Hochschule Esslingen
University of Applied Sciences

Bachelorarbeit

Analyse und Gegenüberstellung von geeigneten Evaluierungstechniken zur Validierung von generativen Algorithmen im Kontext der Bereitstellung von zeitreihen-bezogenen Daten

im Studiengang Technische Informatik
der Fakultät Informationstechnik
im 7. Semester

Roberto Corlito

Zeitraum: 01.09.2020 - 28.02.2020
Prüfer: Prof. Dr.-Ing. Hermann Kull
Zweitprüfer: M. Sc. Nico Schick

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Esslingen, den 9. März 2021 _____

Unterschrift

Zitat

„Die Macht der künstlichen Intelligenz ist so unglaublich, dass sie die Gesellschaft auf tiefgehende Weise verändern wird.“

Bill Gates

Vorwort

Die vorliegende Bachelorarbeit befasst sich mit der *Analyse und Gegenüberstellung von geeigneten Evaluierungstechniken zur Validierung von generativen Algorithmen im Kontext der Bereitstellung von zeitreihen-bezogenen Daten.* Hierbei wurden verschiedenste Evaluierungstechniken analysiert, um deren Eignung zur Validierung von zeitreihen-bezogenen generativen Algorithmen zu bestimmen.

Diese Bachelorarbeit bildet die Abschlussarbeit meines Studiums der Technischen Informatik an der Hochschule Esslingen. Ziel war es, Evaluierungstechniken zu bestimmen, mit denen synthetische Datensätze von sicherheitskritischen Fahrszenarien untersucht werden können. Von September 2020 bis Februar 2021 beschäftigte ich mich intensiv mit der Forschung und dem Schreiben der Bachelorarbeit.

An dieser Stelle möchte ich mich bei all denjenigen bedanken, dich mich während der Erstellung meiner Bachelorarbeit unterstützt haben. Ganz besonders möchte ich M. Sc. Nico Schick für seine intensive Betreuung bedanken. Durch seine Fachkenntnisse im Bereich des autonomen Fahrens sowie der Künstlichen Intelligenz verschaffte er mir wertvolle Einblicke in diese Themengebiete. Ebenso konnte ich, durch seinen Input, meine methodische Vorgehensweise verbessern und meine Bachelorarbeit erfolgreich durchführen. Ich bedanke mich ebenfalls bei Prof. Dr.-Ing. Hermann Kull für seine stets sehr wertvolle Betreuung. Durch seine Anleitung und Erfahrungen habe ich, während dieser Bachelorarbeit, vieles dazu gelernt.

Roberto Corlito

Esslingen, 9. März 2021

Kurz-Zusammenfassung

In den letzten Jahren war ein verstärkter Anstieg an Automatisierung in Fahrzeugen zu verzeichnen. Diese wird durch immer komplexere und intelligentere Assistenzsysteme - oder auch Advanced Driver Assistance Systems - ermöglicht. Zu diesen zählen beispielsweise der Notbremsassistent, durch den die Fahrsicherheit erhöht wird. Hierzu ist eine steigende Anzahl an Sensoren und eine hohe Menge an entsprechenden Daten notwendig. Diese werden von intelligenten System- und Softwarekomponenten verarbeitet. In diesem Kontext werden vermehrt Methoden aus dem Bereich der künstlichen Intelligenz verwendet.

Bei vollständiger Automatisierung wird vom autonomen Fahren gesprochen. Hierbei werden alle Fahraufgaben vom Fahrzeug übernommen. Das autonome Fahren ist inzwischen eine Schlüsseldisziplin des Automobilbereichs, die ein großes Potential aufweist. Hierbei muss die Sicherheit aller Verkehrsteilnehmer gewährleistet werden. Um diese sicherzustellen, müssen die entsprechenden Komponenten ausgiebig validiert und getestet werden. Hierzu sind eine große Menge an Daten für verschiedenste Fahrsituationen erforderlich, um autonome Fahrzeuge auf diese vorzubereiten. Für sicherheitskritische Fahrszenarien sind diese, aufgrund ihrer Kritikalität, relativ selten vorhanden. Hierbei können generative Algorithmen verwendet werden, um zeitreihen-bezogene synthetische Bewegungsdaten, für diese Szenarien, zu generieren.

Die vorliegende Bachelorarbeit behandelt die *Analyse und Gegenüberstellung von geeigneten Evaluierungstechniken zur Validierung von generativen Algorithmen im Kontext der Bereitstellung von zeitreihen-bezogenen Daten*. Hierfür wurde, basierend auf einer umfangreichen Literaturrecherche, eine neuartige Taxonomie für Wahrscheinlichkeitsdichtefunktionen von generativen Algorithmen erstellt. Schließlich werden generative Algorithmen, auf quantitativer Weise, typischerweise validiert, indem die dazugehörigen Wahrscheinlichkeitsdichtefunktionen herangezogen werden.

Hierbei werden anwendungsrelevante Evaluierungstechniken detailliert beschrieben. Anschließend erfolgt, anhand ausgewählter Bewertungskriterien, eine Gegenüberstellung der hoch relevanten Evaluierungstechniken. Darüber hinaus wird eine empfohlene Evaluierungstechnik implementiert und auf synthetische Modelldaten angewandt sowie validiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Im Folgenden wird die Thematik dieser Bachelorarbeit näher beschrieben. Es erfolgt eine kurze Einführung in die Problemstellung und der daraus folgenden Motivation dieser Arbeit. Ebenso wird die genaue Aufgabenstellung definiert.

1.1 Motivation

Täglich sterben ca. 3700 Menschen bei Verkehrsunfällen [5]. Die häufigste Unfallursache ist hierbei menschliches Versagen [33]. Durch das autonome Fahren können Verkehrsunfälle auf ein Minimum reduziert werden. Die Technologien im Automobilbereich entwickeln sich stetig weiter. Derzeit wird verstärkt an der Elektromobilität, dem autonomen Fahren und einer höheren Vernetzung von Fahrzeugen, speziell zwischen Steuergeräten, geforscht. Die hohe Vernetzung ist notwendig, da sich die Anzahl an Sensoren und damit an Assistenzsystemen, wie der klassische Tempomat, erhöht. Mit der daraus folgenden höheren Datenmenge, können intelligentere Assistenzsysteme - oder auch Advanced Driver Assistance Systems - entwickelt werden. Diese können immer komplexere Aufgaben automatisieren. Zu diesen gehören beispielsweise der Notbremsassistent, mit dem die Sicherheit im Straßenverkehr erhöht werden kann. Bei vollständiger Automatisierung bzw. beim autonomen Fahren werden alle Aufgaben vom Fahrzeug übernommen. Um autonome Fahrzeuge auf den Straßenverkehr vorzubereiten, müssen Software- und Systemkomponenten effizient validiert und getestet werden. Hierbei stehen allerdings nur begrenzt Daten für sicherheitskritische Fahrszenarien, wie beispielsweise Überhol- oder Abbiegevorgänge, aufgrund ihrer Kritikalität, zur Verfügung. Mit Hilfe von generativen Algorithmen lassen sich synthetische sicherheitskritische Szenarien generieren. Die Herausforderung hierbei liegt in der Validierung dieser Daten.

1.2 Aufgabenstellung

Es gibt verschiedene Methoden, um zeitreihen-bezogene generative Algorithmen zu validieren. In dieser Arbeit werden speziell Evaluierungstechniken von Wahrscheinlichkeitsdichtefunktionen, die einen quantitativen Ansatz haben, analysiert. Hierbei soll eine neuartige Taxonomie von Evaluierungstechniken für zeitreihen-bezogene generative Algorithmen erstellt werden. Dabei sind die, für den zugrundeliegenden Anwendungsfall, relevanten Evaluierungstechniken, mathematisch zu beschrieben. Anschließend sollen diese, anhand ausgewählter Bewertungskriterien, gegenübergestellt werden. Das Ziel hierbei ist es, eine Empfehlung auszusprechen. Die empfohlene Evaluierungstechnik soll implementiert und anhand von synthetischen Datensätzen, die fahrkritische Szenarien darstellen, validiert werden.

2 Grundlagen

In diesem Kapitel werden elementare Begriffe der Bachelorarbeit erläutert, sowie ein Grundverständnis zum zugrundeliegenden Thema vermittelt. Nachfolgend werden die Themen beschrieben, die zum Kontext dieser Arbeit gehören.

2.1 Autonomes Fahren

Kraftfahrzeuge sind schon seit langem ein essentieller Bestandteil der Gesellschaft. In logistischen Prozessen und Lieferketten tragen diese eine entscheidende Rolle. Ebenso im privaten und öffentlichem Sektor, erfüllen sie einen wichtigen Zweck. Dabei waren stets Innovationen zu vermerken. Schließlich entwickeln sich die Technologien im Automobilbereich stetig weiter.

Darunter zählt ebenso das Autonome Fahren. Durch verschiedene Assistenzsysteme - oder auch Advanced Driver Assistance Systems - bieten autonome Fahrzeuge mehr Sicherheit. Zu diesen gehören beispielsweise der Stop & Go Pilot, Spurhalteassistent, Notbremsassistent oder die Einparkhilfe. Durch diese Fahrassistenzsysteme ist es möglich, dass z. B. ein Fahrzeug schon heute teilweise selbständig auf der Autobahn fahren kann. Bei vollständiger Automatisierung benötigt das Fahrzeug keinen Fahrer mehr. Dies wird als autonomes Fahren bezeichnet. Die verschiedenen Automatisierungsstufen, des autonomen Fahrens, sind in Abb. 2.1 aufgezeigt. Im Folgenden werden diese, auf Basis von [1], näher beschrieben.

Stand der Technik

Beim Level 1 des automatisierten Fahrens - oder auch assistiertem Fahren - hat der Fahrer durchgehend die alleinige Kontrolle über das Fahrzeug. Der Verkehr muss permanent im Blick behalten werden und die Haftung für Verkehrsverstöße oder Schäden liegt beim Fahrer. Assistenzsysteme, wie der Abstandsregeltempomat, der je nach Abstand zum vorausfahrenden Fahrzeug bremst oder beschleunigt, unterstützen heute schon den Fahrer bei entsprechenden Fahraufgaben.

Bei Level 2-Fahrzeugen können, unter definierten Bedingungen, spezielle Aufgaben vom Fahrzeug selbst ausgeführt werden. So kann, ohne das Einwirken des Fahrers, beispielsweise auf der Autobahn die Spur gehalten, gebremst und beschleunigt werden. Dies wird, durch die Kombination von verschiedenen Assistenzsystemen, wie dem automatischen Abstandsregelautomat und dem Spurhalteassistent, möglich. Im Vergleich zu Level 1, kann der Fahrer, während des teilautomatisierten Modus, die Hände kurz vom Steuer nehmen. Dadurch werden Funktionen, wie der Überholassistent oder das automatische Einparken, ermöglicht. Dennoch muss der Fahrer, zu jeder Zeit, die Assistenzsysteme überwachen und bei Fehlfunktionen eingreifen können, um das Fehlverhalten dann zu kompensieren.

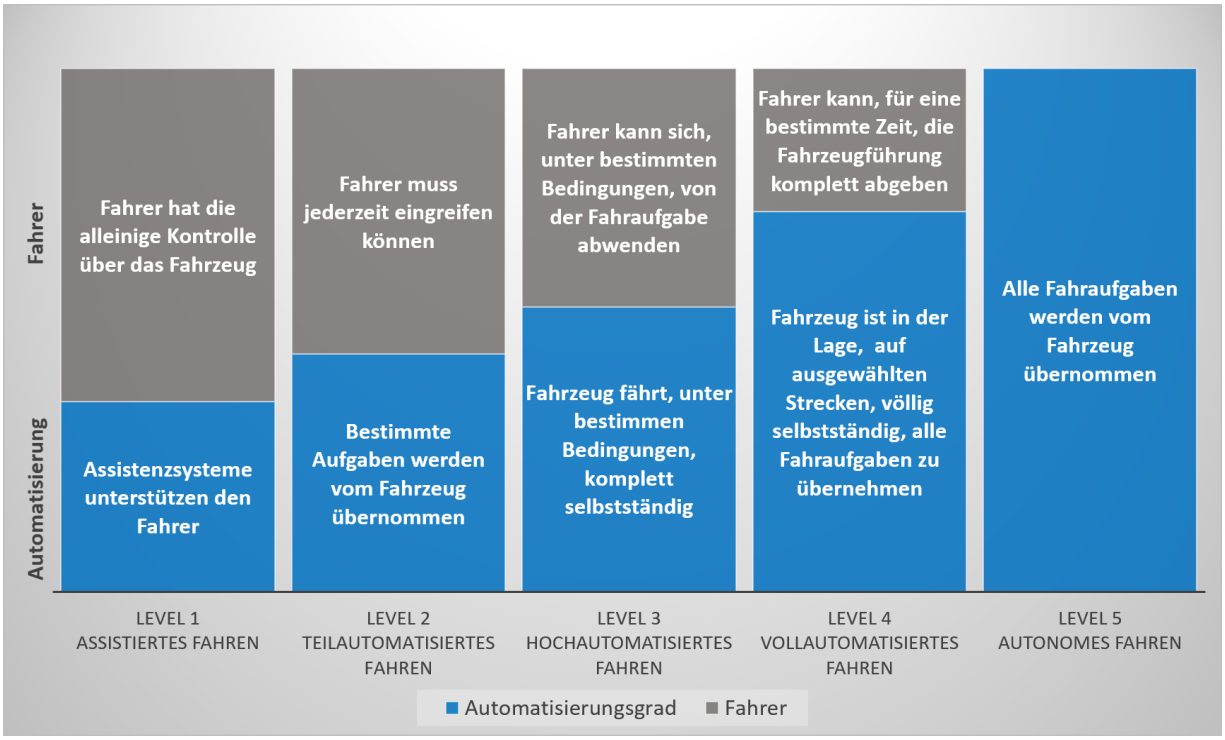

Abb. 2.1: Automatisierungsstufen des autonomen Fahrens

Schließlich liegt, im Falle eines Unfalls, die Verantwortung weiterhin bei ihm.

Bei hochautomatisierten oder Level 3-Fahrzeugen darf sich der Fahrer, bei Hersteller-vorgegebenen Anwendungsfällen, von der Fahraufgabe abwenden. Das Fahrzeug fährt, für diesen begrenzten Zeitraum, selbstständig.

Beim vollautomatisiertem oder Level 4-Fahren kann der Fahrer die Fahrzeugführung komplett abgeben. Bei Verkehrsverstößen, während der vollautomatisierten Fahrt, ist der Fahrer nicht haftbar. Das Fahrzeug ist in der Lage, auf bestimmten Strecken, völlig selbstständig, alle Fahraufgaben für eine längere Zeit, zu übernehmen. Die Insassen können währenddessen, etwaigen Nebentätigkeiten nachgehen. Das Fahrzeug kann allerdings auch ohne Insassen, entsprechende Fahraufgaben übernehmen. So können Level 4-Fahrzeuge in Parkhäusern automatisiert einen Parkplatz suchen oder ebenso bei hohen Geschwindigkeiten auf die Autobahn auffahren, sich in den Verkehr einordnen und eigenständig die Spur und den Abstand halten. Wenn das System während des vollautomatisierten Modus einen Fehler entdeckt, können die Passagiere entweder wieder die Kontrolle übernehmen oder das Fahrzeug begibt sich in einen sicheren Zustand.

Das Level 5 oder autonome Fahren ist die letzte Stufe der Automatisierung bei Fahrzeugen. Es gibt keinen Fahrer mehr, sondern nur noch Passagiere, die nicht mehr für Verkehrsverstöße oder Schäden haften. Die entsprechenden Hersteller, Halter oder Betreiber wären diesbezüglich haftbar beziehungsweise eine Versicherung müsste für eventuelle Schäden aufkommen.

Neben der Einstufung in 5 Level, gibt es noch ein Konzept mit drei Betriebsmodi.

Der erste Betriebsmodus ist das unterstützende Fahren. In diesem muss der Fahrer den Verkehr stets im Blick behalten und ist alleine für die Fahrzeugführung verantwortlich. Fahrassistenzsysteme unterstützen den Fahrer nur. Bei einem Fehlverhalten oder Versagen dieser, haftet der Fahrer für Verkehrsverstöße oder Unfälle. Das automatisierte Fahren stellt den zweiten Betriebsmodus dar. Hierbei fährt das Fahrzeug, ausschließlich in vom Hersteller definierten Anwendungsfällen, selbstständig. Der Fahrer darf währenddessen eine Nebentätigkeit ausüben. Bei Aufforderung des Systems, muss dieser aber kurzfristig die Kontrolle übernehmen, da er sonst haftet. Im dritten Betriebsmodus, dem autonomen Fahren, fährt das Fahrzeug komplett selbstständig. Das System beherrscht kritische Fahrsituationen. Bei etwaigen Betriebsstörungen oder Fehlverhalten, muss ein Betreiber auf diese reagieren können. Dieser überwacht das autonome Fahrzeug nur, ist allerdings nicht selber Fahrer. Die Passagiere haften in diesem Fall nicht.

Ein autonomes Fahrzeug muss seine Umgebung erfassen und die daraus gewonnen Informationen verarbeiten können. Hierzu werden verschiedenste Sensoren verwendet, die in Abb. 2.2 dargestellt sind. Hierbei haben Ultraschallsensoren und Kurzstrecken-Radar eine kleine Reichweite und werden zur Hinderniserkennung sowie zum Parken eingesetzt. Langstrecken-Radar können Objekte in großer Entfernung sowie ihre Position und Geschwindigkeit bestimmen. LIDAR- und Laser-Sensoren erstellen, mit Hilfe von Laser-Licht, ein hochauflösendes 3D-Abbild der Umgebung. Zusätzlich liefern Kamerasysteme, optische Informationen über Textur und Farbe eines Objektes. Infrarot-Sensoren sind vor allem für die Nachtsicht relevant. Durch die Kombination der verschiedenen Sensor-Daten ist eine nahezu exakte Erfassung des Umfelds möglich. Diese Umgebungsdaten können, durch die Informationen von anderen Fahrzeugen sowie der Verkehrs-Infrastruktur, ergänzt werden. Intelligente Softwarealgorithmen verarbeiten, mit Hilfe von leistungsstarken Microcontrollern und Prozessoren, die verschiedenen Daten, um das Fahrzeug autonom Fahren zu lassen. [20]

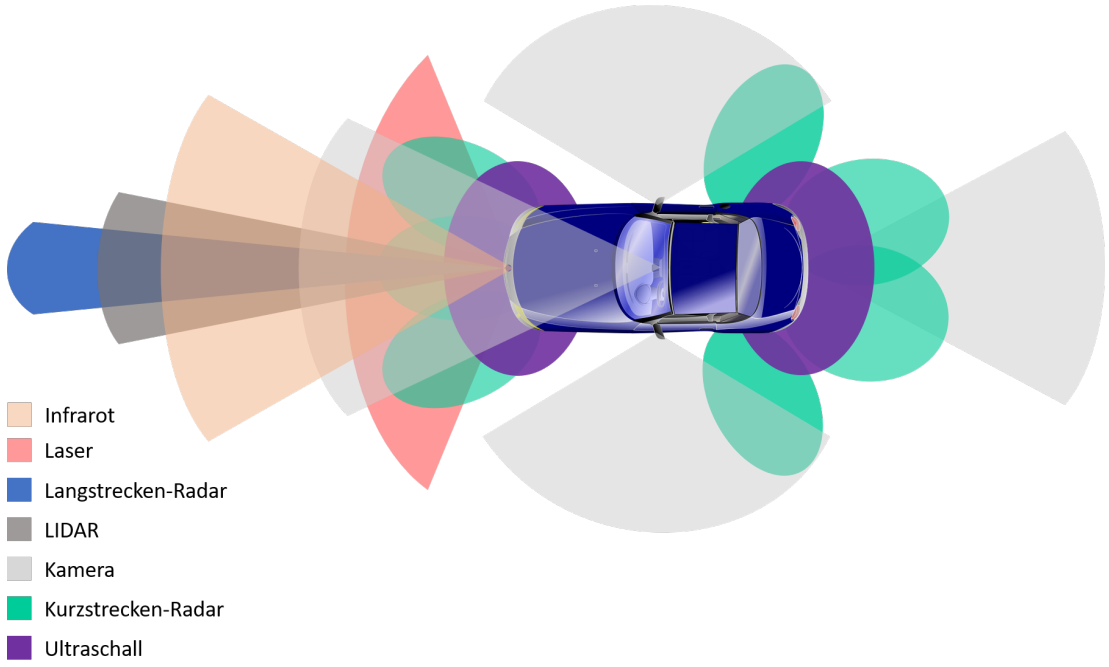

Abb. 2.2: Sensoren in einem Fahrzeug

Motivation und Vorteile

Da bei autonomen Fahrzeugen kein Fahrer mehr notwendig ist, ergeben sich dadurch verschiedene Chancen und Vorteile in den Bereichen Gesellschaft, Wirtschaft und Sicherheit. Durch das Wegfallen eines Fahrers, haben die Passagiere, die gewonnene Fahrtzeit, für andere Aktivitäten zur Verfügung. Ebenso können durch das autonome Fahren Mobilitätseingeschränkte, wie Personen im hohen Alter, mit Behinderungen oder ohne Fahrerlaubnis, durch eine erhöhte Mobilität verstärkt am gesellschaftlichen Leben und Straßenverkehr teilnehmen. [21]

Durch diese erhöhte Nutzer- und Kundengruppen, ergeben sich neue Möglichkeiten für Hersteller und Betreiber, diese durch neue Mobilitätskonzepte, zu erschließen. Eines dieser Konzepte, nämlich das Car-Sharing, gibt es bereits heute und kann auf das autonome Fahren angepasst werden. So könnte sich ein autonomes Fahrzeug selbstständig, vom Nutzer aus, einen Parkplatz suchen und umgekehrt. Alternativ wäre es ebenso denkbar, dass sich das Fahrzeug eigenständig zum gewünschten Zielort des nächsten Nutzers bewegt. [21]

Hinzu kommen, durch das Wegfallen des Fahrers, bei autonomen öffentlichen Verkehrsmitteln, eine höhere Flexibilisierung und Bedarfsorientierung in Bezug auf Einsatzzeit und -gebiet. Durch eine hohe Anzahl an Sensoren, die verschiedene Daten erfassen, ergibt sich eine große Menge an Informationen, wodurch autonome Fahrzeuge präziser durch den Straßenverkehr manövrieren können. Dadurch ergibt sich eine Reduzierung der benötigten Verkehrs- und Parkflächen, insbesondere in Stadtzentren. [21]

Ebenso werden die Kapazitäten im Straßenverkehr effizienter genutzt, da die benötigte Zeit, beispielsweise beim Abbiegevorgang, an Lichtanlagen sowie die Reaktionszeit verringert werden kann. Es werden, im Allgemeinen, höhere Geschwindigkeiten im Straßenverkehr ermöglicht. Aufgrund dieser Aspekte, ergeben sich entsprechende Rationalisierungseffekte. Das Wegfallen eines Fahrers führt beispielsweise im Mobilitäts- und Transportsektor zu Kosteneinsparungen und somit zu Effizienzsteigerungen. Dadurch werden ebenso die Umweltauswirkungen im Straßenverkehr verringert, da Fahrvorgänge optimiert werden und somit der Energie- und Emissionsausstoß sinkt. [21]

In Deutschland waren in 2017 91% der Verkehrsunfälle auf menschliches Versagen zurückzuführen [33]. Dies entsprach der Hauptunfallursache im Straßenverkehr. Bei autonomen Fahrzeugen kann diese als Unfallursache ausgeschlossen werden, welches zu einer Erhöhung der Verkehrssicherheit führt. Hierfür müssen allerdings autonome Fahrzeuge komplexe Verkehrssituationen und fahrkritische Szenarien wahrnehmen und interpretieren können.

Blick in die Zukunft

Derzeit befinden sich auf den deutschen Straßen höchstens Level 2-Fahrzeuge. Diese sind mit Assistenzsystemen ausgerüstet, welche es dem Fahrer erlauben, kurzfristig die Kontrolle über die Lenkung abzugeben. Laut einer Studie von deloitte möchten aktuell 90% der befragten Fahrer aber ebenso jederzeit die Kontrolle über ihr Fahrzeug übernehmen können [13]. Das zeigt auf, dass die Akzeptanz und das Vertrauen in autonome Fahrzeuge noch stark verbessert werden muss, bevor sich diese im deutschen Straßenverkehr durchsetzen können.

Die Sicherheit und Verlässlichkeit des Systems sind hierbei die kritischen Faktoren.

Demgegenüber stehen Aspekte, wie der erhöhte Komfort und die Erschließung neuer Mobilitätskonzepte. Durch autonome Fahrzeuge können Flottenbetreiber individuellere Mobilitäten ermöglichen, wodurch sich ebenso der Trend vom Besitz zum Car-Sharing verstärken könnte. Viele weitere Faktoren und Rahmenbedingungen müssen jedoch erst noch geklärt werden, damit sich die Akzeptanz für das autonome Fahren erhöht und dieses realisierbar wird. Durch diese Herausforderungen ist eine Einführung von Level 4- oder Level 5-Fahrzeugen, voraussichtlich, nicht vor 2040 zu erwarten. [2]

Herausforderungen

Bevor sich autonome Fahrzeuge im Straßenverkehr durchsetzen, gibt es einige Herausforderungen, die vorher zu bewältigen sind. Die Wichtigste stellt hierbei das System für autonome Fahrzeuge an sich dar. Diese müssen im dynamischen Straßenverkehr mit jeder Situation umgehen können. Dazu muss die gesamte Umgebung in Echtzeit erfasst und die Fahrsituation interpretiert werden. Die gesammelten Daten dienen als Grundlage für die Bestimmung der eigenen Fahrzeugbewegung sowie für Prognosen über die Bewegung der weiteren Verkehrsteilnehmer. Diese Anforderungen muss das System, ebenso in komplexen Situationen, regelkonform bewältigen, um die Sicherheit der Insassen und anderer Verkehrsteilnehmer zu gewährleisten. [21]

Zusätzlich muss dieses Systemfehler erkennen und sich bei Bedarf in einen sicheren Zustand begeben können. Um so ein System zu gewährleisten, werden mehr Informationen als bei konventionellen Fahrzeugen, benötigt. Hierzu muss die entsprechende Infrastruktur noch entwickelt bzw. ausgebaut werden, damit die schnelle und fehlerfreie Fahrzeugkommunikation, unter Fahrzeugen, sichergestellt werden kann. Die erhöhte Anzahl an Kommunikationsschnittstellen stellen allerdings auch steigende Angriffsvektoren dar. Personenbezogene Daten wie die Adresse, Fahrtziele oder audiovisuelle Aufzeichnungen müssen sicher und anonymisiert gespeichert werden. Außerdem wächst das Risiko von Cyberangriffen auf die digitale Infrastruktur und autonome Fahrzeuge. Dies zählt zum Bereich Security. [3]

Der langsame Wandel des Straßenverkehrs und der Infrastruktur wird die Einführung des Mischverkehrs, von konventionellen und autonomen Fahrzeugen, unvermeidbar machen, wodurch viele ethische Fragen im Bezug auf die Haftung geklärt werden müssen. Ein zentrales Problem bei autonomen Fahrzeugen, ist das ethisch korrekte Verhalten im Straßenverkehr. Eine hohe Anzahl an Faktoren nehmen auf die Algorithmen autonomer Fahrzeuge Einfluss, wodurch Dilemma-Situationen entstehen können. Eine solche Situation kann auftreten, wenn z. B. eine Person eine rote Ampel überquert. Ein autonomes Fahrzeug, dass durch eine Notbremsung nicht mehr vor dem Fußgänger zum Halten kommen kann, wird hier versuchen, ein Ausweichmanöver durchzuführen. Wenn sich auf der Gegenfahrbahn Fahrzeuge mit weiteren Insassen befinden, bleibt noch die Option auf den Gehweg auszuweichen. Angenommen, dass sich auf diesem auch Personen befinden, kann das autonome Fahrzeug einen Personenschaden nicht verhindern. Wer in solchen Fällen haftet und wie in diesen und weiteren fahrkritischen Szenarien entschieden werden soll, ist ein Kernaspekt, der noch zu klären ist. Um die Akzeptanz für das autonome Fahren, als Technologie bzw. Mobilitätskonzept, zu stärken, ist es essentiell, diese Herausforderungen zu lösen. [7]

2.2 Künstliche Intelligenz und Maschinelles Lernen

Der Begriff künstliche Intelligenz wird in verschiedenen Anwendungsbereichen genannt. Dieser ist allerdings sehr abstrakt. Für die allgemeine Bevölkerung ist es unklar, welches Problem gelöst werden soll und welche Lösungsalgorithmen eingesetzt werden. Der Grund hierfür liegt zum Teil in der Komplexität dieser Algorithmen. Im Folgenden wird diese Thematik näher betrachtet und erläutert.

Definition von künstlicher Intelligenz

Die Informatik enthält viele Teilgebiete. Der Bereich Data Science zählt dazu. Dieses Gebiet befasst sich damit, Erkenntnisse aus großen Datensätzen, zu gewinnen. Hierzu werden meist statistische Methoden angewandt. Ein spezieller Bereich dieses Themenbereichs ist die künstliche Intelligenz. Hierbei wird das kognitive Verhalten des Menschen imitiert. Ein spezieller Fall von künstlicher Intelligenz ist das Machine Learning. Dabei wird, anders als in anderen Teilgebieten der Informatik, kein Lösungsalgorithmus vorgegeben. Die Algorithmen erkennen, durch wiederholtes Ausführen, selbstständig Strukturen und Muster in den Daten und können so spezielle Aufgaben lösen. Das maschinelle Lernen kann in drei Arten eingeteilt werden. Dazu zählen das überwachte (supervised), unbewachte (unsupervised) und bestärkende (reinforcement) Lernen. [12]

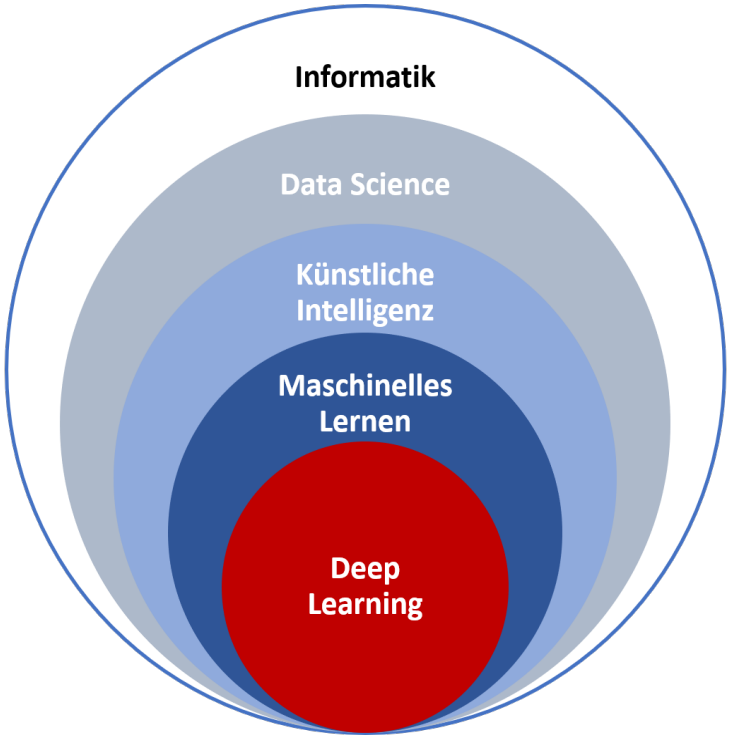

Abb. 2.3: Teilgebiete der Informatik

Supervised Learning

Das Supervised Learning zeichnet sich dadurch aus, dass bereits ein kategorisierter Datensatz vorliegt und so Machine Learning-Algorithmen direktes Feedback erhalten. Es besteht grundsätzlich die Aufgabe, einen gewissen Output zu prognostizieren. Wenn der Input einer bestimmten Klasse zugehörig ist, welche bestimmt werden soll, handelt es sich um eine Klassifizierung. Der Begriff überwachtes Lernen kommt daher, dass die Trainingsdatensätze, mit dem das Modell trainiert wird, bereits Labels enthalten, also in Klassen kategorisiert sind. Ein Beispiel für solche Labels können beispielsweise Buchstaben oder Ziffern sein. Das Modell kann allerdings nur diejenigen Labels bestimmen, die ebenso im Trainingsdatensatz vorhanden sind. Eine sogenannte Regressions-Aufgabe ist gegeben, falls die Ausgabe eines Machine Learning-Modells, die Prognose eines kontinuierlichen Wertes entspricht. Durch diese kontinuierlichen Modelle, lassen sich beispielsweise Vorhersagen über die Zukunft treffen. In beiden Fällen ist das Ziel, ein Modell zu erlernen, welches Prognosen über neue Daten bereitstellt. [29]

Unsupervised Learning

Im Gegensatz zum Supervised Learning, steht beim Unsupervised Learning kein Datensatz, mit bereits bekannten Labels, zur Verfügung.
Diese Modelle und Algorithmen finden eigenständig Muster in Datensätzen und deren Eigenschaften. Eine Kategorie des Unsupervised Learning ist die Cluster-Analyse. Ziel dieser ist es, versteckte Strukturen in Datensätzen zu finden oder diese passend in Clustern zu gruppieren. Ein weiterer Anwendungsfall für das Unsupervised Learning, ist die Sprachverarbeitung oder auch Natural Language Processing. Hierbei wird die Sprache, in ihren verschiedenen Formen, analysiert. Die Sentimentanalyse, also die Analyse der menschlichen Stimmung und das Nutzerverhalten, ist ein Teilgebiet dieser Sprachverarbeitung. Generative Algorithmen gehören ebenso zum Bereich des Unsupervised Learning. Auf diesen Algorithmentyp wird später näher eingegangen. [29]

ReinforcementLearning

Eine weitere Art des maschinellen Lernens ist das Reinforcement Learning. Hierbei werden sogenannte Agenten analysiert, die durch Belohnungen oder Bestrafungen ihr Ziel, in einer bestimmten Umgebung oder Aufgabenstellung, optimieren. [29]

Neuronale Netze

Ein bekanntes Beispiel für ein Machine Learning-Modell bilden neuronale Netze (*engl.: Neural Networks*). Diese sind inspiriert durch den Aufbau der Nervenzellen im menschlichen Gehirn. Wie im menschlichen Gehirn, werden hier künstliche Neuronen bzw. Synapsen, in Form von Reihen aus Datenknoten, miteinander verbunden. Weiterhin werden diese Verbindungen gewichtet. Bei jedem Lerndurchlauf der Daten werden diese Gewichte angepasst, bis das Modell zufriedenstellend arbeitet. Sobald das neuronale Netz, durch den Lernvorgang, die einzelnen Gewichte gelernt hat, kann es ebenso auf neue Daten angewandt werden.

Falls das Netz versteckte Schichten enthält, die nicht direkt mit einer Ein- oder Ausgabeschicht verbunden sind, so handelt es sich um ein tiefes neuronales Netz (*engl.: Deep Neural Network*). Umso mehr Schichten ein Deep Neural Network aufweist, desto komplexere Muster kann es erkennen bzw. erlernen, sowie anspruchsvollere Aufgaben lösen. [29]

Anwendungen

Im Bereich der künstlichen Intelligenz hat es, in den letzten Jahren, enorme Fortschritte gegeben. Dadurch wurden neue Möglichkeiten und Anwendungsgebiete erschlossen. Mithilfe von Machine Vision-Algorithmen können Bilder analysiert und kategorisiert werden. Die extrahierten Informationen bilden die Basis für weitere Analysen. Hierfür finden sich beispielsweise Einsatzgebiete in der medizinischen Diagnostik, sowie bei der Erkennung von Personen und Gesichtern. Ein weiteres Einsatzgebiet, für das maschinelle Lernen, ist die Mustererkennung. Große Datenmengen oder -ströme sind für den Menschen schwer zu interpretieren, wohingegen Machine Learning-Algorithmen leichter diese Muster erkennen und dem Menschen Erkenntnisse liefern können. Mithilfe von künstlicher Intelligenz und den einzelnen Teildisziplinen lassen sich einige, der in Kap. 2.1 genannten Herausforderungen und Anforderungen, lösen. Durch die Bilderkennung lassen sich beispielsweise Personen und Fahrzeuge im Straßenverkehr von autonomen Fahrzeugen identifizieren, wodurch diese ihre entsprechende Fahrtrajektorie planen können. Damit kann die Sicherheit, aller am Straßenverkehr Beteiligten, sichergestellt werden. [16]

2.3 Generative Algorithmen

Die Modelle der künstlichen Intelligenz, die in Kap. 2.2 unter Supervised Learning genannt wurden, werden als diskriminative Algorithmen bezeichnet. So lassen sich, bei neuem Input, Prognosen darüber machen, zu welcher Klasse dieser wahrscheinlich gehört. Anders ausgedrückt, wird also die Wahrscheinlichkeit $p(y|x)$ geschätzt, dass ein Input x der Klasse y zugehörig ist. In den letzten Jahrzehnten wurden vorwiegend in den Bereichen von diskriminativen Algorithmen, speziell in der Klassifizierung, Fortschritte erzielt.

In diesem Kapitel sollen nun generative Algorithmen näher betrachtet werden. Erst in den letzten 5 bis 10 Jahren ergab sich eine stärkere Präsenz der generativen Algorithmen in der Praxis. Als wesentliche Grundlage diente hierbei die Funktionsweise eines sogenannten Autoencoders. Hierbei ermöglicht ein Encoder, hochdimensionale Merkmalsvektoren in einen niederdimensionalen latenten Raum, zu komprimieren. Basierend auf dem latenten Raum kann ein Decoder, mithilfe einer Transformation, den Originaldatensatz rückgewinnen. Ein solcher latenter Raum erzielt ebenso eine höhere Transparenz der wesentlichen Daten. Durch die Verwendung einer Verteilungsfunktion über diesem latenten Raum, erhält der Output eine Varianz, ist aber trotzdem ähnlich wie der Input. Bei der generativen Modellierung sind normalerweise ungelabelte Datensätze die Basis für die Algorithmen. Es handelt sich also um eine Form des Unsupervised Learning. Hierbei wird $p(x)$, also die Wahrscheinlichkeit x zu beobachten, geschätzt. Allerdings gibt es ebenso generative Modelle, die eine Mischform aus Supervised und Unsupervised Learning darstellen, falls ein gelabelter Datensatz vorliegt. In diesem Fall wird die Verteilung $p(x|y)$ geschätzt, also die Wahrscheinlichkeit x, bei gegebenen Labels y, zu beobachten. Bei generativen Algorithmen ist demnach die zugrundeliegende Verteilung des Datensatzes von Interesse. [15]

Durch den Fokus auf eine derartige Verteilung der Daten ist es möglich, neue synthetische Daten zu generieren. Dabei ist davon auszugehen, dass ein Datensatz einer bestimmten Verteilung p_{data} zugrunde liegt. Mit einem generativen Modell wird versucht, diese mit einer neuen Verteilung p_{model} nachzubilden oder abzuschätzen. Dieser Vorgang gilt als erfolgreich, sofern durch p_{model} Daten generiert werden können, die ähnlich wie die zugrundeliegenden Daten sind, sich allerdings so weit davon unterscheiden, dass sie nicht nur Reproduktionen oder aufgeprägtes Rauschen abbilden.[15]

Taxonomie von generativen Algorithmen

Die grundlegenden generativen Algorithmentypen lassen sich, im Rahmen einer allgemeingültigen Taxonomie, gruppieren. Eine derartige Taxonomie ist in Abb. 2.4 dargestellt [18].

Hierbei werden alle Algorithmen von der Maximum Likelihood Estimation abgeleitet. Eine erste Gruppierung erfolgt in Bezug auf die Wahrscheinlichkeitsdichtefunktion der zugrundeliegenden Datenmenge. Zum einen kann diese explizit bzw. als geschlossener Ausdruck angegeben sein. Zum anderen kann diese implizit bzw. während des Trainingsvorgangs, geschätzt werden.

Die expliziten Modelle können weiter in die Tractable Density und die Approximate Density unterteilt werden. Die Tractable Density Modelle können hierbei in Polynomialzeit berechnet werden.

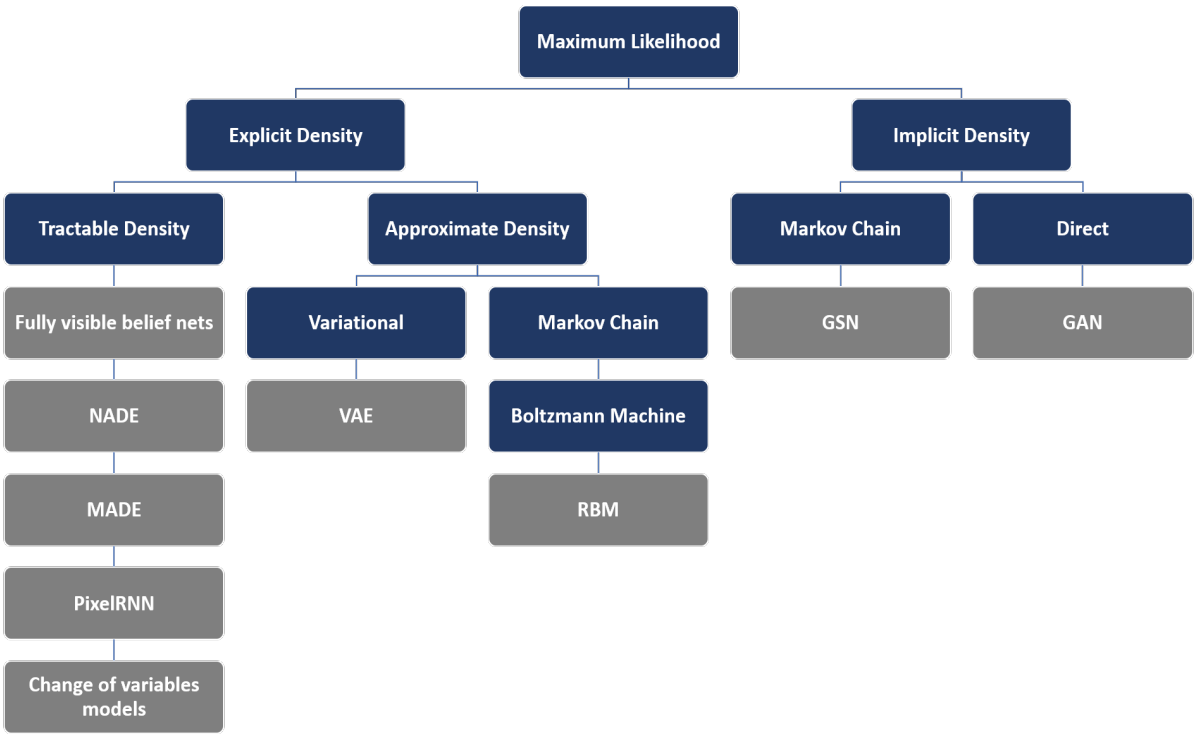

Abb. 2.4: Grundlegende Taxonomie von generativen Algorithmen

Zu diesen gehören generative Algorithmen wie die Fully visible belief nets, Neural Autoregressive Distribution Estimation (NADE), Masked Autoencoder for Distribution Estimation (MADE), Pixel Recurrent Neural Networks (PixelRNN) oder Change of variable models. Die Approximate Modelle schätzen, während des Trainingsvorgangs, eine explizite Dichtefunktion. Diese Approximation erfolgt entweder deterministisch oder stochastisch. Zu diesen Modellen gehören beispielsweise die Variational Autoencoder (VAE) oder Restricted Boltzmann Machines (RBM).

Die impliziten generativen Modelle approximieren die Dichtefunktion eines Trainingsdatensatzes. Zu diesen gehören, die auf Markov-Ketten basierenden Generative Stochastic Networks (GSN) sowie die sogenannten Generative Adversarial Networks (GAN). Letztere erlangten, in den letzten Jahren, eine große Popularität und werden, im Folgenden, näher beschrieben.

Generative Adversarial Networks

Das Generative Adversarial Network ist ein generatives Modell, mit dem es möglich ist, synthetische Daten zu erzeugen. Dieser Algorithmus enthält, wie typische generative Algorithmen, einen Generator, der eine, auf dem Trainingsdatensatz basierende, Verteilung generiert. Neben dem generativem Anteil umfasst es jedoch noch ein diskriminatives Modell. Dieser Diskriminator lernt zu bestimmen, ob Daten aus dem generativen Modell stammen oder reale Daten vorliegen. Während des Lernvorgangs generiert der Generator synthetische Daten, die mit der Zeit immer mehr der Verteilung des zugrundeliegenden Trainingsdatensatzes entsprechen. Der Diskriminator versucht dann, diese korrekt, in reale oder generierte Samples einzuteilen. [29]

Zu Beginn des Trainingsvorgangs wird der Generator mit einem Zufallsvektor z initialisiert. Danach werden abwechselnd der Diskriminator und der Generator trainiert bzw. optimiert. Dabei werden die Parameter der einen Komponente stets festgehalten, solange die jeweils andere trainiert wird. Der Diskriminator versucht, die Wahrscheinlichkeit zu maximieren. Der Generator hingegen verfolgt das Ziel, dessen Verlustfunktion zu minimieren. Diese wird wie folgt beschrieben [19]:

$$\min_G \max_D V(D,G) = E_{x\sim p_{Data}(x)} [\log D(\mathbf{x})] + E_{x\sim p_x(x)} [\log(1 - D(G(z)))] \tag{2.1}$$

Der Ausdruck $E_{x\sim p_{Data}(x)} [\log D(\mathbf{x})]$ stellt den Erwartungswert in Bezug auf die Verteilung des Realdatensatzes dar. $E_{x\sim p_x(x)} [\log(1 - D(G(z)))]$ bezeichnet hierbei den Erwartungswert, bezogen auf die Verteilung des Inputvektors z. [29]

2.4 Zeitreihen

Sequenzielle Daten stellen eine geordnete Liste von Ereignissen dar. Diese Art von Daten findet sich in verschiedenen Anwendungsfällen wieder. So kommen in der Biologie oftmals sequenzielle Daten, z. B. in der DNA oder in Proteinsequenzen, vor. Auch symbolische Sequenzen, wie die logische Folge eines Kundenkaufs in einem Onlineshop, fallen unter diese Kategorie. Eine weitere wichtige Art von sequenziellen Daten sind Zeitreihen.

Dabei handelt es sich um numerische Daten, zwischen denen, normalerweise, ein festes diskretes Zeitintervall liegt. Zeitreihen finden sich in verschiedensten Anwendungsgebieten. Diese kommen in der Natur, in Form von Temperatur oder Klima, sowie in der Wirtschaft, in Form von Aktienkursen, vor. Die in Fahrzeugen gespeicherten Sensordaten bilden ebenfalls Zeitreihen. Mithilfe der Zeitreihenanalyse lässt sich die Struktur der Zeitreihen verstehen. Es lassen sich Anwendungen in den verschiedenen Bereichen simulieren und vergleichen, sowie Prognosen erstellen. [22]

Taxonomie von Zeitreihen

Es gibt verschiedene Arten von Zeitreihen, die sich anhand bestimmter Faktoren, unterscheiden. Im Folgenden werden die wesentlichen Typen von Zeitreihen, auf Basis von [24], näher beschrieben.

Arten von Zeitreihen	**Beschreibung**
Kontinuierlich und Diskret	Die zugrundeliegenden Daten können von kontinuierlicher Natur sein, wie z. B. physikalische Daten, wie die Temperatur. In diesem Fall handelt es sich um kontinuierliche Zeitreihen $x(t)$. Bei diskreten Datenpunkten, die nach bestimmten Intervallen abgerufen werden, z. B. jede Sekunde, handelt es sich um diskrete Zeitreihen x_{t_i}. Kontinuierliche Zeitreihen können durch Abtastung in diskrete umgewandelt werden.
Gleichmäßig und Ungleichmäßig Abgetastet	Bei diskreten Zeitreihen werden die Daten in bestimmten Intervallen erhoben. Wenn dieses Intervall zwischem jeden Datenpunkt gleich ist, so handelt es sich um einen gleichmäßig abgetasteten Datensatz. Bei einem ungleichmäßig abgetasteten Datensatz ist dieses Abtastintervall nicht für alle Zeitpunkte konstant.
Univariat und Multivariat	Eine univariate Zeitreihe $x(t)$ oder x_t, ist durch eine, nach der Zeit, geordnete endliche Sequenz von T Datenpunkten definiert. Für jeden Zeitpunkt t_i gibt es genau einen Datenpunkt x_{t_i}. Eine multivariate Zeitreihe besteht aus $M > 1$ univariaten Zeitreihen. Für jeden Zeitpunkt t_i gibt es also M Datenpunkte x_{u,t_i}. Folgend wird also bei univariaten Zeitreihen nur ein Merkmal und bei multivariaten mehrere betrachtet.

Periodisch und Aperiodisch	Wenn kein periodisches Verhalten festgestellt werden kann, wird diese als aperiodisch bezeichnet. Falls Datenpunkte periodisches Verhalten aufweisen, handelt es sich um periodische Zeitreihen. Bei vollständig periodischen Zeitreihen zeigen alle Datenpunkte periodisches Verhalten auf. Tritt nur in einer Teilmenge von Datenpunkten periodisches Verhalten auf, wird diese als partiell-periodische Zeitreihe bezeichnet. Ein weiterer Faktor ist die Synchronizität der periodischen Muster. Synchrone Muster treten mit einer fixen Periode auf, wohingegen asynchrone Muster keine feste Periode aufweisen.
Stationär und Nicht-stationär	Die statistischen Eigenschaften von stationären Zeitreihen ändern sich nicht über der Zeit. Hierbei wird unter schwacher und starker Stationarität unterschieden. Bei stark stationären Zeitreihen hat die Menge $(x_1, x_2, \ldots, x_n)$, genau dieselben statistischen Eigenschaften, wie die Menge $(x_{1+h}, x_{2+h}, \ldots, x_{n+h})$, für jedes $h > 0$ und $n > 0$. Schwache Stationarität liegt vor, wenn der Erwartungswert und die Kovarianz unabhängig von der Zeit sind. Es muss also gelten $\mu_x(t) = \mu_x$ und $\gamma_x(t + h,t) = cov(x_{t+h},x_t) = E[(x_{t+h} - \mu_{t+h})(x_t - \mu_t)]$, wobei γ_x die Autokovarianzfunktion bezeichnet.
Kurz und Lang	Die Anzahl der Daten in einer Zeitreihe kann stark variieren. Umso mehr Parameter ein Modell enthält, desto größer sollte die Anzahl an Datenpunkten sein, damit dieses richtig interpretiert werden kann. Die theoretische Mindestanzahl an Beobachtungen ist hierbei gleich der Anzahl an vorhandenen Merkmalen. Um genauere Aussagen treffen zu können, muss allerdings eine passende Anzahl an Daten vorliegen. Bei Short Time Series ist die Anzahl an Daten sehr gering, weswegen Erkenntnisse von verschiedenen Analysemethoden eine geringe Aussagekraft haben. Ein anderes Extrembeispiel ist ein zu großer Datensatz. Umso mehr Daten vorhanden sind, desto höher ist die Diskrepanz zwischen diesen. Dadurch wird die Analyse solcher Zeitreihen komplexer.

Ergodisch und Nicht-ergodisch	Bei stationären stochastischen Prozessen stimmt zu jedem beliebigen Zeitpunkt t der Scharmittelwert des Prozesses mit dem Zeitmittelwert jedes Einzelexperimentes überein. Hierdurch können die statistischen Momente, mithilfe der zeitlichen Wiederholung eines einzelnen Zufallsexperiments, bestimmt werden.

Tab. 2.2: Taxonomie von Zeitreihen

Anwendungen

Im Folgenden wird der spezielle Fall einer multivariaten Zeitreihe betrachtet. Bezugnehmend auf Kap. 2.1 werden in Fahrzeugen verschiedenste Sensordaten, wie Positions-, Geschwindigkeits- und Beschleunigungswerte, gespeichert. Diese Werte werden, im Kontext des Machine Learning, ebenso als Merkmale bezeichnet und können als multivariate Zeitreihe dargestellt werden. Die einzelnen Datenpunkte lassen sich in der Matrixform

$$\begin{pmatrix} x_{1,t_1} & x_{1,t_2} & \cdots & x_{1,t_T} \\ \vdots & \ddots & & \vdots \\ x_{M,t_1} & \cdots & & x_{M,t_T} \end{pmatrix}$$

darstellen. Eine multivariate Zeitreihe lässt sich für den reellen Raum wie folgt definieren:

$$f : D \in \mathbb{R} \rightarrow Z \in \mathbb{R}^k, t \mapsto Y \in \{Y^k_{t=t_1}, Y^k_{t=t_2}, \ldots, Y^k_{t=t_N}\},\ \#(t,Y) = N,\ k \in \mathbb{N}^+ \tag{2.2}$$

Hierbei beschreibt t den Zeitvektor. Die Beobachtungen Y werden für k Merkmale in dem Vektor Y^k zusammengefasst. Des Weiteren bezeichnet N die Anzahl der Datenpunkte der multivariaten Zeitreihe.

In Abb. 2.5 ist beispielhaft eine Geschwindigkeit v über der Zeit t dargestellt. Für das konstante Geschwindigkeitssignal von $75\frac{m}{s}$ wurde hierbei ein gaußsches Rauschen mit einer Standardabweichung von $\sigma = 0.01$ angenommen. Dieses Signal ist ein Beispiel für eine stetige univariate Zeitreihe.

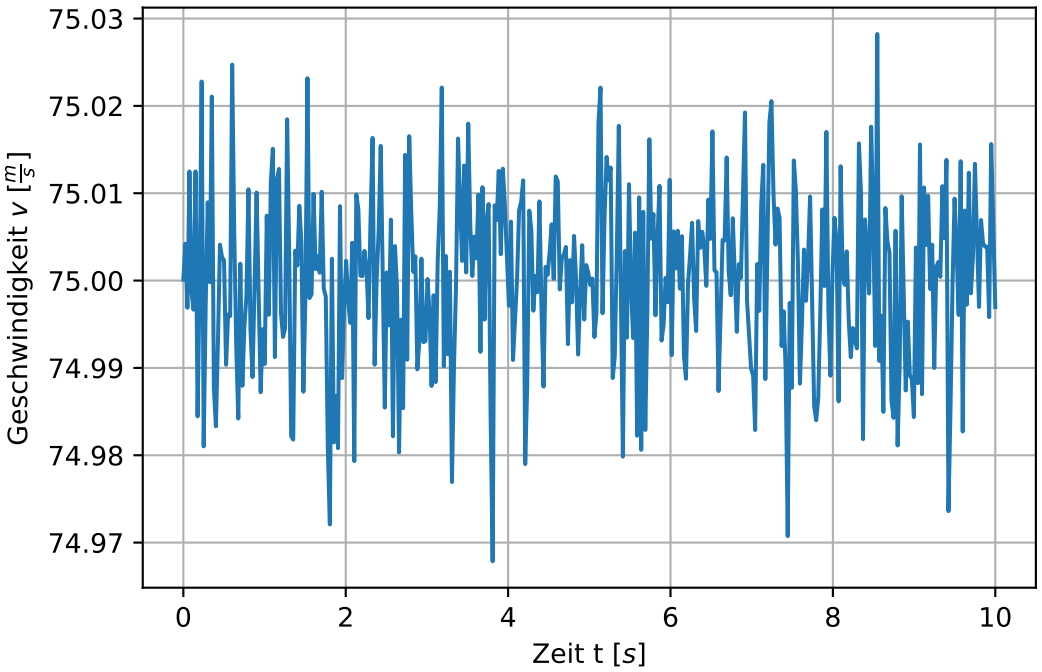

Abb. 2.5: Stetige Univariate Zeitreihe

2.5 Dichtefunktion

Im Alltag gibt es viele Vorgänge, deren Ergebnisse vom Zufall abhängig sind. Bei diesen Prozessen ist es nicht möglich, vorherzusagen, welches Ergebnis eintritt. Man spricht bei solchen Zufallsexperimenten ebenso von nicht deterministischen Prozessen. Ein Beispiel für ein solches Experiment ist das Werfen eines Würfels. Die sogenannte Ergebnismenge ist $\Omega = \{1,2,3,4,5,6\}$ und enthält hierbei alle möglichen Elementarereignisse ω_i, die vorkommen können. Wenn nun ein Zufallsexperiment durchgeführt wird, bei dem ein Würfel zwei mal hintereinander geworfen wird, gibt es eine Menge $\mathbb{D}$, die alle möglichen Resultate für die Augensumme enthält. In diesem Beispiel beinhaltet diese Menge alle Zahlen von $2, \ldots, 12$. Hierbei wird jede mögliche Kombination einer Ergebnismenge $\mathbb{W}$ zugeordnet. [23]

Diese Abbildung auf eine Menge erfolgt mithilfe einer Funktion, die als Zufallsvariable X bezeichnet wird. Eine Zufallsvariable gilt dann als stetig, wenn sie mindestens in einem bestimmten Intervall, jeden beliebigen reellen Wert annehmen kann. Da ein endliches oder unendliches Intervall von reellen Zahlen nicht abzählbar unendlich viele Werte enthält, ist die Wahrscheinlichkeit, dass eine stetige Zufallsvariable genau einen Wert darin annimmt, gleich null. Dabei wird die Wahrscheinlichkeit, einer bestimmten Realisierung der Zufallsvariablen X, durch die Dichtefunktion $f(x)$ bestimmt. Diese stellt die Wahrscheinlichkeitsdichte von X dar.

Für die Dichtefunktion gilt im Allgemeinen [28]:

$$\int_{-\infty}^{+\infty} f(x)\,dx = 1 \quad \wedge \quad f(x) \geq 0; x \in \mathbb{R} \tag{2.3}$$

$$W(a < X \leq b) = \int_{a}^{b} f(x)\,dx \tag{2.4}$$

Kennzahlen

Mithilfe von Dichtefunktionen können Zufallsexperimente -oder variablen beschrieben werden. Allerdings gibt es noch andere Kennzahlen, die diese charakterisieren. Momente und zentrale Momente eines Zufallsexperimente zählen zu den wichtigsten dieser Kennwerte.

Das allgemeine i-te Moment ist durch

$$\alpha_i = E(X^i) = \begin{cases} \int_{-\infty}^{+\infty} x^i \cdot f_X(x)dx & \text{wenn X stetig} \\ \sum_{k\in\mathbb{N}} x(k)^i \cdot f_X(x(k)) & \text{wenn X diskret} \end{cases} \tag{2.5}$$

gegeben [23]. Für $i = 1$ ergibt sich gemäß Gl. 2.5 das erste Moment, welches ebenso als Erwartungswert $E(X)$ oder μ bekannt ist:

$$\alpha_1 = E(X) = \begin{cases} \int_{-\infty}^{+\infty} x \cdot f_X(x)dx & \text{wenn X stetig} \\ \sum_{k\in\mathbb{N}} x(k) \cdot f_X(x(k)) & \text{wenn X diskret} \end{cases} \tag{2.6}$$

Dieser trifft Aussagen über die Lage bzw. das Zentrum der Zufallsvariablen X und stellt das gewogene arithmetische Mittel dar. In praktischen Anwendungen ist die Dichtefunktion $f_X(x)$ oft nicht bekannt, weswegen der Erwartungswert auch durch den arithmetischen Mittelwert $\overline{x}$

$$\overline{x} = \frac{1}{n} \cdot \sum_{k=1}^{n} x(k) \tag{2.7}$$

angenähert wird. Die zentralen Momente einer Zufallsvariablen X sind definiert durch [23]:

$$\mu_i = E\left((X - E(X))^i\right) = \begin{cases} \int_{-\infty}^{+\infty} (X - E(X))^i \cdot f_X(x)dx & \text{wenn X stetig} \\ \sum_{k\in\mathbb{N}} (x(k) - E(X))^i \cdot f_X(x(k)) & \text{wenn X diskret} \end{cases} \tag{2.8}$$

Das zweite zentrale Moment der Zufallsvariablen X

$$Var(X) = E\left((X - E(X))^2\right) = \begin{cases} \int_{-\infty}^{+\infty} (X - E(X))^2 \cdot f_X(x)dx & \text{wenn X stetig} \\ \sum_{k\in\mathbb{N}} (x(k) - E(X))^2 \cdot f_X(x(k)) & \text{wenn X diskret} \end{cases} \tag{2.9}$$

ergibt sich nach Gl. 2.9 für $i = 2$. Sie gibt eine Aussage über die Streuung der Verteilung einer Zufallsvariablen X. In diesem Kontext ist die Standardabweichung das lineare Maß der Varianz:

$$\sigma(X) = \sqrt{Var(X)} \tag{2.10}$$

Neben den bereits genannten Kenngrößen gibt es ebenso Formparameter, die Aussagen über die Gestalt von theoretischen Verteilungen machen. So gibt der Parameter

$$\gamma_1 = \frac{E((X-\mu)^3)}{\sigma^3} \tag{2.11}$$

die theoretische Schiefe der Verteilung von X wieder [25]. Bei $\gamma_1 = 0$ liegt eine symmetrische, bei $\gamma_1 > 0$ eine rechtsschiefe bzw. linkssteile und bei $\gamma_1 < 0$ eine linksschiefe bzw. rechtssteile Verteilung vor. Der weitere Formparameter

$$\gamma_2 = \frac{E((X-\mu)^4)}{\sigma^4} \tag{2.12}$$

gibt die theoretische Wölbung bzw. Kurtosis wieder [25]. Dieser gibt an, wie stark die Flanken einer Verteilung ausgeprägt sind bzw. wie stark die Wahrscheinlichkeitsmasse um den Erwartungswert konzentriert ist.

Normalverteilung

Eine der wohl bekanntesten Dichtefunktionen ist die nach Carl Friedrich Gauß benannte Gaußsche Normalverteilung. Sie ist definiert durch

$$f(x) = \frac{1}{\sigma\sqrt{2\,\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\,\sigma^2}} \tag{2.13}$$

mit μ als den Erwartungswert und σ als die Standardabweichung. In Abb. 2.6 ist diese, für $\mu = 0$ und $\sigma = 1$, dargestellt (Standardnormalverteilung).

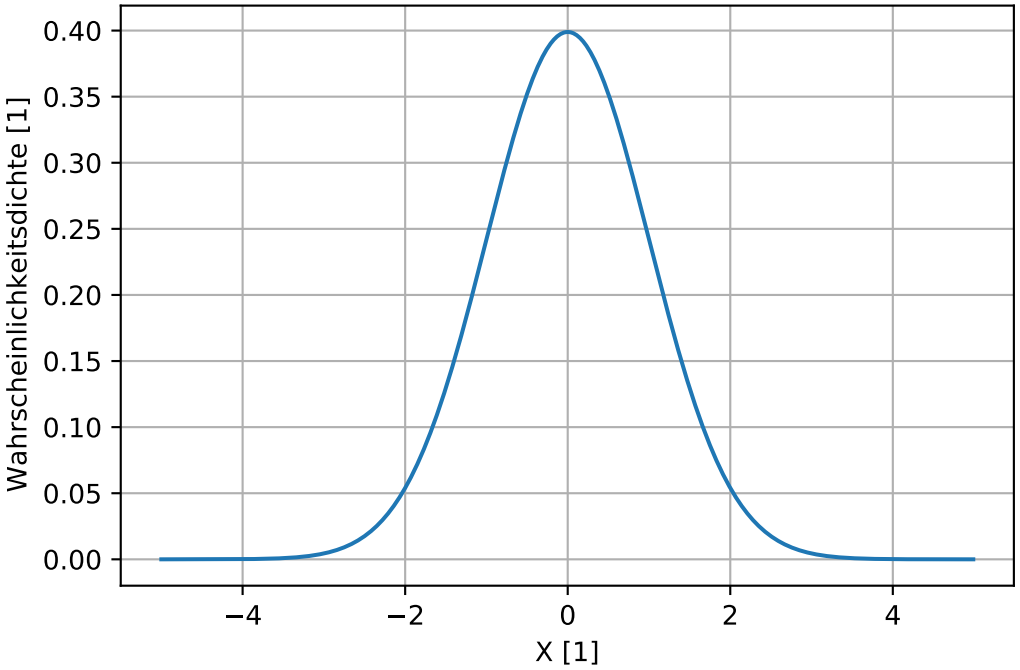

Abb. 2.6: Gaußsche Standardnormalverteilung

Kerndichteschätzer

In der Praxis, wie beispielsweise bei realen Messdaten, ist die Dichtefunktion, der zugrundeliegenden Zufallsvariablen, typischerweise unbekannt. Demnach ist es wichtig, je nach Anwendungsfall, diese nicht nur qualitativ, sondern ebenso quantitativ gut abzuschätzen. In diesem Kontext können sogenannte Kerndichteschätzer (*engl. Kernel Density Estimator, KDE*) eingesetzt werden. Solche Kerndichteschätzer sind wie folgt definiert:

$$\tilde{f}_n(t) = \frac{1}{nh} \sum_{j=1}^{n} k \frac{(t - x_j)}{h} \tag{2.14}$$

Dabei stellt $\tilde{f}_n$ die Approximation der echten Dichtefunktion dar. Die Stichprobe ist hierbei gegeben mit $x_1, \ldots, x_n$ und der Stichprobenumfang mit n. Darüber hinaus ist die Güte des Kerndichteschätzers stark abhängig von der Bandbreite h. Eine zu klein gewählte Bandbreite bewirkt ein Overfittung der Dichtefunktion, wohingehend eine zu große Bandbreite entsprechend zum Underfitting führt. Gemäß dem Satz von Nadaraya existiert eine Bandbreite, so dass die approximierte Dichtefunktion gegen die echte Dichtefunktion konvergiert. Des Weiteren beinhaltet ein Kerndichteschätzer einen sogenannten Kernel k bzw. ein Wahrscheinlichkeitsmaß. Hierbei können verschiedene Kernarten verwendet werden. Der sogenannte Gaußkern (*engl. Gaussian*) ist dabei wie folgt definiert:

$$k(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right) \tag{2.15}$$

Sowohl die Bandbreite als auch der Kernel können mittels Optimierverfahren bestimmt werden. Dadurch wird die Berechnung der Kerndichteschätzer jedoch noch zeitaufwändiger. Je nach Genauigkeitsanforderungen und Rechenzeitaufwand können alternative Ansätze herangezogen werden, um die Wahrscheinlichkeitsdichtefunktion abzuschätzen. Beispielsweise mittels einer sogenannten Histogramm-Spline-Approximation. Hierbei wird eine Spline durch das Histogramm, der zugrundeliegenden Daten, gelegt. Eine solche Approximation erfüllt ebenso die wesentlichen Eigenschaften einer Wahrscheinlichkeitsdichtefunktion. Die Histogramm-Spline-Approximation ist insbesondere vom Grad der Spline und der Anzahl der Bins des Histogrammes abhängig. [32]

2.6 Evaluierungstechniken von generativen Algorithmen

Gemäß Kap. 2.3 lassen sich synthetische Daten mithilfe von generativen Algorithmen erzeugen. Grundsätzlich können diese generierten Daten mit dem Ausgangsdatensatz, auf visueller bzw. qualitativer Ebene, verglichen werden. Ein quantitativer Ansatz ist hingegen der Vergleich der beiden Wahrscheinlichkeitsdichtefunktionen. Hierzu gibt es eine Vielzahl verschiedenster Evaluierungstechniken. Darunter fällt die statistische Distanz zweier Dichtefunktionen. Im Folgenden wird dieser Begriff, auf Basis der Arbeit von M. Deza und E. Deza [14], näher erläutert.

Eine Distanz (*engl. Distance oder Dissimilarity*) $d(x,y)$ ist eine Funktion, die eine Menge X auf den reellen Raum abbildet, d. h. $d : X \times X \to \mathbb{R}$. Hierbei werden $x,y \in X$ durch d auf den reellen Raum abgebildet. Diese besitzt folgende Eigenschaften:

$$\begin{aligned} d(x,y) &\geq 0 && (Nicht - Negativität) \\ d(x,y) &= d(y,x) && (Symmetrie) \\ d(x,x) &= 0 && (Reflexivität) \end{aligned}$$

Eine weitere Art, die Diskrepanz zweier Dichtefunktionen zu bestimmen, ist die Ähnlichkeit (*engl. Similarity*). Ähnlich wie eine Distanz, bildet diese eine Menge auf den reellen Raum ab, d. h. $s : X \times X \to \mathbb{R}$. Neben den Eigenschaften Nicht-Negativität und Symmetrie, muss für eine Similarity gelten $s(x,y) \leq s(x,x)$ für alle $x,y \in X$, außer für $y = x$. Similarities s und Distances d können durch verschiedene Transformationen ineinander umgeformt werden. Beispielsweise durch: $d = 1 - s$, $d = \frac{1-s}{s}$, $d = \sqrt{1-s}$ oder $d = -\ln(s)$ [14]. Umso ähnlicher sich x und y sind, desto höher ist der Wert einer Similarity s. Im Gegensatz ist eine Distanz d umso geringer, desto ähnlicher sich x und y sind. In beiden Fällen können die dazugehörigen Kennzahlen normalisiert werden. Dann gilt $0 \leq s(x,y) \leq 1$ bzw. $0 \leq d(x,y) \leq 1$.

Die Definition einer Distanz kann in eine Semi-Metrik erweitert werden. Hierzu muss, zusätzlich zur Nicht-Negativität, Symmetrie und Reflexivität, für alle $x,y,z \in X$ gelten:

$$d(x,y) \leq d(x,z) + d(z,y) \qquad (Dreiecksungleichung)$$

Eine Metrik erweitert die Semi-Metriken zusätzlich um die Identität (*engl. identity*):

$$\begin{aligned} d(x,y) &\geq 0 && (Nicht - Negativität) \\ d(x,y) &= 0 \Leftrightarrow x = y && (Identität) \\ d(x,y) &= d(y,x) && (Symmetrie) \\ d(x,y) &\leq d(x,z) + d(z,y) && (Dreiecksungleichung) \end{aligned}$$

Eine Menge X, inkl. einer Metrik d, wird als metrischer Raum (X,d) bezeichnet. Diese Evaluierungstechniken bilden den Rahmen, um Dichtefunktionen, im Kontext von generativen Algorithmen, miteinander zu vergleichen. Distanzen, die asymmetrisch sind, können in symmetrische transformiert werden. In Tab. 2.3 befinden sich einige dieser Transformationen [9].

Methode	**Beschreibung**
Addition	$d_{sym}(\mathbb{P},\mathbb{Q}) = d_{asym}(\mathbb{P},\mathbb{Q}) + d_{asym}(\mathbb{Q},\mathbb{P})$
Maximum	$d_{max-sym}(\mathbb{P},\mathbb{Q}) = \max\left(d_{asym}(\mathbb{P},\mathbb{Q}),d_{asym}(\mathbb{Q},\mathbb{P})\right)$
Minimum	$d_{min-sym}(\mathbb{P},\mathbb{Q}) = \min\left(d_{asym}(\mathbb{P},\mathbb{Q}),d_{asym}(\mathbb{Q},\mathbb{P})\right)$
Durchschnitt	$d_{avg-sym}(\mathbb{P},\mathbb{Q}) = avg\left(d_{asym}(\mathbb{P},\mathbb{Q}),d_{asym}(\mathbb{Q},\mathbb{P})\right)$

Tab. 2.3: Techniken zur Symmetrisierung einer Distanz

3 Hauptteil

Dieses Kapitel beschreibt den Hauptteil der zugrundeliegenden Arbeit. Hierbei werden, in einem ersten Schritt, der Anwendungsfall sowie die genaue Herangehensweise, bei der umfangreichen Literaturrecherche, näher erläutert. Anschließend wird eine neuartige Taxonomie der Evaluierungstechniken von Wahrscheinlichkeitsdichtefunktionen vorgestellt, die zur Validierung von generativen Algorithmen, im Kontext der Bereitstellung von zeitreihen-bezogenen Daten, verwendet werden können. Weiterhin wird jede einzelne Evaluierungstechnik in der engeren Auswahl näher beschrieben. Daraufhin werden definierte Bewertungskriterien für eine Gegenüberstellung der Methoden vorgestellt. Abschließend erfolgt die Implementierung und Validierung einer ausgewählten Evaluierungstechnik anhand von synthetisch erzeugten Beispieldatensätzen.

3.1 Anwendungsfall

Wie in Kap. 2.1 bereits beschrieben wurde, müssen autonome Fahrzeuge mit jeder Situation im Straßenverkehr umgehen können. Darunter fallen ebenso fahrkritische Szenarien, wie Überhol- oder Abbiegevorgänge. In Abb. 3.1 ist ein solcher Überholvorgang beispielhaft dargestellt.

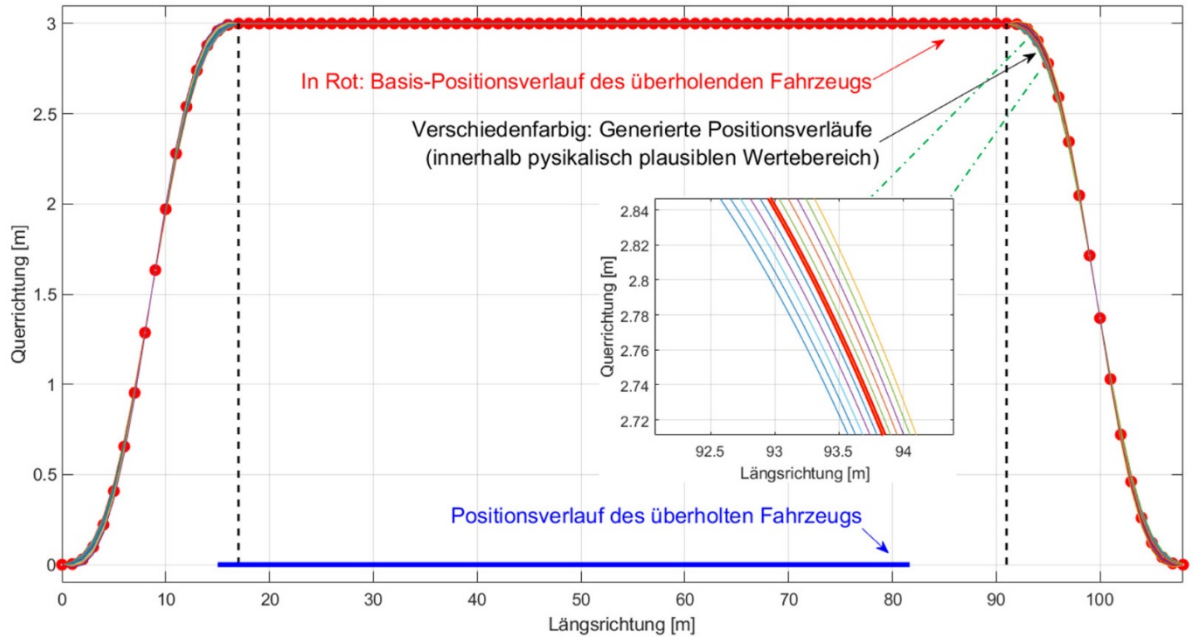

Abb. 3.1: Überholvorgang

Hierbei sind im Koordinatensystem die Bewegungsdaten der Fahrzeuge, für die Längs- und Querrichtung, dargestellt. Der Positionsverlauf des überholten Fahrzeugs ist in blau aufgezeigt. Die rote Kurve stellt den Verlauf des überholenden Fahrzeugs dar. Der gesamte Überholvorgang erstreckt sich über eine Strecke von 110 m in Längs- sowie 3 m in Querrichtung. Die roten Datenpunkte, auf der Kurve des überholenden Fahrzeugs, zeigen die einzelnen Zeitstempel. Hierbei ist anzumerken, dass die Anzahl der Datenpunkte, für den Ein- und Ausschervorgang, geringer ist, als für den restlichen Verlauf. Dies liegt daran, dass die zurückgelegte Strecke in Querrichtung, für diese Verläufe, geringer ist, als für den eigentlichen Überholvorgang. In der Vergrößerung sind generierte synthetische Positionsverläufe aufgezeigt. Diese befinden sich innerhalb eines physikalisch plausiblen Wertebereich.

Um die Algorithmen von autonomen Fahrzeugen, für diese Fälle, trainieren zu können, müssen Daten, für diese, vorhanden sein. In der Realität sind etwaige Realdaten bzw. Messungen kaum vorhanden. Gründe hierfür sind das geringe Auftreten solcher Szenarien oder die seltene Aufnahme dieser, aufgrund der zugrundeliegenden Kritikalität. Eine Abhilfe kann hier die Verwendung von generativen Algorithmen sein, die synthetische Daten generieren können. Durch die Verwendung dieser synthetischen Daten, ist es möglich, autonome Fahrzeuge, für diese fahrkritischen Szenarien, zu trainieren.

Die Daten in Kraftfahrzeugen, die in solchen fahrkritischen Szenarien anfallen, sind zumeist Sensordaten. Diese werden z. B. in Bewegungsvektoren zusammengefasst und enthalten Daten u.a. über die Zeit, Geschwindigkeit, Beschleunigung und Position in x- und y-Richtung. Gemäß der Unterscheidung in Kap. 2.4, bilden diese Art von Daten periodische, multivariate Zeitreihen. Im zugrundeliegenden Anwendungsfall haben diese Bewegungsvektoren ebenso die gleiche Länge. Einerseits gehen Informationen verloren, falls etwaige Datenpunkte entfernt werden. Andererseits können fehlende Datenpunkte, nicht für jeden Fall, kinematisch korrekt nachgebildet werden.

In generativen Algorithmen werden, wie in Kap. 2.3 beschrieben, Dichtefunktionen verwendet, um Datensätze abzubilden und zu generieren. Es liegen jeweils 100 Datensätze zu ausgewählten Szenarien vor. Darunter zählen entsprechende Bewegungsvektoren inkl. Dichtewerte zur Sinusfunktion und zum Überholmanöver mit Gegenverkehr. Die in den Datensätzen befindlichen Dichtefunktionswerte sollen miteinander verglichen werden. Dazu soll eine entsprechende Evaluierungstechnik verwendet werden. Dadurch kann eine Kennzahl bzw. ein Maß ermittelt werden, welche die Übereinstimmung der Dichtefunktionen darstellt. Mithilfe einer solchen Kennzahl ist es möglich, zu entscheiden, ob die synthetisch erzeugten Daten, tatsächlich fahrkritische Szenarien abbilden.

3.2 Vorgehensweise

Dichtefunktionen können auf verschiedenste Arten und Weisen bewertet, evaluiert oder validiert werden. In diesem Kontext existieren eine Vielzahl an Publikationen. Deshalb erfolgte eine systematische Vorgehensweise, um die relevanten Publikationen zu finden. Hierzu wurden spezielle Suchstrings erstellt, die relevante Stichwörter enthalten. Dabei erfolgten ebenso Permutationen der einzelnen Stichwörter, um ein größeres Spektrum an Suchergebnissen zu erhalten.

Die Suchstrings wurden anschließend in Gruppen eingeteilt und dienten als Basis für die Literaturrecherche. Insgesamt erfolgte die beschriebene Literaturrecherche ausschließlich über Suchmaschinen im Internet.

Um die Relevanz der gefundenen Publikationen einzustufen, wurde in einem ersten Schritt, die jeweiligen Einleitungs- und Zusammenfassungs-Kapitel gelesen. Diese geben einen guten Überblick über die zugrundeliegenden Publikationen. Entsprechende Evaluierungstechniken, die in den Publikationen genannt und beschrieben werden, wurden extrahiert und in einer Liste gesammelt. In einem weiteren Schritt wurden die Publikationen näher betrachtet. Dadurch sind, die für den Anwendungsfall irrelevanten Evaluierungstechniken, weiter ausgefiltert worden. Im nächsten Schritt erfolgte eine Gruppierung von ähnlichen Evaluierungstechniken. Damit konnte eine eigene Taxonomie an Evaluierungstechniken, für den zugrundeliegenden Anwendungsfall, erstellt werden. Diese werden später, anhand von festgelegten Bewertungskriterien, evaluiert.

3.3 Taxonomie der Evaluierungstechniken von Wahrscheinlichkeitsdichtefunktionen

Durch die Literaturrecherche ergab sich eine umfangreiche Liste an möglichen Evaluierungstechniken. Diese wurden in bestimmte Teilbereiche zusammengefasst, um eine Taxonomie zu bilden. Die zugrundeliegende mathematische Beschreibung dieser Evaluierungstechniken, war die Basis für diese Unterteilung. In Abb. 3.2 ist diese Taxonomie dargestellt.

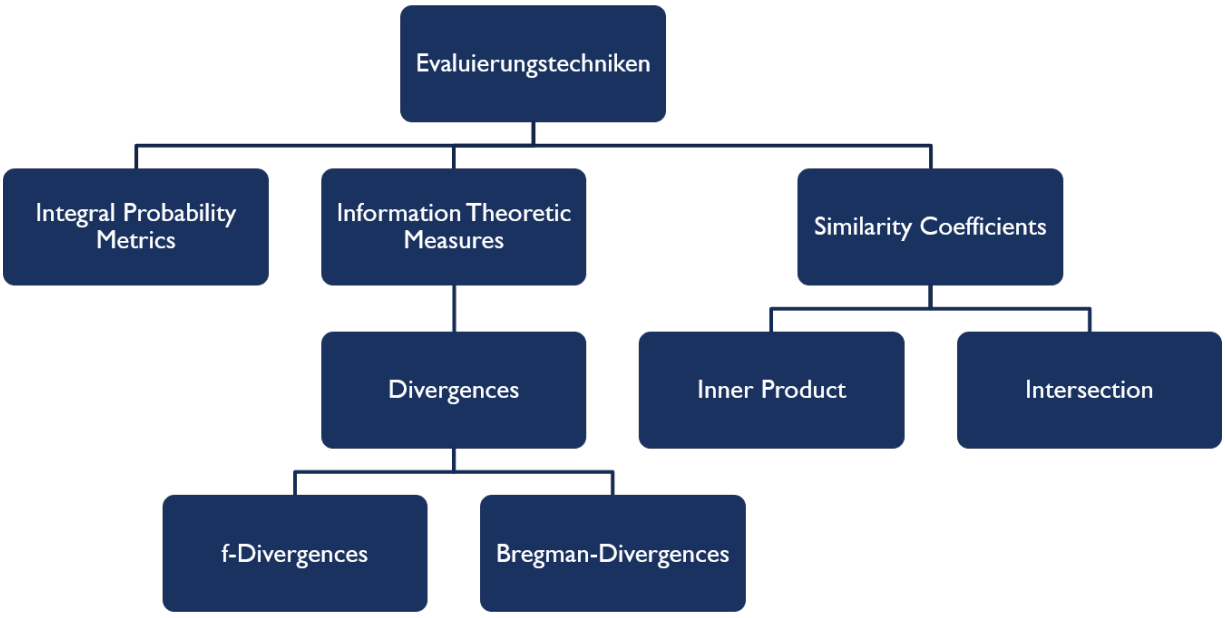

Abb. 3.2: Taxonomie von Evaluierungstechniken in Bezug auf Dichtefunktions-Vergleiche

Eine essenzielle Gruppe an Evaluierungstechniken bilden die Integral Probability Metrics. Die grundlegende Struktur dieser Metriken bildet die Differenz zweier Wahrscheinlichkeitsmaße $\mathbb{P}$ und $\mathbb{Q}$. Sie sind wie folgt definiert [17]:

$$\gamma_F(\mathbb{P},\mathbb{Q}) := \sup_{f\in\mathcal{F}} \left| \int f \, d\mathbb{P} - \int f \, d\mathbb{Q} \right| \tag{3.1}$$

Hierbei ist die Funktion f einer Klasse von reellen und beschränkten Funktionen $\mathcal{F}$ zugehörig.

Eine weitere wichtige Gruppe an Evaluierungstechniken bilden informationstheoretische Maße. Eine essenzielle historische Kennzahl, in diesem Bereich, ist die Entropie. Diese ist ein Maß über den Informationsgehalt bzw. die Unsicherheit einer Zufallsvariablen X. Sie ist wie folgt definiert [11]:

$$H(X) = -\sum_{i=1}^{N} x_i \cdot \log_2(x_i),\ i = 1,\ldots,N \tag{3.2}$$

Viele statistische Distanzen basieren auf der Entropie. Ein ebenfalls bedeutendes Mittel zum Vergleich von Dichtefunktionen sind Divergenzen bzw. Divergenzmaße (*engl. Divergences*). Hierbei ist anzumerken, dass viele der Evaluierungstechniken, wie z. B. Divergenzen, keine Metriken, wie in Kap. 2.6 definiert, darstellen. Das schließt sie jedoch nicht als passendes Evaluierungsmittel aus. Die hohe Anzahl an Nennungen in Literaturen rechtfertigt die Relevanz dieser Evaluierungstechniken für den zugrundeliegenden Anwendungsfall.

Eine der wohl bekanntesten Familie an Divergenzmaßen ist die f-Divergence. Diese beinhaltet eine hohe Anzahl an bekannten statistischen Distanzen. Sie ist definiert durch:

$$d_f(p,q) = \sum_x q(x) f\left(\frac{p(x)}{q(x)}\right) \tag{3.3}$$

Hierbei ist f eine beliebige Funktion, die konvex über dem Definitionsbereich $(0,\infty)$ ist und für die gilt $f(1) = 0$. Des Weiteren sind f-divergences stets nicht-negativ und nur dann null, wenn die zwei Dichtefunktionen $p(x)$ und $q(x)$ übereinstimmen [10].

Eine weitere wichtige Gruppe von Evaluierungstechniken sind die Bregman-Divergenzen. Diese messen die Diskrepanz zwischen zwei Werten von Dichtefunktionen $p(x)$ und $q(x)$ [10]:

$$d_\varphi(p,q) = \varphi(p) - \varphi(q) - (p-q)\varphi'(q) \tag{3.4}$$

Die gesamte Diskrepanz zwischen den Dichtefunktionen $p(x)$ und $q(x)$, lässt sich, mit Hilfe der Bregman-Divergenz, wie folgt beschreiben:

$$d_\varphi(\mathbb{P},\mathbb{Q}) = \int \left[\varphi(p(x)) - \varphi(q(x)) - (p(x)-q(x))\varphi'(q(x))\right] \, dx \tag{3.5}$$

Gl. 3.5 lässt sich, wie in folgender Rechenvorschrift, diskretisieren:

$$d_\varphi(\mathbb{P},\mathbb{Q}) = \sum_{i=1}^{N} \left[\varphi(p_i) - \varphi(q_i) - (p_i - q_i)\varphi'(q_i)\right] \tag{3.6}$$

Hierbei muss die Funktion $\varphi(t)$ streng konvex und reell sein. $\varphi'(\mathbb{Q})$ beschreibt die Ableitung nach $\mathbb{Q}$. Bregman-Divergenzen besitzen ebenso die Eigenschaft der Nicht-Negativität (vgl. Kap. 2.6). Die Symmetrie ist abhängig von $\varphi(t)$.

Mithilfe von sogenannten Similarity Coefficients können ebenso zwei Dichtefunktionen, in Bezug auf ihrer Ähnlichkeit, miteinander verglichen werden. Hierbei werden die beiden Dichtefunktionen bzw. Datensätze $\mathbb{P}$ und $\mathbb{Q}$, mittels unterschiedlichen Operationen, gegenübergestellt. Zu diesen zählen Kennzahlen, die auf dem Inner Product oder der Intersection basieren. Folgende Definitionen beruhen auf den Erkenntnissen in [9].

Hierzu zählen unter anderem Kennzahlen, die auf dem Inner Product basieren. Hierbei treten die zwei Dichtefunktionen P und Q in der folgenden Form auf:

$$s = P \bullet Q = \sum_{i=1}^{d} = P_i Q_i \tag{3.7}$$

Das Inner Product von zwei Vektoren wird ebenso als Skalarprodukt bezeichnet.

Des Weiteren gibt es noch Kennzahlen, die auf der Überschneidung von Dichtefunktionen aufbauen. In der Form einer Similarity treten diese wie folgt auf:

$$s = \sum_{i=1}^{N} \min(P_i, Q_i) \tag{3.8}$$

Durch die bereits in Kap. 2.6 genannte Umformung $d = 1 - s$, lässt sich die Intersection ebenso als Distanz formulieren:

$$d = \frac{1}{2} \sum_{i=1}^{N} |P_i - Q_i| \tag{3.9}$$

3.4 Mathematische Beschreibung der Evaluierungstechniken von Wahrscheinlichkeitsdichtefunktionen

Im Folgenden werden einige Evaluierungstechniken bzw. Algorithmen, gemäß ihrer Definition, diskret oder kontinuierlich beschrieben.

3.4.1 Kolmogorov-Smirnov Metrik

Die Kolmogorov-Smirnov Distanz - oder auch Kolmogorov-Smirnov Metrik - gehört zu der Familie der Integral Probability Metrics. Sie ist, im Normalfall, als die L_1 Norm (vgl. Kap. 3.4.4) zwischen zwei kumulativen Dichtefunktionen definiert:

$$d_{KS}(\mathbb{P}, \mathbb{Q}) = \sup_{x \in \mathbb{R}} |F_p(x) - F_q(x)| \tag{3.10}$$

In Gl. 3.10 stehen $\mathbb{P}$ und $\mathbb{Q}$ für kumulative Verteilungsfunktionen und *sup* für die Supremum Funktion. Diese gibt die kleinste obere Schranke einer Punktedifferenz, also die maximale Differenz, wieder. Zusätzlich ist die Kolmogorov-Smirnov Distanz auf [0,1] beschränkt, wodurch das Ergebnis leichter interpretiert werden kann. Sie wird, im Normalfall, dazu verwendet, um einen Kolmogorov-Smirnov Test durchzuführen. Hierbei werden zwei eindimensionale kumulative Verteilungsfunktionen verglichen. Dies geschieht durch den Hypothesentest $F_p(x) = F_q(x)$ [17]. In diskreter Form und auf Dichtefunktionen bezogen, lässt sich diese, basierend auf [30], wie folgt definieren:

$$d_{KS}(\mathbb{P},\mathbb{Q}) = \max_i(|p_i - q_i|) \tag{3.11}$$

Hierbei kann, für diskrete Dichtefunktionen, die jeweilige kumulative Verteilungsfunktion iterativ berechnet werden. Für den diskreten Fall ist sie somit ähnlich aufgebaut wie die L_∞ Metrik (vgl. Gl. 3.19). Die Kolmogorov-Smirnov Distanz entspricht, im Falle von diskreten kumulativen Verteilungen, dem maximalen Abstand zwischen $\mathbb{P}$ und $\mathbb{Q}$. Sie eignet sich somit nicht für Datensätze oder Dichtefunktionen mit Ausreißern.

3.4.2 Wasserstein Distance

Die Wasserstein-1 Distance ist ein Maß für die Distanz zwischen zwei Dichtefunktionen. Sie wird ebenso als die Earth Mover's (EM) Distance bezeichnet. Sie ist definiert durch [35]:

$$W(\mathbb{P},\mathbb{Q}) = \inf_{\gamma \in \prod(\mathbb{P},\mathbb{Q})} \mathbb{E}_{(x,y)\sim\gamma} \left[\| x - y \|\right] \tag{3.12}$$

In Gl. 3.12 ist $\prod(\mathbb{P},\mathbb{Q})$ die Menge aller gemeinsamen Wahrscheinlichkeitsverteilungen zwischen $\mathbb{P}$ und $\mathbb{Q}$. $\gamma \in \prod(\mathbb{P},\mathbb{Q})$ beschreibt eine der Transformationen, um $\mathbb{P}$ in $\mathbb{Q}$ zu überführen. Die EM-Distanz stellt die „Kosten" dar. Mit Hilfe der Kantorovich-Rubinstein Dualität lässt sich die Wasserstein Distanz ebenso als Integral Probability Metric, wie folgt, definieren [4]:

$$W(\mathbb{P},\mathbb{Q}) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}}\left[f(x)\right] - \mathbb{E}_{x\sim\mathbb{Q}}\left[f(x)\right] \tag{3.13}$$

In Gl. 3.13 umfasst die Menge der Funktionen $\mathcal{F}$ (vgl. Gl. 3.1) alle 1-Lipschitz Funktionen. Der Ausdruck $\mathbb{E}_{(x,y)\sim P}\left[f(x)\right]$ stellt den Erwartungswert in Bezug auf die Verteilung P der Funktion $f(x)$ dar. Analog dazu lässt sich der Ausdruck $\mathbb{E}_{(x,y)\sim Q}\left[f(x)\right]$ erklären. Es ist zu beachten, dass die Definition einer Integral Probability Metric, von der in Gl. 3.1, abweicht. Der Übergang von Integralen zu Erwartungswerten erfolgt ähnlich wie in Gl. 2.5.

3.4.3 Cramér Distance

Eine weitere Integral Probability Metric ist die Cramér Distance [6]. Diese ist definiert durch:

$$L_2^2(\mathbb{P},\mathbb{Q}) := \int_{-\infty}^{\infty} \left(F_P(x) - F_Q(x)\right)^2 dx \tag{3.14}$$

In Gl. 3.14 ist zu erkennen, dass die Cramér Distance keine vollwertige Metrik darstellt. Allerdings stellt ihre Wurzel eine dar und gehört zur L_P Familie der Metriken (vgl. Kap. 3.4.4).

3.4.4 Minkowski

Die Minkowski Metrik stellt die Familie der L_p Metriken dar. Sie ist definiert durch [6]:

$$L_p(\mathbb{P},\mathbb{Q}) := \left(\int_{-\infty}^{\infty} |F_P(x) - F_Q(x)|^p \, dx \right)^{\frac{1}{p}} \tag{3.15}$$

Für den diskreten Fall kann sie, wie folgt, beschrieben werden [9]:

$$d_{L_p}(\mathbb{P},\mathbb{Q}) = \sqrt[p]{\sum_{i=1}^{n} |P_i - Q_i|^p} \tag{3.16}$$

Die Minkowski Metrik ist eine Verallgemeinerung von verschiedenen anderen Distanzen. Diese ergeben sich durch die entsprechenden Werte für den Parameter p. Beispielsweise führt $p = 1$ zur L_1 Metrik. Diese wird ebenso als City-Block oder Manhattan-Distanz bezeichnet:

$$d_{L_1}(\mathbb{P},\mathbb{Q}) = \sum_{i=1}^{N} |P_i - Q_i| \tag{3.17}$$

Durch die Wahl $p = 2$ ergibt sich die Euklidische Distanz:

$$d_{L_2}(\mathbb{P},\mathbb{Q}) = \sqrt[2]{\sum_{i=1}^{N} |P_i - Q_i|^2} \tag{3.18}$$

Für $p = \infty$ ergibt sich die Chebyshev-Distanz, die als die maximale Differenz zweier Datensätze definiert ist:

$$d_{L_\infty}(\mathbb{P},\mathbb{Q}) = \max_i |P_i - Q_i| \tag{3.19}$$

Grundlegend lässt sich sagen, dass durch die Wahl eines höheren Werts für p, die größte Distanz bzw. Differenz zweier Elemente höher gewichtet wird. Es ist der Zusammenhang zwischen den L_p und Wasserstein Metriken anzumerken. Für den Fall $p = 1$ sind diese identisch. Ebenso wie Wasserstein Metriken, können L_p Metriken, in der Form einer Integral Probability Metric, beschrieben werden [6]:

$$d_{Mk}(\mathbb{P},\mathbb{Q}) = \sup_{f \in \mathbb{F}_q} |\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x)| \tag{3.20}$$

In Gl. 3.20 wird hier das Supremum über die absolut stetigen Funktionen $\mathbb{F}_q$ gebildet. Der Ausdruck $\mathbb{E}_{x\sim P} f(x)$ stellt den Erwartungswert in Bezug auf die Verteilung P der Funktion $f(x)$ dar. Analog dazu lässt sich der Ausdruck $\mathbb{E}_{x\sim Q} f(x)$ erklären.

3.4.5 Maximum Mean Discrepancy

Die Maximum Mean Discrepancy (MMD) gehört zur Klasse der Integral Probability Metrics, wenn in Gl. 3.1 für $\mathcal{F} = \{f : \| f \|_{\mathcal{H}} \leq 1\}$ gilt, wobei $\mathcal{H}$ den reproduzierenden Kernel Hilbertraum (*engl.: Reproducing Kernel Hilbert Space (RKHS)*) bezeichnet. Sie bewertet die Differenz zweier Distributionen mit Hilfe einer Glättungsfunktion. Diese Funktion erzeugt für Stichproben aus $\mathbb{P}$ große und für $\mathbb{Q}$ kleine Werte. Für Samples x und y, entsprechend aus $\mathbb{P}$ und $\mathbb{Q}$, ist die MMD wie folgt definiert [17]:

$$d_{MMD}(\mathbb{P},\mathbb{Q}) = \sup_{f \in \mathcal{F}} |\mathbb{E}_x [f(x)] - \mathbb{E}_y [f(y)]| \tag{3.21}$$

Die Gl. 3.21 liefert Werte für die beiden Stichproben, wobei die Differenz zwischen den Mittelwerten dieser Funktionswerte, dem MMD entspricht. D. h. dem maximalen Abstand zwischen den Mittelwerten nach der Transformation durch die Glättungsfunktion f. Zu der Funktionsklasse $\mathcal{F}$ gehören alle Funktionen, die einen RKHS produzieren, sogenannte reproduzierende Kernels. Es gibt eine Reihe von Kernels $k(\cdot,\cdot)$, die in der MMD gewählt werden können. Eine beliebte Wahl ist der radiale Basiskernel:

$$k(x,x') = e^{\frac{\|x-x'\|^2}{2\sigma^2}} \tag{3.22}$$

Hierbei sind x und x' jeweils Samples aus den entsprechenden Dichtefunktionen $p(x)$ und $q(x)$. Der Parameter σ wird oft durch den Median der paarweisen Abstände zwischen den gemeinsamen Daten bestimmt.

3.4.6 Total Variation Metric

Die Total Variation Metric nimmt eine besondere Stellung unter den Evaluierungstechniken ein. Sie kann zum einen, durch die Wahl von $f(t) = \frac{|t-1|}{2}$ in Gl. 3.3, als f-Divergence beschrieben werden. Zum anderen kann sie, durch die Wahl von $\mathcal{F} = \{f : \| f \|_{\infty} \leq 1\}$ in Gl. 3.1, als Integral Probability Metric definiert werden. Eine Möglichkeit, diese zu beschreiben, ist wie folgt [17]:

$$d_{TV}(\mathbb{P},\mathbb{Q}) = \sqrt{\frac{1}{2} \int |p(x) - q(x)| \, dx} \tag{3.23}$$

Es ist anzumerken, dass der Wertebereich der Gleichung Gl. 3.23 auf [0,1] beschränkt ist.

3.4.7 Shannon Entropie

Die wohl bekannteste und am häufigsten verwendete f-Divergence ist die Kullback-Leibler Divergenz. In Gl. 3.3 muss hierbei für $f(t) = t \ln(t)$ eingesetzt werden. Die diskrete Variante der Kullback-Leibler Divergence lautet [9]:

$$d_{KL}(\mathbb{P},\mathbb{Q}) = \sum_{i=1}^{N} p_i \ln\left(\frac{p_i}{q_i}\right) \tag{3.24}$$

Die Gl. 3.24 ist ein Maß der relativen Entropie (vgl. Gl. 3.2) und gibt die Menge an Information an, die benötigt wird, um die Änderung der Wahrscheinlichkeit von $\mathbb{Q}$ nach $\mathbb{P}$ zu kodieren. Der Grund für ihre Popularität ist die leichte Optimierung und die Verbindung zur *Maximum Likelihood Estimation*, die bei generativen Algorithmen von Bedeutung ist [6]. Sie ist ebenfalls skaleninvariant. Es ist anzumerken, dass die KL-Divergence keine Metrik darstellt, wie in Kap. 2.6 beschrieben. Sie ist nicht symmetrisch und erfüllt nicht die Dreiecksungleichung. Mit Hilfe von Transformationen, wie z. B. die in Tab. 2.3 aufgeführten, lässt sich eine symmetrische Form erzeugen. So ist die Jensen-Shannon Distanz eine symmetrische Version der KL-Divergence und eine vollwertige Metrik [17]:

$$d_{JS}(\mathbb{P},\mathbb{Q}) = \sqrt{\frac{1}{2}d_{KL}\left(\mathbb{P},\frac{\mathbb{P}+\mathbb{Q}}{2}\right) + \frac{1}{2}d_{KL}\left(\mathbb{Q},\frac{\mathbb{P}+\mathbb{Q}}{2}\right)} \tag{3.25}$$

Aus Anwendersicht stellt sich hierbei die Frage, wann die Kullback-Leibler und wann die Jensen-Shannon Divergence verwendet werden soll. Ein Vergleich der beiden Evaluierungstechniken ist in Abb. 3.3 für 99 Samples dargestellt.

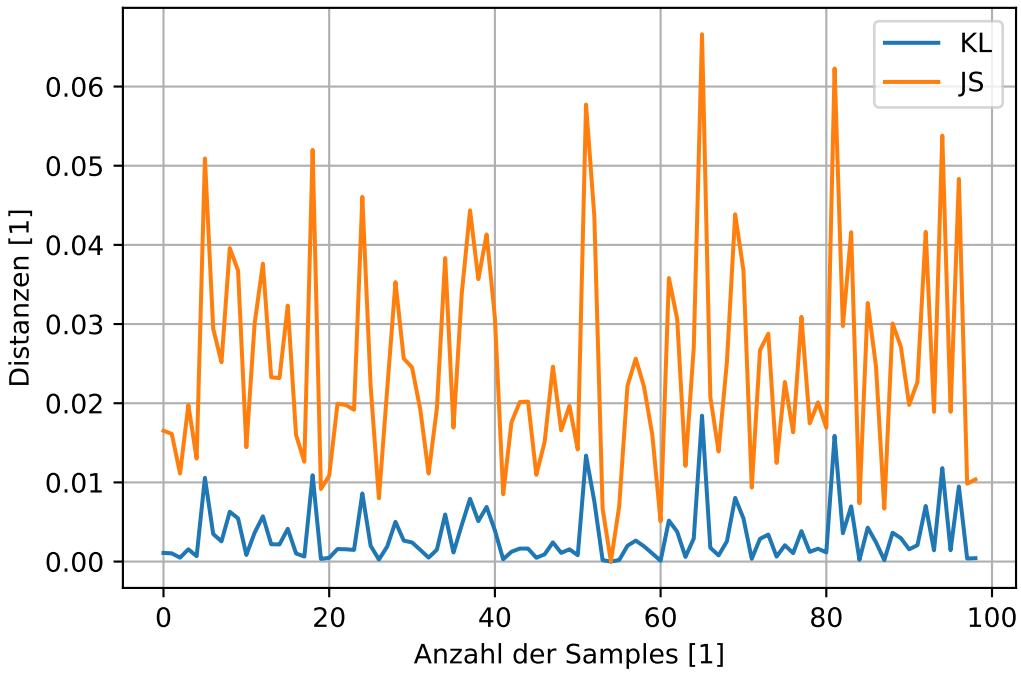

Abb. 3.3: Vergleich der Kullback-Leibler und Jensen-Shannon Divergence über 99 Samples

Es ist zu erkennen, dass der Verlauf der beiden Evaluierungstechniken sehr ähnlich ist. Die Jensen-Shannon Divergence ist hierbei stärker skaliert, als die Kullback-Leibler Divergence.

3.4.8 χ^2-Distance

Eine weitere f-Divergence ist die χ^2-Distance. Gemäß Gl. 3.3 wird hierbei $f(t) = (t-1)^2$ verwendet. Dafür ergibt sich Pearson χ^2 [9]:

$$d_{\chi^2}(\mathbb{P},\mathbb{Q}) = \sum_{i=1}^{N} \frac{(P_i - Q_i)^2}{Q_i} \tag{3.26}$$

Die Gl. 3.26 kann ebenso in symmetrischer Form auftreten:

$$d_{\chi^2}(\mathbb{P},\mathbb{Q}) = \sum_{i=1}^{N} \frac{(P_i - Q_i)^2}{P_i + Q_i} \tag{3.27}$$

3.4.9 Hellinger Distance

Die Hellinger Distanz gehört zu den f-Divergences, wenn in Gl. 3.3 für $f(t) = (\sqrt{t}-1)^2$ eingesetzt wird. Sie ist nach Kap. 2.6 eine vollwertige Metrik [17]. Ebenso ist sie, analog zur Euklidischen Distanz in Gl. 3.18, eine L_2-Norm für Wahrscheinlichkeitsmaße. Sie ist durch folgende Rechenvorschrift definiert:

$$d_H(\mathbb{P},\mathbb{Q}) = \sqrt{\sum_{i=1}^{N} \left(\sqrt{P_i} - \sqrt{Q_i}\right)^2} \tag{3.28}$$

3.4.10 Mahalanobis Distance

Die Mahalanobis Distanz gehört zu der Klasse der Bregman-Divergences. Hierbei wird in Gl. 3.4 für $\varphi(t) = \frac{1}{2} x A x^T$ eingesetzt. Sie ist definiert durch [34]:

$$\| x - y \|_A = \sqrt{(\det A)^{\frac{1}{n}} (x-y) A^{-1} (x-y)^T} \tag{3.29}$$

In Gl. 3.29 sind x und y Vektoren, jeweils mit der Länge n. Die Matrix A ist üblicherweise die Kovarianzmatrix. Eine wichtige Eigenschaft ist, dass sie skalen- und translationsinvariant ist.

3.4.11 Itakura-Saito Distance

Die Itakura-Saito Distanz gehört zur Familie der Bregman-Divergenzen. Hierbei wird in Gl. 3.6 für $\varphi(t) = -\log(t)$ eingesetzt [26]. Dadurch ergibt sich:

$$d_\varphi(\mathbb{P},\mathbb{Q}) = \sum_{i=1}^{N} \left(\frac{p_i}{q_i} - \log\left(\frac{p_i}{q_i}\right) - 1 \right) \tag{3.30}$$

3.4.12 Structural Similarity Index

Der Structural Similarity Index ist, im Normalfall, eine Maßzahl, die Aussagen über die Ähnlichkeit zweier Bilder trifft. Es wurde gezeigt, dass diese ebenso für vektorielle bzw. sequenzielle Datensätze x und y, mit gleicher Länge, verwendet werden kann [27]. Die folgenden Definitionen, in Bezug auf den Structural Similarity Index, basieren auf [36]. Die formale Beschreibung eines Ähnlichkeitsmaß ist definiert durch:

$$S(x,y) = f\left(l(x,y),c(x,y),s(x,y)\right) \tag{3.31}$$

Hierbei wird die Ähnlichkeit in drei Komponenten aufgeteilt, die relativ unabhängig voneinander sind. $S(x,y)$ ist symmetrisch und auf [0,1] beschränkt. Der Fall $S(x,y) = 1$ liegt nur im Fall $x = y$ vor, d. h. wenn die zwei Datensätze identisch sind. Im Kontext der Bildverarbeitung beschreibt in Gl. 3.31 $l(x,y)$ den Lumineszenz-Vergleich, $c(x,y)$ den Kontrast-Vergleich und $s(x,y)$ den Struktur-Vergleich zweier Bilder.

Der Lumineszenz-Vergleich ist abhängig von den Mittelwerten μ_x und μ_y der entsprechenden Datensätze:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{3.32}$$

In Gl. 3.32 dient die Konstante zur Stabilisierung des Nenners, falls $\mu_x^2 + \mu_y^2$ sehr nahe an null ist. Eine Möglichkeit wäre die Wahl zu $C_1 = (K_1 L)^2$. Hierbei ist L die Spannweite der möglichen Werte. Im Falle von Pixel-Werten beträgt diese für 8-bit Graustufenbilder 255. K_1 ist eine sehr kleine Konstante $K_1 \ll 1$.

Der Kontrast-Vergleich $c(x,y)$ ist ähnlich beschrieben wie $l(x,y)$, basiert allerdings auf den Standardabweichungen σ_x und σ_y der entsprechenden Datensätze:

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{3.33}$$

Ähnlich wie in Gl. 3.32 dient die Konstante C_2 zur Stabilisierung und kann durch $C_2 = (K_2 L)^2$, mit $K_2 \ll 1$, bestimmt werden.

Analog zum Lumineszenz- und Kontrast-Vergleich kann der Struktur-Vergleich $s(x,y)$ beschrieben werden. Dieser bezieht die Korrelation σ_{xy} als Strukturfaktor mit ein:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\mu_x^2 + \sigma_y + C_3} \tag{3.34}$$

Wenn $l(x,y)$, $c(x,y)$ und $s(x,y)$ zusammengefasst werden, lässt sich der Structural Similarity Index formal, wie folgt, definieren:

$$S(x,y) = [l(x,y)]^\alpha \cdot [c(x,y)]^\beta \cdot [s(x,y)]^\gamma \tag{3.35}$$

In Gl. 3.35 gilt α, β, $\gamma > 0$. Diese Parameter können dazu verwendet werden, um die einzelnen Komponenten zu gewichten.

Für $\alpha = \beta = \gamma = 1$ (gleiche Gewichtung) sowie $C_3 = \frac{C_2}{2}$, ergibt sich folgende Form des Structural Similarity Index:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.36}$$

Es ist anzumerken, dass Gl. 3.36 ausschließlich von statistischen Kennzahlen der Datensätze x und y abhängig ist und dementsprechend eine Art von Similarity Coefficient darstellt.

3.4.13 Dynamic Time Warping

Beim Dynamic Time Warping Algorithmus werden zwei Zeitreihen $T = \{t_1, t_2, \ldots, t_n\}$ und $S = \{s_1, s_2, \ldots, s_m\}$, unabhängig ihrer Längen n und m, miteinander verglichen [8]. Hierbei liegt der Fokus auf der Ähnlichkeit ihrer Formen und nicht auf dem Zeitpunkt ihres Datenpunktes. Dadurch kann, trotz beispielsweise möglicher Verschiebungen, ähnliche Formen der beiden Datensätze, erkannt werden. Der Dynamic Time Warp kann ebenso auf allgemeine vektorielle Datensätze, wie beispielsweise zwei diskrete Dichtefunktionen $\mathbb{P}$ und $\mathbb{Q}$, angewandt werden [27].

Zu Beginn wird die $m \times n$ - Distanzmatrix aufgestellt, welche die Distanzen $d(\mathbb{P}_i, \mathbb{Q}_j)$ (vgl. Gl. 3.38) beinhaltet.

$$distMatrix = \begin{pmatrix} d(P_1,Q_1) & d(P_1,Q_2) & \cdots & d(P_1,Q,m) \\ d(P_2,Q_1) & d(P_2,Q_2) & & \vdots \\ \vdots & & \ddots & \\ d(P_n,Q_1) & \cdots & & d(P_n,Q_m) \end{pmatrix} \tag{3.37}$$

Hierbei entspricht $d(P_i,Q_j)$ der Distanz zwischen den Punkten P_i und Q_i mit $1 \leq i \leq n$ und $1 \leq j \leq m$. Ziel des DTW ist es, den kürzesten *Warping Path* $W = \{w_1, w_2, \ldots, w_K\}$, mit $\max(n,m) < K < m + n - 1$ und $w_k = distMatrix(i,j)$, zu finden. Für die einzelnen w_k bzw. Distanzen in der Distanzmatrix, lassen sich beliebige Distanzfunktionen $d(\cdot\,,\cdot\,)$ verwenden. Eine solche Distanzfunktion kann z. B., wie folgt beschrieben, definiert werden:

$$w(\mathbb{P},\mathbb{Q}) = \sum_{i,j} (P_i - Q_j)^2 \tag{3.38}$$

Der Algorithmus beginnt bei $w_1 = (1,1)$ und endet maximal bei $w_K = (n,m)$. In jedem Schritt wird, innerhalb der Distanzmatrix, die geringste Distanz aller anliegenden Punkte als nächstes w_k bestimmt, so dass der *Warping Path* W, wie in Gl. 3.39 beschrieben, minimiert wird.

$$DTW(\mathbb{P},\mathbb{Q}) = \min \sqrt{\sum_{k=1}^{K} w_k} \tag{3.39}$$

Für die einzelnen Distanzen $w_k = (i,j)$ bzw. $w_{k-1} = (i',j')$, mit $i,i' \leq n$ und $j,j' \leq m$, muss gelten:

- $w_1 = (1,1) \ \wedge \ w_K = (n,m)$
- $i - i' \leq 1 \ \wedge \ j - j' \leq 1$
- $i - i' \geq 0 \ \wedge \ j - j' \geq 0$

3.5 Gegenüberstellung der Evaluierungstechniken

Im weiteren Verlauf werden die in Kap. 3.4 beschriebenen Evaluierungstechniken, anhand ausgewählter Bewertungskriterien, bewertet. Nach einer Gegenüberstellung erfolgt die Empfehlung einer Evaluierungstechnik sowie deren Implementierung. Anschließend wird diese auf ausgewählte generierte Daten angewandt und die Ergebnisse validiert.

3.5.1 Bewertung der Evaluierungstechniken

Im Folgenden werden die in Kap. 3.4 beschriebenen Evaluierungstechniken, auf Basis von ausgewählten Bewertungskriterien, qualitativ bewertet. Die hierbei verwendeten Kriterien werden nachfolgend genannt und beschrieben.

Effektivität (E) Die Effektivität einer Evaluierungstechnik ist stellvertretend für die Qualität bzw. Güte des zugrundeliegenden Ähnlichkeitsmaßes. Im zugrundeliegenden Anwendungsfall kann diese als wichtigstes Kriterium betrachtet werden. Die Güte ist dabei, teils stark, von den entsprechenden Daten abhängig. Ein weiterer Aspekt von hoher Relevanz ist die Entwicklung bzw. der Verlauf der Ähnlichkeitswerte. Diese sollten plausible Werte über den gesamten Wertebereich liefern.

Komplexität (K) Zur Komplexität der Evaluierungstechniken gehört die Zeitkomplexität. Diese kann anhand der jeweiligen Implementierung bestimmt werden. Hierbei wird die Big-O Notation - oder auch Landau Symbol $\mathcal{O}$ - verwendet. Daraus folgt ebenso die entsprechende Laufzeit bzw. Effizienz der Evaluierungstechnik. Diese ist von der Anzahl an Datenpunkten abhängig.

Anwendbarkeit (A) Die Anwendbarkeit einer Evaluierungstechnik ist davon abhängig, wie einfach diese auf eine Wahrscheinlichkeitsdichtefunktion anwendbar ist. Es ist hierbei zu beachten, ob die zugrundeliegende Implementierung mit einem hohen Aufwand verbunden ist. Beispielsweise in Form von Optimierungsprozessen.

Transparenz (T) Die Transparenz ist ein Maß für die Klarheit und den Umfang der Evaluierungstechnik. Insbesondere ob genügend Literaturquellen diese Methode fundiert und ausführlich beschreiben.

Robustheit (R) Die Robustheit einer Evaluierungstechnik beschreibt, inwiefern diese mit Messungenauigkeiten umgehen kann. Hierzu zählen beispielsweise Peaks, Rauschen oder Ausreißer der Daten.

Parametrisierbarkeit (P) Die Parametrisierbarkeit gibt an, ob die Evaluierungstechnik Parameter enthält und falls dieses der Fall ist, inwiefern diese, nach Qualitätsaspekten, einzustellen sind. Beispielsweise existieren Parameter, die typischerweise über Optimierungsprozesse oder empirische Methoden bestimmt werden.

Interpretierbarkeit (I) Die Interpretierbarkeit gibt an, wie gut die Daten bzw. Ergebnisse der Evaluierungstechnik interpretiert werden können. Methoden, die standardisiert, auf einen Wertebereich beschränkt sind oder gegen einen bestimmten Wert konvergieren, haben hierbei eine hohe Aussagekraft. Falls dies nicht der Fall ist, können die Ergebnisse, auf Basis von Expertenwissen oder Erfahrungswerten, qualitativ bewertet bzw. interpretiert werden.

Nachfolgend werden die genannten Bewertungskriterien in drei verschiedene Stufen eingeteilt. Die beschriebenen Kriterien können, anhand der Symbole „+“ als positive, „o“ als neutrale sowie „-“ als negative Eigenschaft, auftreten. Ein genauere Beschreibung der Unterteilung erfolgt in Tab. 3.1.

Weiterhin erfolgt eine Gegenüberstellung der Evaluierungstechniken, auf Basis der beschriebenen Bewertungskriterien, in Tab. 3.2.

Effektivität	+	Evaluierungstechnik weißt eine hohe Qualität auf.
	o	Evaluierungstechnik weißt eine mäßige Qualität auf.
	-	Evaluierungstechnik weißt eine niedrige Qualität auf.
Komplexität	+	Evaluierungstechnik besitzt eine niedrige Zeitkomplexität und Laufzeit.
	o	Evaluierungstechnik besitzt eine mäßige Zeitkomplexität und Laufzeit.
	-	Evaluierungstechnik besitzt eine hohe Zeitkomplexität und Laufzeit.
Anwendbarkeit	+	Evaluierungstechnik lässt sich grundsätzlich leicht umsetzen oder (Referenz-)Implementierung(en) sind vorhanden.
	o	Evaluierungstechnik lässt sich nur bedingt einfach umsetzen. Es stehen keine (Referenz-)Implementierungen zur Verfügung.
	-	Evaluierungstechnik lässt sich grundlegend schwierig umsetzen und (Referenz-)Implementierungen stehen nicht zur Verfügung.
Transparenz	+	Evaluierungstechnik ist transparent beschrieben und verständlich.
	o	Evaluierungstechnik ist eher mäßig transparent beschrieben (z. B. mathematische Beschreibung nicht vollständig dokumentiert).
	-	Evaluierungstechnik setzt ein vertieftes mathematisches Wissen voraus und/oder dessen Beschreibung ist größtenteils unvollständig.
Robustheit	+	Evaluierungstechnik ist robust gegenüber Ungenauigkeiten der Daten (z. B. Messfehler, Rauschen oder Ausreißer).
	o	Die Evaluierungstechnik ist nicht vollkommen robust gegenüber Ungenauigkeiten der Daten.
	-	Die Evaluierungstechnik ist nicht robust gegenüber Ungenauigkeiten der Daten.
Parametrisierbarkeit	+	Die Evaluierungstechnik beinhaltet keine oder nur wenige Parameter. Falls diese Parameter umfasst, dann sind sie leicht einstellbar.
	o	Die Evaluierungstechnik beinhaltet mehrere Parameter. Diese sind nicht alle einfach einstellbar.
	-	Die Evaluierungstechnik beinhaltet mehrere bis viele Parameter. Zumindest die meisten davon sind nicht einfach einstellbar.
Interpretierbarkeit	+	Das Ergebnis der Evaluierungstechnik ist leicht interpretierbar.
	o	Das Ergebnis der Evaluierungstechnik ist mäßig interpretierbar.
	-	Das Ergebnis der Evaluierungstechnik lässt sich schwierig interpretieren.

Tab. 3.1: Beschreibung der Bewertungskriterien

Evaluierungstechnik	**E**	**K**	**A**	**T**	**R**	**P**	**I**
Kolmogorov-Smirnov	+	+ $\mathcal{O}(n)$	o	+	-	-	+
Wasserstein	+		o	o		-	
Cramér	o	+ $\mathcal{O}(n)$	o	+	o	+	+
Minkowski	o	+ $\mathcal{O}(n)$	+	+	o	o	+
Maximum Mean Discrepancy	o		o	o	+	-	
Total Variation Metric	+	+ $\mathcal{O}(n)$	+	+	o	+	+
Kullback-Leibler	+	+ $\mathcal{O}(n)$	+	+	+	+	+
Jensen-Shannon	+	+ $\mathcal{O}(n)$	+	+	+	+	+
χ^2	+	+ $\mathcal{O}(n)$	+	+	-	+	o
Hellinger	+	+ $\mathcal{O}(n)$	+	+	o	+	+
Mahalanobis	o	+	+	+	+	o	o
Itakura-Saito	+	+ $\mathcal{O}(n)$	+	o	o	+	o
Structural Similarity Index	o	+	+	+	o	o	+
Dynamic Time Warping	+	o $\mathcal{O}(n^2)$	+	o	+	o	o

Tab. 3.2: Bewertung der Evaluierungstechniken

Die beiden Farben in Tab. 3.2 haben die folgende Bedeutung:

- Cyan: Das Kriterium kann, aufgrund der mathematischen Beschreibung der zugrundeliegenden Evaluierungstechnik, nicht zweckdienlich eingestuft werden.
- Orange: Das Kriterium (zugehörig zur jeweiligen Evaluierungstechnik) kann, aufgrund begrenzter Literatur, nicht zweckdienlich eingestuft werden.

Die Implementierung der Kolmogorov-Smirnov Distanz erfolgt, wie in Gl. 3.11 beschrieben, durch die simple Ermittlung der maximalen Differenz zweier kumulativer Verteilungsfunktionen. Diese muss jedoch zunächst, für die übergebenen Wahrscheinlichkeitsdichtefunktionen, iterativ berechnet werden.

Die Wasserstein-1 Distanz - oder auch Earth Mover's Distance - beschreibt die „Kosten“, um eine Wahrscheinlichkeitsdichtefunktion, mit Hilfe einer Transformation, in eine andere zu überführen. Hierbei wird die statistische Geometrie der Dichtefunktionen berücksichtigt. Daraus folgt eine hohe Qualität der hieraus resultierenden Diskrepanzen. Die Eigenschaften der Wasserstein Distanz sind stark abhängig von der gewählten Transformation. Daraus folgt eine hohe Parametrisierbarkeit, mit der Folge, dass die Wahl dieser in einem Optimierungsproblem resultiert. Es lassen sich hierbei keine allgemeinen Aussagen über die entsprechende Komplexität der Transformation treffen. Diese bestimmt ebenso die Anwendbarkeit bzw. wie leicht sie zu implementieren ist. Zudem bestimmt diese die Robustheit gegenüber etwaigen Messungenauigkeiten.

Die Cramér Distanz entspricht der quadrierten Differenz von kumulativen Verteilungsfunktionen. Dementsprechend muss diese, in der Implementierung, iterativ berechnet werden.

Die Minkowski Metrik ist eine verallgemeinerte Evaluierungstechnik, die verschiedene Distanzen beinhaltet. Die Werte sind hierbei nicht normalisiert und müssen, in einem weiteren Schritt, qualitativ bewertet werden. Im Allgemeinen besitzen die entsprechenden Distanzen, da es sich um Metriken handelt (vgl. Kap. 2.6), eine hohe Qualität. In Gl. 3.16 sind die Spezialfälle, durch die Wahl des Parameters p, bestimmt. Ein größeres p bewirkt hierbei eine höhere Gewichtung der größten Differenz zweier Wahrscheinlichkeitsdichtefunktionen. Somit kann die Robustheit gegenüber Messungenauigkeiten, wie beispielsweise Ausreißern, angepasst werden. Die Wahl eines passenden Wertes für p stellt allerdings einen Optimierungsprozess dar.

Die Maximum Mean Discrepancy ist, wie viele andere Integral Probability Metrics, durch eine hohe Parametrisierbarkeit gekennzeichnet. Sie ist durch die Wahl einer reproduzierenden Kernelfunktion definiert. Die Zeitkomplexität ist hierbei von dem gewählten Kernel abhängig. Dieser bestimmt ebenso die Anwendbarkeit. Es handelt sich um eine komplexe Evaluierungstechnik, die ein vertieftes Verständnis in der Mengenlehre, speziell in Bezug auf den reproduzierenden Kernel Hilbertraum, voraussetzt. Die Kernelfunktion besitzt, auf Basis ihrer Definition, einen Glättungseffekt, wodurch sich eine hohe Robustheit gegenüber Messungenauigkeiten ergibt.

Die Total Variation Metric ist auf [0,1] beschränkt und verfügt dementsprechend über eine hohe Aussagekraft. Sie kann als f-Divergence sowie als Integral Probability Metric beschrieben werden.

Die Kullback-Leibler Divergence ist ein Maß der relativen Entropie und besitzt eine hohe Qualität. Um diese Divergence besser nachvollziehen zu können, ist ein grundlegendes Wissen über informationstheoretische Maße, speziell die Entropie, vorauszusetzen. Dennoch ist sie leicht zu implementieren. Durch ihre Skaleninvarianz ist sie Robust gegenüber Ausreißern. Sie ist Parameterunabhängig. Da es sich hierbei um ein Divergenzmaß handelt, sind die Ergebnisse leicht zu interpretieren. Umso näher das Ergebnis bei null liegt, desto ähnlicher sind sich die übergebenen Wahrscheinlichkeitsdichtefunktionen. Bei identischen Dichtefunktionen ergibt sich für die Kullback-Leibler Divergence ein Wert von null.

Die Jensen-Shannon Divergence ist eine symmetrische Version der Kullback-Leibler Divergence. Folglich besitzt sie ähnliche Eigenschaften. Die Evaluierungstechnik weist ebenfalls eine hohe Qualität auf. Sie ist robust gegenüber Messungenauigkeiten, leicht verständlich und leicht zu implementieren. Das Ergebnis lässt, sich analog zur Kullback-Leibler Divergence, interpretieren.

Die χ^2-Distance kann in symmetrischer oder asymmetrischer Form beschrieben werden. In beiden Fällen ist sie leicht verständlich und implementierbar.

Die Hellinger Distanz ist eine vollwertige Metrik und auf [0,1] beschränkt. Aufgrund der metrischen Eigenschaften, besitzt sie eine hohe Güte. Werte, die näher bei null liegen, weisen hierbei auf ähnliche Wahrscheinlichkeitsdichtefunktionen. Folglich lässt sie sich leicht interpretieren.

Die Mahalanobis Distanz ist eine Bregman-Divergence und nicht normalisiert bzw. auf einen bestimmten Wertebereich beschränkt. Umso näher ein Wert bei null liegt, desto ähnlicher sind sich die übergebenen Datensätze. Das Ergebnis muss folglich qualitativ bewertet werden. Sie besitzt den Parameter A, der meistens als die Kovarianzmatrix gewählt wird. Durch ihre Skalen- und Translationsinvarianz ist sie robust gegenüber Messungenauigkeiten. Es wird ein grundlegendes Verständnis der linearen Algebra und Statistik benötigt, um sie zu verstehen. Dennoch ist die Implementierung leicht durchzuführen.

Die Itakura-Saito Distanz erfordert, um sie zu verstehen, ein vertieftes Wissen über Divergenzmaße, speziell über die Bregman-Divergences. Sie ist dennoch leicht anzuwenden.

Der Structural Similarity Index ist auf [0,1] beschränkt. Hierbei ergibt sich, für den Fall von identischen Datensätzen, ein Wert von eins. Für den Fall, dass die Datensätze sich für keinen Punkt ähneln, ergibt sich ein Wert gegen null. Folglich lässt sich das Ergebnis leicht interpretieren. Da nur verschiedene statistische Kennzahlen miteinander verrechnet werden, weißt dieses Ähnlichkeitsmaß eine begrenzte Qualität auf. Die Evaluierungstechnik ist durch eine hohe Parametrisierbarkeit charakterisiert. Diese müssen, in einem Optimierungsprozess, bestimmt werden. Es handelt sich um ein sehr spezielles Ähnlichkeitsmaß aus der Bildverarbeitung, zu dem nicht viel Literatur zur Verfügung steht.

Der Dynamic Time Warping Algorithmus ist nicht normalisiert oder auf einen bestimmten Wertebereich beschränkt. Ein Ergebnis, dass nah bei null liegt, weißt auf eine hohe Ähnlichkeit der Wahrscheinlichkeitsdichtefunktionen. Da es sich allerdings um kein Divergenzmaß handelt, muss das Ergebnis noch qualitativ bewertet werden. Der Algorithmus verwendet eine weitere Methode der Distanzberechnung. Dadurch ergibt sich eine hohe Parametrisierbarkeit. Es muss dementsprechend, vor der Implementierung, eine passende Methode bestimmt werden.

Ebenso ist der Algorithmus robust gegenüber Messungenauigkeiten. Der Grund hierfür liegt, unter anderem, in der Translationsinvarianz dieser Methode. Zudem wird, bei der Berechnung, die statistische Geometrie der Dichtefunktionen miteinbezogen. Hierdurch ergibt sich eine hohe Qualität.

Anhand der in Tab. 3.2 durchgeführten Bewertung und Gegenüberstellung der Evaluierungstechniken, ergeben sich drei äußerst relevante Methoden. Diese sind namentlich die Kullback-Leibler und Jensen-Shannon Divergence sowie die Hellinger Distance. Jeder dieser Evaluierungstechniken weißt, in Bezug auf die genannten Bewertungskriterien, ausschließlich positive Einstufungen auf. Bei einer solchen Gleichheit (Patt-Situation) ist es eine bewährte Methode, eine Endauswahl, auf Basis der Literaturnennungen, im (angrenzenden) Anwendungsfall, zu treffen. Dies trifft auf die Kullback-Leibler Divergenz zu. Aus diesem Grund wird die Kullback-Leibler Divergenz, als entsprechende Evaluierungstechnik, empfohlen.

3.5.2 Implementierung

Die Evaluierung von generativen Algorithmen, unter Verwendung der Wahrscheinlichkeitsdichtefunktionen, erfolgt mit Hilfe der in Kap. 3.5.1 empfohlenen Kullback-Leibler Divergenz. Hierzu wird die Evaluierungstechnik implementiert. Die Komplettumsetzung bzw. Pipeline der angrenzenden Doktorarbeit verwendet die Programmiersprache Python [32]. Aus diesem Grund wird diese ebenso für die Implementierung verwendet. Python umfasst hierbei eine Vielzahl an Package bzw. Bibliotheken, die im Kontext der Künstlichen Intelligenz und deren Teilgebiete, verwendet werden können [29][15]. Der Funktionsprototyp wird wie folgt definiert:

$$\mathbf{[y1,\ y2,\ y3] = f(P,Q)}$$

Hierbei werden der Funktion zwei Parameter **P** und **Q** übergeben. Diese entsprechen den Wahrscheinlichkeitsdichtefunktionen. Der Rückgabewert **y1** stellt die Kullback-Leibler Divergenz dar. Neben dieser sind ebenso zwei weitere Kennzahlen für die eigentliche Validierung vorgesehen. Hierbei wird die Kullback-Leibler Divergenz iterativ berechnet. Die Größe **y2** beschreibt den maximalen Einzelwert der Kullback-Leibler Divergenz zwischen zwei Punkten der beiden Wahrscheinlichkeitsdichtefunktionen (maximale Diskrepanz). Dieser macht Aussagen über die höchste Diskrepanz von zwei Werten der entsprechenden Wahrscheinlichkeitsdichtefunktionen. Die Größe **y3** gibt den maximalen Gradienten der jeweiligen Einzelwerte an. Die zwei Rückgabewerte **y2** und **y3** dienen zur Validierung, inwieweit die beiden Wahrscheinlichkeitsdichtefunktionen in plausiblen Wertebereichen liegen.

Die Implementierung erfolgt in List. 3.1. Hierbei werden in einem ersten Schritt, mit Hilfe der *where* Funktion des NumPy Packages, die Einzelwerte der Kullback-Leibler Divergenz bestimmt. Die Ermittlung der eigentlichen Kullback-Leibler Divergenz erfolgt mit Hilfe der *sum* Funktion. Der maximale Gradient wird durch die Verkettung der *max* und *gradient* Funktionen bestimmt. Abschließend werden die drei genannten Kennzahlen zurückgegeben.

```python
def kullback_leibler(p,q):
    """

    """
    # Einzelwerte der Kullback-Leibler Divergenz
    kl_vals = np.where(q != 0, p * np.log(p / q), 0)
    # Kullback-Leibler Divergenz
    kl_div = np.sum(kl_vals)
    # maximaler Einzelwert
    max_val = np.max(kl_vals)
    # maximaler Gradient
    max_gradient = np.max(np.gradient(kl_vals))
    return kl_div, max_val, max_gradient
```

List. 3.1: Python Code: Kullback-Leibler Divergence

3.5.3 Datenanalyse

Im Folgenden wird die implementierte Kullback-Leibler Divergenz auf verschiedene synthetisch generierte Datensätze angewandt, mit dem Ziel, die Ergebnisse zu validieren. Hierbei werden zwei Fälle betrachtet. In einem ersten Schritt werden synthetisch generierte Sinusfunktionen, als sogenannte Baseline (Referenz), analysiert. Darauffolgend werden synthetisch generierte Überholvorgänge betrachtet. In beiden Fällen werden die spezifischen Variationsparameter nur geringfügig verändert. Dadurch fällt es leichter, eine qualitative Aussage über die generierten Daten zu machen, da diese leichter zu interpretieren sind. Die hierbei verwendeten Wahrscheinlichkeitsdichtefunktionen werden allesamt mittels der Histogramm-Spline-Approximation bestimmt.

Sinusfunktion

In Abb. 3.4 sind 100 synthetisch generierte Sinusfunktionen als Zeitreihe dargestellt. Diese wurden mit Hilfe des Python Codes in List. A.1 generiert. Es wurden hierbei 1000 Datenpunkte verwendet. Diese sind durch folgende Rechenvorschrift definiert:

$$f(t) = a \cdot sin(b \cdot (t + c)) + d \tag{3.40}$$

Die Variationsparameter a, b, c und d besitzen jeweils eine Abweichung von $\pm 0{,}01$ mit einer Schrittweite von 0,001. Durch diese Variation ergeben sich verschiedene Verläufe für die Funktionen. In Abb. 3.5 sind diese Varianzen, für einen Ausschnitt, genauer dargestellt.

Die Wahrscheinlichkeitsdichtefunktionen der Sinusfunktionen sind, in Abb. 3.6, über der Auslenkung y, dargestellt. Die entsprechenden Dichtefunktionen wurden mittels der Histogramm-Spline-Approximation bestimmt (vgl. Kap. 2.5). Hierbei wurden sieben Bins für das zugrundeliegende Histogramm und ein Annäherungspolynom 5. Grades verwendet.

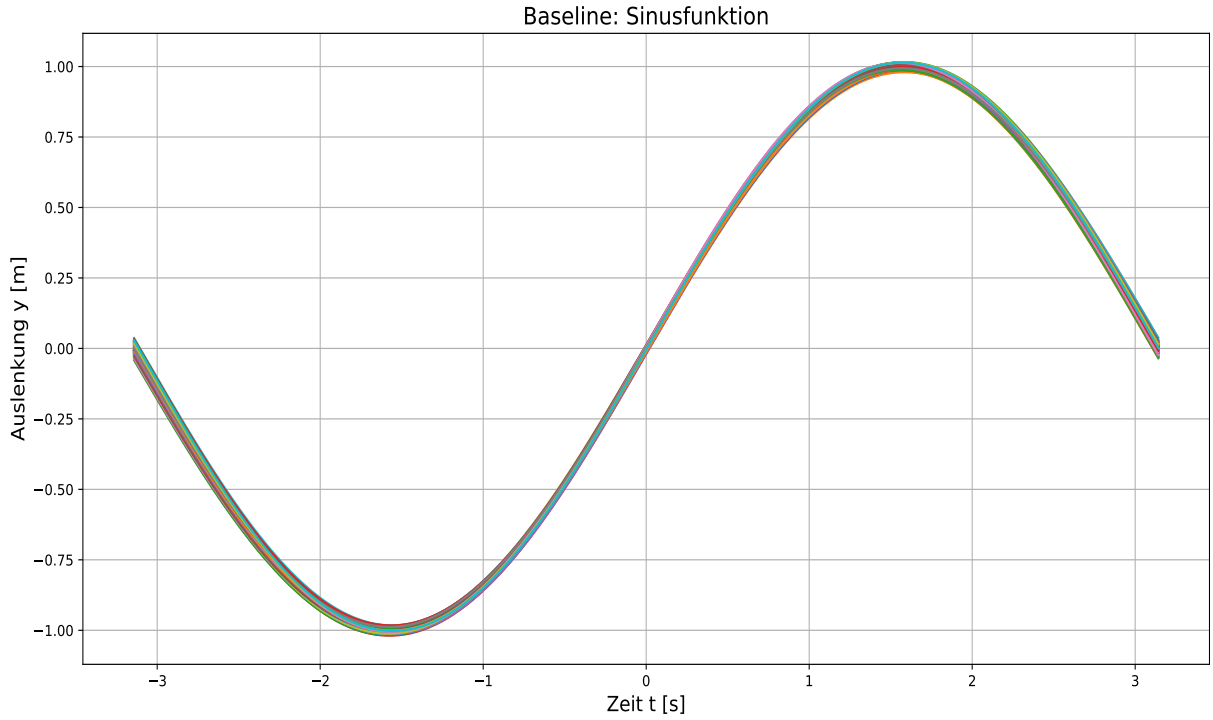

Abb. 3.4: Sinusfunktionen

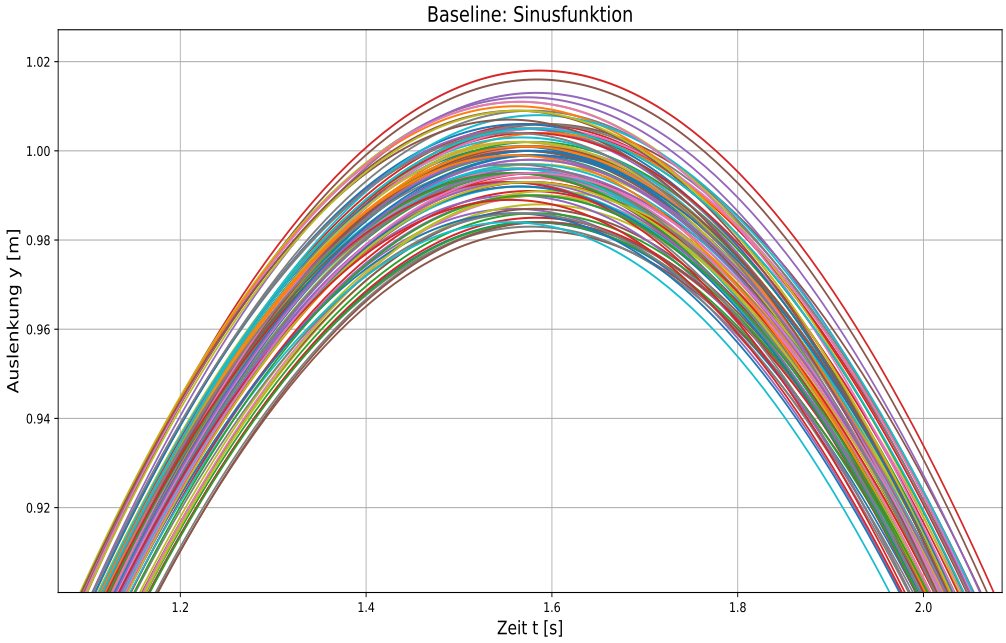

Abb. 3.5: Ausschnitt der Sinusfunktionen

In Bezug auf Sinusfunktionen sind hierbei drei Punkte von hoher Relevanz. Diese sind namentlich der Hochpunkt, Tiefpunkt sowie der Nulldurchgang. Für die Extrempunkte ergeben sich hierbei hohe Varianzen. Da die Steigung um diese Punkte gering ist, ändert sich die Auslenkung nur schwach.
Folglich sind für diese Auslenkungen die meisten Datenpunkte vorhanden. Dies spiegelt sich ebenso in den Dichtefunktionen in Abb. 3.6 wieder. Hierbei sind für die Auslenkungen $y = \pm 1{,}00$ die höchsten relativen Wahrscheinlichkeiten zu verzeichnen. Die Dichtefunktionen weisen hierbei ebenso hohe Varianzen auf.

In den Nulldurchgängen besitzen die, in Abb. 3.4 dargestellten Sinusfunktionen, die höchste Steigung. Folglich ändert sich die Auslenkung, um diese Stellen, am schnellsten, über der Zeit. Hierbei ergeben sich, bei Variation der Parameter, hohe Varianzen. Diese sind ebenso in den Wahrscheinlichkeitsdichtefunktionen in Abb. 3.6 bzw. Abb. 3.7 zu verzeichnen.

Für die verschiedenen Dichtefunktionen sind einige Punkte mit einer höheren Konzentration zu beobachten. Dies wird vor allem in Abb. 3.7, für die Auslenkungen $y \approx \pm 2{,}5$, deutlich. Der Grund hierfür liegt in der Histogramm-Spline-Approximation selbst. Hierbei werden Splines durch die Klassenmitten des zugrundeliegenden Histogrammes gelegt. Dadurch ergeben sich ebenso Punkte mit höherer Konzentration bei den Klassenmitten.

In Abb. 3.8 sind die Kullback-Leibler Distanzen für die Wahrscheinlichkeitsdichtefunktionen von 99 Sinusfunktionen dargestellt. Eine der 100 Dichtefunktionen wurde hierbei als Referenz-Dichtefunktion verwendet. Es ergeben sich ausschließlich Werte nahe null. Dies spricht für eine sehr hohe Ähnlichkeit der Kurven.

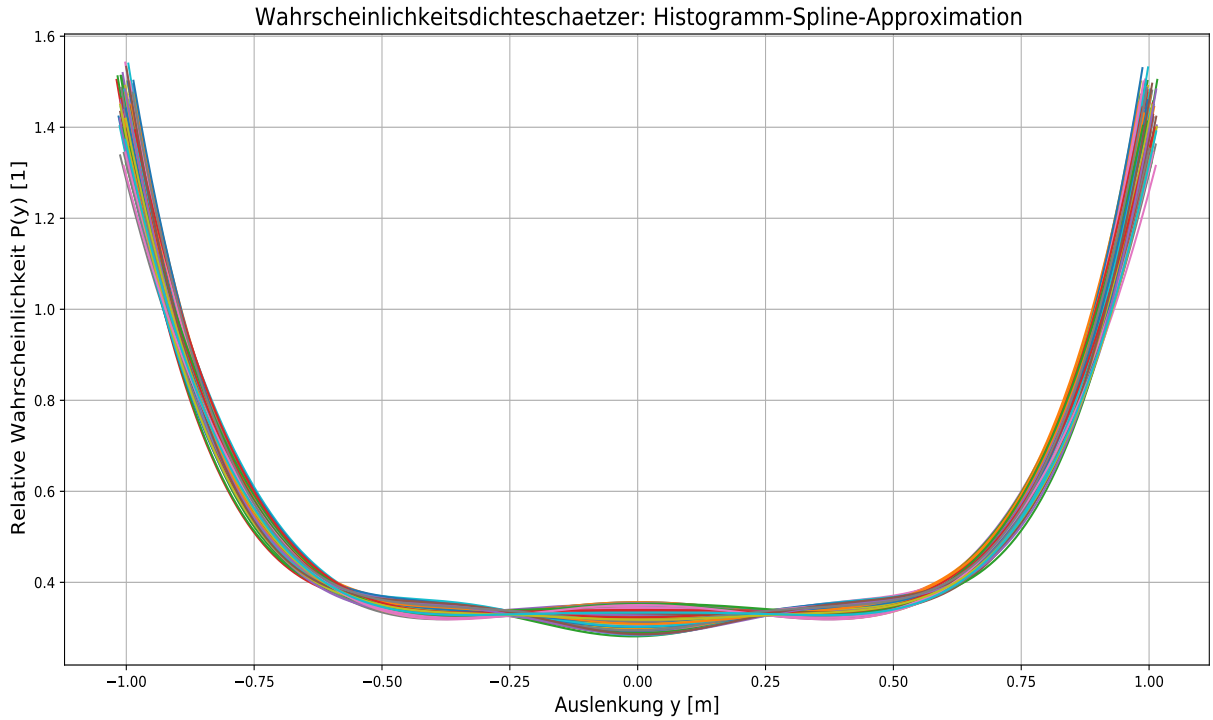

Abb. 3.6: Wahrscheinlichkeitsdichtefunktionen von Sinusfunktionen

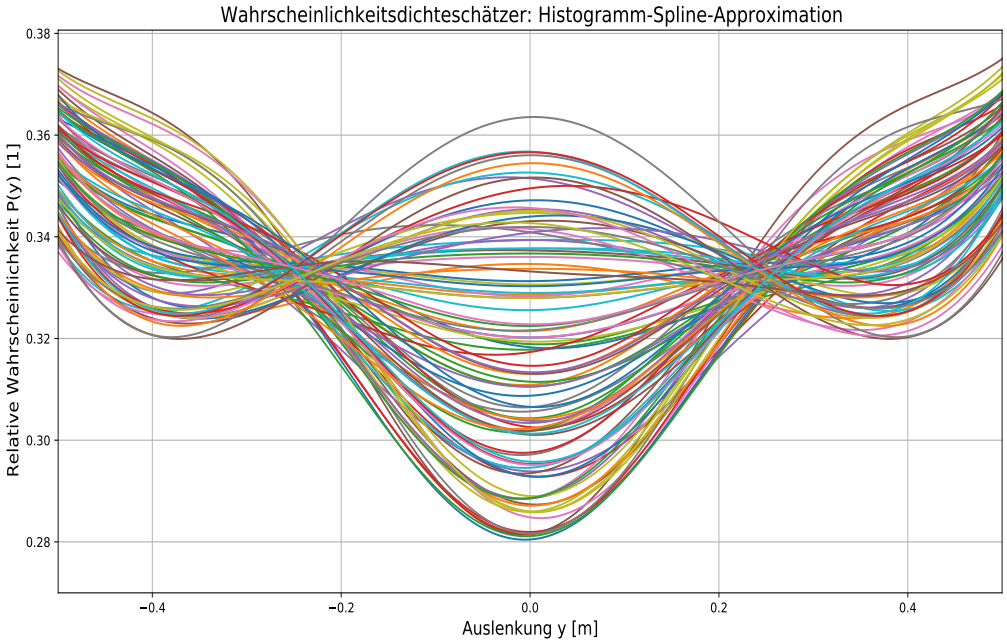

Abb. 3.7: Ausschnitt der Wahrscheinlichkeitsdichtefunktionen von Sinusfunktionen

Neben den eigentlichen aufsummierten Kullback-Leibler Divergenzen sind deren Einzelwerte relevant. Durch diese lassen sich die Verläufe der Wahrscheinlichkeitsdichtefunktionen validieren. In Abb. 3.9 sind die Einzelwerte für 99 Samples dargestellt.
Es sind die verschiedenen Bereiche mit hoher Varianz zu erkennen, die mit den Varianzen aus Abb. 3.6 übereinstimmen. Es sind ebenfalls die sieben Bins des zugrundeliegenden Histogramms zu erkennen, die für die Histogramm-Spline-Approximation verwendet werden. Die Grenzen dieser Histogramme liegen an den Stellen mit einer höheren Konzentration.

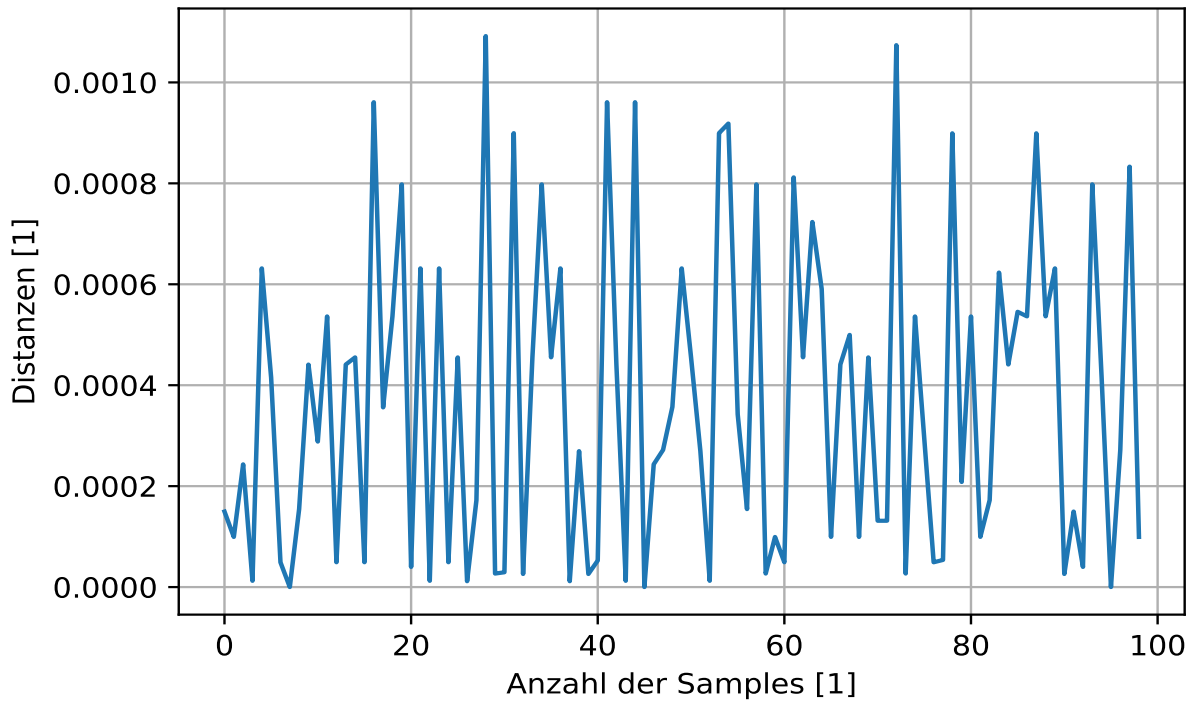

Abb. 3.8: Kullback-Leibler Distanzen für Dichtefunktionen der Sinusfunktionen

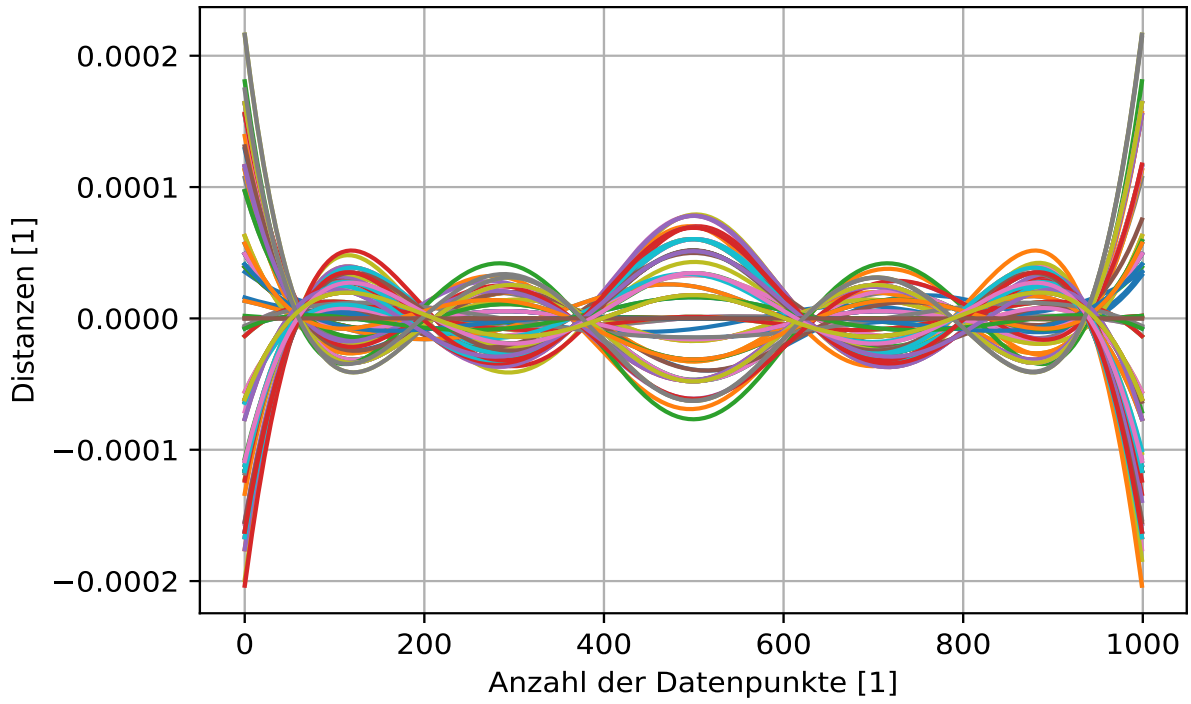

Abb. 3.9: Einzelwerte der Kullback-Leibler Distanzen für Dichtefunktionen der Sinusfunktionen

Überholvorgang

Nachfolgend werden synthetisch generierte Überholvorgänge, vergleichbar mit denen in Abb. 3.1, analysiert. Die dazugehörigen Bewegungsdaten sind in Abb. 3.10 als Zeitreihen, für die Längs- und Querrichtung, dargestellt. Diese wurden mit Hilfe des Python Codes in List. A.2 generiert. Es wurden hierbei 1602 Datenpunkte verwendet.

Die entsprechenden Verläufe basieren auf den kinematischen bzw. physikalischen Beschreibungen in [31]. Hierbei besitzen die Variationsparameter γ_1 und γ_2 jeweils eine Abweichung von $\pm 0{,}1$ mit einer Schrittweite von 0,001. Durch diese Variation der Parameter ergeben sich unterschiedliche Bewegungsverläufe des überholenden Fahrzeugs.

Es sind deutliche Varianzen in den Bewegungsverläufen für die Ein- und Ausschervorgänge zu erkennen. In Abb. 3.11 sind diese beispielhaft für den Ausschervorgang dargestellt. Im Gegensatz dazu sind, während des Überholvorgangs auf der Gegenfahrbahn, die Variation der einzelnen Verläufe gering.

In Abb. 3.12 sind die Wahrscheinlichkeitsdichtefunktionen über der Querrichtung dargestellt. Diese wurden ebenso mit Hilfe der Histogramm-Spline-Approximation bestimmt. Hierbei wurden 13 Bins für das zugrundeliegende Histogramm und ein Annäherungspolynom 11. Grades verwendet. Insgesamt werden die Variationen der einzelnen Wahrscheinlichkeitsdichtefunktionen deutlich. Diese ergeben sich durch die unterschiedlichen Bewegungsverläufe.

Für eine Querrichtung nahe null befindet sich das Fahrzeug auf der ursprünglichen Fahrbahn.

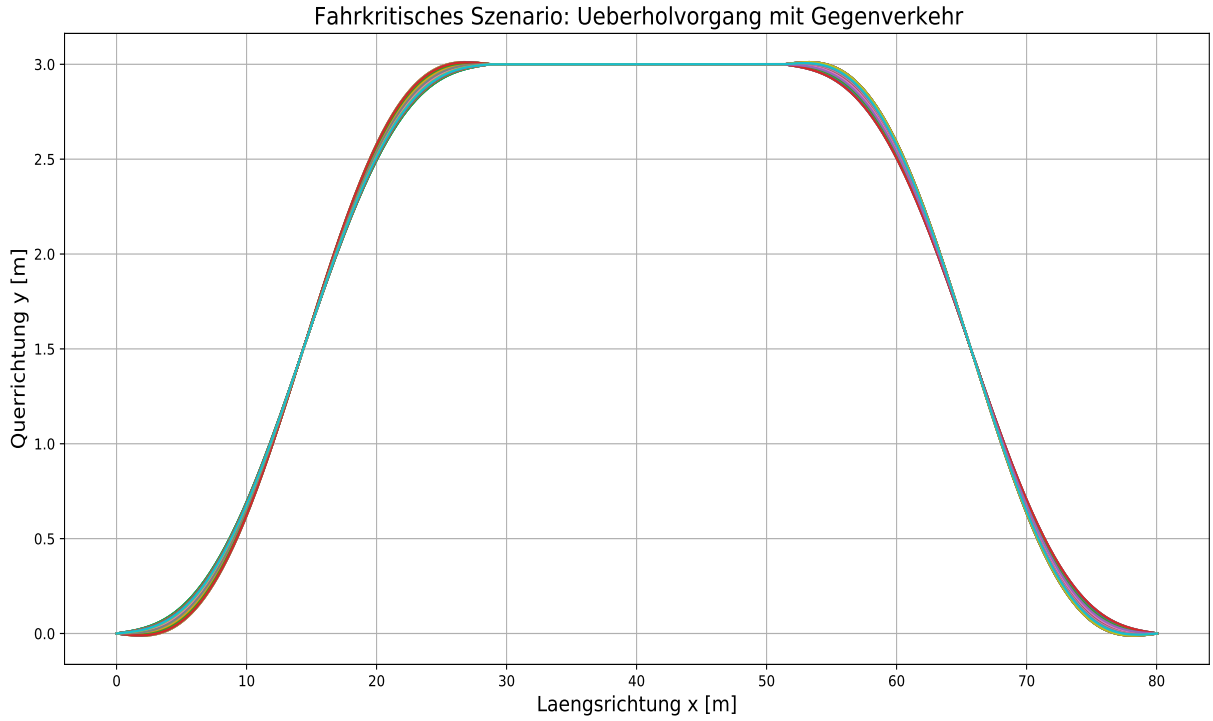

Abb. 3.10: Bewegungsdaten von synthetisch generierten Überholvorgängen

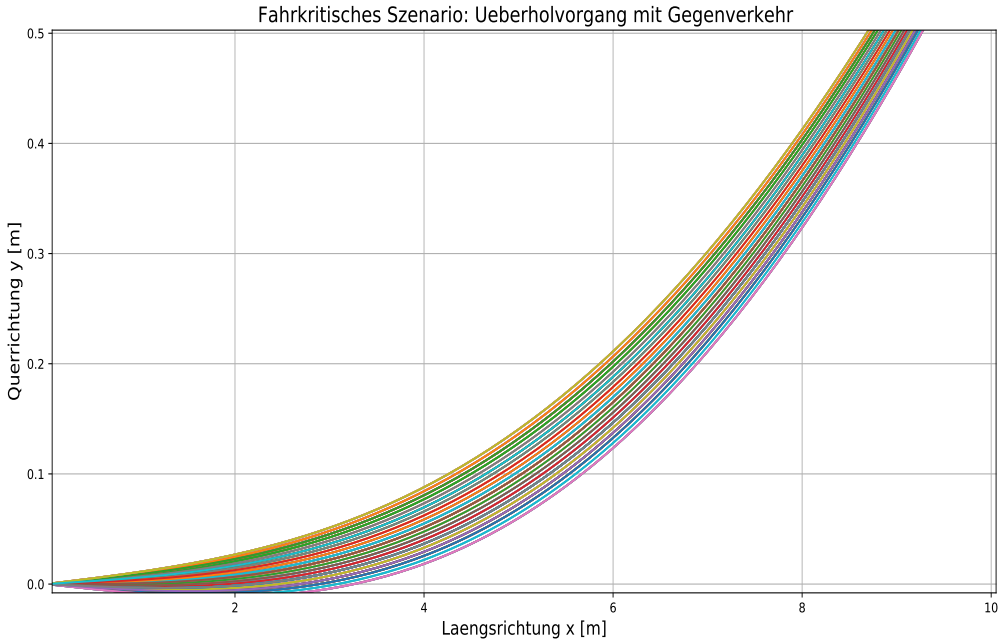

Abb. 3.11: Ausschnitt der Bewegungsdaten von synthetisch generierten Überholvorgängen

Hierbei ergeben sich relativ hohe Werte für die relative Wahrscheinlichkeit. Ebenso sind hohe Varianzen zu beobachten, die aus den unterschiedlichen Bewegungsverläufen resultieren. Für eine Querrichtung nahe drei befindet sich das Fahrzeug auf der Gegenfahrbahn. Hierbei sind, aufgrund der geringen Variation in den Bewegungsverläufen während des Überholvorgangs, die höchsten relativen Wahrscheinlichkeiten zu verzeichnen. Die Dauer des Ein- und Ausschervorgangs ist, relativ zum Überholvorgang, gering. Aus diesem Grund ergibt sich, für eine Auslenkung in Querrichtung von 0,5 bis 2,5 m, eine relative Wahrscheinlichkeit nahe null.

In Abb. 3.13 ist ein Ausschnitt der Dichtefunktionen über der Querrichtung, zugehörig zur Überholspur des Überholvorgangs auf der Gegenfahrbahn, dargestellt. Die Variationen der einzelnen Bewegungsverläufe sind auf dieser gering. Dadurch sind die relativen Wahrscheinlichkeiten dort entsprechend größer. Die Variationen der einzelnen Wahrscheinlichkeitsdichtefunktionen werden hierbei wiederum verdeutlicht. Der Grund hierfür liegt in den unterschiedlichen Phasen der Ein- und Ausschervorgängen.

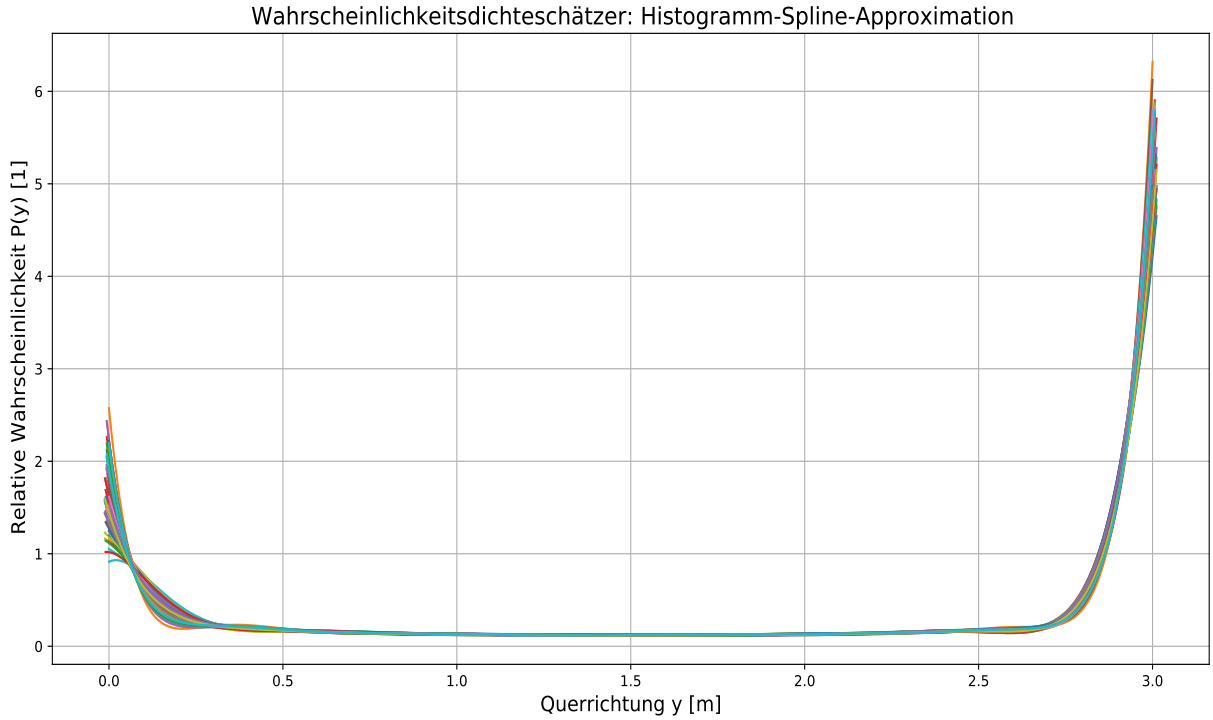

Abb. 3.12: Dichtefunktionen von synthetisch generierten Überholvorgängen

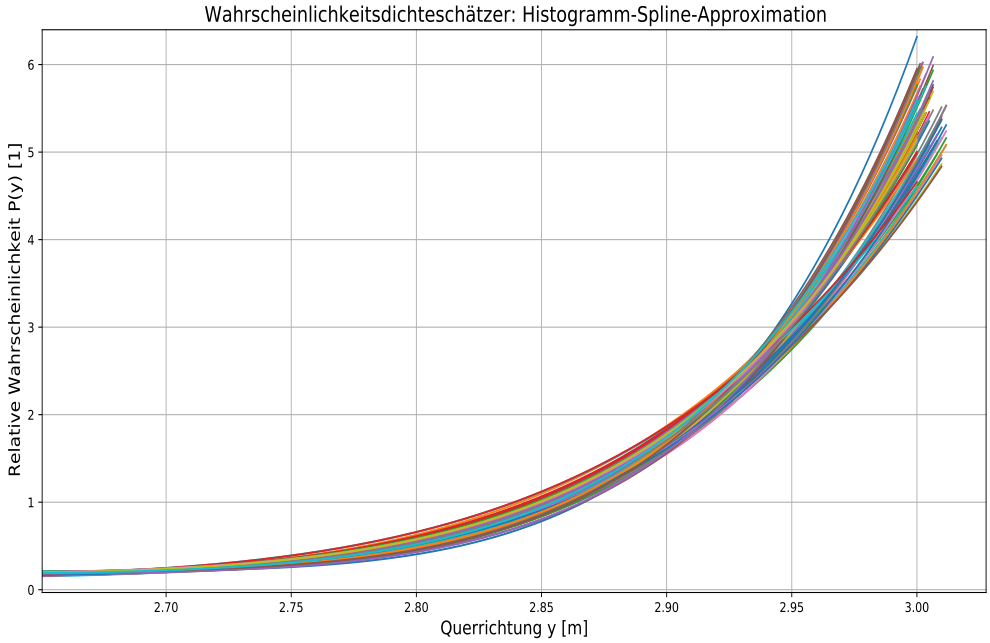

Abb. 3.13: Ausschnitt der Dichtefunktionen von synthetisch generierten Überholvorgängen

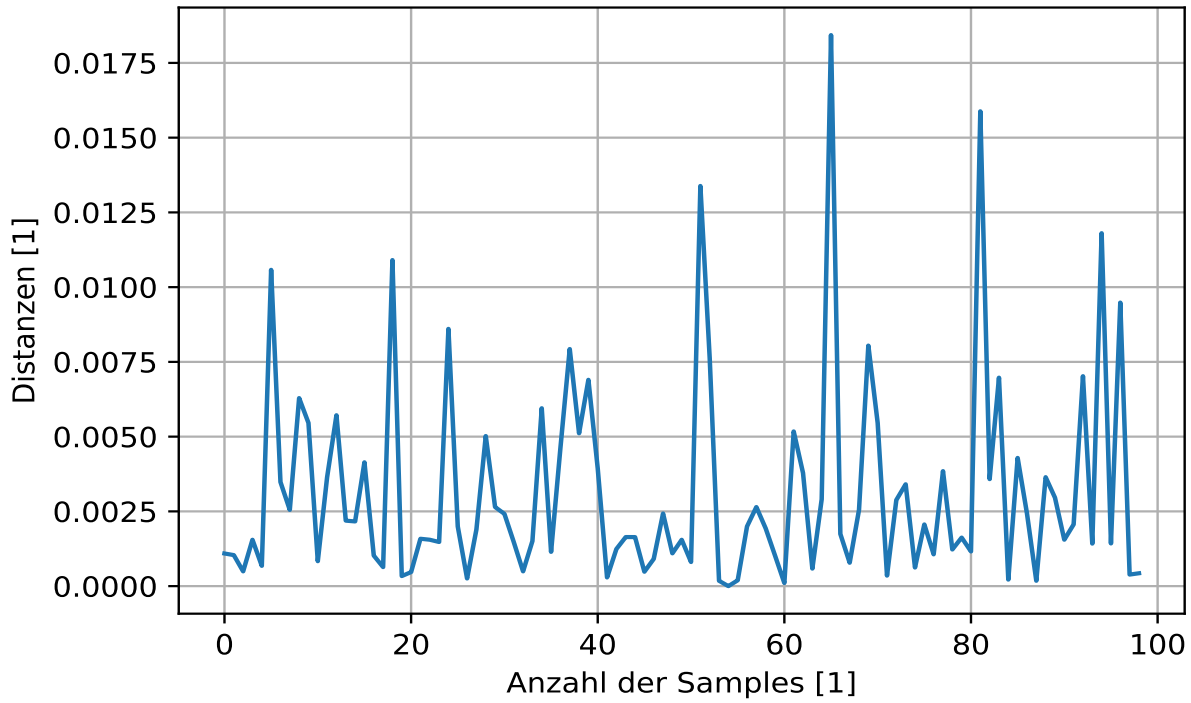

Abb. 3.14: Kullback-Leibler Distanzen für Dichtefunktionen der synthetisch generierten Überholvorgängen

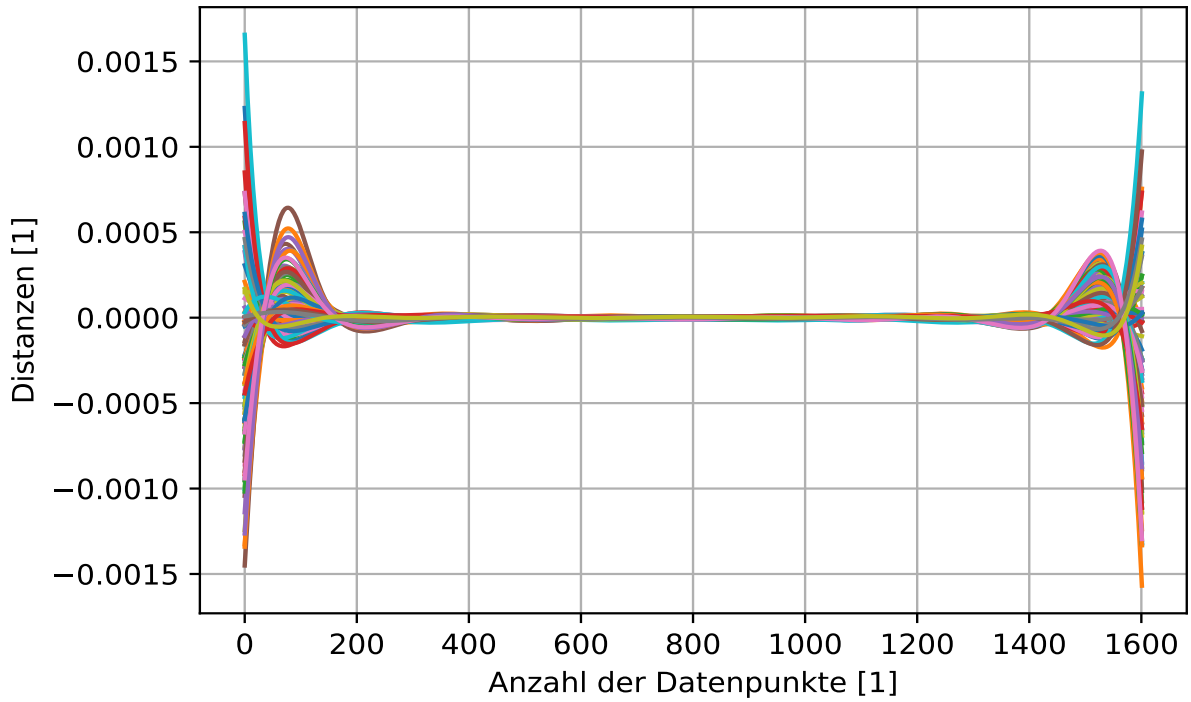

Abb. 3.15: Kullback-Leibler Distanzen für Dichtefunktionen der synthetisch generierten Überholvorgängen

Für die Ermittlung der Kullback-Leibler Distanzen wurde, aus den 100 Wahrscheinlichkeitsdichtefunktionen der Überholvorgänge, eine als Referenz verwendet. In Abb. 3.14 sind die Kullback-Leibler Distanzen für 99 Samples dargestellt. Es ergibt sich stets ein Wert nahe null. Diese Werte sprechen für eine hohe Ähnlichkeit der Dichtefunktionen.

Die Einzelwerte der Kullback-Leibler Divergenzen sind in Abb. 3.15 dargestellt. Die Verläufe zeigen eine hohe Varianz für den Ein- und Ausschervorgang sowie für den Überholvorgang. Für den Bereich zwischen den Fahrbahnen ergibt sich, wie in den entsprechenden Dichtefunktionen, ein annähernd linearer Verlauf. Es sind hierbei, wie im Falle der Sinusfunktionen, verschiedene Punkte mit einer höheren Konzentration zu beobachten. Diese resultieren ebenso aus der Histogramm-Spline-Approximation.

Es ist zu beachten, dass die Werte hierbei um das zehnfache höher sind, als die Kullback-Leibler Distanzen der Dichtefunktionen von Sinusfunktionen. Dies lässt sich anhand der zugrundeliegenden mathematischen Beschreibung erklären. Die Definition einer Sinuskurve ist relativ simpel. Sie besitzt, in Abhängigkeit der Variationsparameter, stets einen ähnlichen periodischen Verlauf. Im Gegensatz dazu erfolgt die mathematische Beschreibung eines Überholvorgangs wesentlich komplexer [31].

4 Schluss

Im Folgenden wird eine Zusammenfassung der Bachelorarbeit sowie zukünftig aufbauende Arbeiten vorgestellt.

4.1 Zusammenfassung

Jährlich sterben etwa 1,3 Millionen Menschen bei Verkehrsunfällen. Die häufigste Unfallursache ist hierbei menschliches Fehlverhalten. Aus diesem Grund besitzt das autonome Fahren das Potential, diese auf ein Minimum zu reduzieren. Hierbei werden alle Fahraufgaben vom Fahrzeug übernommen. Diese müssen, um die Sicherheit aller Verkehrsteilnehmer zu gewährleisten, jede Situation, im dynamischen Straßenverkehr, regelkonform bewältigen können. Ebenso müssen die dazugehörigen System- und Softwarekomponenten ausgiebig getestet werden. Hierzu sind eine große Menge an Daten für verschiedenste Fahrsituationen erforderlich, um autonome Fahrzeuge auf diese vorzubereiten. Für sicherheitskritische Fahrszenarien sind diese, aufgrund ihrer Kritikalität, relativ selten vorhanden. Hierbei können generative Algorithmen verwendet werden, um synthetische Bewegungsdaten für diese Szenarien zu generieren.

Die generierten Datensätze müssen, um physikalisch plausible sicherheitskritische Fahrszenarien zu gewährleisten, anhand von Evaluierungstechniken, validiert werden. Hierbei wird typischerweise die Wahrscheinlichkeitsdichtefunktion des zugrundeliegenden Datensatzes betrachtet. Im Rahmen dieser Arbeit wurden Evaluierungstechniken für derartige Wahrscheinlichkeitsdichtefunktionen untersucht. Dazu wurde, auf Basis einer umfangreichen Literaturrecherche, eine Liste mit existierenden Evaluierungstechniken, erstellt. Hierbei wurden insgesamt ca. 200 Evaluierungstechniken herangezogen.

In einem weiteren Schritt sind die, für den Anwendungsfall irrelevanten, Evaluierungstechniken ausgefiltert worden. Die daraus resultierende Liste enthielt 13 hoch relevante Evaluierungstechniken. Diese konnten, auf Basis ihrer mathematischen Beschreibung, in verschiedene Gruppen eingeteilt werden. Daraus entstand eine neuartige Taxonomie von Evaluierungstechniken für Wahrscheinlichkeitsdichtefunktionen.

Die einzelnen Evaluierungstechniken wurden detailliert mathematisch beschrieben. In einem weiteren Schritt erfolgte, anhand von definierten Bewertungskriterien, für den zugrundeliegenden Anwendungsfall, eine Gegenüberstellung dieser Evaluierungstechniken. Hierbei konnte eine qualitative Empfehlung ausgesprochen werden. Abschließend wurde die empfohlene Evaluierungstechnik implementiert und anhand von synthetischen Datensätzen validiert.

4.2 Ausblick

Die empfohlene Evaluierungstechnik wird in einer weiterführenden Arbeit verwendet. Hierbei wird die entsprechende Python-Implementierung in die Pipeline eines generativen Algorithmus eingebettet. Dieser generiert synthetische Bewegungsdaten von sicherheitskritische Fahrszenarien. Die Kullback-Leibler Divergenz kann hierbei verwendet werden, um diese Datensätze, gemäß der zugrundeliegenden Wahrscheinlichkeitsdichtefunktionen, zu validieren.

In diesem Zusammenhang ist ebenso eine stärkere Prüfung von anderen, nicht direkt empfohlenen, Evaluierungstechniken denkbar. Speziell die Wasserstein-1 Metrik oder Maximum Mean Discrepancy besitzen eine hohe Qualität. Hierbei müssen allerdings die entsprechenden Parameter bestimmt werden. Dies ist mit einem weiteren Optimierungsprozess sowie einer vertieften Literaturrecherche verbunden.

Im Rahmen dieser Arbeit wurden ausschließlich synthetische Daten analysiert. Diese weisen geringe Messungenauigkeiten auf und besitzen einen festen Wertebereich. Ebenso ist die Auswirkung von Realdaten sowie verschiedenen Wertebereichen auf die Evaluierungstechnik relevant. Schließlich soll diese reale sicherheitskritische Szenarien validieren können.

A Anhang

A.1 Sinusfunktion (Baseline)

```
# Bibliotheken einbinden
from numpy import asarray, arange, linspace, ones_like, histogram,
    diff, insert, cumsum, trapz
from math import pi, sin
from random import gauss, choice
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline

# Berechnung des Integralwerts unter einer 2D-Kurve
def integrateT(x, y):
    area = trapz(y=y, x=x)
    return area

# Histogramm-Spline-Approximation
def histogramToSplinePDF(data,k=5,bc_type='not-a-knot'):
    weights = ones_like(data)/float(len(data))
    heights, bin_borders =
        histogram(data,bins=7,normed=True,weights=weights)
    centers = bin_borders[:-1] + diff(bin_borders) / 2
    bin_width = centers[1]-centers[0]
    t = linspace( centers[0]-bin_width/2, centers[-1]+bin_width/2,
        len(centers)+1 )
    dt = diff(t)
    cumsumTemp = heights * dt
    cumsumTemp = insert(cumsumTemp,0,0)
    Fvals = cumsum(cumsumTemp)
    spl = make_interp_spline(t, Fvals, k=k, bc_type=bc_type)
    splD = spl.derivative()
    t2 =
        linspace(centers[0]-bin_width/2,centers[-1]+bin_width/2,len(data))
    y2 = splD(t2)
    print('Integral-Wert:␣' + str(integrateT(t2,y2)))
    return t2,y2

# Hilfsfunktion - Datensaetze in CSV-Dateien schreiben
def dataWriterHelperCSV(f,key,value,N):
    k = 0
    f.write(key + ',')
    for i in value:
        f.write(str(i))
        if N == 1:
```

```
38              f.write("")
39          if k < N-1:
40              f.write(",")
41          k = k + 1
42      f.write("\n")
43
44  # Datensaetze in CSV-Dateien schreiben
45  def writeDataIntoCSV(filename,x,y):
46      N = len(x)
47      with open(filename,'w') as f:
48          dataWriterHelperCSV(f,'x',x,N)
49          dataWriterHelperCSV(f,'y',y,N)
50
51  # Funktion zur Bestimmung der Sinusfunktion als Baseline
52  def callBaselineSinusFunction(alpha1,alpha2,alpha3,alpha4):
53      a = 1+alpha1
54      b = 1+alpha2
55      c = 0+alpha3
56      d = 0+alpha4
57      tMin = -1*pi
58      tMax = 1*pi
59      tN = 1000
60      t = linspace(tMin,tMax,tN)
61      y = []
62      for _, val in enumerate(t):
63          y.append( a*sin(b*(val+c))+d )
64      t = asarray(t)
65      y = asarray(y)
66      return t, y
67
68  # Ansatz der Datensaetze bzw. Samples
69  N = 100
70
71  # Variationswerte (initial)
72  alpha1_init = 0
73  alpha2_init = 0
74  alpha3_init = 0
75  alpha4_init = 0
76
77  # Vektor fuer die Variationswerte (zu Beginn gleich der Initialwerte)
78  alpha1V = [alpha1_init]
79  alpha2V = [alpha2_init]
80  alpha3V = [alpha3_init]
81  alpha4V = [alpha4_init]
82
83  # Wertebereiche der Variationswerte
84  alpha1 = arange(start=-0.01, stop=0.01, step=0.001)
85  alpha2 = arange(start=-0.01, stop=0.01, step=0.001)
86  alpha3 = arange(start=-0.01, stop=0.01, step=0.001)
87  alpha4 = arange(start=-0.01, stop=0.01, step=0.001)
88
89  # Vektoren anlegen (der zu generierenden Datensaetze)
90  ttt = []
```

```
91  yyy = []
92  Y_Probs = []
93  ProbsY = []
94
95  # Basisdatensatz
96  tt0, yy0 = callBaselineSinusFunction(
        alpha1_init,alpha2_init,alpha3_init,alpha4_init )
97
98  # Werte der Dichtefunktionen + Reshaping (Basisdatensatz)
99  tt0 = tt0.reshape(-1,1)
100 yy0 = yy0.reshape(-1,1)
101
102 # Histogramm-Spline-Approximation (Basisdatensatz)
103 y_probs0,probs0Y = histogramToSplinePDF(yy0)
104
105 # Basisdatensatz dem spaeter finalen Datensatz hinzufuegen
106 ttt.append(tt0)
107 yyy.append(yy0)
108 Y_Probs.append(y_probs0)
109 ProbsY.append(probs0Y)
110
111 # Normierung der relativen Wahrscheinlichkeitsdichtewerte
        (Basisdatensatz)
112 normProbs0Y = probs0Y/sum(probs0Y)
113
114 # Basisdatensatz in CSV-Datei schreiben
115 fileNameBase = 'dataSynthesizedSinusfunktion'
116 writeDataIntoCSV(fileNameBase + '0.csv',y_probs0,normProbs0Y)
117
118 # Iteriere ueber die Anzahl an Samples
119 for i in range(1,N):
120
121     # Variationswerte zufaellig aus Vektoren herausziehen
122     alpha1V.append(choice(alpha1))
123     alpha2V.append(choice(alpha2))
124     alpha3V.append(choice(alpha3))
125     alpha4V.append(choice(alpha4))
126
127     # Zusaetzlicher Datensatz (pro Iteration)
128     tt, yy = callBaselineSinusFunction(
            alpha1V[i],alpha2V[i],alpha3V[i],alpha4V[i] )
129
130     # Werte der Dichtefunktionen + Reshaping (Zusaetzlicher Datensatz)
131     tt = tt.reshape(-1,1)
132     yy = yy.reshape(-1,1)
133
134     # Histogramm-Spline-Approximation (Zusaetzlicher Datensatz)
135     y_probs,probsY = histogramToSplinePDF(yy)
136
137     # Datensatz dem spaeter finalen Datensatz hinzufuegen
138     ttt.append(tt)
139     yyy.append(yy)
140     Y_Probs.append(y_probs)
```

```
141     ProbsY.append(probsY)
142
143     # Normierung der relativen Wahrscheinlichkeitsdichtewerte
            (Zusaetzlicher Datensatz)
144     normProbsY = probsY/sum(probsY)
145
146     # Zusaetzlicher Datensatz in CSV-Datei schreiben
147     writeDataIntoCSV(fileNameBase + str(i) + '.csv',y_probs,normProbsY)
148
149 # Plot -> Fahrkritisches Szenario: Ueberholvorgang mit Gegenverkehr
150 fontsizeBaseline = 14
151 plt.figure(figsize=(12,8))
152 for i in range(len(ttt)):
153     plt.plot(ttt[i], yyy[i])
154 plt.title('Baseline: Sinusfunktion', fontsize=fontsizeBaseline+2)
155 plt.xlabel("Zeit t [s]", fontsize=fontsizeBaseline)
156 plt.ylabel("Auslenkung y [m]", fontsize=fontsizeBaseline)
157 plt.grid()
158 plt.show()
159
160 # Plot -> Wahrscheinlichkeitsdichtefunktionen
161 plt.figure(figsize=(5,5))
162 for i in range(len(ttt)):
163     plt.plot(Y_Probs[i],ProbsY[i])
164 plt.title('Wahrscheinlichkeitsdichteschaetzer:
        Histogramm-Spline-Approximation', fontsize=fontsizeBaseline+2)
165 plt.grid()
166 plt.xlabel("Auslenkung y [m]", fontsize=fontsizeBaseline)
167 plt.ylabel("Relative Wahrscheinlichkeit P(y) [1]",
        fontsize=fontsizeBaseline)
168 plt.show()
```

List. A.1: Python Code: Sunusfunktion (Baseline)

A.2 Überholvorgang

```
# Bibliotheken einbinden
from numpy import asarray, arange, ones, concatenate, zeros, linspace,
    ones_like, histogram, diff, insert, cumsum, trapz
from math import pi, sqrt, sin
from random import gauss, choice
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline

# Berechnung des Integralwerts unter einer 2D-Kurve
def integrateT(x, y):
    area = trapz(y=y, x=x)
    return area

# Histogramm-Spline-Approximation
def histogramToSplinePDF(data,k=11,bc_type='not-a-knot'):
    weights = ones_like(data)/float(len(data))
    heights, bin_borders =
        histogram(data,bins=13,normed=True,weights=weights)
    centers = bin_borders[:-1] + diff(bin_borders) / 2
    bin_width = centers[1]-centers[0]
    t = linspace( centers[0]-bin_width/2,
        centers[-1]+bin_width/2,len(centers)+1 )
    dt = diff(t)
    cumsumTemp = heights * dt
    cumsumTemp = insert(cumsumTemp,0,0)
    Fvals = cumsum(cumsumTemp)
    spl = make_interp_spline(t, Fvals, k=k, bc_type=bc_type)
    splD = spl.derivative()
    t2 =
        linspace(centers[0]-bin_width/2,centers[-1]+bin_width/2,len(data))
    y2 = splD(t2)
    print('Integral-Wert:␣' + str(integrateT(t2,y2)))
    return t2,y2

# Hilfsfunktion - Datensaetze in CSV-Dateien schreiben
def dataWriterHelperCSV(f,key,value,N):
    k = 0
    f.write(key + ',')
    for i in value:
        f.write(str(i))
        if N == 1:
            f.write("")
        if k < N-1:
            f.write(",")
        k = k + 1
    f.write("\n")

# Datensaetze in CSV-Dateien schreiben
def writeDataIntoCSV(filename,x,y):
    N = len(x)
```

```
47      with open(filename,'w') as f:
48          dataWriterHelperCSV(f,'x',x,N)
49          dataWriterHelperCSV(f,'y',y,N)
50
51  # Berechnung der Laenge einer 2D-Kurve
52  def arcLengthCurve(x,y):
53      s = zeros(len(x)-1)
54      s[0] = sqrt((x[1] - x[0])**2 + (y[1] - y[0])**2)
55      for i in range(1,len(x)-1):
56          s[i] = s[i-1] + sqrt((x[i+1] - x[i])**2 + (y[i+1] - y[i])**2)
57      return s[-1]
58
59  # Funktion zur Abbildung eines ueberholvorgangs mit Gegenverkehr
60  def ueberholvorgangFkt(alpha1,alpha2):
61
62      # Variationsparameter des Sicherheitsabstandes des Gegenverkehrs
63      alpha3 = 0
64
65      # Schrittweite in Laengsrichtung
66      xIter=0.05
67
68      # Geschwindigkeit des Ueberholten [m/s]
69      v1=50/3.6
70
71      # Geschwindigkeit des Ueberholenden [m/s]
72      v2=100/3.6
73
74      # Geschwindigkeit Gegenverkehr [m/s]
75      v3=30/3.6
76
77      # Spurbreite [m]
78      sv=3
79
80      # Querbeschleunigung [m/s^2]
81      aq=5
82
83      # Empirischer Faktor
84      K=2.67
85
86      # Sicherheitsabstand vor Ueberholvorgang
87      aVor=15
88
89      # Sicherheitsabstand nach Ueberholvorgang
90      aNach=17
91
92      # Fahrzeuglaenge Ueberholender
93      L1=4
94
95      # Fahrzeuglaenge Ueberholter
96      L2=4
97      sSafe=0+alpha3
98
99      # Geschwindigkeitsunterschied
```

```
100     dv=v2-v1
101
102     # Einscherzeit = Ausscherzeit
103     tE=K*sqrt(sv/aq)
104
105     # Einscherweg = Ausschwerweg
106     sEinscher=tE*dv
107     sAusscher=sEinscher
108
109     # Ueberholweg
110     sa=aVor+L1+aNach+L2
111
112     # Ueberholzeit
113     tUeb=sa/dv
114
115     # Eigentliche Ueberholstrecke
116     sUeb=v2*tUeb
117
118     # Zuruecklegende Strecke Gegenverkehr
119     sGegen=v3*tUeb
120
121     # Safety Margin Safety-Critical
122     SWerf=sUeb+sGegen+sSafe
123
124     # Ueberholstrecke Gegenfahrbahn
125     sP=sUeb-sAusscher-sEinscher
126
127     # Vektor Ausschwerweg Laengsrichtung
128     xA = arange(start=0, stop=sAusscher, step=xIter)
129
130     # Vektor Ausschwerweg Querrichtung
131     y1 = []
132     for _, val in enumerate(xA):
133         y1.append( val*(sv/sAusscher) -
                (1+alpha1)*(sv/(2*pi))*sin((val*2*pi)/sAusscher) )
134     y1 = asarray(y1)
135
136     # Vektor Gegenfahrbahn Laengsrichtung
137     x2 = arange(start=xA[-1]+xIter, stop=sP+xA[-1]+xIter, step=xIter)
138
139     # Vektor Gegenfahrbahn Querrichtung
140     y2 = y1[-1]*ones(len(x2))
141
142     # Vektor Einschwerweg Laengsrichtung
143     x3 = arange(start=x2[-1]+xIter, stop=x2[-1]+xIter+sEinscher,
            step=xIter)
144
145     # Vektor Einschwerweg Querrichtung
146     xE = arange(start=0, stop=sEinscher, step=xIter)
147     y3 = []
148     for _, val in enumerate(xE):
149         y3.append( -((sv/sEinscher)*val -
                (1+alpha2)*(sv/(2*pi))*sin((2*pi*val)/sEinscher)) + sv )
```

```
150     y3 = asarray(y3)
151
152     # Einschwerweg + Gegenfahrbahn + Ausschwerweg (Laengsrichtung &
            Querrichtung)
153     xUeb = concatenate([xA,x2,x3])
154     yUeb = concatenate([y1,y2,y3])
155
156     # Gesamtweg Ueberholvorgang
157     s2Ges = arcLengthCurve(xUeb,yUeb)
158
159     # Gesamtzeit Ueberholvorgang
160     t2Ges = s2Ges/v2
161     t = linspace(0,t2Ges,len(xUeb))
162
163     # Gesamtweg Ueberholte (Laengsrichtung & Querrichtung)
164     xHE = v1*tUeb+aVor
165     xH = linspace(aVor,xHE,len(xUeb))
166     yH = zeros(len(xUeb))
167
168     # Gesamtweg Gegenverkehr (Laengsrichtung & Querrichtung)
169     xG = linspace(SWerf,SWerf-sGegen,len(xUeb))
170     yG = sv*ones(len(xG))
171
172     return xUeb,yUeb
173
174 # Ansatz der Datensaetze bzw. Samples
175 N = 100
176
177 # Variationswerte (initial)
178 alpha1_init = 0
179 alpha2_init = 0
180
181 # Vektor fuer die Variationswerte (zu Beginn gleich der Initialwerte)
182 alpha1V = [alpha1_init]
183 alpha2V = [alpha2_init]
184
185 # Wertebereiche der Variationswerte
186 alpha1 = arange(start=-0.1, stop=0.1, step=0.01)
187 alpha2 = arange(start=-0.1, stop=0.1, step=0.01)
188
189 # Vektoren anlegen (der zu generierenden Datensaetze)
190 ttt = []
191 yyy = []
192 Y_Probs = []
193 ProbsY = []
194
195 # Basisdatensatz
196 tt0, yy0 = ueberholvorgangFkt(alpha1_init,alpha2_init)
197
198 # Werte der Dichtefunktionen + Reshaping (Basisdatensatz)
199 tt0 = tt0.reshape(-1,1)
200 yy0 = yy0.reshape(-1,1)
201
```

```
202 # Histogramm-Spline-Approximation (Basisdatensatz)
203 y_probs0,probs0Y = histogramToSplinePDF(yy0)
204
205 # Basisdatensatz dem spaeter finalen Datensatz hinzufuegen
206 ttt.append(tt0)
207 yyy.append(yy0)
208 Y_Probs.append(y_probs0)
209 ProbsY.append(probs0Y)
210
211 # Normierung der relativen Wahrscheinlichkeitsdichtewerte
        (Basisdatensatz)
212 normProbs0Y = probs0Y/sum(probs0Y)
213
214 # Basisdatensatz in CSV-Datei schreiben
215 fileNameBase = 'dataSynthesizedUeberholvorgang'
216 writeDataIntoCSV(fileNameBase + '0.csv',y_probs0,normProbs0Y)
217
218 # Iteriere ueber die Anzahl an Samples
219 for i in range(1,N):
220
221     # Variationswerte zufaellig aus Vektoren herausziehen
222     alpha1V.append(choice(alpha1))
223     alpha2V.append(choice(alpha2))
224
225     # Zusaetzlicher Datensatz (pro Iteration)
226     tt, yy = ueberholvorgangFkt(alpha1V[i],alpha2V[i])
227
228     # Werte der Dichtefunktionen + Reshaping (Zusaetzlicher Datensatz)
229     tt = tt.reshape(-1,1)
230     yy = yy.reshape(-1,1)
231
232     # Histogramm-Spline-Approximation (Zusaetzlicher Datensatz)
233     y_probs,probsY = histogramToSplinePDF(yy)
234
235     # Datensatz dem spaeter finalen Datensatz hinzufuegen
236     ttt.append(tt)
237     yyy.append(yy)
238     Y_Probs.append(y_probs)
239     ProbsY.append(probsY)
240
241     # Normierung der relativen Wahrscheinlichkeitsdichtewerte
            (Zusaetzlicher Datensatz)
242     normProbsY = probsY/sum(probsY)
243
244     # Zusaetzlicher Datensatz in CSV-Datei schreiben
245     writeDataIntoCSV(fileNameBase + str(i) + '.csv',y_probs,normProbsY)
246
247 # Plot -> Fahrkritisches Szenario: Ueberholvorgang mit Gegenverkehr
248 fontsizeBaseline = 14
249 plt.figure(figsize=(12,8))
250 for i in range(len(ttt)):
251     plt.plot(ttt[i], yyy[i])
252 plt.title('Fahrkritisches␣Szenario:␣Ueberholvorgang␣mit␣Gegenverkehr',
```

```
    fontsize=fontsizeBaseline+2)
plt.xlabel("Laengsrichtung x [m]", fontsize=fontsizeBaseline)
plt.ylabel("Querrichtung y [m]", fontsize=fontsizeBaseline)
plt.grid()
plt.show()

# Plot -> Wahrscheinlichkeitsdichtefunktionen
plt.figure(figsize=(5,5))
for i in range(len(ttt)):
    plt.plot(Y_Probs[i],ProbsY[i])
plt.title('Wahrscheinlichkeitsdichteschaetzer:
    Histogramm-Spline-Approximation', fontsize=fontsizeBaseline+2)
plt.grid()
plt.xlabel("Querrichtung y [m]", fontsize=fontsizeBaseline)
plt.ylabel("Relative Wahrscheinlichkeit P(y) [1]",
    fontsize=fontsizeBaseline)
plt.show()
```

List. A.2: Python Code: Überholvorgang

Literaturverzeichnis

[1] ADAC e.V. Autonomes Fahren: Die 5 Stufen zum selbstfahrenden Auto, 2018. URL https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/grundlagen/autonomes-fahren-5-stufen/?redirectId=quer.5stufen.

[2] ADAC e.V. Autonomes Fahren: Digital entspannt in die Zukunft, 2020. URL https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/technik-vernetzung/aktuelle-technik/.

[3] ADAC e.V. Autonomes Fahren: Gefahr durch Hacker?, 2021. URL https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/recht/autonomes-fahren-hacker-angriff/.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN, 2017.

[5] Association for Safe International Road Travel. Road Safety Facts, 2021. URL https://www.asirt.org/safe-travel/road-safety-facts/.

[6] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos. The Cramer Distance as a Solution to Biased Wasserstein Gradients, 2017.

[7] Bundesministerium für Verkehr und digitale Infrastruktur. Ethik-Kommission: Automatisiertes und vernetztes Fahren, 2017. URL https://www.bmvi.de/SharedDocs/DE/Publikationen/DG/bericht-der-ethik-kommission.pdf?__blob=publicationFile.

[8] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti. *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. 09 2012. ISBN 978-953-51-0748-4. doi: 10.5772/49941.

[9] S.-H. Cha. Comprehensive Survey on Distance/Similarity Measures Between Probability Density Functions. *Int. J. Math. Model. Meth. Appl. Sci.*, 1, 01 2007.

[10] A. Cichocki and S.-i. Amari. Families of Alpha- Beta- and Gamma- Divergences: Flexible and Robust Measures of Similarities. *Entropy*, 12, 06 2010. doi: 10.3390/e12061532.

[11] I. Csiszár. Axiomatic Characterizations of Information Measures. *Entropy*, 10(3):261–273, Sept. 2008. doi: 10.3390/e10030261. URL https://doi.org/10.3390/e10030261.

[12] I. Döbel, M. Leis, M. Vogelsang, D. Neustroev, H. Petzka, A. Riemer, S. Rüping, A. Voss, M. Wegele, and J. Welz. Maschinelles Lernen: Eine Analyse zu Kompetenzen, Forschung und Anwendung, 2018. URL https://www.bigdata-ai.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Fraunhofer_Studie_ML_201809.pdf.

[13] Deloitte. Autonomes Fahren in Deutschland - wie Kunden überzeugt werden, 2016. URL https://www2.deloitte.com/content/dam/Deloitte/de/Documents/consumer-industrial-products/Autonomes-Fahren-komplett-safe-Sep2016.pdf.

[14] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-30958-8. URL https://doi.org/10.1007/978-3-642-30958-8.

[15] D. Foster. *Generatives Deep Learning - Maschinen das Malen, Schreiben und Komponieren beibringen*. O'Reilly, Sebastopol, 2020. ISBN 978-3-960-10356-1.

[16] Fraunhofer-Institut für Kognitive Systeme IKS. Künstliche Intelligenz (KI) und maschinelles Lernen, 2021. URL https://www.iks.fraunhofer.de/de/themen/kuenstliche-intelligenz.html.

[17] P. Gardner, C. Lord, and R. Barthorpe. An Evaluation of Validation Metrics for Probabilistic Model Outputs. page V001T06A001, 05 2018. doi: 10.1115/VVS2018-9327.

[18] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks, 2017.

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks, 2014.

[20] Infineon Technologies AG. Vom assistierten zum automatisierten Fahren. URL https://www.infineon.com/cms/de/discoveries/adas-to-ad/.

[21] F. Kallmeyer. Autonomes Fahren - Chancen und Herausforderungen, 12 2019. URL https://www.zukunft-mobilitaet.net/170765/strassenverkehr/autonomes-fahren-chancen-und-herausforderungen-sae-level5/.

[22] J.-P. Kreiß and G. Neuhaus. *Einführung in die Zeitreihenanalyse*. Springer-Verlag, Berlin Heidelberg New York, 2006. ISBN 978-3-540-33571-9.

[23] R. Marchthaler and S. Dingler. *Kalman-Filter*. Springer Fachmedien Wiesbaden, 2017. doi: 10.1007/978-3-658-16728-8. URL https://doi.org/10.1007/978-3-658-16728-8.

[24] D. M. Meko. Time Series Analysis. URL https://imedea.uib-csic.es/master/cambioglobal/Modulo_V_cod101615/Theory/TSA_theory_part1.pdf.

[25] H.-J. Mittag and K. Schüller. *Statistik*. Springer Berlin Heidelberg, 6 edition, 2020. doi: 10.1007/978-3-662-61912-4. URL https://doi.org/10.1007/978-3-662-61912-4.

[26] A. Muñoz, G. Martos, and J. González. Level Sets Based Distances for Probability Measures and Ensembles with Applications, 2015.

[27] D. Parthasarathy, K. Bäckstrom, J. Henriksson, and S. Einarsdóttir. Controlled time series generation for automotive software-in-the-loop testing using GANs. In *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 39–46, 2020. doi: 10.1109/AITEST49225.2020.00013.

[28] J. Puhani. *Statistik*. Springer Fachmedien Wiesbaden, 13 edition, 2020. doi: 10.1007/978-3-658-28955-3. URL https://doi.org/10.1007/978-3-658-28955-3.

[29] S. Raschka and V. Mirjalili. *Python Machine Learning - Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd., Birmingham, UK, 3 edition, 2019. ISBN 978-1-789-95829-4.

[30] Y. Rubner, C. Tomasi, and L. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40:99–121, 11 2000. doi: 10.1023/A:1026543900054.

[31] N. Schick. *Modeling of specific safety-critical driving scenarios for data synthesis in the context of autonomous driving software*. Cuvillier Verlag, Göttingen, 2020. ISBN 978-3-736-96246-0.

[32] N. Schick. *Entwurf eines KI-Systems zur automatisierten Erzeugung von realitätsnahen kritischen Fahrsituationen für SW-Algorithmen des autonomen Fahrens*. Dissertation, FernUniversität Hagen, 2021 (wird noch veröffentlicht).

[33] Statistisches Bundesamt. Unfallentwicklung auf deutschen Straßen 2017, 2018. URL https://www.destatis.de/DE/Presse/Pressekonferenzen/2018/Verkehrsunfaelle-2017/statement-nfallentwicklung.pdf?__blob=publicationFile.

[34] D. J. Weller-Fahy, B. J. Borghetti, and A. A. Sodemann. A Survey of Distance and Similarity Measures Used Within Network Intrusion Anomaly Detection. *IEEE Communications Surveys Tutorials*, 17(1):70–91, 2015. doi: 10.1109/COMST.2014.2336610.

[35] L. Weng. From GAN to WGAN, 2019.

[36] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. doi: 10.1109/TIP.2003.819861.