing: length>200m or breadth>25m
18	Tug	(52) Tug
19	Other ship type	(20-24) Wing in ground (WIG) (34) Diving ops (35) Military ops (50) Pilot Vessel (51) Search and Rescue vessel (53) Port Tender (54) Anti Pollution Equipment (55) Law Enforcement (56, 57) Spare - Local Vessel (58) Medical Trasport (59) Noncombatant ship according to RR Resolution No. 18 (90-99) Other Type
20	Default ship type	(0) Not available (default)

Tabelle A11: Abbildung der Ausprägungen des Felder Ship Type (vgl. Def. 11) zur One-Hot-Codierung

Tabelle A12 zeigt die verwendeten Ausprägungen des Feldes Navigational Status (vgl. Def. 8) des AIS Datensets. Die Index-Spalte repräsentiert die Stelle des Datums in den Trainingsdaten. Die ursprüngliche Ausprägung gemäß der AIS Dokumentation[4] beziffert die Status-Nummer in Klammern, gefolgt von ihrem Beschreibungstext.

Index	Verwendete Ausprägung	Ursprüngliche Ausprägung
21	Under way using engine	(0) Under way using engine
22	At anchor	(1) At anchor
23	Not under command	(2) Not under command
24	Restricted maneuverability	(3) Restructed manoeuverability
25	Constrained by her draught	(4) Constrained by her draught
26	Moored	(5) Moored
27	Aground	(6) Aground
28	Engaged in Fishing	(7) Engaged in fishing
29	Under way sailing	(8) Under way sailing
30	Reserved for future amendment of Navigational Status for HSC	(9) Reserved for future amendment of Navigational Status for HSC
31	Reserved for future amendment of Navigational Status for WIG	(10) Reserved for future amendment of Navigational Status for WIG
32	Reserved for future use	(11), (12), (13) Reserved for future use
33	AIS-SART is active	(14) AIS-SART is active
34	Other navigational status	No category found
35	Default navigational status	(15) Not defined (default)

Tabelle A12: Abbildung der Ausprägungen des Feldes Navigational Status (vgl. Def. 8) zur One-Hot-Codierung

[4] api.vtexplorer.com/docs/ref-navstat.html

Die Abbildungen 17 - 27 zeigen die den MAE der Basismodelle, gruppiert nach den tatsächlichen Ankunftszeiten (Labels= der Testbeispiele.

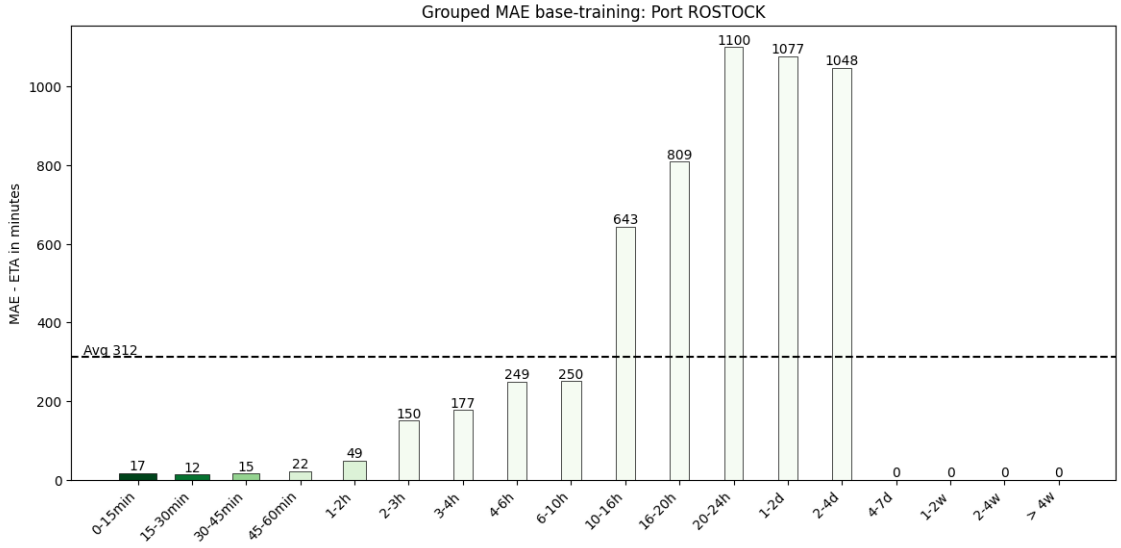

Abbildung 17: Gruppierter MAE des Basismodells vom Hafen Rostock.

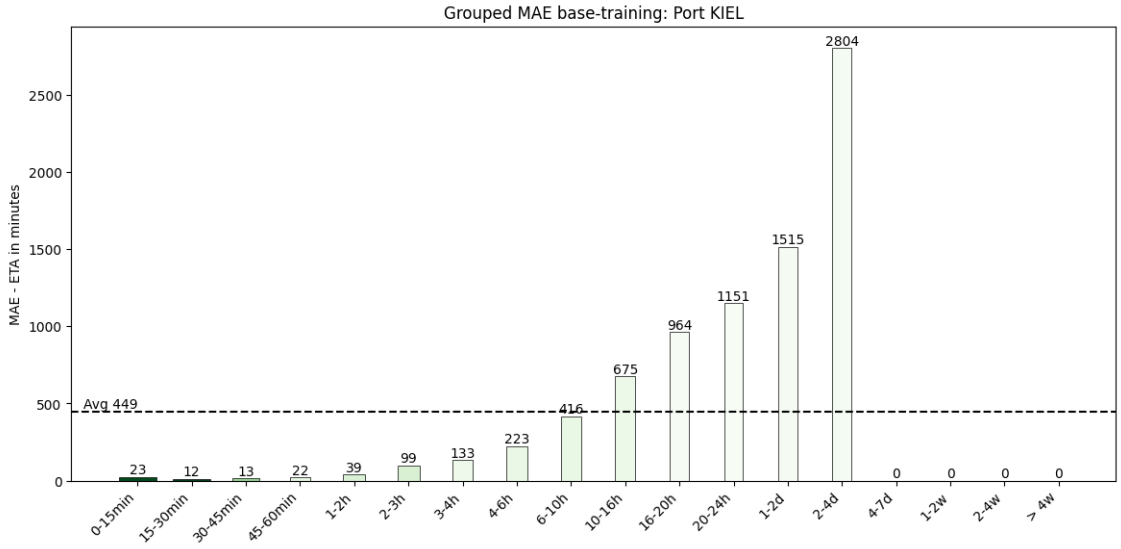

Abbildung 18: Gruppierter MAE des Basismodells vom Hafen Kiel.

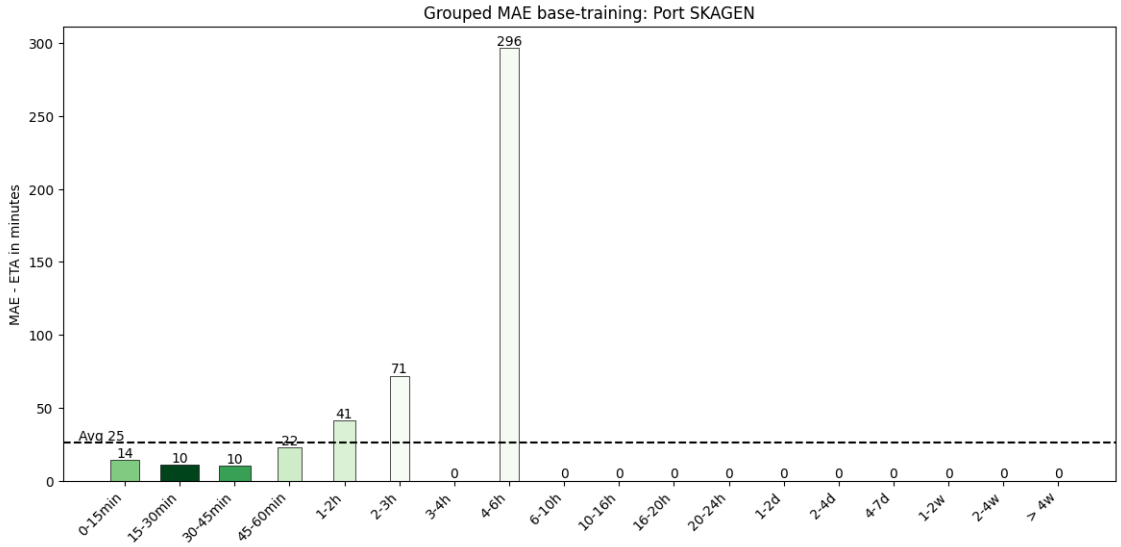

Abbildung 19: Gruppierter MAE des Basismodells vom Hafen Skagen.

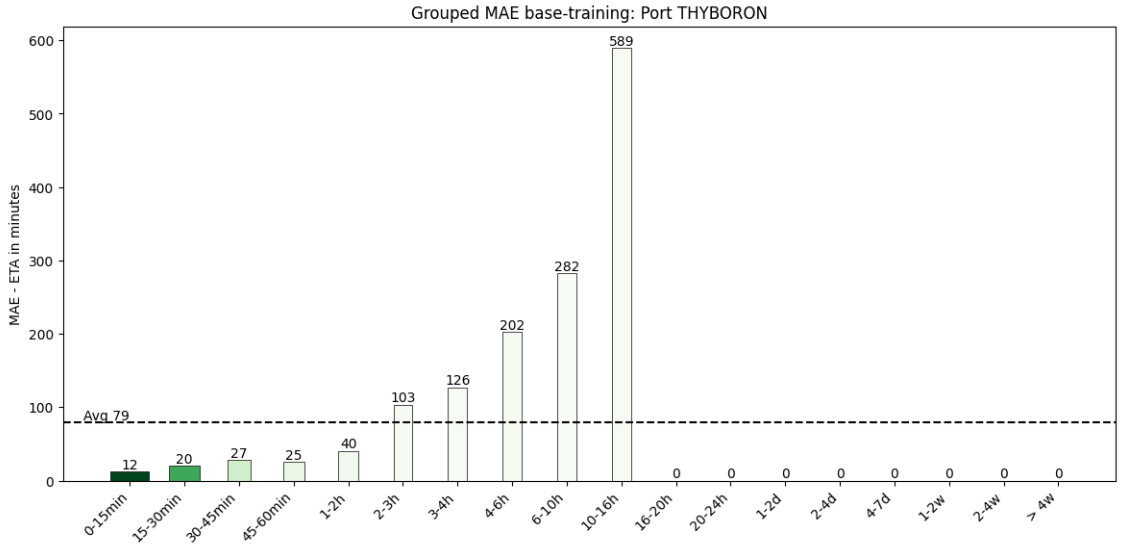

Abbildung 20: Gruppierter MAE des Basismodells vom Hafen Thyboron.

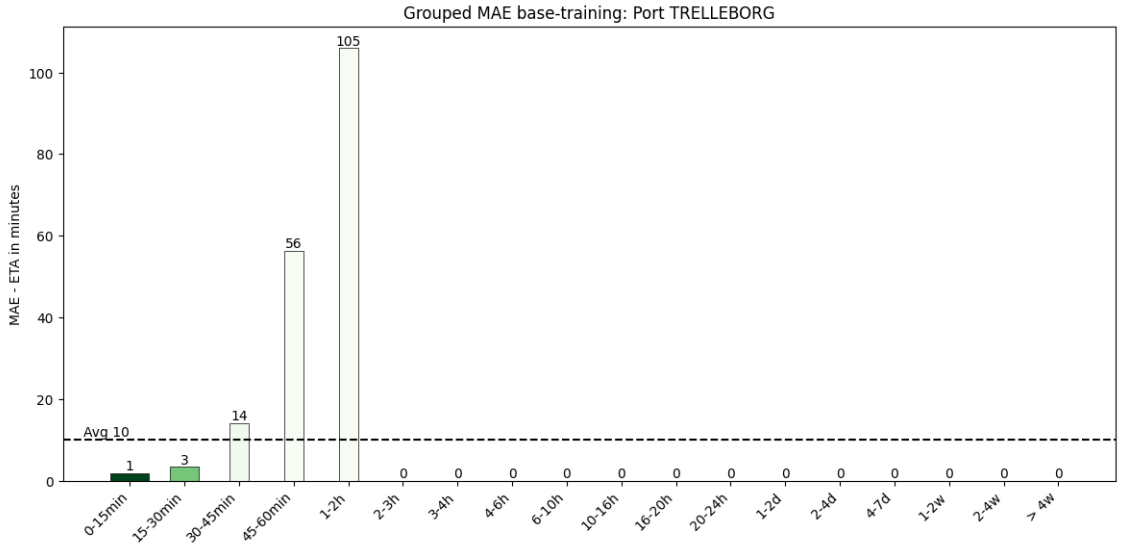

Abbildung 21: Gruppierter MAE des Basismodells vom Hafen Trelleborg.

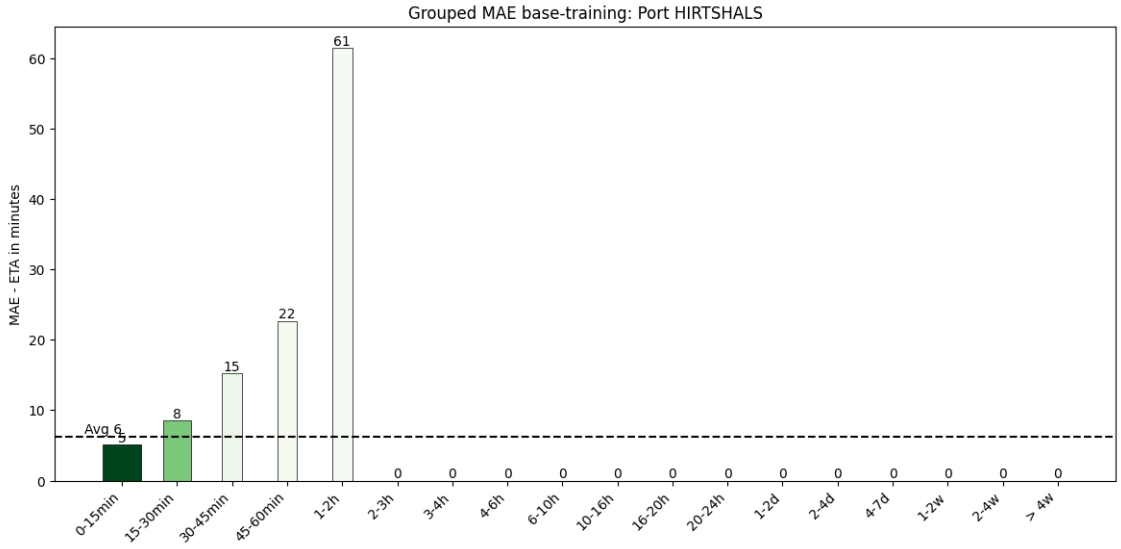

Abbildung 22: Gruppierter MAE des Basismodells vom Hafen Hirtshals.

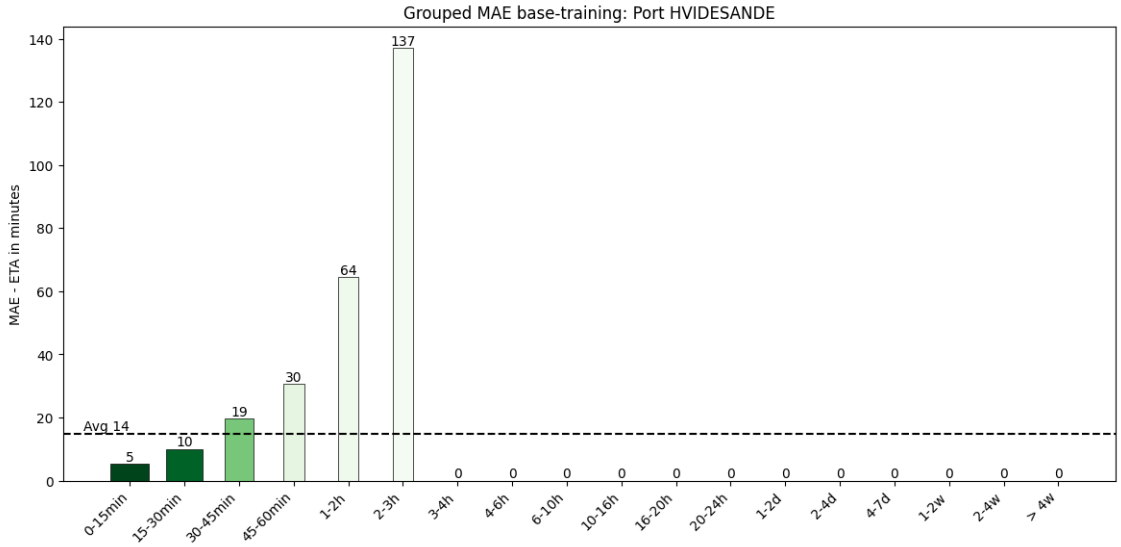

Abbildung 23: Gruppierter MAE des Basismodells vom Hafen Hvidesande.

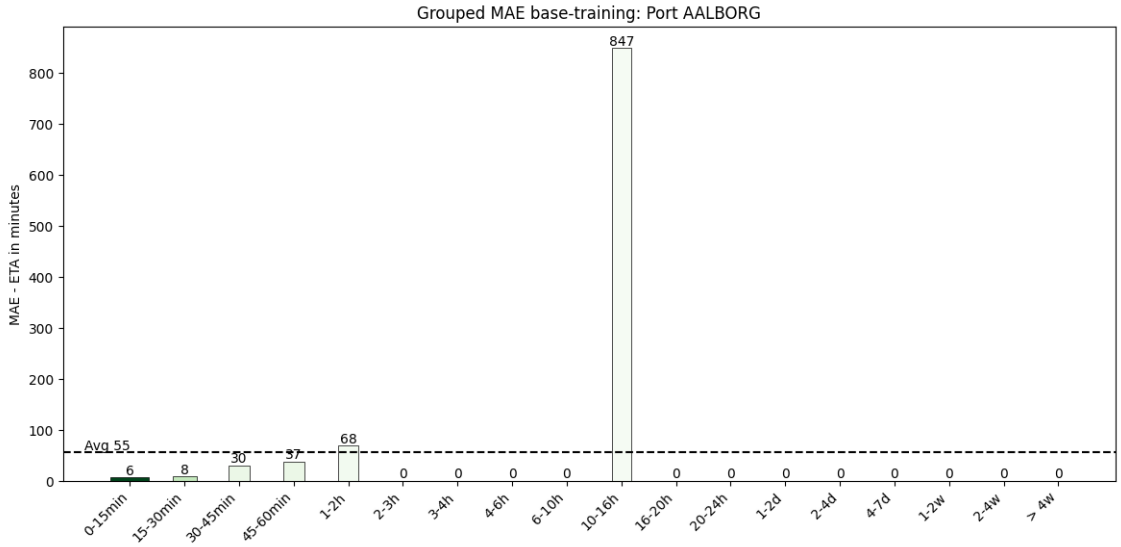

Abbildung 24: Gruppierter MAE des Basismodells vom Hafen Aalborg.

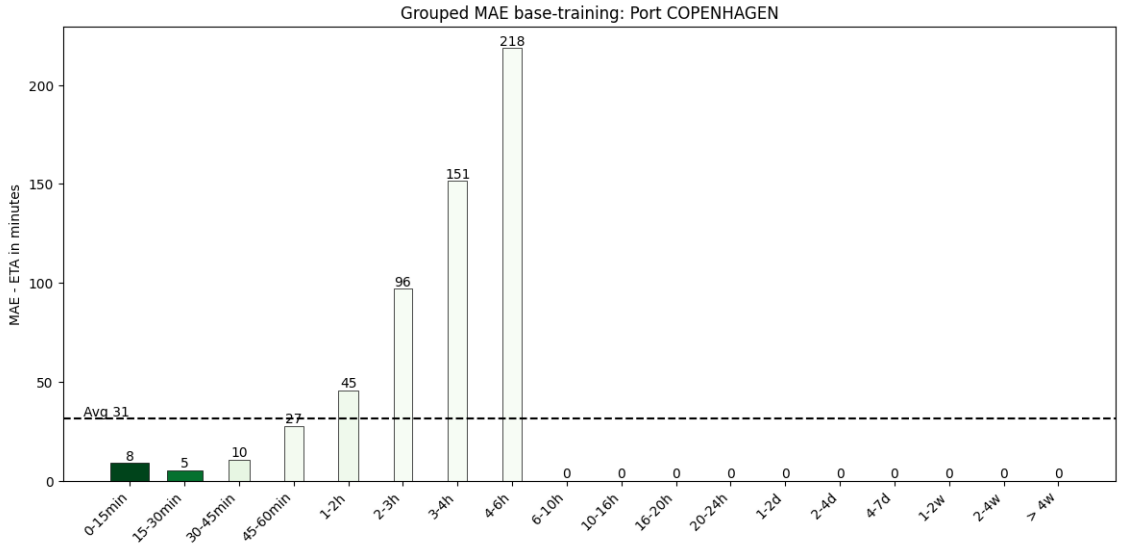

Abbildung 25: Gruppierter MAE des Basismodells vom Hafen Copenhagen.

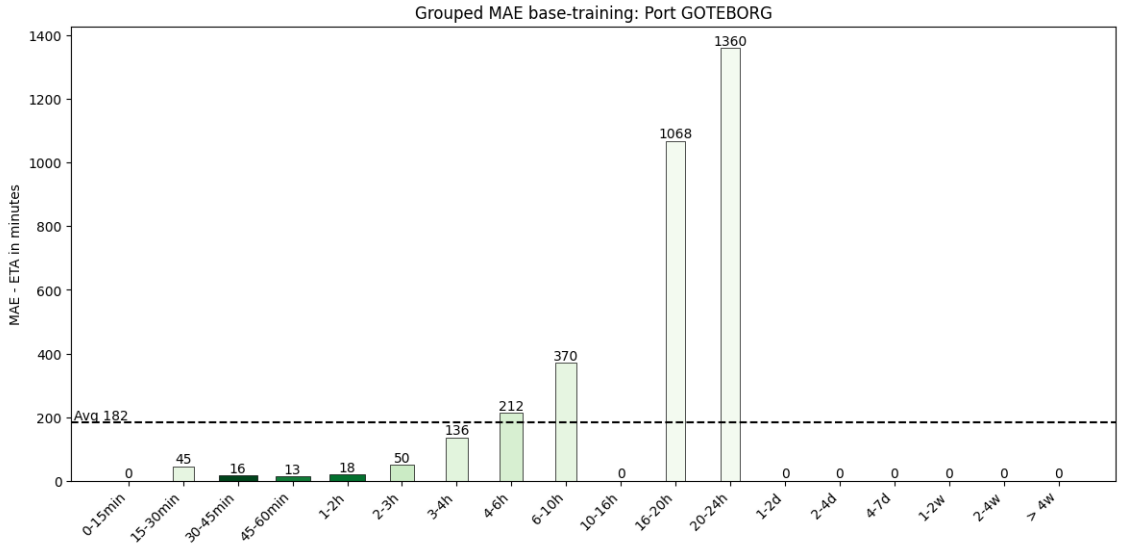

Abbildung 26: Gruppierter MAE des Basismodells vom Hafen Goteborg.

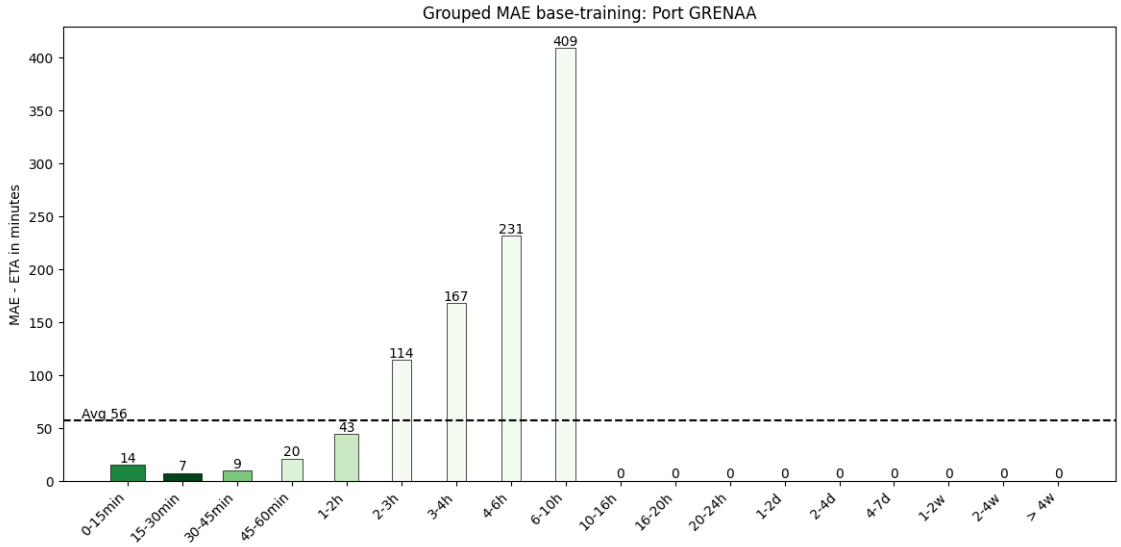

Abbildung 27: Gruppierter MAE des Basismodells vom Hafen Grenaa.

Die Abbildungen 28 - 32 zeigen die durchschnittliche Performance der Transfermodelle pro Hafen.

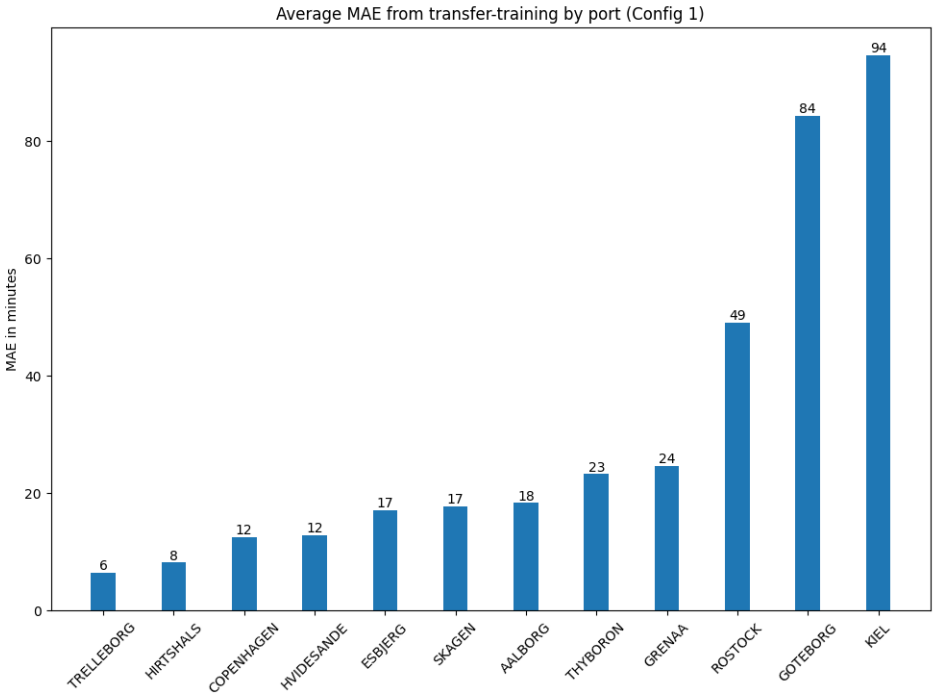

Abbildung 28: Durchschnitt aller MAEs der Transfermodelle pro Hafen unter Anwendung der Transferstrategie Nr. 1.

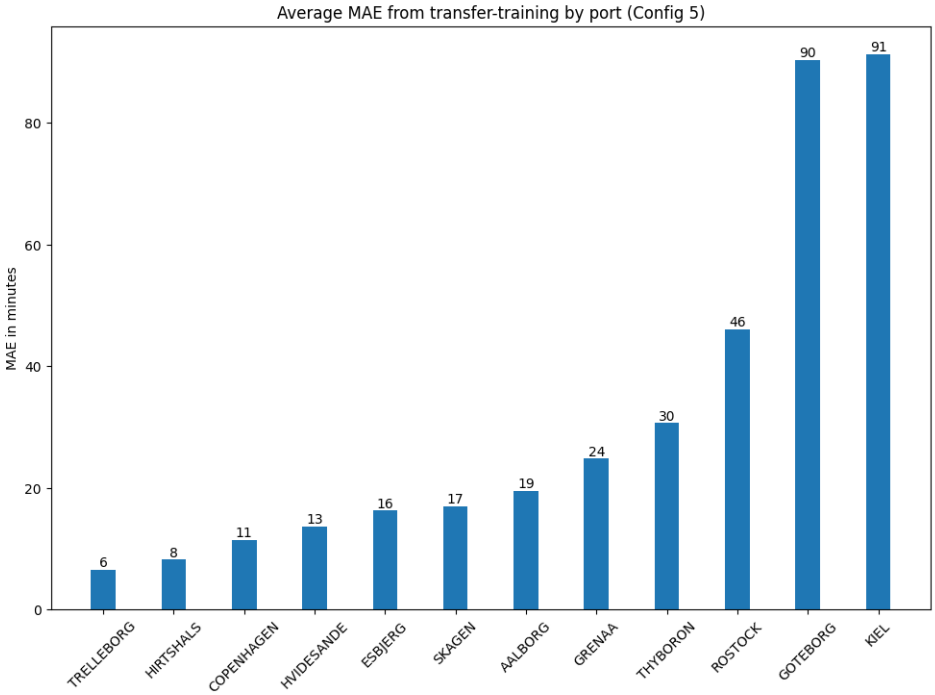

Abbildung 29: Durchschnitt aller MAEs der Transfermodelle pro Hafen unter Anwendung der Transferstrategie Nr. 5.

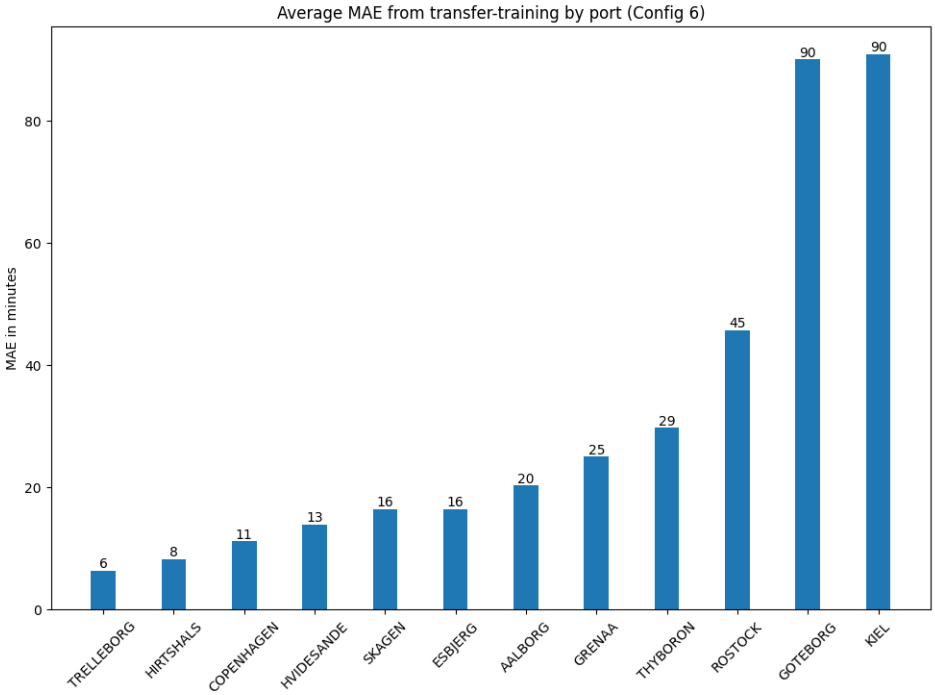

Abbildung 30: Durchschnitt aller MAEs der Transfermodelle pro Hafen unter Anwendung der Transferstrategie Nr. 6.

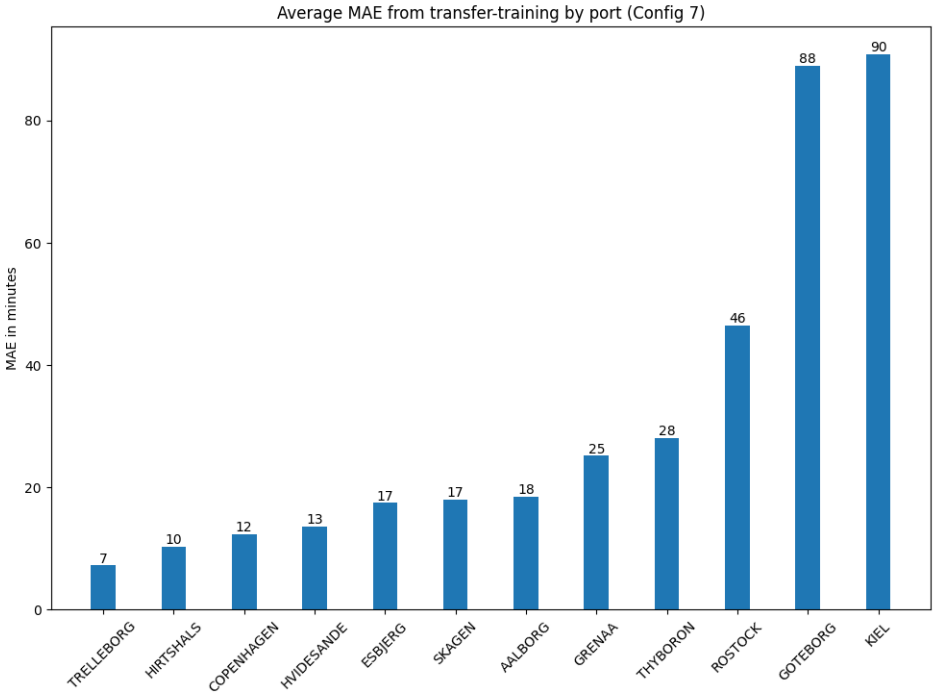

Abbildung 31: Durchschnitt aller MAEs der Transfermodelle pro Hafen unter Anwendung der Transferstrategie Nr. 7.

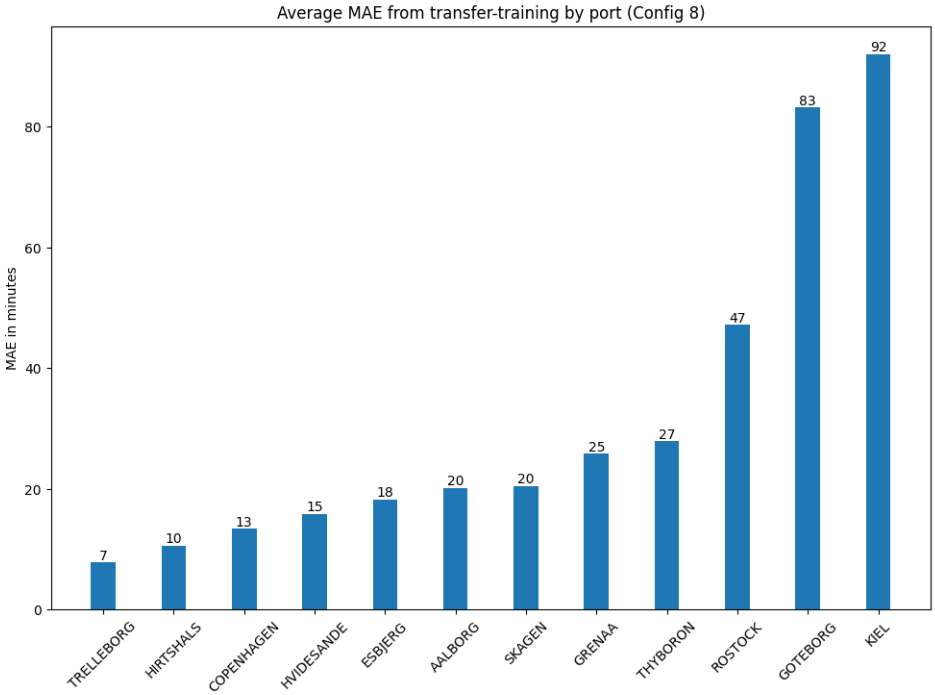

Abbildung 32: Durchschnitt aller MAEs der Transfermodelle pro Hafen unter Anwendung der Transferstrategie Nr. 8.

Die nachfolgenden Abbildungen 33 - 36 zeigen Transfer-Matrizen hinsichtlich der angewandten Transferstrategie (vgl. Def. 23).

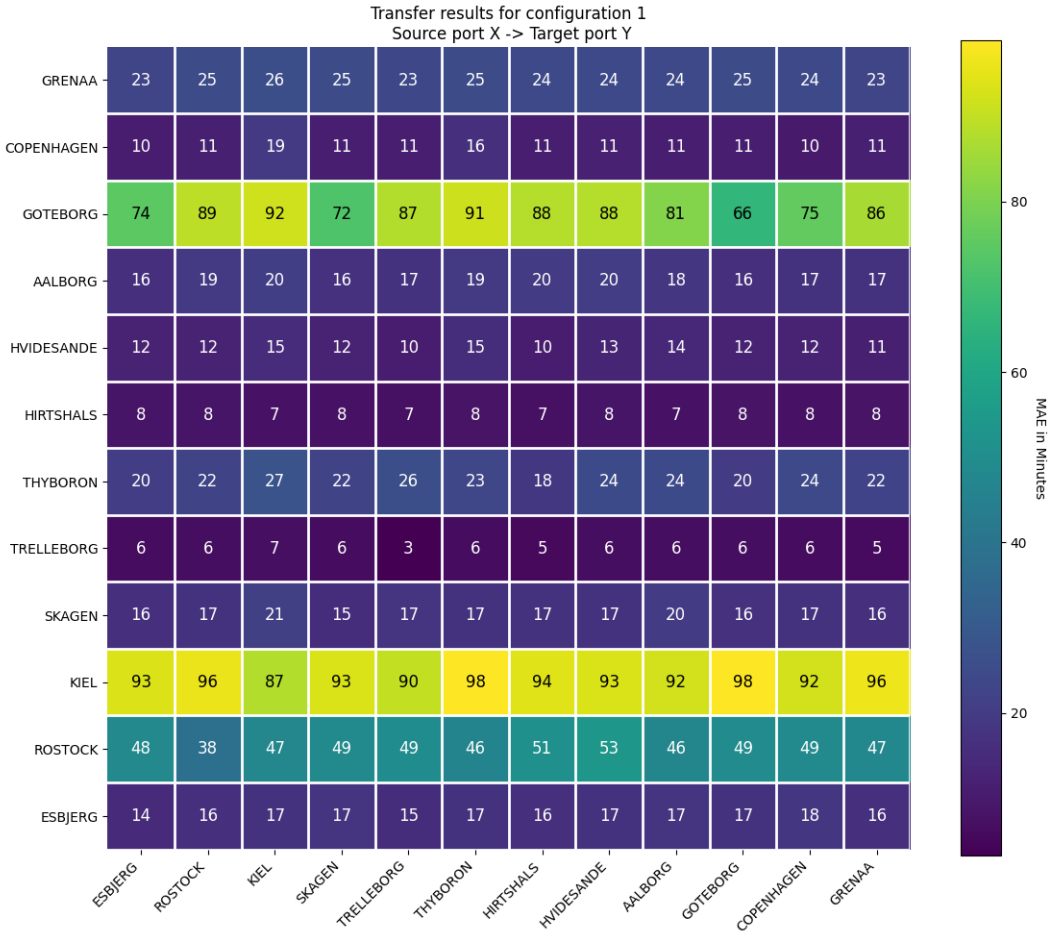

Abbildung 33: Die Transfer-Matrix zeigt die aus Transferstrategie Nr. 1 (vgl. Tab. 5) resultierenden MAEs alles Basis- und Transfermodelle Hafen-Paare (vgl. Kap. 5.6.1). Die x-Achse bezeichnet den Hafen, dessen Basismodell auf den jeweiligen Hafen der y-Achse transferiert wurde. Auf der *Anti-Diagonalen* (von links unten nach rechts oben) befinden sich die MAEs der Basismodelle, alle anderen Felder zeigen den MAE eines Transfermodells. Die Farbskala weist jedem Feld eine Farbe zur Darstellung der Qualität des Modells entsprechend der Höhe des MAE zu. Dunkles lila repräsentiert niedrige, blau mittlere und gelb hohe MAEs.

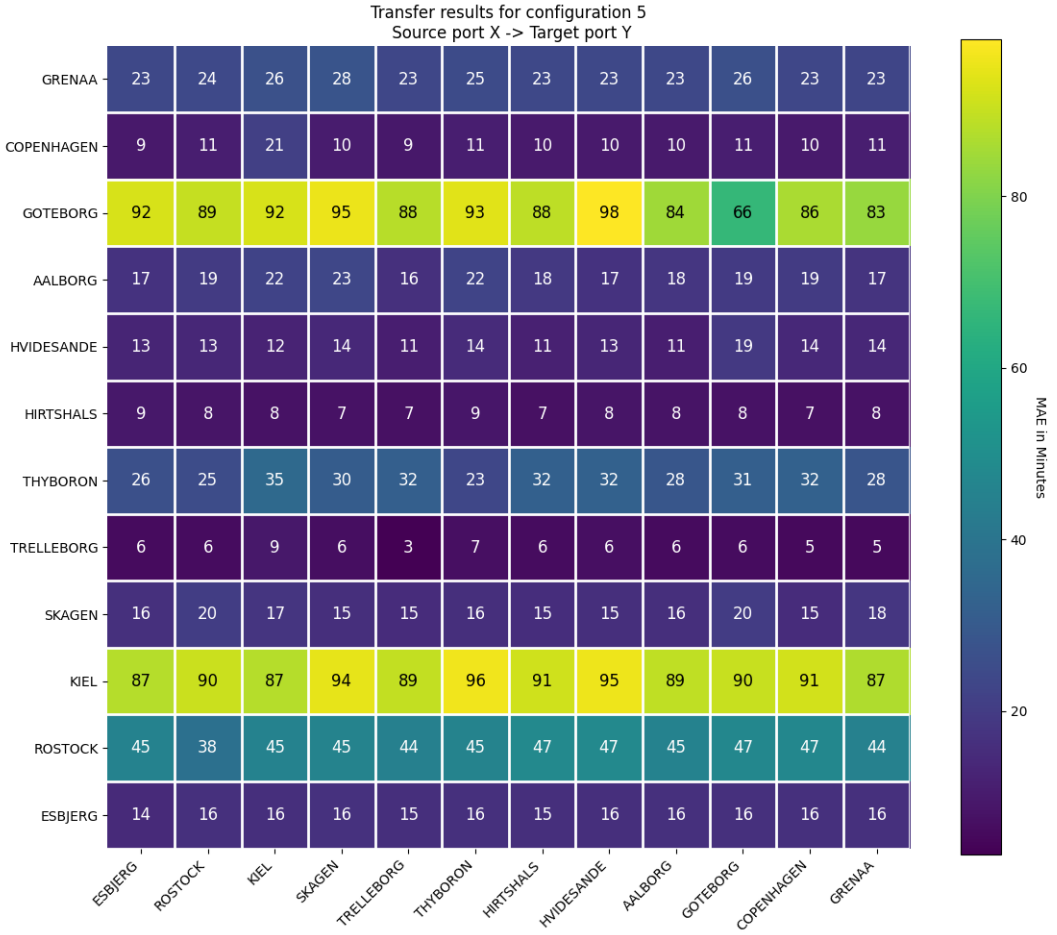

Abbildung 34: Die Transfer-Matrix zeigt die aus Transferstrategie Nr. 5 (vgl. Tab. 5) resultierenden MAEs alles Basis- und Transfermodelle Hafen-Paare (vgl. Kap. 5.6.1). Die x-Achse bezeichnet den Hafen, dessen Basismodell auf den jeweiligen Hafen der y-Achse transferiert wurde. Auf der *Anti-Diagonalen* (von links unten nach rechts oben) befinden sich die MAEs der Basismodelle, alle anderen Felder zeigen den MAE eines Transfermodells. Die Farbskala weist jedem Feld eine Farbe zur Darstellung der Qualität des Modells entsprechend der Höhe des MAE zu. Dunkles lila repräsentiert niedrige, blau mittlere und gelb hohe MAEs.

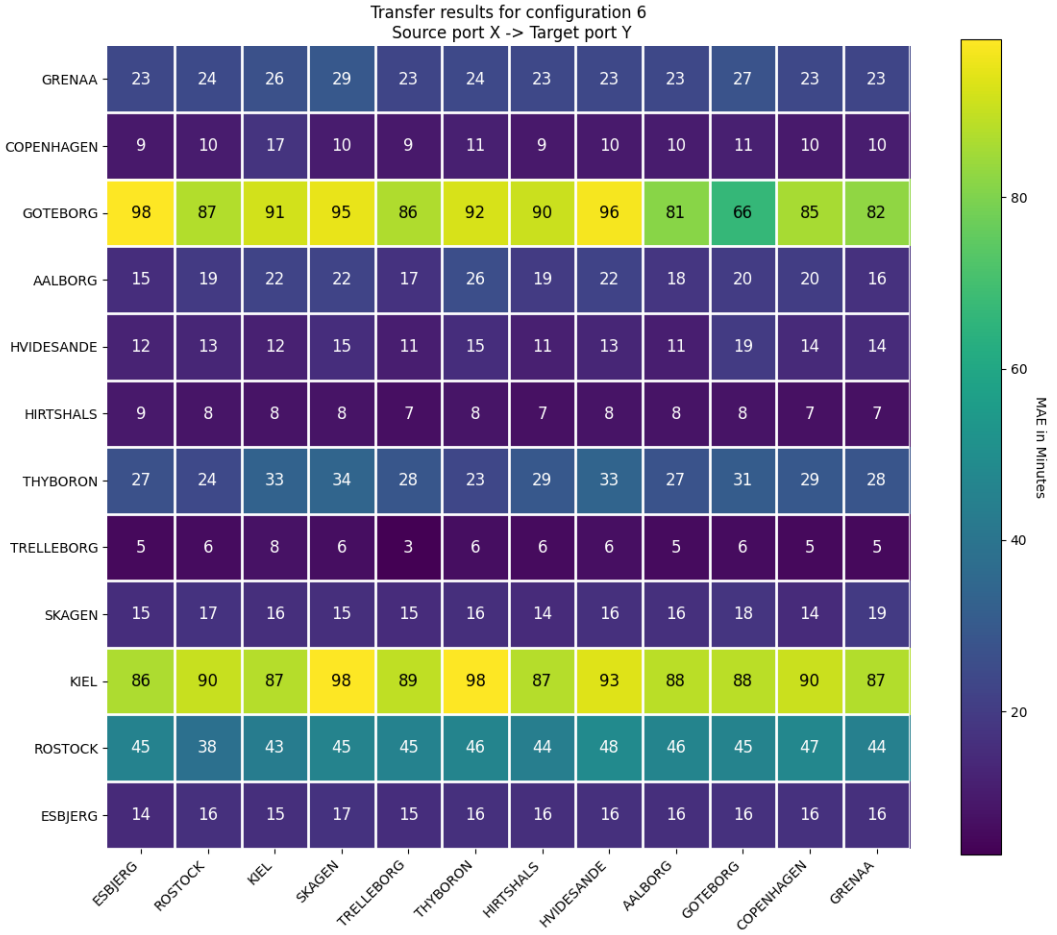

Abbildung 35: Die Transfer-Matrix zeigt die aus Transferstrategie Nr. 6 (vgl. Tab. 5) resultierenden MAEs alles Basis- und Transfermodelle Hafen-Paare (vgl. Kap. 5.6.1). Die x-Achse bezeichnet den Hafen, dessen Basismodell auf den jeweiligen Hafen der y-Achse transferiert wurde. Auf der *Anti-Diagonalen* (von links unten nach rechts oben) befinden sich die MAEs der Basismodelle, alle anderen Felder zeigen den MAE eines Transfermodells. Die Farbskala weist jedem Feld eine Farbe zur Darstellung der Qualität des Modells entsprechend der Höhe des MAE zu. Dunkles lila repräsentiert niedrige, blau mittlere und gelb hohe MAEs.

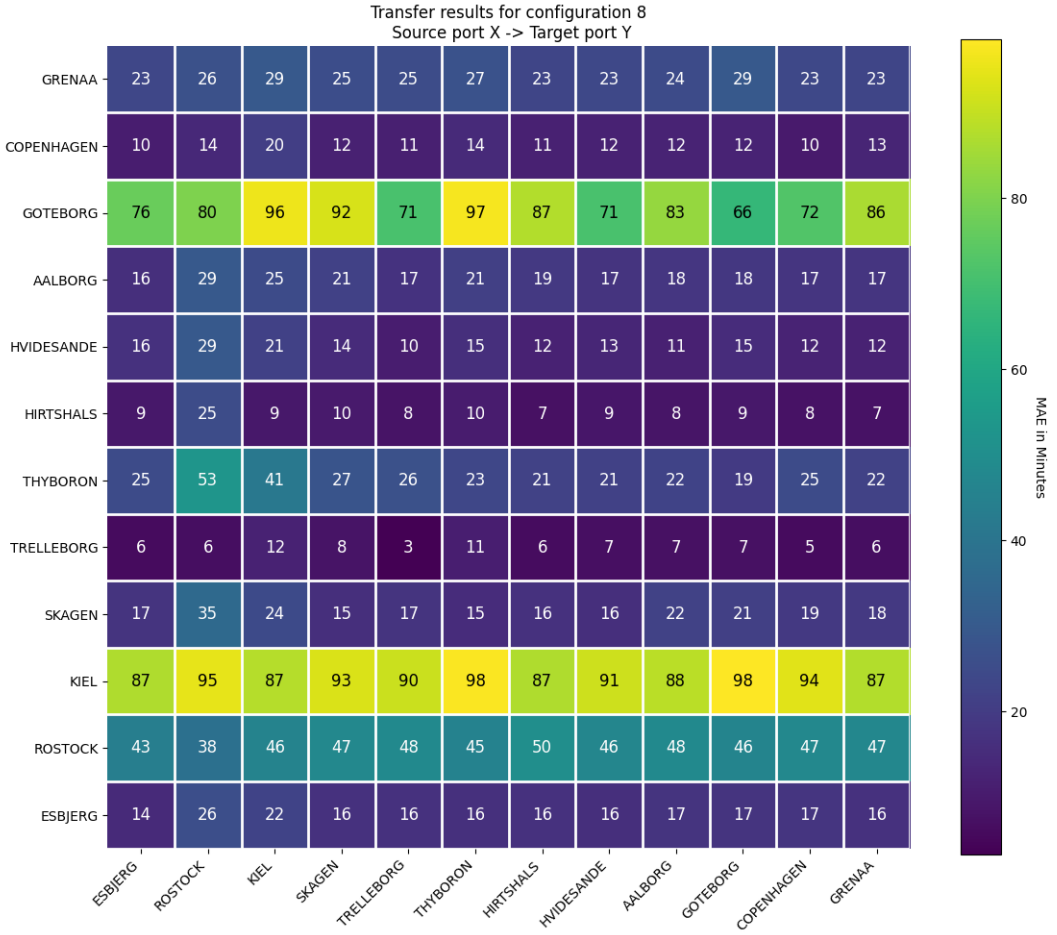

Abbildung 36: Die Transfer-Matrix zeigt die aus Transferstrategie Nr. 8 (vgl. Tab. 5) resultierenden MAEs alles Basis- und Transfermodelle Hafen-Paare (vgl. Kap. 5.6.1). Die x-Achse bezeichnet den Hafen, dessen Basismodell auf den jeweiligen Hafen der y-Achse transferiert wurde. Auf der *Anti-Diagonalen* (von links unten nach rechts oben) befinden sich die MAEs der Basismodelle, alle anderen Felder zeigen den MAE eines Transfermodells. Die Farbskala weist jedem Feld eine Farbe zur Darstellung der Qualität des Modells entsprechend der Höhe des MAE zu. Dunkles lila repräsentiert niedrige, blau mittlere und gelb hohe MAEs.

Literatur

[Alessandrini u. a. 2019] Alessandrini, Alfredo ; Mazzarella, Fabio ; Vespe, Michele: Estimated Time of Arrival Using Historical Vessel Tracking Data. In: *IEEE Transactions on Intelligent Transportation Systems* 20 (2019), Nr. 1, S. 7–15. http://dx.doi.org/10.1109/TITS.2017.2789279. – DOI 10.1109/TITS.2017.2789279

[National Customs Brokers & Forwarders Association of America (NCBFAA) 2021] America (NCBFAA), Inc. National Customs Brokers & Forwarders Association o.: *Port Congestion: An Economic Threat.* zuletzt abgerufen: 20.06.2021. https://www.ncbfaa.org/Scripts/4Disapi.dll/4DCGI/cms/review.html?Action=CMS_Document&DocID=18042. Version: 2021

[Araujo u. Sancricca 2021] Araujo, Samir ; Sancricca, Michele: *Using machine learning to predict vessel time of arrival with Amazon SageMaker.* https://aws.amazon.com/blogs/machine-learning/using-machine-learning-to-predict-vessel-time-of-arrival-with-amazon-sagemaker/, 2021. – zuletzt abgerufen: 16.06.2021

[Area Central Europe 2021] Area Central Europe, ACE: *SURCHANRGES: EXPORT OVERVIEW.* Website Surcharge & Feeds, zuletzt abgerufen: 18.06.2021. https://www.hamburgsud-line.com/liner/de/liner_services/country_information/germany/deu_forms_procedures/index.html. Version: 2021

[Aswiga u. a. 2021] Aswiga, R. ; Aishwarya, R. ; Shanthi, A.: Multistage transfer learning technique for classifying rare medical datasets. In: *Journal of Ambient Intelligence and Humanized Computing* (2021), 03. http://dx.doi.org/10.1007/s12652-021-02989-1. – DOI 10.1007/s12652–021–02989–1

[Bagnall u. a. 2016] Bagnall, Anthony J. ; Bostrom, Aaron ; Large, James ; Lines, Jason: The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version. In: *CoRR* abs/1602.01711 (2016). http://arxiv.org/abs/1602.01711

[Balster u. a. 2020] Balster, Andreas ; Hansen, Ole ; Friedrich, Hanno ; Ludwig, André: An ETA Prediction Model for Inermodal Transport Networks Based on Machine Learning. In: *Business & Information Systems Engineering* 62 (2020), S. 403–416

[Barker 2016] Barker, Ian: *Dark data and why you should worry about it.* https://betanews.com/2016/02/19/big-dark-data, 2016. – zuletzt abgerufen: 21.06.2021

[Baskerville 2001] Baskerville, R.: Conducting Action Research: High Risk and High Reward in Theory and Practice. (2001), 04

[BBC 2019] BBC: *World's largest container ship arrives at Felixstowe.* https://www.bbc.com/news/av/uk-england-suffolk-49610297, 9 2019. – zuletzt abgerufen: 18.06.2021

[BBC News 2021] BBC News, Justin H.: *Suez blockage is holding up $9.6bn of goods a day.* https://www.bbc.com/news/business-56533250, 2021. – zuletzt abgerufen: 06.05.2021

[Berthold 2019] Berthold, K. R.: Supply Chain Management: A Descriptive Copnception. In: *International Journal For Empirical Educatoin and Research* (2019), 3, S. 42–65. http://dx.doi.org/10.35935/edr/32.5642. – DOI 10.35935/edr/32.5642

[Bjorck u. a. 2018] Bjorck, Johan ; Gomes, Carla ; Selman, Bart ; Weinberger, Kilian Q.: *Understanding Batch Normalization.* 2018

[Bodunov u. a. 2018] Bodunov, Oleh ; Schmidt, Florian ; Martin, André ; Brito, Andrey ; Fetzer, Christof: Real-time Destination and ETA Prediction for Maritime Traffic. In: *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems* (2018), Jun. http://dx.doi.org/10.1145/3210284.3220502. – DOI 10.1145/3210284.3220502. ISBN 9781450357821

[Bottou 1998] Bottou, Léon: Online Algorithms and Stochastic Approximations. Version: 1998. http://leon.bottou.org/papers/bottou-98x. In: Saad, David (Hrsg.): *Online Learning and Neural Networks.* Cambridge, UK : Cambridge University Press, 1998. – revised, oct 2012

[Breiman u. a. 1984] Breiman, L. ; Friedman, J. ; Stone, C.J. ; Olshen, R.A.: *Classification and Regression Trees.* Taylor & Francis, 1984 https://books.google.de/books?id=JwQx-WOmSyQC. – ISBN 9780412048418

[Breiman 2001] Breiman, Leo: Random Forests. In: *Machine Learning* 45 (2001), 10, S. 5–32. http://dx.doi.org/10.1023/A:1010933404324. – DOI 10.1023/A:1010933404324

[Brownlee 2019] Brownlee, Jason: *Difference Between Classification and Regression in Machine Learning.* https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/, 2019. – zuletzt abgerufen: 21.06.2021

[Buzoianu 2020] Buzoianu, Gabriela: *Delivery Deval | What Causes Late Shipping Delivery?* https://www.eurosender.com/blog/en/delayed-delivery/, 2020. – zuletzt abgerufen: 30.06.2021

[Chalapathy u. Chawla 2019] Chalapathy, Raghavendra ; Chawla, Sanjay: *Deep Learning for Anomaly Detection: A Survey*. 2019

[Chollet 2020] Chollet, Francois: *Transfer learning & fine-tuning*. https://keras.io/guides/transfer_learning/, 2020. – zuletzt abgerufen: 29.06.2021

[Czermański u. a. 2021] Czermański, Ernest ; Cirella, Giuseppe T. ; Oniszczuk-Jastrząbek, Aneta ; Pawłowska, Barbara ; Notteboom, Theo: An Energy Consumption Approach to Estimate Air Emission Reductions in Container Shipping. In: *Energies* 14 (2021), Nr. 2. http://dx.doi.org/10.3390/en14020278. – DOI 10.3390/en14020278. – ISSN 1996–1073

[Danish Maritime Authority 2021] Danish Maritime Authority, DK: *AIS data*. https://www.dma.dk/SikkerhedTilSoes/Sejladsinformation/AIS/Sider/default.aspx, 2021. – zuletzt abgerufen: 22.06.2021

[Deng u. Yu 2014] Deng, Li ; Yu, Dong: Deep Learning: Methods and Applications / Microsoft. Version: May 2014. https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/. 2014 (MSR-TR-2014-21). – Forschungsbericht

[Dertat 2017] Dertat, Arden: *Applied Deep Learning - Part 4: Convolutional Neural Networks*. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2, 2017. – zuletzt abgerufen: 23.06.2021

[Doshi 2019] Doshi, Sanket: *Various Optimization Algorithms For Training Neural Network*. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6, 2019. – zuletzt abgerufen: 15.06.2021

[EMSA 2014] EMSA: *Vessel Tracking Gobally. Unterstanding LRIT*. Brochure, zuletzt abgerufen: 17.06.2021. http://www.emsa.europa.eu/lrit/items.html?cid=273&id=256. Version: 2014

[Foot Locker 2021] Foot Locker, Inc. (.: *Foot Locker, Inc. Report 220 Fourth Quarter And Full Year Results*. zuletzt abgerufen: 20.06.2021. https://www.prnewswire.com/news-releases/foot-locker-inc-reports-2020-fourth-quarter-and-full-year-results-301236242.html. Version: 2 2021. – PR Newswire

[Gamboa 2017] Gamboa, John Cristian B.: *Deep Learning for Time-Series Analysis.* 2017

[Goodfellow u. a. 2016] *Kapitel* 6.5. In: Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning.* MIT Press, 2016. – `http://www.deeplearningbook.org`

[Google 2021a] Google: *Google Maps.* `https://goo.gl/maps/fnaPkKAWhCpFCUiGA`, 2021. – zuletzt abgerufen: 29.06.2021

[Google 2021b] Google: *Google Maps.* `https://www.google.de/maps`, 2021. – zuletzt abgerufen: 30.06.2021

[Gould u. Vrba 1982] Gould, S. ; Vrba, E.: Exaptation-a missing term in the science of form. In: *Paleobiology* 8 (1982), S. 4–15

[Gregor u. Hevner 2013] Gregor, Shirley ; Hevner, Alan R.: Positioning and Presenting Design Science Research for Maximum Impact. In: *MIS Quarterly* 3 (2013), Nr. 2, S. 337–355

[Haldar 2015] Haldar, Malay: *How much training data do you need?* `https://medium.com/@malay.haldar/how-much-training-data-do-you-need-da8ec091e956`, 2015. – zuletzt abgerufen: 21.06.2021

[Harland 1996] Harland, C.: Supply chain management, purchasing and supply management, logistics, vertical integration, materials management andsupply chain dynamics. In: *Blackwell Encyclopedic Dictionary of Operations Management*, 1996

[He u. a. 2015] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: *Deep Residual Learning for Image Recognition.* 2015

[Heilig u. a. 2019] Heilig, Leonard ; Stahlbock, Robert ; Voss, Stefan: *From Digitalization to Data-Driven Decision Making in Container Terminals.* 2019

[Hevner u. Chatterjee 2010] Hevner, Alan ; Chatterjee, Samir: *Design Research in Information Systems.* Springer Science + Business Media, LLC 2010, 2010. – ISBN 9781441956521

[Hevner 2007] Hevner, Alan R.: A Three Cycle View of Design Science Research. In: *Scandinavian Journal of Information Systems* 19 (2007), 1, Nr. 2, S. 87–92

[Hitchcock 1941] Hitchcock, Frank L.: The Distribution of a Product form Several Sources to Numerous Localities. In: *Journal of Mathematics and Physics* (1941), 4, S. 224–230. `http://dx.doi.org/10.1002/sapm1941201224`. – DOI 10.1002/sapm1941201224

[Incze 2019] Incze, Raul: *The Cost of Machine Learning Projects.* https://medium.com/cognifeed/the-cost-of-machine-learning-projects-7ca3aea03a5c, 2019. – zuletzt abgerufen: 22.06.2021

[Ismail Fawaz u. a. 2020] Ismail Fawaz, H. ; Lucas, B. ; Frestier, G. et a.: InceptionTime: Finding AlexNet for time series classification. In: *Data Mining and Knowledge Discovery* 34 (2020), Juni, 1936-1962. http://dx.doi.org/10.1007/s10618-020-00710-y. – DOI 10.1007/s10618–020–00710–y

[Jin u. Han 2010] Jin, Xin ; Han, Jiawei: *K-Means Clustering.* Boston, MA : Springer US, 2010. – 563–564 S. http://dx.doi.org/10.1007/978-0-387-30164-8_425. http://dx.doi.org/10.1007/978-0-387-30164-8_425. – ISBN 978–0–387–30164–8

[Kannan u. a. 2018] Kannan, Kalapriya ; Ananthanarayanan, Rema ; Mehta, Sameep: *What is my data worth? From data properties to data value.* 2018

[Keskar u. a. 2017] Keskar, Nitish S. ; Mudigere, Dheevatsa ; Nocedal, Jorge ; Smelyanskiy, Mikhail ; Tang, Ping Tak P.: *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.* 2017

[Khandelwal 2020] Khandelwal, Renu: *Convolutional Neural Network: Feature Map and Filter Visualization.* https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c, 2020. – zuletzt abgerufen: 22.06.2021

[Kingma u. Ba 2017] Kingma, Diederik P. ; Ba, Jimmy: *Adam: A Method for Stochastic Optimization.* 2017

[Korn u. Korn 2013] Korn, Granini A. ; Korn, Theresa M.: *Appendix B: B-9. Plane and Sperical Trigonometry: Formulas Expressed in Terms of the Haversine Function.* Mineola, New York : Dover Publications, 2013. – 892 S. https://books.google.de/books?id=A4XCAgAAQBAJ. – ISBN 978–0–486–41147–7

[Krizhevsky u. a. 2017] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Commun. ACM* 60 (2017), Mai, Nr. 6, 84–90. http://dx.doi.org/10.1145/3065386. – DOI 10.1145/3065386. – ISSN 0001–0782

[Kuntz 2020] Kuntz, Juan: *Markov chains revisited.* 2020

[Leonard 2021] Leonard, Matt: *Foot Locker cites port congestion for nearly 24% drop in inventory.* zuletzt abgerufen: 20.06.2021. https://www.supplychaindive.com/news/foot-locker-port-congestion-

inventory-retail-ocean-shipping/596270/. Version: 3 2021. – Industy Dive - Supply Chain Dive

[Lexico.com 2021a] *Kapitel* 0. In: LEXICO.COM: *Definition of Markov chain in English*. Oxford University Press, 2021. – https://www.lexico.com/en/definition/markov_chain

[Lexico.com 2021b] *Kapitel* 0. In: LEXICO.COM: *Meaning of demurrage in English*. Oxford University Press, 2021. – https://www.lexico.com/definition/demurrage

[Lim u. Zohren 2021] LIM, Bryan ; ZOHREN, Stefan: Time-series forecasting with deep learning: a survey. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379 (2021), Feb, Nr. 2194, 20200209. http://dx.doi.org/10.1098/rsta.2020.0209. – DOI 10.1098/rsta.2020.0209. – ISSN 1471–2962

[Liu u. a. 2014] LIU, James N. ; HU, Yan-Xing ; YOU, J. ; CHAN, P.: Deep neural network based feature representation for weather forecasting, 2014

[Luo u. a. 2017] LUO, Wenjie ; LI, Yujia ; URTASUN, Raquel ; ZEMEL, Richard: *Understanding the Effective Receptive Field in Deep Convolutional Neural Networks*. 2017

[MathVault 2021] MATHVAULT: *Mean Squared Error (MSE)*. https://mathvault.ca/hub/higher-math/math-symbols/probability-statistics-symbols/, 2021. – zuletzt abgerufen: 28.06.2021

[McLaren u. a. 2011] MCLAREN, Tim ; HEAD, Milena ; YUAN, Yufei ; CHAN, Yolande: A Multilevel Model for Measuring Fit Between A Firm's Competitive Strategies and Information Systems Capabilities. In: *MIS Quarterly* 35 (2011), 12, S. 909–929. http://dx.doi.org/10.2307/41409966. – DOI 10.2307/41409966

[Mitchell 1997] MITCHELL, T.: *Machine Learning*. McGraw-Hill Education Ltd, 1997. – ISBN 0070428077

[Mitsa 2019] MITSA, Theophano: *How Do You Know You Have Enough Training Data?* https://towardsdatascience.com/how-do-you-know-you-have-enough-training-data-ad9b1fd679ee, 2019. – zuletzt abgerufen: 21.06.2021

[Navingo 2016] NAVINGO: *Terminal Congestion Increased Pollution at POLB in 2015*. https://www.offshore-energy.biz/terminal-congestion-increased-pollution-at-polb-in-2015/, 2016. – zuletzt abgerufen: 18.06.2021

[Neupert u. a. 2021] Neupert, Titus ; Fischer, Mark H. ; Greplova, Eliska ; Choo, Kenny ; Denner, Michael: *Introduction to Machine Learning for the Sciences*. 2021

[Ocean Insights 2021a] Ocean Insights, GmbH: *Port Congestion. All you need to know About Port Congestion 2021*. https://www.ocean-insights.com/port-congestion/, 2021. – zuletzt abgerufen: 18.06.2021

[Ocean Insights 2021b] Ocean Insights, GmbH: *Proactively Managing Demurrage and Detention with Increased Visibility*. https://www.ocean-insights.com/demurrage-detention/, 2021. – zuletzt abgerufen: 18.06.2021

[O'Shea u. Nash 2015] O'Shea, Keiron ; Nash, Ryan: An Introduction to Convolutional Neural Networks. In: *ArXiv e-prints* (2015), 11

[Pang u. a. 2021] Pang, Guansong ; Shen, Chunhua ; Cao, Longbing ; Hengel, Anton Van D.: Deep Learning for Anomaly Detection. In: *ACM Computing Surveys* 54 (2021), Apr, Nr. 2, 1–38. http://dx.doi.org/10.1145/3439950. – DOI 10.1145/3439950. – ISSN 1557–7341

[Pani 2014] Pani, C.: Managing vessel arrival uncertainty in container terminals: a machine learning approach, 2014

[Pani u. a. 2013] Pani, C. ; Cannas, M. ; Fadda, P. ; Fancello, G. ; Frigau, L. ; Mola, F.: Delay prediction in container terminals: A comparison of machine learning methods, 2013

[Pani u. a. 2015] Pani, Claudia ; Vanelslander, Thierry ; Fancello, Gianfranco ; Cannas, Massimo: Prediction of late/early arrivals in container terminals - A qualitative approach. In: *European Journal of Transport and Infrastructure Research* 15 (2015), 09, S. 536–550. http://dx.doi.org/10.18757/ejtir.2015.15.4.3096. – DOI 10.18757/ejtir.2015.15.4.3096

[Park u. a. 2021] Park, Kikun ; Sim, Sunghyun ; Bae, Hyerim: Vessel estimated time of arrival prediction system based on a path-finding algorithm. In: *Maritime Transport Research* 2 (2021), 100012. http://dx.doi.org/https://doi.org/10.1016/j.martra.2021.100012. – DOI https://doi.org/10.1016/j.martra.2021.100012. – ISSN 2666–822X

[Peduzzi u. a. 1996] Peduzzi, Peter ; Concato, John ; Kemper, Elizabeth ; Holford, Theodore R. ; Reinstein, Alvan R.: A simulation study of the number of events per variable in logistic regression analysis. In: *Journal of Clinical Epidemiology* 49 (1996), Nr. 12, 1373-1379. http://dx.doi.org/https://doi.org/10.1016/S0895-4356(96)00236-3. – DOI https://doi.org/10.1016/S0895–4356(96)00236–3. – ISSN 0895–4356

[Priceonomics 2019] Priceonomics, Data S.: *Companies Collect a Lot of Data, But How Much Do Tey Actually Use?* https://priceonomics.com/companies-collect-a-lot-of-data-but-how-much-do/, 2019. – zuletzt abgerufen: 21.06.2021

[Qui u. a. 2016] Qui, Junfei ; Wu, Qihui ; Xu, Yuhua ; Feng, Shuo: A survey of machine learning for big data processing. In: *EURASIP Journal on Advances in Signal Processing* 2016 (2016), 05, S. 67–83. http://dx.doi.org/10.1186/s13634-016-0355-x. – DOI 10.1186/s13634–016–0355–x

[Roger u. a. 2020] Roger, Vincent ; Farinas, Jérôme ; Pinquier, Julien: *Deep Neural Networks for Automatic Speech Processing: A Survey from Large Corpora to Limited Data.* 2020

[Rosenblatt 1958] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological Review* 65 (1958), Nr. 6, S. 386–408. http://dx.doi.org/https://doi.org/10.1037/h0042519. – DOI https://doi.org/10.1037/h0042519

[Roy 2020] Roy, Baijayanta: *All about Feature Scaling.* https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35, 2020. – zuletzt abgerufen: 25.06.2021

[Ruder 2017] Ruder, Sebastian: *An overview of gradient descent optimization algorithms.* 2017

[Samuel 1959] Samuel, A. L.: Some Studies in Machine Learning Using the Game of Checkers. In: *IBM Journal of Research and Development* 3 (1959), Nr. 3, S. 210–229. http://dx.doi.org/10.1147/rd.33.0210. – DOI 10.1147/rd.33.0210

[Schölkopf u. a. 2021] Schölkopf, Bernhard ; Locatello, Francesco ; Bauer, Stefan ; Ke, Nan R. ; Kalchbrenner, Nal ; Goyal, Anirudh ; Bengio, Yoshua: *Towards Causal Representation Learning.* 2021

[Shao u. a. 2021] Shao, Wei ; Prabowo, Arian ; Zhao, Sichen ; Koniusz, Piotr ; Salim, Flora D.: *Predicting Flight Delay with Spatio-Temporal Trajectory Convolutional Network and Airport Situational Awareness Map.* 2021

[Shiani 2019] Shiani: *Port Congestion - An Industry threat.* Blog, zuletzt abgerufen: 20.06.2021. https://container-xchange.com/blog/port-congestion/. Version: 8 2019. – xChange Solutions GmbH

[Simon 1996] Simon, Herbert A.: *The Science of the Artifical, 3rd Edition.* MIT Press, 1996. – ISBN 9780262193740

[Simonyan u. Zisserman 2015] Simonyan, Karen ; Zisserman, Andrew: *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015

[Srivastava u. a. 2015] Srivastava, Rupesh K. ; Greff, Klaus ; Schmidhuber, Jürgen: *Highway Networks*. 2015

[Stokes 2011] Stokes, D.E.: *Pasteur's Quadrant: Basic Science and Technological Innovation*. Brookings Institution Press, 2011 https://books.google.de/books?id=TLKDbvJX86YC. – ISBN 9780815719076

[Stone 1974] Stone, M.: Cross-Validatory Choice and Assessment of Statistical Predictions. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36 (1974), Nr. 2, 111-133. http://dx.doi.org/https://doi.org/10.1111/j.2517-6161.1974.tb00994.x. – DOI https://doi.org/10.1111/j.2517–6161.1974.tb00994.x

[Sutton u. Barto 2020] Sutton, Richard S. ; Barto, Andrew G.: *Reinforcement Learning: An Introduction (2nd Edition)*. MIT Press, 2020 http://incompleteideas.net/book/RLbook2020.pdf. – ISBN 9780262039246

[Szegedy u. a. 2015] Szegedy, Christian ; Vanhoucke, Vincent ; Ioffe, Sergey ; Shlens, Jonathon ; Wojna, Zbigniew: *Rethinking the Inception Architecture for Computer Vision*. 2015

[Tensorflow 2021a] Tensorflow: *EarlyStopping*. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping, 2021. – zuletzt abgerufen: 29.06.2021

[Tensorflow 2021b] Tensorflow: *ReduceLROnPlateau*. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateaur, 2021. – zuletzt abgerufen: 29.06.2021

[Tensorflow 2021c] Tensorflow: *TimeseriesGenerator*. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator, 2021. – zuletzt abgerufen: 28.06.2021

[Trade Finance Global 2021a] Trade Finance Global, TFG: *Demurrage - What is Demurrage?* https://www.tradefinanceglobal.com/freight-forwarding/demurrage/, 2021. – zuletzt abgerufen: 18.06.2021

[Trade Finance Global 2021b] Trade Finance Global, TFG: *Demurrage versus Detention - What's the Difference? Shipping 101*. https://www.tradefinanceglobal.com/freight-forwarding/demurrage-versus-detention/, 2021. – zuletzt abgerufen: 18.06.2021

[UNCTAD 2020] UNCTAD: *World seaborne trade by types of cargo and by group of economies, annual.* https://unctadstat.unctad.org/wds/TableViewer/tableView.aspx?ReportId=32363, 2020. – zuletzt abgerufen: 06.05.2021

[UNCTAD 2021] UNCTAD: *Review of maritime transport.* https://unctad.org/topic/transport-and-trade-logistics/review-of-maritime-transport, 2021

[vesseltracker.com 2021] VESSELTRACKER.COM: *Woodmac Vessertracker™ Data Services.* https://www.vesseltracker.com/de/products/dataServices.html, 2021. – zuletzt abgerufen: 17.06.2021

[VT Explorer 2021a] VT EXPLORER, Ltd.: *AIS Navigational Status.* https://api.vtexplorer.com/docs/ref-navstat.html, 2021. – zuletzt abgerufen: 25.06.2021

[VT Explorer 2021b] VT EXPLORER, Ltd.: *AIS Ship Type.* https://api.vtexplorer.com/docs/ref-aistypes.html, 2021. – zuletzt abgerufen: 25.06.2021

[Walls u. a. 1992] WALLS, J.G. ; G., Joseph ; WIDMEYER ; R., George ; EL SAWY, Omar A. ; A., Omar: Building an Information System Design Theory for Vigilant EIS. In: *Information Systems Research* 3 (1992), 03, S. 36–59. http://dx.doi.org/10.1287/isre.3.1.36. – DOI 10.1287/isre.3.1.36

[Wang u. a. 2019] WANG, Senzhang ; CAO, Jiannong ; YU, Philip S.: *Deep Learning for Spatio-Temporal Data Mining: A Survey.* 2019

[Willmott u. Matsuura 2005] WILLMOTT, Cort J. ; MATSUURA, Kenji: Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. In: *Climate Research* 30 (2005), 79-82. http://dx.doi.org/https://doi.org/10.3354/cr030079. – DOI https://doi.org/10.3354/cr030079

[Zhuang u. a. 2019] ZHUANG, Fuzhen ; QI, Zhiyuan ; DUAN, Keyu ; XI, Dongbo ; ZHU, Yongchun ; ZHU, Hengshu ; XIONG, Hui ; HE, Qing: A Comprehensive Survey on Transfer Learning. In: *CoRR* abs/1911.02685 (2019). http://arxiv.org/abs/1911.02685